Ph.D. DISSERTATION

Adaptive Deep Image Signal Processor for Practical Applications

실용적인 애플리케이션을 위한 적응형 심층 이미지 시그널 프로세서

BY

HEEWON KIM

FEBRUARY 2023

DEPARTMENT OF
ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Adaptive Deep Image Signal Processor for Practical Applications

실용적인 애플리케이션을 위한
적응형 심층 이미지 시그널 프로세서

지도교수 이 경 무
이 논문을 공학박사 학위논문으로 제출함
2023 년 2 월

서울대학교 대학원
전기·정보공학부
김 희 원

김희원의 공학박사 학위논문을 인준함
2023 년 2 월

Abstract

Cameras have become indispensable in everyday life for history recording, social network services, video meetings, and personal broadcasting. Smartphone manufacturers (*e.g.*, Samsung and Apple) adopt more and bigger camera lenses for new generations, given that camera performance plays a key role in smartphone market share. In an image pipeline of digital cameras, an image sensor converts light from an object into a digital signal called a raw image, and an Image Signal Processor (ISP) transforms the raw image into a human-readable RGB image. An ISP performs a variety of image processing tasks related to restoration and enhancement, where image restoration aims to restore an original image from its corrupted version and image enhancement aims to retouch images attractive.

The challenge in ISPs addresses designing and implementing a general model across all image degradation or all image styles. Specifically, image details are corrupted by many types of degradation such as noise, blur, and compression. A general model to handle all degradation not only requires a sufficiently large number of model parameters but also is not optimal for each degradation. Moreover, attractive image styles are subjective and vary depending on many factors such as personal experience, atmosphere, and mood. Recent deep enhancement models usually generate a single image style for an input image which is limited to satisfy user preferences.

This dissertation proposes adaptive deep ISPs for practical applications. Specifically, we introduce the three adaptation methods for three applications: (1) adaptive data synthesis for camera image denoising, (2) adaptive neural architecture search for controllable image restoration, and (3) adaptive ISP parameter estimation for controllable image enhancement. First, in camera image denoising, it is difficult to obtain noisy-clean image pairs for supervised learning. The proposed method generates RGB noisy-clean image pairs at low-resolution from raw-RGB noisy image pairs and allows the accurate training of general CNN-based denoisers. Second, controllable image restoration is a new application for unknown degradation that aims to generate outputs for predetermined restoration tasks and select the desired result for user preferences. The proposed method automatically finds a neural network architecture that is efficient for multiple inferences to generate different restoration outputs. The searched network shares the early layers and adapts the remaining layers to each task. Third, the proposed method learns style representation that can generate multiple high-quality image styles to satisfy subjective user preferences. Users select styles in the latent representation, and a neural network decodes the style into ISP parameters. Style generation is efficient through a plug-and-play ISP.

The proposed methods improve significant performances in each practical computer vision application, and empirical analyses and ablation studies show the effectiveness of the proposed methods. We present the improvement of image quality and model efficiency in benchmarks widely used in each task and real images.

Key words: Image Signal Processor, Image Restoration, Image Enhancement, Deep Learning, Data Synthesis, Neural Architecture Search, ISP Parameter Estimation

Student number: 2017-27265

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Digital Camera

Digital cameras, capturing and storing photographs in digital form rather than on film, have become very popular in recent decades with a number of advantages over traditional film cameras. For one, they allow viewing the images immediately after capturing them. This is particularly useful for obtaining better image quality by taking multiple shots and choosing the best one. Digital cameras also allow sharing images or videos (*i.e.*, multiple consecutive images) easily, either by uploading them to the internet or transferring them to a computer. Smartphones have built-in digital cameras that are portable, convenient, and easy to share images for social network services, video meetings, and personal broadcasting. Camera performances have been playing a key role in the smartphone market, with many consumers considering it in their purchasing decision. Smartphone manufacturers (*e.g.*, Samsung and Apple) have responded by investing heavily in camera technology and offering advanced features such as multiple lenses, large image sensors, and software-based image processing.

When the camera takes a picture, light from the scene is focused onto an image sensor through the lens. The image sensor is a light-sensitive electronic component that converts light into electrical signals. It is made up of millions (recently, billions) of tiny photocells, also known as pixels, that are responsible for capturing the light entering the camera. For capturing color images, a color filter array (*e.g.*, Bayer pattern) passes a specific color (*e.g.*, red, green, or blue) to each photocell. The photocells are activated by the incoming light and generate electrical signals in proportion to the intensity of the light. An image signal processor converts these electric signals, also called raw data or raw images, into human-readable color images through a variety of image processing tasks.

1.2 Image Signal Processor (ISP) and Its Challenges

An image signal processor (ISP) converts raw image data from image sensors into usable formats for imaging devices. Specialized chips (*e.g.*, Apple A16 Bionic) typically incorporate ISPs for high-speed and low-power processing. ISPs perform a variety of image processing tasks including denoising, deblurring, demosaicing, compression artifact removal, white balancing, tone mapping, and retouching, which fall into two categories: image restoration and enhancement with their own challenges.

1.2.1 Image Restoration: Unknown Degradation

Image restoration aims to restore the original appearance of a real-world scene that has been degraded. Various factors occur degradation such as hardware limitations in image capture and software processing for storage or bandwidth optimization.

There are four main types of degradation: First, the discretization of continuous analog signals (light) into digital signals, where high-resolution images with more pixels can better represent realistic textures than low-resolution images. Second, image sensors produce noise during light-to-electrical signal conversion, due to an inconsistent number of photons and random fluctuations in electron counts indicating light intensity. Third, digital images can be blurred by out-of-focus or camera/object movement during exposure. Fourth, lossy compression methods reduce image file size at the cost of image quality.

Deep neural networks have achieved breakthroughs in image restoration by leveraging supervised learning with training datasets of original-degraded image pairs. However, the degradation of test images is unknown. Models learned from these datasets fail to restore test images that are degraded differently from the datasets. Additionally, a model that is capable of restoring all types of degradation requires a sufficiently large model size and is not optimal for specific types of degradation.

1.2.2 Image Enhancement: Subjective User Preferences

The goal of image enhancement is to improve the visual appeal of an image by adjusting various factors such as contrast, intensity, colors, and tones. ISPs perform image enhancement to transform raw images into RGB images, which are visually similar or more pleasing to the original scene, through the following steps:

First, scaling the pixel values to the appropriate range of the image format. Second, adjusting the colors with respect to illumination and the RGB spectra of the color filters. Third, allocating more bits to display low- and high-intensity for human eye perception and high dynamic range.

Similar to image restoration, deep neural networks have recently made significant progress in image enhancement through supervised learning. They learn the image transformation from low-quality images to high-quality images, as defined by

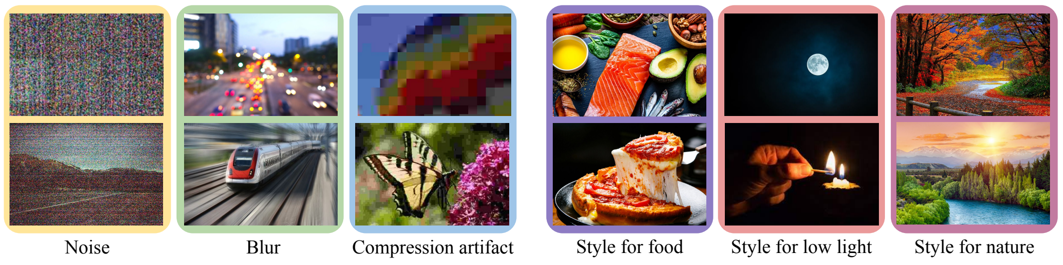


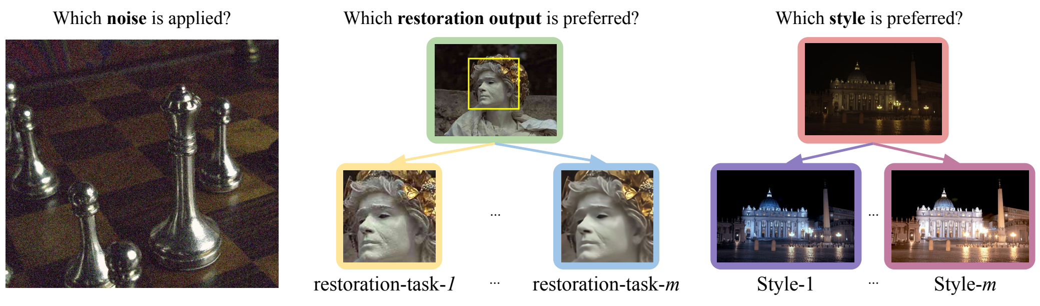
Figure 1.1: Example images for various degradations and styles.

training datasets, and generate an enhanced image for a given test image. However, the single image output is limited in satisfying user preferences as each user has different desired styles and even the same user prefers different styles in different moods and environments.

1.3 Adaptation in Image Signal Processor

The ISP is responsible for restoring and enhancing digital images, which incorporate various degradations and have diverse visually appealing styles visualized in Figure 1.1. A general-purpose ISP with fixed parameters leads to suboptimal results as it only generates a single output restoration or style for a given test image, given that an image has multiple possible solutions depending on the degradation applied and the user preference. Conversely, an ISP that has specialized parameters for each degradation or style achieves high image quality but requires too many parameters and computations to generate all possible output restorations or styles.

This dissertation introduces an adaptive ISP to overcome these limitations. The adaptive ISP adjusts its parameters, features, or architectures based on the specific degradation or style. This adaptation allows the ISP to efficiently handle a wide range of degradations and styles with one model. The research process of the adaptive


Which **noise** is applied?
Which **restoration output** is preferred?
Which **style** is preferred?
restoration-task-*1*
...
restoration-task-*m*
Style-1
...
Style-*m*

(a) Camera Image Denoising (b) Controllable Image Restoration (c) Controllable Image Enhancement

Figure 1.2: Examples and key questions about practical applications of interest.

ISP involves determining the image processing tasks in an ISP pipeline, analyzing the nature of the task, and designing the ISP components for adaptation. This dissertation presents various deep-learning-based approaches for the adaptive ISP and aims to inspire future research.

1.4 Practical Applications

This dissertation explores three practical applications of image restoration and enhancement for adaptive ISPs, each addressing unique image processing tasks with different degradations or styles. Figure 1.2 presents examples and key questions about the applications, and this section briefly describes each application.

1.4.1 Camera Image Denoising

Camera image denoising aims to eliminate noise from RGB images. This is a practical application as all digital images contain camera noise, which is caused by the image sensor, camera setting, and image styles. Camera manufacturers typically estimate the noise model of their cameras and develop a denoiser for it. This dissertation considers a denoiser tailored to the noise in test images as an adaptive ISP for camera image denoising.

1.4.2 Controllable Image Restoration

Camera noise can be accurately estimated based on strong noise assumptions (*e.g.*, Poisson-Gaussian noise), but most image degradations are hard to model. Instead of estimating degradations, controllable image restoration has recently gained interest as a practical application for handling various unknown degradations. This application defines a set of restoration tasks and generates restoration outputs for each task, allowing users to choose their preferred output. This dissertation regards a model for controllable image restoration as an adaptive ISP.

1.4.3 Controllable Image Enhancement

Controllable applications are intuitively understandable in image enhancement. The models in this area generate multiple styles for an input image by adjusting some control factors, allowing users to select their preferred style. This approach is widely accepted in image editing tools, as it caters to individual preferences. Recent works focus on the image quality of output styles, ease of generating high-quality styles, and model efficiency using deep neural networks. This dissertation proposes a controllable image enhancement model with an adaptive ISP.

1.5 Outline of the Dissertation

This dissertation introduces three deep-learning-based approaches that facilitate adaptive ISPs for practical computer vision applications: camera image denoising, controllable image restoration, and controllable image enhancement. Each proposed approach tackles three different current issues associated with adaptive ISPs, through data synthesis, neural architecture search, and ISP parameter estimation,

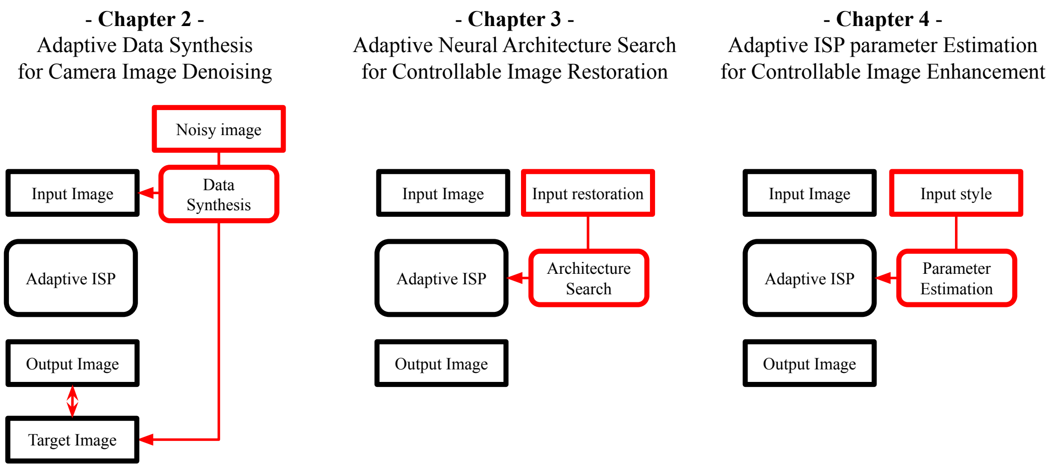


Figure 1.3: Overview of the proposed adaptive ISP methods.

to take one step closer to deep neural networks that can promptly produce visually-pleasing images in mobile devices. Figure 1.3 summarizes the proposed approaches.

In Chapter 2, we aim to train accurate denoising networks for smartphone/digital cameras from raw-RGB noisy image pairs. Downscaling is commonly used as a practical denoiser for low-resolution images. Based on this processing, we found that the pixel variance of natural images is more robust to downscaling than the pixel variance of camera noise. Intuitively, downscaling removes high-frequency noise more easily than natural textures. To utilize this property, we can adopt noisy/clean image synthesis at low-resolution to train camera denoisers. On this basis, we propose a new solution pipeline (NERDS) that estimates camera noise and synthesizes noisy-clean image pairs from only noisy images. In particular, it first models the noise in raw-sensor images as Poisson-Gaussian distributions, then estimates noise parameters using the difference of pixel variances by downscaling. We formulate the noise estimation as a gradient-descent-based optimization problem through a reparametrization trick. We further introduce a new RAW2RGB conversion estimation method that enables denoiser training in a human-readable RGB space by transforming the downscaled raw images to the style of a given RGB noisy image. The noise and RAW2RGB conversion estimations utilize rich augmentation to synthesize image pairs for denoiser training. Experiments show that NERDS can accurately train CNN-based denoisers (*e.g.*, DnCNN, ResNet-style network) outperforming previous noise-synthesis-based and self-supervision-based denoisers in real datasets.

In Chapter 3, we present a novel framework for controllable image restoration that can effectively restore multiple types and levels of degradation of a corrupted image. The proposed model, named TASNet, is automatically determined by our neural architecture search algorithm, which optimizes the efficiency-accuracy trade-off of the candidate model architectures. Specifically, we allow TASNet to share the early layers across different restoration tasks and adaptively adjust the remaining layers with respect to each task. The shared task-agnostic layers greatly improve the efficiency while the task-specific layers are optimized for restoration quality, and our search algorithm seeks for the best balance between the two. We also propose a new data sampling strategy to further improve the overall restoration performance. As a result, TASNet achieves significantly faster GPU latency and lower FLOPs compared to the existing state-of-the-art models, while also showing visually more pleasing outputs.

In Chapter 4, we present a plug-and-play Image Signal Processor (ISP) for image enhancement to better produce diverse image styles than the previous works. Our proposed method, *ContRollable* Image Signal Processor (CRISP), explicitly controls the parameters of the ISP that determine output image styles. ISP parameters for high-quality (HQ) image styles are encoded into low-dimensional latent codes, allowing fast and easy style adjustments. We empirically show that CRISP covers a wide range of image styles with high efficiency. On the MIT-Adobe FiveK dataset, CRISP can very closely estimate the reference styles produced by human experts and achieves better MOS with diverse image styles. Compared with the state-of-the-art method, our ISP comprises only 19 parameters, allowing CRISP to have 2$\times$ smaller parameters and 100$\times$ reduced FLOPs for image output. CRISP outperforms previous works in PSNR and FLOPs with several scenarios for style adjustments.

This dissertation concludes with a summary and discussions for future research in Chapter 5.

Chapter 2

Adaptive Data Synthesis for Camera Denoiser Training

2.1 Introduction

Image denoising is a conventional machine learning problem restoring original colors and patterns from noisy images. Deep-learning-based approaches have achieved breakthroughs in recent decades due to the power of neural networks. Early works [127, 84, 101] have successfully removed additive white Gaussian noise (AWGN), which allows network training under supervision by synthesizing noisy-clean image pairs.

Nevertheless, denoising images captured by smartphone/digital cameras poses an obstacle, as it is difficult to obtain clean images for noisy images with pixel-level alignment. Several works [2, 13] constructed datasets with the noisy-clean pairs for real-world images. Using these pairs (Figure 2.1(a)), many supervised-learning-based denoisers [121, 63, 122, 47, 25] restore crisp images on benchmarks from the datasets. However, constructing such datasets requires tightly controlled capturing environments, complicated post-processing, and massive human labor.

To overcome the drawback of plain supervised learning, two major types of research have been studied. The first line of works synthesizes the realistic noise from clean images to utilize supervised denoiser training as visualized in Figure 2.1(b). Several approaches [23, 20, 45, 54] adopt generative models using *unpaired* noisy-clean images based on GAN [38], but they achieve limited accuracy on real noises. Some other works synthesize more realistic noises using existing noisy-clean image pairs [123, 1] or metadata for real cameras [12, 41], but they are limited in generalization for unseen noises. The second category aims to learn denoisers *without clean images*. The first work [73] in this category proposed the learning framework using multiple noisy images. After that, many self-supervised-learning approaches [9, 68, 16] use single noisy images (Figure 2.1(c)), which enable easy data collection and the denoiser adaptation to the test noises. However, they are still limited in real-world applications due to the requirements of custom network architectures and strong statistical noise assumptions.

To address the above limitations on camera denoiser training, we propose a new solution pipeline, namely NERDS, to perform noisy-clean image pair synthesis and accurate denoiser training from single noisy images. The framework composes three parts–noise estimation, RAW2RGB conversion estimation, and denoiser training. We found that the pixel variance of natural clean images is robust to the image downscaling, which is a widely-used denoiser used for low-resolution images [89]. The noise estimation adopts a Poisson-Gaussian noise model for the raw images from sensors and optimizes the noise parameters by the pixel variances of downscaled images. The downscaled images and the estimated noise parameters enable the synthesis of noisy-clean image pairs at low-resolution (Figure 2.1(d)). Training denoisers on human-readable RGB images from real cameras has another issue: the conversion

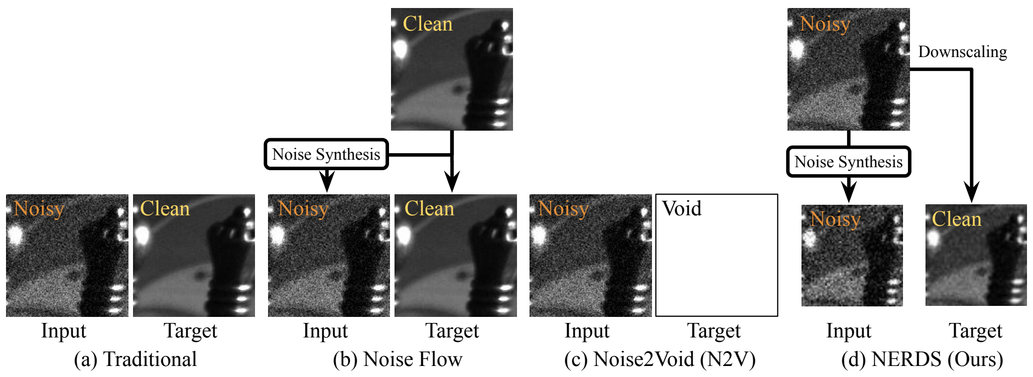


Figure 2.1: Different training schemes for CNN-based camera denoisers. (a) Traditionally, training denoisers requires pairs of noisy and clean images. However, clean target images are difficult to obtain from smartphone/digital cameras. (b) Noise Flow [1] generates realistic synthetic noise from clean images by learning the noise distributions using existing pairs of real images. (c) N2V [68] enables more practical training from single noisy images without clean targets but requires custom network architectures. (d) Our NERDS synthesizes noisy-clean image pairs at low-resolution by utilizing image downscaling as a general denoiser and noise estimation through gradient-descent-based optimization.

from the raw images to the RGB images is a black box. Our RAW2RGB conversion estimation enables the noise synthesis on the RGB space by learning the RAW2RGB conversion using the raw-RGB image pairs[1]. Our denoiser training can utilize rich data augmentation based on estimated noise parameters and ISPs. Specifically, we introduce two techniques for this framework. First, a reparametrization trick allows estimating noise parameters through a gradient-descent-based optimizer. Second, a technique for style disentanglement from raw-RGB noisy image pairs.

We summarize our contributions as follows:

- To the best of our knowledge, this is the first work to synthesize RGB noisy-clean image pairs at low-resolution for accurate camera denoiser training from single noisy images.

- We formulate noise estimation for Poisson-Gaussian noises as an optimization

[1]Major camera manufacturers (*e.g.*, Samsung, Apple, Xiaomi, Cannon, and Sonny) provide raw and RGB image pairs on their devices.

problem, and a novel reparameterization trick allows to estimate accurate noise parameters through gradient-descent.

- We propose a neural network that estimates the RAW2RGB conversions used for given raw-RGB noisy image pairs. The RAW2RGB conversion estimation generates realistic RGB noisy-clean image pairs from raw images.
- Our frameworks can train general CNN-based denoisers (*e.g.* DnCNN, ResNet-style network) accurately for given test noisy images by performing noise synthesis using them.

2.2 Related Works

2.2.1 Blind Image Denoising

Traditional methods. Classical methods usually denoise noisy images without training data using wavelet [32], filtering [14] including BM3D [28], optimization [33, 83, 39], and effective prior [134]. However, they perform limited accuracy compared to the recent deep-learning-based approaches.

Supervised-learning-based methods. SIDD [2] and NIND [13] captured real noisy-clean image pairs to enable supervised-learning for real camera denoisers. However, the capturing procedure is unacceptably expensive and cumbersome. We describe the literature on realistic noise synthesis methods for the supervised-learning. DnCNN [127] introduced a neural network to remove additive white Gaussian noise (AWGN) for the first time. However, Guo *et al.* [41] demonstrated the limitation of AWGN denoisers to signal-dependent or spatially-correlated noises, which are known as the characteristics of real-world images. To alleviate this problem, two lines of works have been researched. The first category uses generative models for noise synthesis with stable learning [23], various noise characteristics [54], knowledge distillation [109], self-supervised-learning [20], and conditional adversarial networks [45]. Nevertheless, the scene statistics mismatched between clean and noisy datasets make it difficult to train accurate denoisers in practice.

The second category investigates camera noise modeling. CBDNet [41] transforms AWGN into realistic noise by using signal-dependent noise parameters and simulating in-camera ISP functions, such as gamma correction and demosaicing. UPI [12] converts RGB images to raw images and simulates noises using metadata of specific cameras. CycleISP [123] trains neural networks for both RAW2RGB and RGB2RAW conversions on large-scale datasets with specific ISP and noise settings. Noise Flow [1] synthesizes raw noisy images using normalizing flow and metadata. The work on [21] proposed a GAN-based framework for noise generation in raw images which is adaptive to the camera. In [130], the authors claimed that the noise levels in metadata are inaccurate. Recently, the method in [67] models the RGB noise distribution using normalizing flow. SCUNet [126] proposed a practical noise model for a general-purpose denoiser.

However, all the above methods require real noisy-clean image pairs, ISPs, metadata, or high-quality clean images which are not always available. More importantly, the noise synthesis using predetermined training datasets leads to poor denoising performances on unseen noises. In contrast, the proposed method (NERDS) synthesizes realistic noisy-clean image pairs from only noisy images, which enables accurate denoiser training specialized in the test images.

Self-supervised-learning-based methods. Noise2Noise [73] introduced a framework that trains denoisers using noisy images *only* for the first time. Noise2Void [68], Noise2Self [9], Noise2Same [111], and Neighbor2Neighbor [50] adopted advanced approaches which can train denoisers with single images corrupted by the i.i.d noise. Noisier2Noise [87], NAC [113], and R2R [90] add additive noises to the given noisy images to make auxiliary training pairs by using prior knowledge (or assumptions) of the noise distribution. Notably, the blind-spot network (BSN) in [68] has been improved by efficient architectures with small receptive fields [69] and dilated convolutions [109]. Using the advanced BSNs, FBI-D [16] adopts a denoiser specialized in Poisson-Gaussian noise for real-world raw image denoising. AP-BSN [72] breaks the spatially-correlated noise of real-world RGB images by pixel-shuffle downsampling.

Although the above self-supervised-learning methods use practical training datasets (only noisy images), they require custom network architectures or strong noise assumptions. In contrast, NERDS performs noise synthesis that allows training general CNN-based denoisers based on the noise modeling for smartphone/digital cameras.

2.2.2 Noise Estimation

Prior knowledge of noise distribution supports accurate restoration in most methods for image denoising, but it is not generally available in practice. Noise estimation methods alleviate this problem, especially for the additive white Gaussian noise (AWGN) and the Poisson-Gaussian noise. Principal component analysis (PCA) based approaches [80, 94, 22] perform accurate AWGN estimation. For the Poisson-Gaussian noise model, most existing methods [3, 102] including [34, 81] adopt two-step approaches; estimating the local means and variances, then adopting maximum likelihood estimation (MLE) to fit the noise model. Foi *et al.*[34] proposed the

Poisson-Gaussian noise model for the first time and a noise estimation algorithm based on wavelet decomposition. Liu *et al.* [81] adopted iterative patch selection for the generalized source-dependent noise. Recently, PGE-Net [16] adopted neural networks for accurate and fast optimization based on Generalized Anscombe Transformation (GAT) [8]. In contrast, we proposed a Poisson-Gaussian noise estimation method using natural scene statistics with gradient-descent-based optimization.

2.3 Preliminary: Noise Modeling

In a digital camera, an image sensor converts light into a digital signal (or a raw image), and an image signal processor (ISP) converts it into a human-readable RGB image. We regard that the noise of RGB images originates from the image sensor and is transformed by the ISP. Thus, we model the noise distribution of raw images and the RAW2RGB conversion including ISPs.

Raw Image: Poisson-Gaussian (P-G) Noise. A common noise model for raw images follows the Poisson-Gaussian distribution [34], defined as:

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{z}, \beta_1^2 \boldsymbol{z} + \beta_2^2), \tag{2.1}$$

which is a heteroscedastic Gaussian where $\boldsymbol{z}$ is the true signal, $\boldsymbol{x}$ is a raw noisy image observed on real image sensors, and β_1, $\beta_2 \geq 0$ are signal-dependent and signal-independent noise parameters. Real cameras provide $\boldsymbol{x}$ as well as β_1 and β_2 in metadata. However, the noise parameters in the metadata are often inaccurate [130]. Section 2.6.1 discusses the noise distribution of raw images and the noise parameters in metadata further.

RGB Image: Transformed Noise. ISPs transform raw images into RGB images using nonlinear functions, such as demosaicing, white balancing, color correction, and tone mapping. The transformation changes the noise distribution, such as breaking the i.i.d property in (2.1). Moreover, users can retouch image tones and colors, altering the noise distribution further. We define such RAW2RGB conversion ($\boldsymbol{T}$) as a function of the image style ($\boldsymbol{s}$),

$$\boldsymbol{y} \equiv T(\boldsymbol{x}; \boldsymbol{s}), \tag{2.2}$$

where $\boldsymbol{x}$ is a raw noisy image obtained from real image sensors and $\boldsymbol{y}$ is the RGB noisy image with custom tones and colors. The real noise datasets [2, 93] used a simple and open-source ISP for the entire dataset without any image retouching processes. However, modern smartphones and digital cameras use custom ISPs with hidden internal functions. They provide multiple image styles depending on the scenes with user edits.

The following section provides a general framework to estimate P-G noise parameters (β_1, β_2), the RAW2RGB conversion ($\boldsymbol{T}$), and the image style ($\boldsymbol{s}$) using only noisy images ($\boldsymbol{x}$, $\boldsymbol{y}$).

2.4 Proposed Method

2.4.1 Overview

For a given noisy pair of raw ($\boldsymbol{x}$) and RGB ($\boldsymbol{y}$) images, the proposed method composes three steps to restore the clean RGB image without auxiliary training data. First, we estimate noise parameters (β_1, β_2) from $\boldsymbol{x}$ (Section 2.4.3). Second, we learn

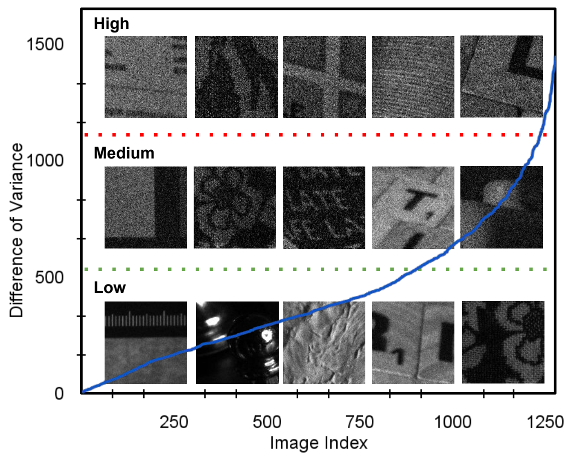


Figure 2.2: The ranked curve for the difference of pixel variances through downscaling on images from SIDD validation set and the visualization of noise levels.

the conversion from $\boldsymbol{x}$ to $\boldsymbol{y}$ while disentangling the image style $\boldsymbol{s}$ (Section 2.4.4). Third, we synthesize diverse pairs of noisy-clean images to train arbitrary RGB denoisers with a general supervised-learning framework (Section 2.4.5).

2.4.2 Observation

We first illustrate our observation on raw noisy images with downscaling. Specifically, we investigate the statistical characteristics variances of 256×256 images in SIDD validation dataset [2]. To evaluate their noise levels, we downscale all raw images for each color channel with the scaling factor of 2, and rank these images according to the differences of pixel variances through downscaling. As visualized in Figure 2.2, we show these values in a blue curve and separate them into three levels –low, medium, high. It is observed that the images with high differences are very noisy, while the images with low differences present clean textures. This phenomenon indicates that the true signal $\boldsymbol{z}$ is robust to the pixel variances through downscaling than real noises. That is why we propose the following method, which uses downscaled images and pixel variances to estimate the noise parameters of original images.

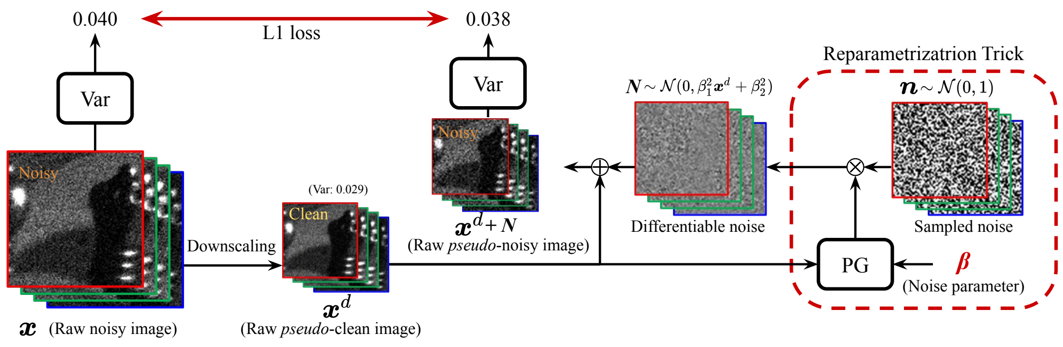

0.040
L1 loss
0.038
Var
Var
Reparametrizatrion Trick
$\boldsymbol{N} \sim \mathcal{N}(0, \beta_1^2 \boldsymbol{x}^d + \beta_2^2)$
$\boldsymbol{n} \sim \mathcal{N}(0,1)$
Noisy
(Var: 0.029)
Clean
Noisy
$\boldsymbol{x}^d + \boldsymbol{N}$
(Raw *pseudo*-noisy image)
Differentiable noise
Sampled noise
Downscaling
PG
$\boldsymbol{\beta}$
(Noise parameter)
$\boldsymbol{x}^d$
(Raw *pseudo*-clean image)
$\boldsymbol{x}$ (Raw noisy image)

Figure 2.3: Noise estimation from a raw noisy image with a reparametrization trick.

2.4.3 Poisson-Gaussian Noise Estimation

We aim to estimate noise for $\boldsymbol{x}$ without any information other than $\boldsymbol{x}$. For this, we find an additive noise ($\boldsymbol{N}$) for the downscaled image ($\boldsymbol{x}^d$) by solving the following optimization problem,

$$\min_{\boldsymbol{N}} |Var(\boldsymbol{x}) - Var(\boldsymbol{x}^d + \boldsymbol{N})|, \tag{2.3}$$

where $\boldsymbol{N} \sim \mathcal{N}(0, \beta_1^2 \boldsymbol{x}^d + \beta_2^2)$ is the Poisson-Gaussian noise discussed in (2.1) and $Var(\cdot)$ denotes a function that outputs the pixel variance of the input image. We can approximate β_1 and β_2 as the noise parameters for $\boldsymbol{x}$, given that downscaling reduces noise levels and that the pixel variance of an image is correlated with the variance of the noise distribution, as discussed in Section 2.4.2. Section 2.6.2 analyzes the downscaling effect further. However, (2.3) is difficult to optimize due to the non-differentiable process of noise sampling. To alleviate this problem, we introduce a reparameterization trick that separates the noise sampling into learnable parameters and the sampling from the normal distribution. Formally, we reformulate (2.3) as

$$\min_{\boldsymbol{\beta}} |Var(\boldsymbol{x}) - Var(\boldsymbol{x}^d + PG(\boldsymbol{\beta}, \boldsymbol{x}^d) \times \boldsymbol{n})|, \tag{2.4}$$

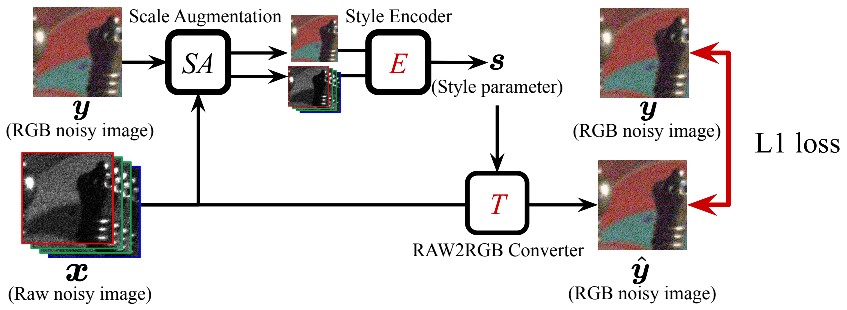


Figure 2.4: RAW2RGB conversion estimation from a pair of raw-RGB noisy images with style disentanglement.

where $PG(\boldsymbol{\beta}, \boldsymbol{x}^d) = (\beta_1^2 \boldsymbol{x}^d + \beta_2^2)^{0.5}$ denotes the Poisson-Gaussian converter and $\boldsymbol{n} \sim \mathcal{N}(0, 1)$ is the additive noise with the normal distribution. Given that the only learnable parameters in (2.4) are β_1 and β_2, a gradient-descent optimizer (*e.g.*, Adam [64]) easily finds the noise parameters. Figure 2.3 visualizes the optimization problem used as noise estimation for $\boldsymbol{x}$.

We are surprised that such a simple optimization problem estimates accurate noise parameters, while downscaled images can contain remaining noise or over-smoothing textures. This result is because the well-designed optimizer learns the characteristics of signal-dependent and signal-independent noises over the entire image content.

So far we have discussed noise estimation and synthesis on raw images. For more practical applications, the following section describes RGB noise synthesis by estimating RAW2RGB conversion.

2.4.4 RAW2RGB Conversion Estimation

We aim to learn a RAW2RGB conversion from $\boldsymbol{x}$ to $\boldsymbol{y}$. However, a naìve network training from $\boldsymbol{x}$ to $\boldsymbol{y}$ can easily overfit to the image contents. Moreover, to generate multiple styles of RGB images, training a neural network for each style is resource intensive. For this, we disentangle the styles from RGB images, allowing a network to learn multiple-style generation.

Specifically, our RAW2RGB conversion estimation composes two networks, the style encoder (E) and the RAW2RGB converter (T) (Figure 2.4). E identifies the image style as a style parameter ($\boldsymbol{s}$) from the raw noisy image ($\boldsymbol{x}$) and the RGB noisy image ($\boldsymbol{y}$) while T uses $\boldsymbol{x}$ and $\boldsymbol{s}$ as input to generate an RGB image ($\hat{\boldsymbol{y}}$). Then, we can recall the equation (2.2) as

$$\hat{\boldsymbol{y}} = T(\boldsymbol{x}; E(\boldsymbol{x}, \boldsymbol{y})), \tag{2.5}$$

where $\boldsymbol{s} = E(\boldsymbol{x}, \boldsymbol{y})$ while training E and T to minimize the L1 difference between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$. E composes 6 residual blocks, global pooling, and a fully connected layer and T composes 4 residual blocks and ISP functions of digital gain, white balance, color correction, gamma correction, and tone mapping. We use the residual blocks of two convolutional layers and one ReLU activation with 64 filters and 3×3 kernels. The output of E is channel attention coefficients of residual features and the parameters of ISP functions in T. The training data can compose a single pair or multiple pairs of raw-RGB noisy images. To avoid learning identity mapping from the input $\boldsymbol{y}$ to the output $\hat{\boldsymbol{y}}$, we first adopt image scale augmentation (SA) that changes the image resolution. Each scaling factor transforms the noise of $\boldsymbol{x}$ and $\boldsymbol{y}$ while SA randomly samples the scaling factor to prevent the encoder from learning noises. Formally, we redefine the style parameter ($\boldsymbol{s}$) as follows,

$$\boldsymbol{s} \equiv E(SA(\boldsymbol{x}, \boldsymbol{y})). \tag{2.6}$$

Second, we design a bottleneck structure that lowers the dimension of $\boldsymbol{s}$ to avoid encoding of information about image contents. E reduces channels and resolutions of $\boldsymbol{s}$ ($\boldsymbol{s} \in \mathbb{R}^{3\times1\times1}$ in this chapter). $\boldsymbol{s}$ conditions T by channel attention and ISP parameter generation via fully connected layers.

2.4.5 Denoiser Training

We synthesize noisy-clean RGB image pairs at low-resolution to train general RGB denoisers. The noisy/clean images are the downscaled raw image ($\boldsymbol{x}^d$) with/without additive noise $\boldsymbol{N}$ transformed to RGB space using T. Formally, the denoiser (D) training minimizes the following objective function,

$$\min_D |T(\boldsymbol{x}^d; \boldsymbol{s}) - D(T(\boldsymbol{x}^d + \boldsymbol{N}); \boldsymbol{s}))|, \tag{2.7}$$

where $\boldsymbol{s}$ is the style parameter in (2.6). Our denoising framework allows rich augmentation for data synthesis. First, scaling and intensity augmentation (SIA) increases the content diversity by changing the image resolution and pixel values from $0.5\times$ to $1.5\times$. Second, we randomly scale the noise parameters, β_1 and β_2, from $0.5\times$ to $1.5\times$. This augmentation alleviates the noise estimation error in Section 2.4.3. Lastly, when multiple noisy images are given, we can augment the noise parameters and the style parameters across different images. Overall, the objective function of our denoiser training (2.7) becomes

$$\min_D |T(SIA(\boldsymbol{x}^d); \boldsymbol{s}') - D(T(SIA(\boldsymbol{x}^d) + \boldsymbol{N}'; \boldsymbol{s}'))|, \tag{2.8}$$

where SIA is the scale and intensity augmentation, $\boldsymbol{s}'$ is the augmented style parameter, and $\boldsymbol{N}'$ is the additive noise with augmented noise parameters. At testing, the denoiser takes the RGB noisy image ($\boldsymbol{y}$) as input, just like the conventional CNN-based approaches.

2.5 Experiments

2.5.1 Implementation Details

Downscaling. We downscale a raw image $\boldsymbol{x}$ to $\boldsymbol{x}^d$ by bicubic interpolation after asymmetric 2D Gaussian blurring with the kernel size of 21×21. We randomly select the standard deviation of the Gaussian blur from $0.25 \times ds$ to $0.75 \times ds$ for each dimension, where ds denotes the downscaling factor randomly selected in the range of $[1.5, 2.5]$. We use the same hyper-parameters for noise estimation, RAW2RGB conversion estimation, and denoiser training unless otherwise specified.

Optimization. We use Adam [64], 128×128 image patches, and a batch size of 64 for all experiments. Noise estimation adopts $2.5 \times e^4$ iterations with the initial learning rate of $1 \times e^{-4}$ which becomes a tenth part in every $5 \times e^3$ iterations. We use the same initial learning rates for RAW2RGB conversion estimation and denoiser training for $5 \times e^5$ iterations without the learning rate decay.

Denoiser. We use two simple networks as denoiser to present the generalizability of the proposed training scheme. Specifically, NERDS+DnCNN uses DnCNN [127] and NERDS+D uses a ResNet-style architecture, composing a global skip connection and 32 residual blocks. Each block has two convolutional layers and one ReLU activation with 64 filters and 3×3 kernels.

Dataset. We use BSD68 [85] which consists of 68 gray-scale images to evaluate the performance of noise estimation. For real image denoising, we use raw noisy and RGB noisy images on SIDD [2], DND [93], and MIT-Adobe FiveK [15]. SIDD [2] consists of training, validation, and benchmark datasets. We use 1,280 256×256 patches from 40 noisy images on the benchmark for denoiser training. In the setting of extra images, we use 50 noisy images on the training dataset with low ISO levels of 100 and downscale the noisy images with the scaling factor from 1.1 to 1.2. For DND [93], we use 1,000 512×512 patches from 50 noisy images on the benchmark. We evaluate the denoising performances on the benchmarks by submitting the results to the public websites for SIDD and DND. MIT-Adobe FiveK [15] consists of 5,000 raw images and paired RGB images retouched by 5 photographers. Each RGB image has its own RAW2RGB conversion using Adobe Lightroom while the raw image contains real camera noises. We demonstrate an extreme scenario where only 5 noisy images are available for denoiser training.

2.5.2 Results for Noise Estimation and Synthesis

We first validate our noise estimation on additive Poisson-Gaussian noise to images on BSD68 [85], and then visualize the noise synthesis at low-resolution from noisy images on MIT-Adobe FiveK [15].

Poisson-Gaussian noise estimation. We demonstrate the effectiveness of our noise estimation on Poisson-Gaussian noises (NERDS-raw) by comparing with Foi *et al.* [34], Liu *et al.* [81], and PGE-Net [16] which enable Poisson-Gaussian noise estimation from noisy images. Table 2.1 presents the average of the estimated values of (β_1, β_2) in four different noise levels. In most cases, NERDS-raw estimates the most accurate noise parameters.

Table 2.1: Performance comparison of Poisson-Gaussian noise estimation. The reported scores are average values of ($\hat{\beta}_1$, $\hat{\beta}_2$) estimated from BSD68 with additive Poisson-Gaussian noise level of (β_1, β_2). Bold denotes the best result.

Noise level (β_1, β_2)	Foi *et al.* [34] ($\hat{\beta}_1$, $\hat{\beta}_2$)	Liu *et al.* [81] ($\hat{\beta}_1$, $\hat{\beta}_2$)	PGE-Net [16] ($\hat{\beta}_1$, $\hat{\beta}_2$)	NERDS-raw (Ours) ($\hat{\beta}_1$, $\hat{\beta}_2$)
(0.100, 0.0200)	(0.096, 0.042)	(0.072, 0.045)	(0.098, 0.0030)	(**0.100**, **0.0229**)
(0.100, 0.0002)	(0.097, 0.035)	(0.071, 0.044)	(0.095, **0.0001**)	(**0.101**, **0.0001**)
(0.050, 0.0200)	(0.049, 0.031)	(0.040, 0.040)	(0.052, 0.0001)	(**0.051**, **0.0245**)
(0.050, 0.0002)	(**0.051**, 0.018)	(0.039, 0.034)	(**0.051**, 0.0001)	(0.052, **0.0002**)

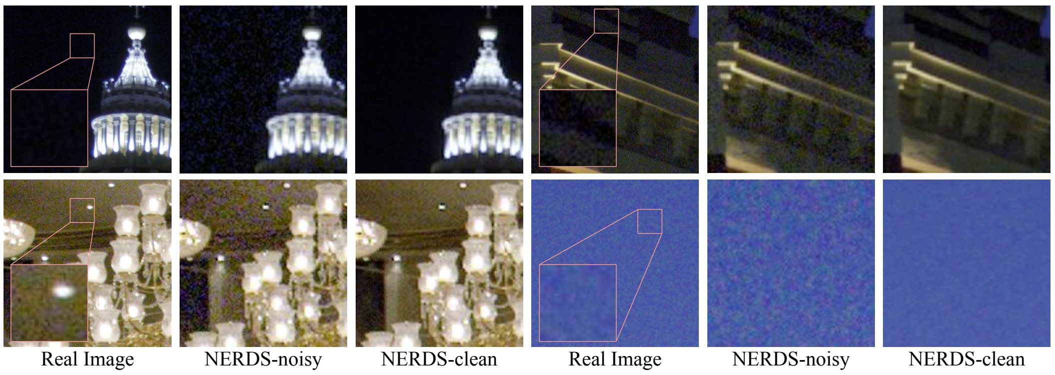


Figure 2.5: Noise synthesis results using NERDS. NERDS can synthesize noisy-clean RGB image pairs at low-resolution from single noisy images. NERDS-noisy and NERDS-clean denote synthetic RGB images with and without noises. This example uses the downscaling factor of 2. (*Zooming-in for the best view.*)

Noisy-clean RGB image synthesis. Figure 2.5 shows noise synthesis results using NERDS. Each real image from MIT-Adobe FiveK has its own RAW2RGB conversion. NERDS-noisy contains realistic noises specialized to each real image, such as blue/dark noises in dark areas with none i.i.d characteristics. To the best of our knowledge, NERDS is the first work to generate the realistic noisy images and the paired clean images for the unknown RAW2RGB conversions (or ISPs) by using only raw-RGB noisy image pairs. Existing works for noise synthesis [12, 126, 123, 1, 67] require real noisy-clean image pairs, metadata, and ISPs. Thus, naı̀ve comparisons between NERDS and the existing methods are unfair, but we present them in Section 2.6.4.1.

Table 2.2: Performance comparison with SotA denoising methods. The reported scores are PSNR (dB)/SSIM on RGB images from the SIDD and DND benchmarks. Bold denotes the best result. ✓denotes accessible types of images at training. Extra images denote that images other than the noisy images of the benchmark are used for training.

Dataset	GAT+BM3D	N2V	AP-BSN	FBI-D	C2N+DIDN	SCUNet	NERDS+DnCNN	NERDS+D	
Clean images	-	-	-	-	✓	✓	-	-	-
Extra images	-	-	-	✓	✓	✓	-	-	✓
Synthetic pairs	-	-	-	-	✓	✓	✓	✓	✓
SIDD	34.61/0.879	32.85/0.847	36.91/0.931	38.07/0.942	35.35/0.937	22.89/0.797	36.42/0.923	37.40/0.941	**38.28/0.949**
DND	37.98/0.920	35.82/0.902	38.09/0.937	38.98/0.945	37.28/0.924	-	38.21/0.941	**39.34/0.950**	-

2.5.3 Comparisons to the SotA Denoising Methods

Results on benchmarks. Given that NERDS trains denoisers without clean images, we compare our denoisers with the SotA denoising methods that *do not use the pairs of real noisy-clean images.* Specifically, we compare GAT+BM3D [28], N2V [68], AP-BSN [72], FBI-D [16], C2N+DIDN [54], and SCUNet [126] on SIDD and DND benchmarks. Table 2.2 presents the effectiveness of the proposed denoisers, NERDS+DnCNN and NERDS+D. C2N+DIDN [54] synthesizes realistic noises using *unpaired* noisy-clean images, while our NERDS+DnCNN outperforms it with the simpler denoiser (DIDN *vs.* DnCNN). SCUNet [126] synthesizes realistic noises using high-quality clean images and the predetermined noise models including specific noise levels and ISP pipelines. The other works require only noisy images via prior-based filtering [28], self-supervised learning [68, 72, 109], and noise estimation [16]. The works without clean images perform denoising in raw image space and received the results in RGB image space by submitting the denoised raw images to the public websites [2, 93]. They often fail to remove noises in RGB image space, given that they assume strong noise characteristics such as Poisson-Gaussian distributions and the i.i.d property that do not hold in the RGB image space. Section 2.6.3 presents a generalization test for NERDS+D with latency analysis to denoise a test image from scratch. Section 2.6.4.2 provides the comparisons with the noise synthesis methods

Table 2.3: Ablation study on data augmentation to train NERDS+D. PSNR (dB)/SSIM on RGB images from SIDD validation.

Augmentation	(1)	(2)	(3)	(4)	(5)	(6)
Image scale	-	✓	✓	✓	✓	✓
Image intensity	-	-	✓	✓	✓	✓
Noise parameter	-	-	-	✓	✓	✓
Style parameter	-	-	-	-	✓	✓
Extra images	-	-	-	-	-	✓
SIDD	35.63/0.897	36.79/0.928	37.16/0.935	37.84/0.944	38.02/0.946	**38.51**/**0.950**

which use the pairs of real noisy-clean images. Figure 2.10, 2.11, and 2.12 visualize more denoising results on SIDD validation, DND, and MIT-Adobe FiveK.

Ablation Study. Our NERDS enables rich and effective augmentation for denoiser training. Table 2.3 demonstrates the ablation study on the data augmentation. The setting without image scale augmentation (Table 2.3(1)) uses a fixed downscaling factor of 2. Each component improves the restoration performances. In particular, noise parameter augmentation improves over than 0.6 dB, given that it can alleviate noise estimation error. The extra images enable high-quality clean image synthesis by downscaling images with low noise levels and small scaling factors.

Results on MIT-Above FiveK. We evaluate CycleISP [123], SCUNet [126], and NERDS+D on retouched images from MIT-Adobe FiveK [15]. CycleISP [123] employs supervised learning on SIDD training dataset with additional synthetic data using the predetermined noise levels and ISP pipeline. Nevertheless, CycleISP fails to denoise the retouched image as visualized in Figure 2.6. SCUNet [126] is trained on high-quality images with practically designed additive noise, but SCUNet often oversmooths textures. These results are due to different noise distributions between the training images and the test images. In contrast, NERDS+D removes severe noise while maintaining image details. Figure 2.12 presents more qualitative results.

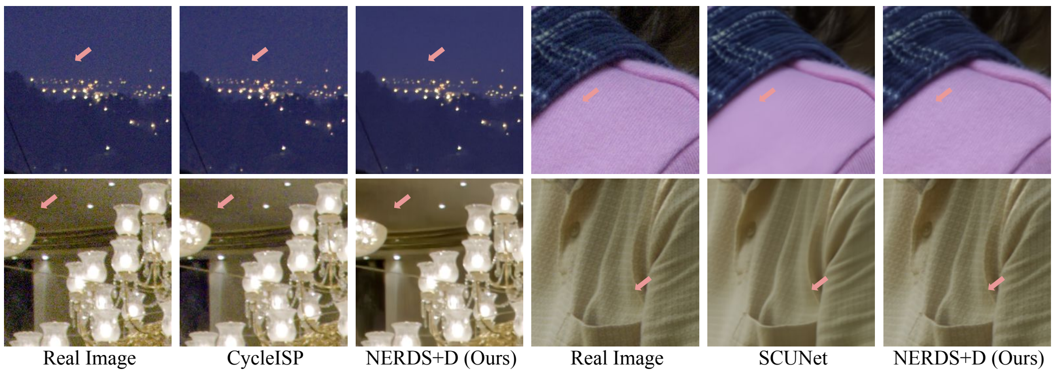


Figure 2.6: Qualitative comparisons of denoising results on MIT-Adobe FiveK [15]. Pink arrows indicate the remaining noise of CycleISP and the oversmooth textures of SCUNet while our denoiser generates crisp images. (*Zooming-in for the best view.*)

2.6 Discussions

2.6.1 Noise Distribution of Raw Images

Our NERDS assumes that the raw images have the noise that follows the Poisson-Gaussian (P-G) distribution. However, the image sensor of a smartphone or a digital camera is a black box. We do not know what kind of post-processing has been applied to the raw images, or whether the noise level (parameters) in the metadata represent the proper parameters of Poisson-Gaussian distribution. For instance, one of the noise levels in the metadata of Galaxy S6 (β_2 in Figure 2.7) equals zero for all images, which is theoretically impossible for both shot and read noise parameters of the Poisson-Gaussian model. Nonetheless, Figure 2.7 presents a similar tendency between the noise levels in the metadata (β_1, β_2) and the estimated noise parameters ($\hat{\beta}_1$, $\hat{\beta}_2$). Thus, we approximate the noise distribution in raw images as P-G noise.

To show the effectiveness of accurate P-G noise parameter estimation, Table 2.4 presents an ablation study of the values of P-G noise parameters. NERDS-raw, Metadata, and Random represent the restoration performance of an RGB image

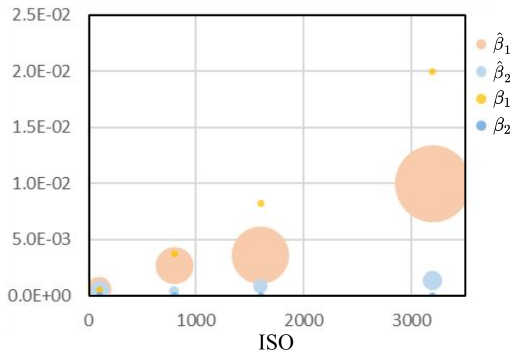


Figure 2.7: Noise distribution of images from Galaxy S6. β_1 and β_2 are the noise levels of metadata, while $\hat{\beta}_1$ and $\hat{\beta}_2$ are the estimated noise parameters for Poisson-Gaussian distribution using NERDS-raw. While (β_1, β_2) have specific values at each ISO level, the radius of the circles for $\hat{\beta}_1$ and $\hat{\beta}_2$ represents the standard deviation of the estimated values for each image on SIDD training dataset.

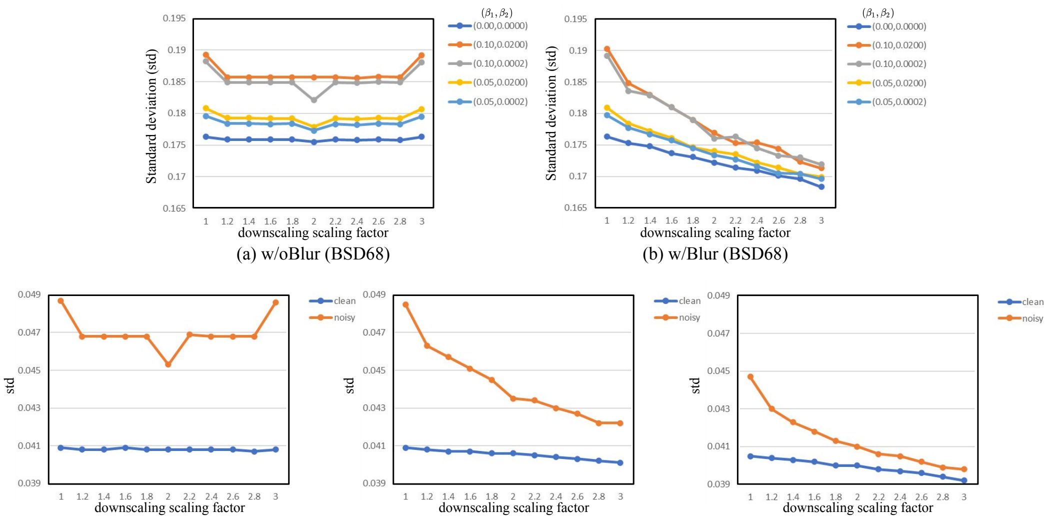
Table 2.4: Ablation study of P-G noise parameters to train denoisers. Each method provides different values of noise parameters for the same denoiser architecture, augmentation techniques, and training schemes. The reported scores indicate restoration performances on RGB images from the SIDD validation. NERDS-raw achieves the best PSNR score.

	Metadata	Random	NERDS-raw (Ours)
PSNR (dB)/SSIM	37.56/0.937	37.92/0.941	**38.51/0.950**

denoiser trained with the noise parameters from each method. NERDS-raw indicates the model in Table 2.3(6), Metadata uses the noise levels from the metadata, and Random uniformly samples noise parameters between the maximum and minimum values in NERDS-raw. Metadata and Random use the same denoiser architecture (D), augmentation techniques, and training schemes with NERDS-raw except the values of noise parameters for a fair comparison.

2.6.2 Clean Image via Downscaling

The *optimal* clean image via downscaling is a noise-free low-resolution image that has the same statistics as the true signal (clean high-resolution image). Empirically, we regard the raw images downscaled after burring (low-pass filtering) as raw *pseudo*-clean images. However, blurring images breaks the optimal setting when im-



(a) w/oBlur (BSD68) (b) w/Blur (BSD68)

(c) w/oBlur (SIDD) (d) w/Blur (SIDD) (e) w/LargeBlur (SSID)

Figure 2.8: Standard deviation (std) of downscaled images. Blue lines denote the std of clean images while the others denote the std of noisy images. Downscaling without pre-blurring maintains the std values through scaling on both BSD68 ((a)) and SIDD ((c)). In contrast, blurring before downscaling reduces the std values of noisy images drastically than clean images ((b) and (d)). (e) Large blur kernels (2× larger than the kernels used in NERDS-raw) reduce the std values more steeply.

Table 2.5: $\mathcal{D}_{KL}$ comparison for ablation study of NERDS-raw with the different sizes of blur kernels. NERDS-raw w/Blur, which is the original NERDS-raw, achieves the best performance.

NERDS-raw	w/oBlur	w/Blur (Ours)	w/LargeBlur
$\mathcal{D}_{KL}$	0.1710	**0.0344**	0.0535

age structures are too small (*e.g.*, 1-pixel dots) or the noise is too severe compared to the size of the blur. Figure 2.8 shows the evidence and limitations of the utility of pseudo-clean images used in NERDS-raw. First, the images downscaled without pre-blurring have similar statistics to those of the images before downscaling. This indicates that image scaling can be regarded as capturing an image at different distances between an object and a camera. Second, the blurring operation reduces the std values of noisy images drastically more than clean images. This is why we use downscaled images as pseudo-clean images. However, blurring also reduces the std values of clean images, making it challenging to find *optimal* clean images. To alleviate this difficulty, NERDS estimates noise by gradient-descent-based optimization

Table 2.6: Latency analysis of the processes for NERDS and NERDS+D.

	Noise Estimation	ISP Estimation	Denoiser Training	Denoiser Testing
Latency	30 m	1 h	6 h	0.1 s

Table 2.7: Generalization test for NERDS+D on different training datasets without clean images. The reported scores are PSNR (dB)/SSIM on RGB images from the SIDD validation and the DND benchmark. Bold denotes the best result.

NERDS+D		Taining	
		SIDD	DND
Testing	SIDD	**38.51/0.950**	36.26/0.923
	DND	39.14/0.949	**39.34/0.950**

and adopts augmentation of noise parameter scaling.

We further analyze the effectiveness of blur kernels by measuring KL divergence between synthesized noisy images and real noisy images. The lower value, the better. Table 2.5 visualizes an ablation study on the size of blur kernels on the SIDD dataset. NERDS-raw w/Blur achieves the best performance compared to both cases of w/oBlur and w/LargeBlur. These results indicate that the downscaled images work as clean images well with the proper size of blur kernels.

2.6.3 Efficient Inference at Test Time

The proposed method composes three steps (noise estimation, ISP estimation, and denoiser training) and an additional step of denoiser testing. Each step described above has the latency described in Table 2.6. We use GeForce RTX 2080 Ti GPU and an HD image for testing.

The noise estimation, ISP estimation, and denoiser training using noisy test images are time-consuming. To skip the processes at test time, Table 2.7 presents a generalization test for NERDS+D trained on different datasets. NERDS+D trained on DND has accurate restoration performance on DND but performs 2 dB lower PSNR on SIDD than the model trained on SIDD. In contrast, NERDS+D trained on SIDD performs accurate restoration on both datasets. This phenomenon indicates

Table 2.8: $\mathcal{D}_{KL}$ comparison with SotA noise synthesis methods. NERDS-raw w/clean image outperforms all compared methods.

	Calibrated P-G [130]	Noise Flow [1]	Camera-Aware [21]	NERDS-raw (Ours)	
Clean data	✓	✓	✓	✓	
$\mathcal{D}_{KL}$	1.5147	0.0481	0.0144	**0.0079**	0.0344

that SIDD contains noise distributions similar to DND and that well-designed noisy images enable generalized denoiser training. For instance, camera manufacturers can collect training datasets of only noisy images concerning the image sensor, ISP, and expected image retouching.

2.6.4 Comparisons to Noise Synthesis Methods using Real Image Pairs

The recently proposed methods [1, 21, 130, 67] synthesize noisy images with novel noise models. These methods use noise/clean image pairs and metadata that are not accessible at test time, such as the DND dataset. Although the proposed method is a general framework for denoising training without a clean image, we present comparisons with these methods by evaluating the noise estimation results with denoising performance and KL divergence.

2.6.4.1 Results for Noise Estimation and Synthesis

Table 2.8 presents the comparisons of KL divergence ($\mathcal{D}_{KL}$) between synthesized noisy images and real noisy images. The lower values, the better. The proposed method outperforms all compared methods when using clean images. We could not reproduce the results of RGB Noise Flow [67] since the source code was not available. The score reported in the paper is 0.044 for RGB Noise Flow, where Noise Flow scores 0.198. The scores of KL divergence are dependent on hyperparameters.

Table 2.9: Raw image denoising comparison with SotA noise synthesis methods on the SIDD benchmark. NERDS-raw achieves the best PSNR score. We use DnCNN as a denoiser for NERDS-raw.

	Gaussian	Noise Flow [1]	Camera-Aware [21]	NERDS-raw (Ours)
PSNR (dB)/SSIM	43.63/0.968	48.52/0.992	48.71/**0.993**	**48.93**/0.985

Table 2.10: RGB image denoising comparison with SotA noise synthesis methods on the SIDD benchmark. NERDS achieves the best PSNR/SSIM scores. All methods use DnCNN as the denoiser.

	Gaussian	Noise Flow [1]	C2N [54]	RGB Noise Flow [67]	NERDS (Ours)
PSNR (dB)/SSIM	32.72/0.873	33.81/0.894	33.76/0.901	34.74/0.912	**36.42**/**0.923**

2.6.4.2 Results for Raw/RGB Image Denoising

Table 2.9 and 2.10 present denoising performances for raw/RGB images on the SIDD benchmark, where NERDS-raw and all methods for RGB images use DnCNN as a denoiser. Although NERDS-raw and NERDS do not use clean images for noise estimation, noise synthesis, and denoiser training, NERDS-raw and NERDS achieve the best PSNR scores in each table. When converting the denoised raw images using NERDS to RGB images, the PSNR is 35.74 dB which is lower than RGB image denoising (36.42 dB).

2.7 Additional Ablation Study

Ablation Study for Noise Estimation (NERDS-raw). For noise estimation (NERDS-raw), we design ablation studies on blurring strengths and downscaling factors on BSD68 in Table 2.11 and 2.12. The setting of w/LargeBlur uses two times larger blur kernels than NERDS-raw. The settings of w/Blur and DF 1.5∼2.5, which indicates NERDS-raw in Table 2.1, present better performance than compared settings. The settings of w/oBlur or DF 1 perform worse than those of w/LargeBlur or DF 2.5∼4.5. These ablation studies show the effectiveness of image downscaling after blurring for NERDS-raw.

Table 2.11: Ablation study for noise estimation on blurring strengths.

Noise level (β_1, β_2)	w/oBlur ($\hat{\beta}_1$, $\hat{\beta}_2$)	w/Blur (Ours) ($\hat{\beta}_1$, $\hat{\beta}_2$)	w/LargeBlur ($\hat{\beta}_1$, $\hat{\beta}_2$)
(0.100, 0.0200)	(0.080, 0.144)	(**0.100**, **0.0229**)	(0.119, 0.0220)

Table 2.12: Ablation study for noise estimation on downscaling factors (DF).

Noise level (β_1, β_2)	DF 1 ($\hat{\beta}_1$, $\hat{\beta}_2$)	DF 1.5∼2.5 (Ours) ($\hat{\beta}_1$, $\hat{\beta}_2$)	DF 2.5∼4.5 ($\hat{\beta}_1$, $\hat{\beta}_2$)
(0.100, 0.0200)	(0.065, 0.0085)	(**0.100**, **0.0229**)	(**0.100**, 0.0279)

Table 2.13: Ablation study for denoiser training on blurring strengths.

	w/oBlur	w/Blur (Ours)	w/LargeBlur
PSNR (dB)/SSIM	26.99/0.642	**38.02/0.946**	38.01/0.944

Table 2.14: Ablation study for denoiser training on blurring strengths.

	DF 1	DF 1.5∼2.5 (Ours)	DF 2.5∼4.5
PSNR (dB)/SSIM	37.66/0.942	**38.02/0.946**	37.89/0.943

Ablation Study for Denoiser Training (NERDS+D). Table 2.13 and 2.14 present ablation studies on blurring strengths and downscaling factors for denoiser training on the SIDD validation. This experiment uses SIDD validation to visualize the effectiveness of downscaling and blurring to noisy images. The setting of w/LargeBlur uses two times larger blur kernels than NERDS+D. Results show that our settings for NERDS+D perform the best accuracy at the diverse DFs and blurring strengths. The setting of w/oBlur performs poor PSNR/SSIM given that the downscaled images still constrain severe noise as demonstrated in Figure 2.8(c). Instead, the comparable results between w/Blur and w/LargeBlur indicate that the denoiser training is robust to blurring strengths. The settings for diverse DFs perform similar results given that blurring transforms noise from the P-G distribution. The transformed noise can be regarded as high-frequency textures that allow denoisers training for P-G noise.

2.8 Additional Qualitative Results

Synthesized Noisy-Clean Image Pairs. Figure 2.9 visualizes noise synthesis results using NERDS. NERDS-clean contains low-level noise while NERDS-noisy presents severe noise similar to the real noisy images. We have empirically found that denoisers *do not* learn to remove noise in NERDS-clean. This phenomenon is because the blurring and downscaling process in NERDS distorts the noise of raw images while denoisers learn to remove Poisson-Gaussian noise in the raw images. Moreover, given that NERDS does not require any clean data, NERDS successfully synthesizes noisy-clean image pairs on DND and MIT-Adobe FiveK datasets which are not available to access clean images.

Denoised Images. We present more images denoised by NERDS+D on SIDD (Figure 2.10), DND (Figure 2.11), and MIT-Adove FiveK (Figure 2.12). While AP-BSN [72], FBI-D [16], SCUNet [126], C2N [54]+DIDN [119], and CycleISP [123] often fail to restore image patterns in GT, NERDS+D successfully recover the original structures. AP-BSN uses pixel shuffle for self-supervised learning to decorrelate spatially but also loses spatial information for denoising. FBI-D learns raw image denoising for Poisson-Gaussian noise, but denoising raw images is an indirect approach for the human visual system (or RGB space). SCUNet over-smooth or over-sharpen images. This restoration style helps to generate readable characters, but it can distort the patterns in GT (See Figure 2.10). C2N+DIDN is the denoiser trained by generating noisy images from clean images using both noisy and clean images on SIDD datasets. For DND benchmarks, C2N+DIDN sometimes generates artifacts or noise that significantly drops the restoration accuracy (See Figure 2.11). In contrast, NERDS+D uses only noisy images on DND benchmarks to generate noisy-clean image pairs for accurate denoiser training. CycleISP is the denoiser trained by paired noisy-clean images on SIDD datasets and paired images synthesized by the metadata of DND benchmarks. CycleISP performs the highest PSNR and SSIM on both SIDD and DND benchmarks compared to the methods in Table 2.2. However, CycleISP fails to remove the noise on MIT-Adobe FiveK (See Figure 2.12). This failure comes from the noise distribution of RGB images mismatched between MIT-Adobe FiveK and SIDD (or DND) datasets.

2.9 Conclusion

We present a general framework to train denoisers from noisy images, called NERDS. The framework composes noise estimation, RAW2RGB conversion estimation, and denoiser training. For noise synthesis, we estimate Poisson-Gaussian noise in raw images and ISP (or RAW2RGB conversion) for each RGB image. NERDS allows rich data augmentation for accurate denoiser training. Experimental results show the state-of-the-art restoration accuracy on real noise benchmarks.

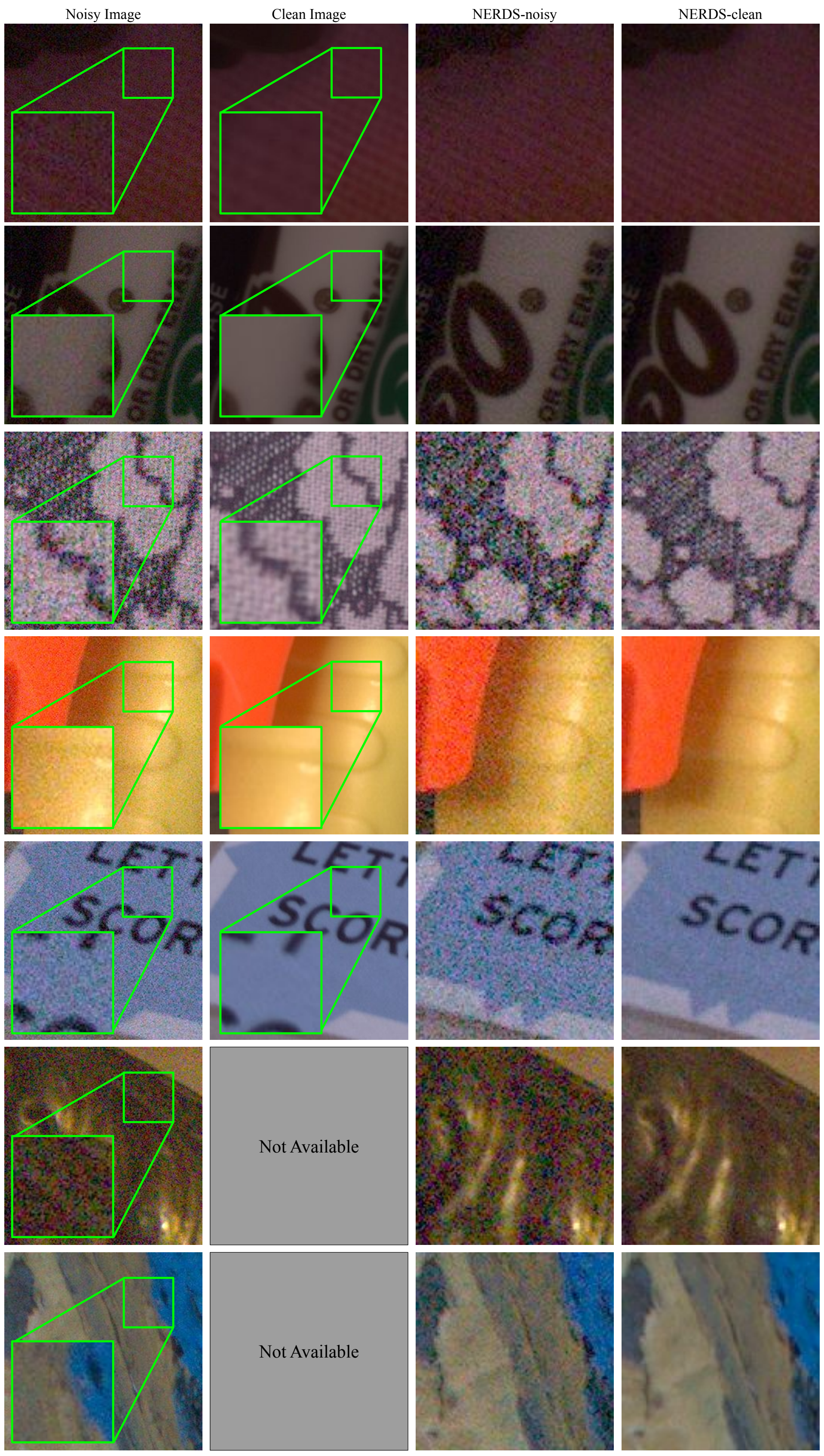


Figure 2.9: Examples of noisy synthesis results using NERDS. We upscale NERDS-noisy, NERDS-clean, and green boxes with a scaling factor of 2 for visualization.

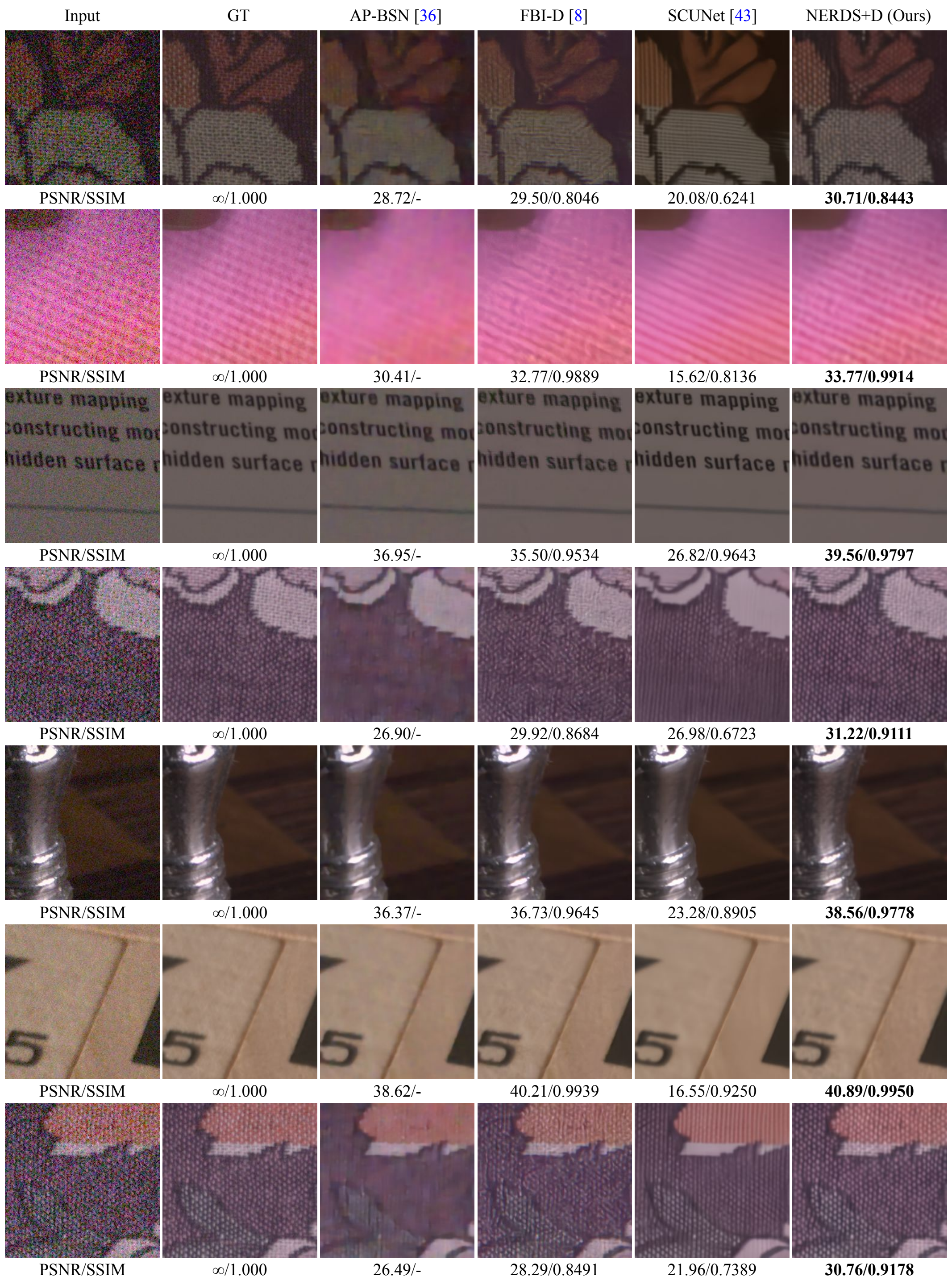

Input
GT
AP-BSN [36]
FBI-D [8]
SCUNet [43]
NERDS+D (Ours)
PSNR/SSIM ∞/1.000 28.72/- 29.50/0.8046 20.08/0.6241 **30.71/0.8443**
PSNR/SSIM ∞/1.000 30.41/- 32.77/0.9889 15.62/0.8136 **33.77/0.9914**
exture mapping
constructing mo
hidden surface
PSNR/SSIM ∞/1.000 36.95/- 35.50/0.9534 26.82/0.9643 **39.56/0.9797**
PSNR/SSIM ∞/1.000 26.90/- 29.92/0.8684 26.98/0.6723 **31.22/0.9111**
PSNR/SSIM ∞/1.000 36.37/- 36.73/0.9645 23.28/0.8905 **38.56/0.9778**
5
PSNR/SSIM ∞/1.000 38.62/- 40.21/0.9939 16.55/0.9250 **40.89/0.9950**
PSNR/SSIM ∞/1.000 26.49/- 28.29/0.8491 21.96/0.7389 **30.76/0.9178**

Figure 2.10: Qualitative denoising results and PSNR (dB)/SSIM on the SIDD validation.

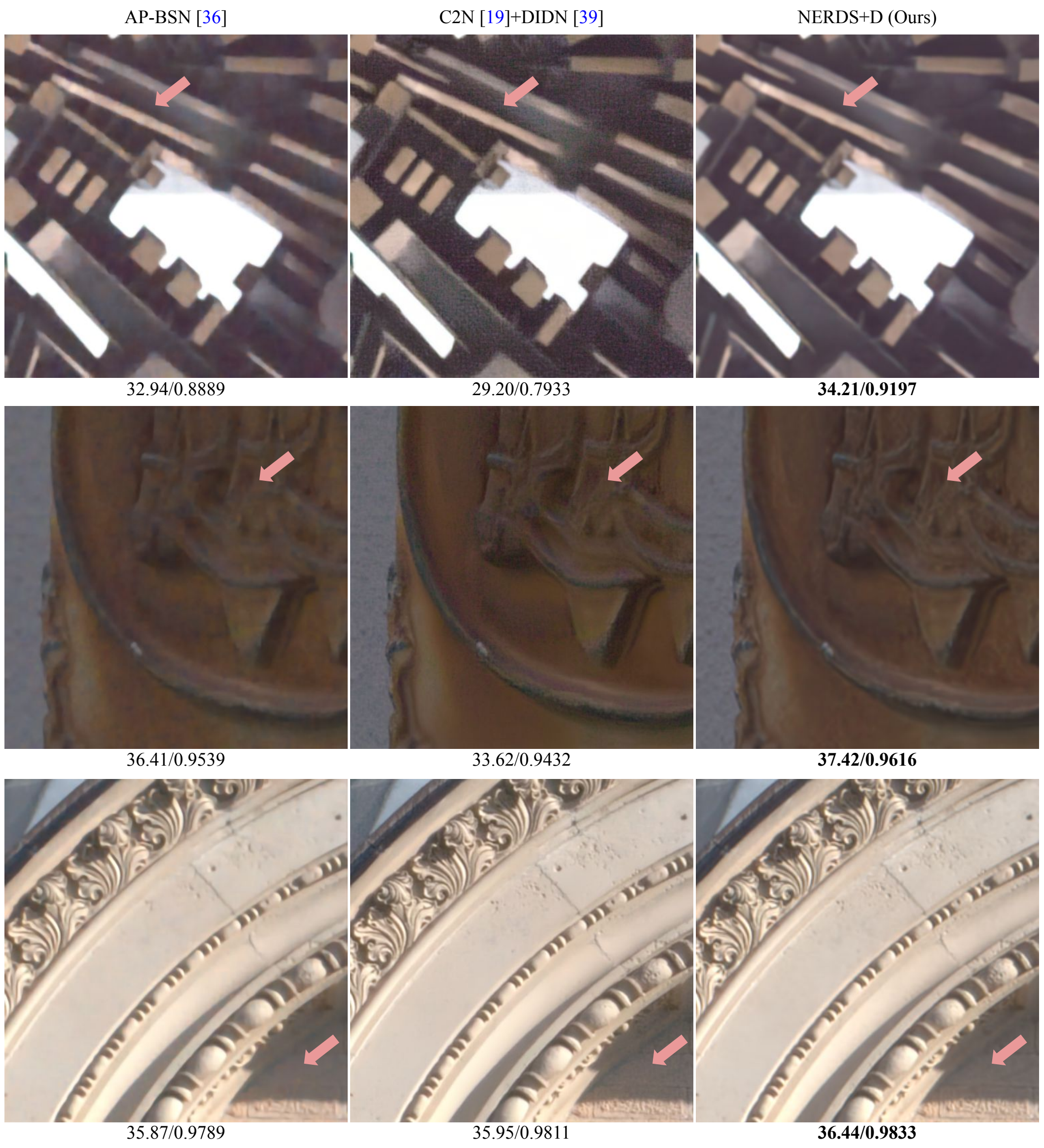


Figure 2.11: Qualitative denoising results and PSNR (dB)/SSIM on the DND benchmark.

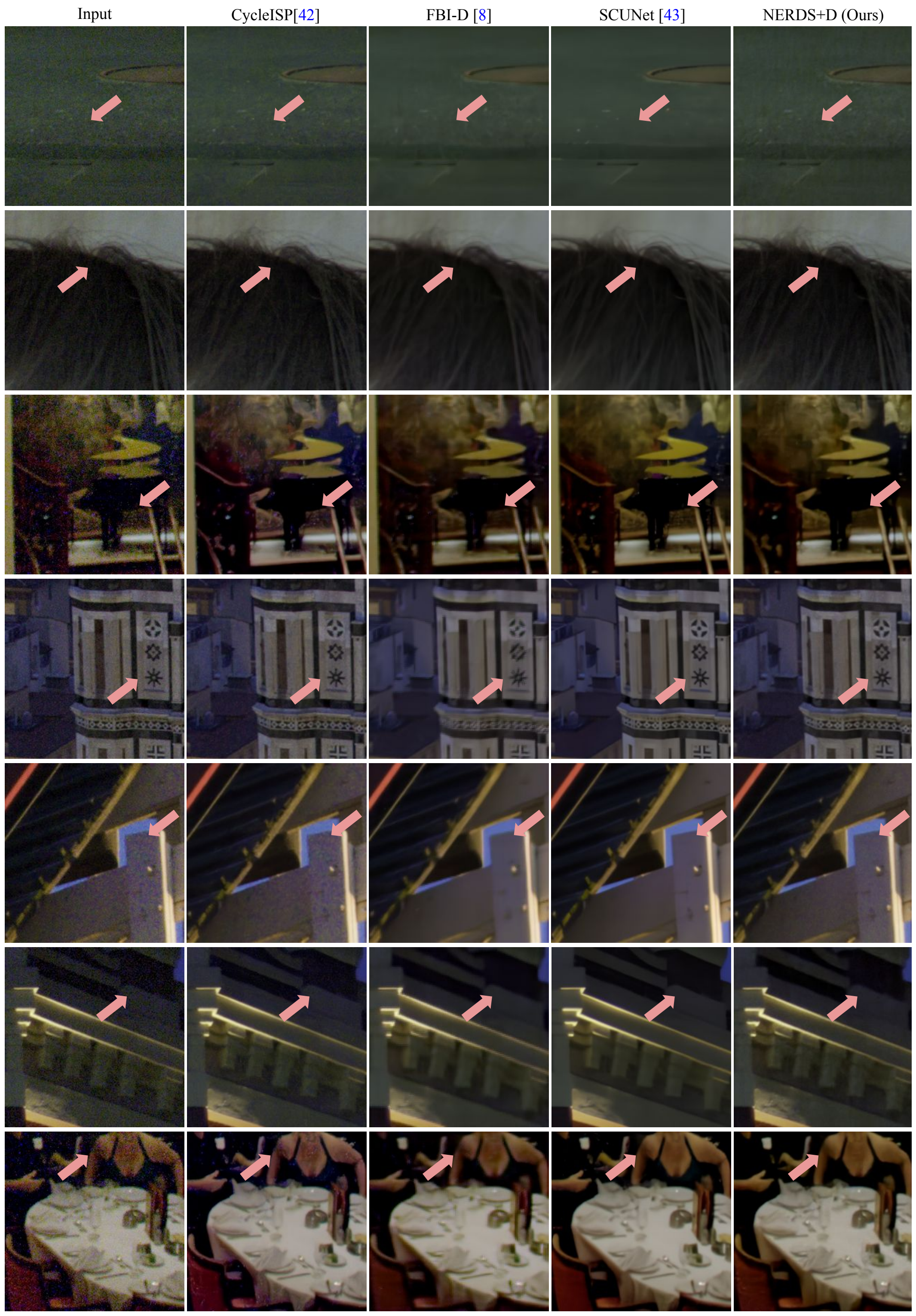


Figure 2.12: Qualitative results on MIT-Adobe FiveK. CycleISP often fails to remove noise, while FBI-D and SCUNet over-smooth or over-sharpen images. Note that NERDS+D uses only 5 test noisy images and estimated noise parameters from them for training. (*Zooming-in for the best view.*) We use T of NERDS to convert raw images denoised by FBI-D to RGB images.

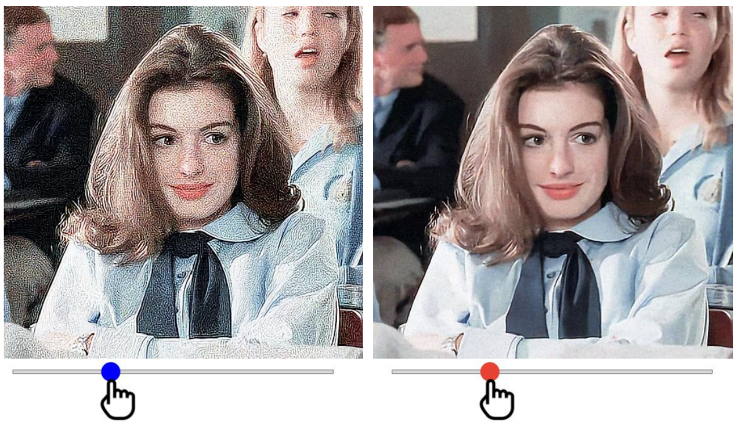
Chapter 3

Adaptive Neural Architecture Search for Controllable Image Restoration

3.1 Introduction

Restoration of real-world corrupted images is a challenging problem since the types and the severity (or level) of degradation are unknown. Previous works on *blind* image super-resolution [10, 96] or blind deblurring [114, 37, 4] tackle this problem by learning to predict the unknown degradation kernel, and then using the predicted kernel to restore clean images. Recently, *controllable* image restoration has been gaining increased attention as alternative approaches. In this scenario, instead of accepting a single restored image given by the final model, users can control the output restoration to generate multiple images and choose the output image that best fits their preferences.

Early works on controllable image restoration (CIR) [42, 97, 106, 107] mostly consider a single type of degradation and modulate the levels of restoration. For instance, the denoising model from [42] allows continuous modulation of denoising a

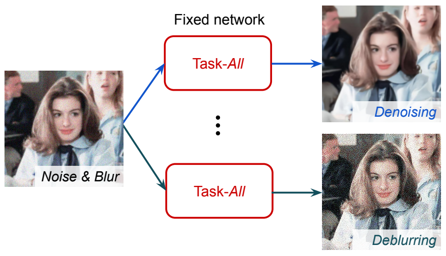


(a) CResMD [43] (b) TASNet (Ours)

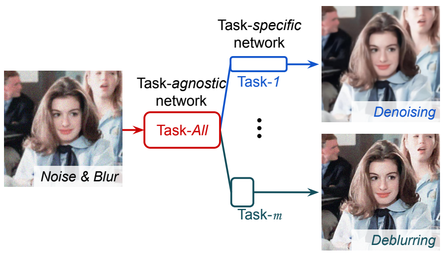
Figure 3.1: An example of controllable image restoration. Our model generates visually more pleasing outputs while adjusting restoration levels with **3 times faster GPU latency** and **95.7% reduced FLOPs** compared to CResMD [43].

Gaussian noise with $\sigma = 15 \sim 75$. More recently, CResMD [43] proposed an extended framework that learns multiple types of degradation (Gaussian blur, Gaussian noise, and JPEG compression) jointly with a single network, so that users can interactively adjust not only the level but also the type of degradation. However, as more flexible control is enabled, two new challenges arise for the practical application of CIR models: 1) the high computation cost of generating multiple images to choose from, and 2) the difficulty of finding the true types and the levels of degradation, in which failing to do so may lead to significantly deteriorated outputs.

To alleviate these limitations, we present **TASNet**, a novel deep-learning-based CIR model that is optimized to achieve better visual quality and substantially reduced computational complexity. Figure 3.1 demonstrates a sample result. Our TASNet consists of two parts: task-agnostic layers and task-specific layers, where we denote "task" as a restoration problem *w.r.t.* a combination of degradation types and levels. The task-agnostic part is composed of the early layers of the baseline super-



(a) CResMD [43]



(b) TASNet (Ours)

Figure 3.2: The overview of our efficient architecture for controllable image restoration. (a) CResMD [43] has a fixed network across all tasks and requires separate inference through the full model whenever the target restoration task becomes different. (b) Our task-agnostic and task-specific network (TASNet) shares the early layers to facilitate feature reuse. When we perform inference for multiple tasks, the task-agnostic part requires only a single computation, of which the output feature can be reused multiple times as the input for the task-specific network. The architecture of the task-specific network is adaptively adjusted for each given task. The width and the height of boxes represent the number of layers and channels of neural networks, respectively. In this example, two popular restoration tasks of denoising and deblurring are visualized, where different colors represent the corresponding inference path.

network, where the parameters of the layers are shared across all tasks. Sharing the early layers greatly improves the efficiency of CIR model, since we do not need to re-compute the output of the shared layers each time a user changes the task (the type or the level of degradation). On the other hand, the remaining layers that consist of the task-specific parts are differently adjusted for each task. The main concept is summarized in Figure 3.2. However, deciding the architectural hyperparameters that balances the efficiency and the performance is still a very challenging problem.

To this end, we propose a new supernetwork-based neural architecture search (NAS) algorithm that can automatically search for the task-agnostic and task-specific architectures on the efficiency-accuracy trade-off curve. Since we need to consider a large number of tasks for continuously varying levels of restoration, the search space of our algorithm should be able to represent a diverse set of architectures. This is why our algorithm allows channel-level selection for each layer as well as layer-wise decision of whether to share its parameters or not. Specifically, the proposed NAS algorithm selects: 1) the number of layers to share (task-agnostic part), 2) the important channels for the shared layers, and 3) the important channels for each task-specific layer, where these task-specific channel selection is adaptive for each task. We also formulate the overall learning objective to be differentiable for efficient end-to-end training of our searching framework, which results in the final TASNet. Moreover, we propose a new data sampling strategy to reduce the visual artifacts, which is empirically shown to be effective for cases when the task given by the user is very different from the true degradation of an input image.

Experimental results show that TASNet runs 3.7 times faster than the state-of-the-art CIR model on modern high-end GPUs with 95.7% FLOPs reduction when generating 4K images with 27 modulations. Also, the visual quality of the generated restoration using TASNet is much better than the previous approaches with significantly less artifacts.

Overall, our contributions can be summarized as follows:

- We present a novel neural network, named TASNet, for controllable image restoration (CIR) that remarkably improves the model efficiency and output image quality.

- We propose a supernetwork-based NAS algorithm that finds efficient CIR networks in a differentiable manner.

- We introduce a new data sampling strategy to improve the generated image quality in CIR problems.

- The proposed TASNet outperforms the state-of-the-art models in image quality and computation costs of FLOPs and CPU/GPU latency.

3.2 Related Work

3.2.1 Image Restoration

Image restoration, including denoising, deblurring, super-resolution, and compression artifact removal, is a classical topic in computer vision that aims at restoring the original image from its degraded versions. Deep-learning-based image restoration networks [29, 30, 31, 62, 71, 76, 128, 131, 57] have achieved breakthroughs in restoring accurate image details. While the conventional approaches specialize in dealing with a single degradation type, general image restoration aims to restore the corrupted image with multiple types of degradation. Notable approaches include learning a policy to determine a specialized restoration network for the input image [117, 118], or using an operation-wise attention module to produce the specialized feature maps *w.r.t.* each degradation type [100]. However, these approaches cannot control the diverse restoration levels for the input images, and sometimes generate overly smooth or sharpened outputs.

On the other hand, controllable image restoration is recently gaining interests from the computer vision research community, to control the output restoration of an image corrupted by unknown degradation. Existing works learn to control restoration levels for a single type of degradation [42, 97, 106, 107]. In particular, AdaFM [42], CFSNet [106], and Dynamic-Net [97] design their network architectures with tuning modules, which modulate the feature maps layer-wise [42] or block-wise [106, 97] with respect to the tasks of interest at test time. Instead, DNI [107] interpolates all parameters of the differently trained networks for modulation. For the general controllable image restoration, CResMD [43] controls restoration levels in multiple types of degradation with a block-wise tuning module. While the prior works may have provided good performance to control the restoration levels, they have solely focused on the image quality and do not consider computational efficiency. By contrast, using CResMD as the baseline, the proposed TASNet significantly reduces the computations and running time.

3.2.2 Efficient CNNs for Image Restoration

To make the image restoration models efficient with less computation cost, several novel architectures have been developed for diverse restoration tasks. The early works downscale the spatial resolution of the input image to reduce the computation costs of the convolution operations for denoising [129] and super-resolution [31]. More recently, CARN [6] presents a cascading residual block with multiple skip connections, leading to a fast and light-weight super-resolution network. For deblurring, a method using spatially variant deconvolution is proposed in [120] to achieve accurate performance with its efficient backbone network. Meanwhile, FALSR [26], ESRN [98], and FGNAS [59] adopt neural architecture search (NAS) algorithms for efficient super-resolution networks. Path-Restore [118] and AdaDSR [79] save computation costs via adaptive inference for general image restoration and super-resolution, respectively. Prior works also employ network quantization [27, 112] or pruning [35, 74, 82, 103, 124], but they are not task-adaptive.

On the other hand, we study the network acceleration approaches for controllable image restoration for the first time, especially when it requires a large number of inference passes per image. A neural architecture accelerated by our algorithm can be considered as a special instance of multi-task learning [19, 132], a network design paradigm that uses a shared network for multiple tasks or optimization. The main difference from the previous multi-task learning approaches is employing NAS

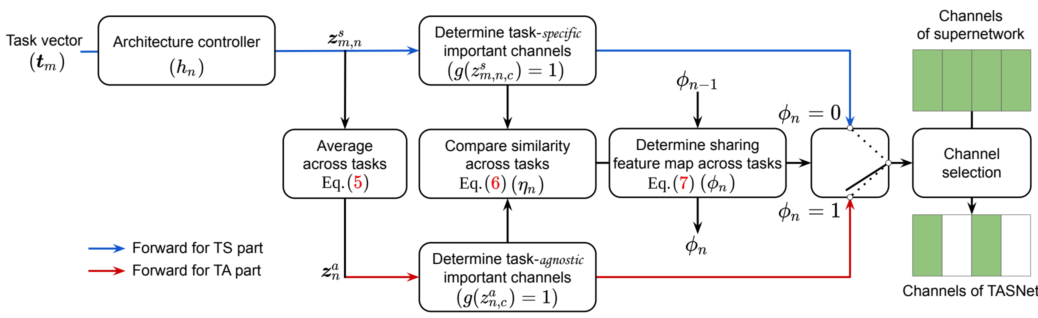

Task vector (t_m)
Architecture controller (h_n)
$z^s_{m,n}$
Determine task-specific important channels ($g(z^s_{m,n,c}) = 1$)
Average across tasks Eq.(5)
Compare similarity across tasks Eq.(6) (η_n)
Determine sharing feature map across tasks Eq.(7) (ϕ_n)
ϕ_{n-1}
$\phi_n = 0$
$\phi_n = 1$
ϕ_n
Channels of supernetwork
Channel selection
Channels of TASNet
Forward for TS part
Forward for TA part
z^a_n
Determine task-agnostic important channels ($g(z^a_{n,c}) = 1$)

Figure 3.3: The neural architecture search process for each layer of TASNet. Our algorithm automatically determines the number of shared layers and channels in each feature map from the supernetwork. Task-specific (TS) part ($\phi_n = 0$) adaptively selects channels based on the given task (**blue arrow**). By contrast, task-agnostic (TA) part ($\phi_n = 1$) selects fixed channels across tasks (**red arrow**). A feature map is determined to be shared if the channel importance is similar among tasks and the previous feature map is shared. All processes are differentiable via a straight-through estimator (g). During the inference, ϕ, $\boldsymbol{z^a}$, and thus task-agnostic (TA) part are fixed.

for continuously varying tasks from an input image. Our search algorithm is a variant of supernetwork-based NAS methods [59, 78], which aim to find an efficient or performance-wise optimal network by pruning from a supernetwork. Our search process is performed over a search space of channels and shared layers across tasks, each combination of which provides a candidate network derived from a supernetwork.

3.3 Method

Controllable image restoration (or modulation) aims to control the restoration levels of a corrupted image. Following the CResMD setting, we formulate multi-dimensional restoration levels to be controllable. Formally, given D number of degradation types, $\boldsymbol{t} \in \mathbb{R}^D$ denotes a task vector, where $t_d \in [0, 1]$ encodes the restoration level for the d-th degradation type. For instance, a task vector of (1,0,0) for three degradation types (*e.g.*, blur, noise, JPEG compression) indicates that the task requires the maximum level of deblurring but no denoising or compression artifact removal. During training, a training image pair (input and target) determines the corresponding values of the task vector, which are controlled by users at inference time.

3.3.1 Efficient Architecture Design

Unknown degradation of real images demands interactive image restoration with adjustable restoration levels. In this scenario, a network for image modulation computes its operations multiple times for a single input image with different task vectors. Formally, the total computation cost for M times of inferences is given by,

$$\mathcal{R}_{\text{total}}(f, \boldsymbol{x}, \boldsymbol{t}) = \sum_{m=1}^{M} \mathcal{R}(f, \boldsymbol{x}, \boldsymbol{t}_m), \tag{3.1}$$

where $\mathcal{R}(f, \boldsymbol{x}, \boldsymbol{t}_m)$ denotes FLOPs or latency to generate an output with the network architecture f, the input image $\boldsymbol{x}$, and the m-th task vector $\boldsymbol{t}_m$. Architectures used in CResMD and other previous works [42, 97, 106, 107] follow the computation cost of Eq. (3.1), as outlined in Figure 3.2(a).

Our goal is to design a network architecture which is efficient under the aforementioned multiple -inference scenario. To this end, we propose **TASNet** that shares the feature map of early layers with the remaining task-specific architecture, as described in Figure 3.2(b). The task-agnostic shared layers facilitate feature reuse for repeated inferences from a single image. On the other hand, our task-specific architecture *adaptively* transforms itself to be efficient as it is difficult to find a single fixed network that is efficient for continuously varying restoration levels.

For TASNet, we reformulate Eq. (3.1) and divide the network f into the early layers f^a and the remaining layers f^s. Then, the total computation cost becomes:

$$\begin{aligned} \mathcal{R}_{\text{total}}(f, \boldsymbol{x}, \boldsymbol{t}) &= \sum_{m=1}^{M} [\mathcal{R}(f^a, \boldsymbol{x}) + \mathcal{R}(f^s, \tilde{\boldsymbol{x}}, \boldsymbol{t}_m)] \\ &\geq \mathcal{R}(f^a, \boldsymbol{x}) + \sum_{m=1}^{M} \mathcal{R}(f^s_m, \tilde{\boldsymbol{x}}, \boldsymbol{t}_m), \end{aligned} \tag{3.2}$$

where $\tilde{\boldsymbol{x}} = f^a(\boldsymbol{x})$ and $\mathcal{R}(f^a, \boldsymbol{x})$ is the computation cost of *a single inference* for $f^a(\boldsymbol{x})$, and f_m denotes the transformed task-specific architecture. Although Eq. (3.2) should theoretically reduce the computational redundancy, designing efficient architectures (f^a and f^s_m) is still an open problem.

3.3.2 Search Formulation

Overview. In order to find efficient TASNet architectures, we propose a supernetwork-based neural architecture search algorithm. Our search algorithm determines 1) the number of early layers that are shared across tasks, 2) the important channels for task-agnostic layers, and 3) the important channels for each task-specific layer, where the channels are selected from the supernetwork CResMD [43]. TASNet aims to minimize both restoration error and computation cost of Eq. (3.2) via following rules, as illustrated in Figure 3.3 (Please also see Figure 3.12 for architecture details):

- Learn task-specific channel importance ($\boldsymbol{z}^s_m$).
- Learn task-agnostic channel importance ($\boldsymbol{z}^a$).
- Share a feature map across tasks ($\phi_n = 1$), when important channels are similar across tasks ($\eta_n = 1$) and the feature map of its previous layer is shared ($\phi_{n-1} = 1$).

- Maximize the number of shared layers (Eq. (3.10)).
- Prune unimportant channels across tasks ($g(z^a_{n,c}) = 0$) in shared feature maps ($\phi_n = 1$).
- Adaptively select important channels ($g(z^s_{m,n,c}) = 1$) to the task $\boldsymbol{t}_m$ in non-shared feature maps ($\phi_n = 0$).

Channel selection. Variants of straight-through estimator [11] have been widely used for differentiable NAS approaches [108, 17]. To *select* or *de-select* each channel from the supernetwork, channel selection virtually multiplies a binary value to the channel. Our straight-through estimator enables this process differentiable, given by,

$$g(z) = \begin{cases} \mathbb{I}\left[z > 0\right] & \text{if forward} \\ \text{sigmoid}(z) & \text{if backward,} \end{cases} \tag{3.3}$$

where $z \in \mathbb{R}$, and $\mathbb{I}\left[\cdot\right]$ is an indicator function that returns 1 when its input is true (and 0 otherwise). We introduce two types of z which determine task-specific and task-agnostic channels, respectively, in the following.

Task-specific channel importance. To learn channel importance for a given task $\boldsymbol{t}_m$, we introduce architecture controller $\boldsymbol{h}$, formally given by,

$$\boldsymbol{z}^s_{m,n} \equiv h_n(\boldsymbol{t}_m), \tag{3.4}$$

where $z^s_{m,n,c} \in \mathbb{R}$ denotes the importance of c-th channel to the task vector $\boldsymbol{t}_m$ in the n-th feature map of the supernetwork. h_n is the architecture controller, composed of few fully connected layers, for the n-th feature map.

Task-agnostic channel importance. To learn general channel importance across tasks, we simply average the values of the task-specific channel importance as:

$$z^a_{n,c} \equiv \frac{1}{M} \cdot \sum_{m=1}^{M} z^s_{m,n,c}, \tag{3.5}$$

where $z^a_{n,c} \in \mathbb{R}$ denotes the task-agnostic channel importance and M is a large enough number of inference. Empirically, we adopt exponential moving average over iterations with the small mini-batch size.

Channel importance similarity across tasks. To determine whether a feature map should be shared across tasks, we compute the agreement criterion via the similarity between selected channels from $\boldsymbol{z^a}$ and $\boldsymbol{z^s}$ as follows:

$$\frac{1}{M} \cdot \sum_{m=1}^{M} \sum_{c=1}^{C} g(z^s_{m,n,c}) \cdot g(z^a_{n,c}) > \gamma \cdot \sum_{c=1}^{C} g(z^a_{n,c}), \tag{3.6}$$

where γ is a threshold hyperparameter. Whether Eq. (3.6) holds is represented by a boolean variable η_n. If the equation holds ($\eta_n = 1$), a large number of tasks have an agreement on the channel importance for a given layer, and thus this layer is likely to be shared across all tasks.

Recursive layer sharing. To facilitate feature reuse across tasks, the shared layers need to be located together at the initial stage of the network. To this end, the n-th feature map is shared if the n-th and all previous feature maps satisfy the agreement criterion on the position of pruning across tasks ($\eta_i = 1$), formally,

$$\phi_n = \begin{cases} 1 & \text{if } \eta_i = 1, \forall i = 1, 2, ..., n \\ 0 & \text{otherwise,} \end{cases} \tag{3.7}$$

where $\boldsymbol{\phi} \in \mathbb{Z}_2^N$ denotes a decision variable, in which the n-th element ϕ_n is 1 if the n-th feature map is task-agnostic.

Objective function. By using all equations above, we can formulate the objective function with differentiable resource regularization terms. Let $\mathcal{L}(\cdot, \cdot)$ denote a standard ℓ_1 loss function for image restoration tasks. The overall objective function is formally given by,

$$\min_{\theta,\psi} \mathcal{L}(\theta, \psi) + \lambda_1 \cdot \mathcal{R}_1(\psi) + \lambda_2 \cdot \mathcal{R}_2(\psi), \tag{3.8}$$

where θ and ψ are learnable parameters in the supernetwork and architecture controller, respectively. The first resource regularizer $\mathcal{R}_1(\cdot)$ penalizes FLOPs of currently searched architectures by *de-selecting* channels, formally defined as:

$$\begin{aligned}\mathcal{R}_1(\psi) &= \mathcal{R}_{\text{FLOPs}}(f^a, \boldsymbol{x}) + \sum_{m=1}^{M} \mathcal{R}_{\text{FLOPs}}(f^s, \tilde{\boldsymbol{x}}, \boldsymbol{t}_m) \\ &= 2\sum_{n=1}^{N} \mathrm{K}_n^2 H_n W_n \cdot [\phi_n \cdot \sum_{c=1}^{C} g(z_{n,c}^a) \cdot \sum_{c=1}^{C} g(z_{n-1,c}^a) \\ &\quad + (1-\phi_n) \cdot \sum_{m=1}^{M} \{\sum_{c=1}^{C} g(z_{m,n,c}^s) \cdot \sum_{c=1}^{C} g(z_{m,n-1,c}^s)\}],\end{aligned} \tag{3.9}$$

where K_n is the kernel size of convolution operation to generate the n-th feature map, H_n and W_n are the height and the width of the n-th feature map, respectively, and $z_{0,c}^a$ and $z_{m,0,c}^s$ the channel of input images and are fixed to be 1. The second regularizer $\mathcal{R}_2$ enforces the network to maximize the number of the early shared layers by penalizing the *disagreement* of selected channels across tasks as follows:

$$\mathcal{R}_2(\psi) = \sum_{n=1}^{N} \phi_{n-1} \cdot \sum_{c=1}^{C} \sum_{m=1}^{M} \left\| g(z_{m,n,c}^s) - g(z_{n,c}^a) \right\|_1, \tag{3.10}$$

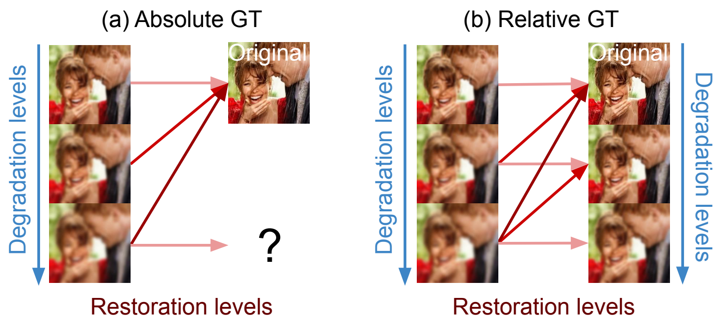


Figure 3.4: Absolute ground truth *vs.* relative ground truth. (a) Mapping from all degraded versions to its original image. (b) Mapping from degraded versions to relatively higher-quality images.

where layer at $n = 0$ denotes an input image and $\phi_0 \equiv 1$ since the input image is fixed over tasks. The hyperparameters λ_1 and λ_2 balance three terms.

3.3.3 Data Sampling Strategy

Degradation level *vs.* restoration level. Previous works train a network to restore the original image from the degraded images with arbitrary degradation level (see Figure 3.4(a)). However, CIR algorithms should be able to restore images to various extents to facilitate better user interaction experience. Thus, we redefine a restoration level as a mapping from more degraded images (input) to less degraded images (relative GT) (see Figure 3.4(b)).

Task vector with relative GT. A task vector $\boldsymbol{t}$ is a model input that encodes restoration levels. In training, an input-GT image pair (sampled with two different multi-dimensional degradation levels) determines its task vector as follows:

$$t_d \equiv l_d^{in} - l_d^{gt}, \tag{3.11}$$

where $l_d^{in} \in [0,1]$ and $l_d^{gt} \in [0,1]$ denote the levels of d-th degradation type for the input and GT images, respectively. We assume GT images are less degraded than input ($l_d^{in} \geq l_d^{gt}$). Each image pair randomly selects the number of degradation types (single or multiple) and the granularity of degradation levels (continuous or binary).

3.4 Experiments

In this section, we present the experimental results and comparisons between TAS-Net and CResMD in terms of network computation cost and output image quality. Then, we thoroughly analyze the effectiveness of our proposed algorithm with the ablation studies. Implementation details are described in Section 3.5.

3.4.1 Dataset

In this work, we use DIV2K [5] dataset for training and CBSD68 [85] dataset for testing, unless specified otherwise. DIV2K consists of 800 clean 2K-resolution training images and 100 validation images while CBSD68 consists of 68 clean HVGA-resolution test images. Following the degradation setting in CResMD [43], to create degraded images, we use three types of degradation: Gaussian blur, Gaussian noise, and JPEG compression. Each degradation is sequentially applied to the clean images. For Gaussian blur, we use the kernel size of 21×21 with the radius $r \in [0,4]$. The covariance for the Gaussian noise is denoted as $\sigma \in [0,50]$. The JPEG compression quality factor is denoted as $q \in [10,100]$ (We also include images with no JPEG compression as in CResMD). The training dataset is constructed by applying the degradation levels with a stride of 0.1, 1, and 2 for r, σ, and q, respectively.

Table 3.1: The average computation cost comparison. TASNet outperforms CResMD [43] *w.r.t.* all measures and resolutions.

Cost metric	Resolution	CResMD	TASNet
	HD	1,124.3 G	**45.2 G**
FLOPs$_\downarrow$	2K	2,698.4 G	**108.4 G**
	4K	10,119.2 G	**406.7 G**
	HD	22.8 s	**5.5 s**
CPU latency (single)$_\downarrow$	2K	55.6 s	**13.5 s**
	4K	209.3 s	**55.5 s**
	HD	5.1 s	**1.7 s**
CPU latency (multi)$_\downarrow$	2K	11.7 s	**4.2 s**
	4K	40.6 s	**13.1 s**
	HD	144.4 ms	**68.4 ms**
GPU latency$_\downarrow$	2K	280.8 ms	**99.2 ms**
	4K	930.0 ms	**250.7 ms**

Table 3.2: Quantitative image quality comparison for non-blind restoration on CBSD68.

Method	PSNR$_\uparrow$	SSIM$_\uparrow$	NIQE$_\downarrow$	BRISQUE$_\downarrow$	FLOPs$_\downarrow$
CResMD	**25.86 dB**	**0.8194**	6.7165	54.13	189.1 G
TASNet	25.64 dB	0.8137	**6.6301**	**50.60**	**7.5 G**

3.4.2 Computation Cost Comparison

Latency and FLOPs reduction. Table 3.1 presents the average computation cost of TASNet (ours) and the state-of-the-art network, CResMD [43], across diverse image resolutions and devices. The experiments are performed for the controllable image restoration setting, in which a multiple number ($M = 27$) of inferences are performed for each input image. While displaying similar image restoration performance (as described in the next section), TASNet manages to reduce 95.7% FLOPs from CResMD and shows faster speed on all reported devices: $\times 3.8$ on a single-core CPU, $\times 3.1$ on a multi-core CPU, and $\times 3.7$ on a GPU, when generating 4K (3840×2160) images, compared with CResMD. Notably, TASNet only requires 0.07s to restore an HD (1280×720) image. We also observe that the latency difference between two models becomes small in the case of low-resolution images. As the input resolution decreases, the size of each feature map also decreases, reducing the benefit of selecting channels or shared layers to some extent.

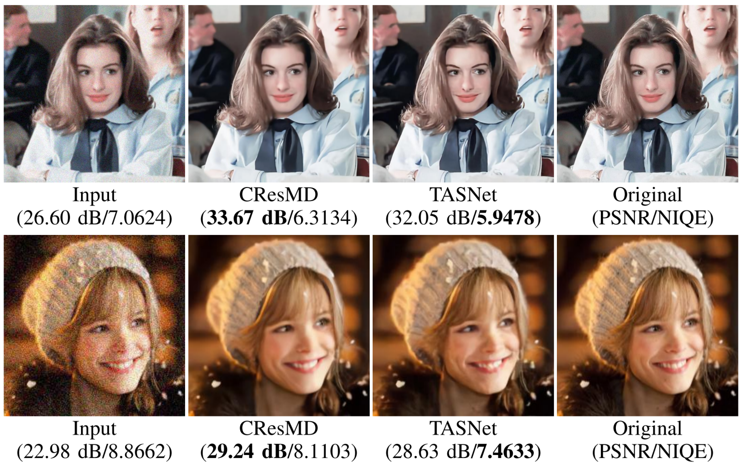


Figure 3.5: Qualitative image quality comparison for non-blind restoration. TASNet produces sharper images with better NIQE scores than CResMD.

3.4.3 Image Quality Comparison

Non-blind setting. Table 3.2 illustrates the quantitative image quality comparisons in a non-blind image restoration setting, where the degradation type and level of input images are known. This setting allows models to generate their best results with a single inference. The results demonstrate that the images restored by TASNet have lower PSNR than the images generated by CResMD but better NIQE, which means that TASNet in general restores sharper image details, as illustrated in Figure 3.5.

Blind setting. Controllable image restoration (CIR) algorithms aim to tackle a blind setting, in which the types and levels of degradation are unknown. CResMD [43] struggles to handle such challenging scenario, generating images with artifact (Figure 3.6) or over-smoothing effect (Figure 3.7) and restoring images unevenly across continuously varying restoration levels (Figure 3.8). Figure 3.6 presents images re-

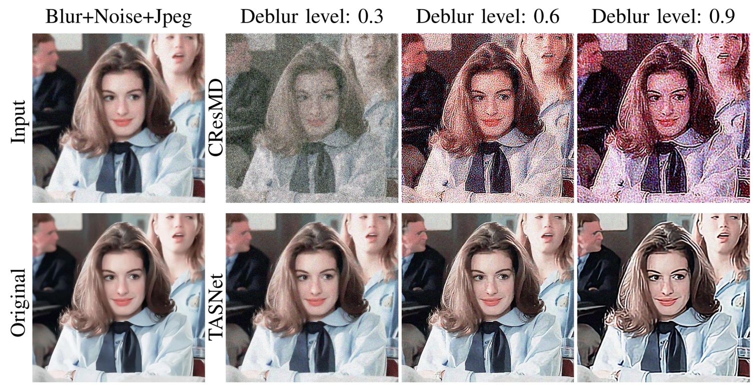


Figure 3.6: Blind setting (artifact). CResMD often produces significant artifacts when deburring images that are corrupted with noise or compression artifacts. By contrast, TASNet successfully reduces the blur in input images.

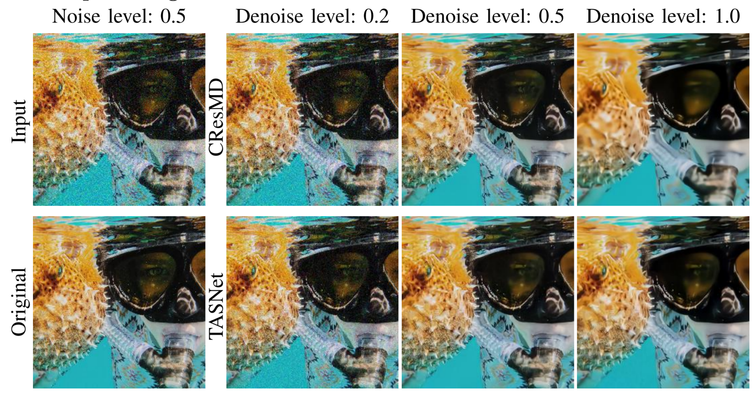


Figure 3.7: Blind setting (over-smoothing). When denoise levels are higher than the actual noise levels of input images, CResMD over-smoothes images whereas TASNet noticeably removes noise.

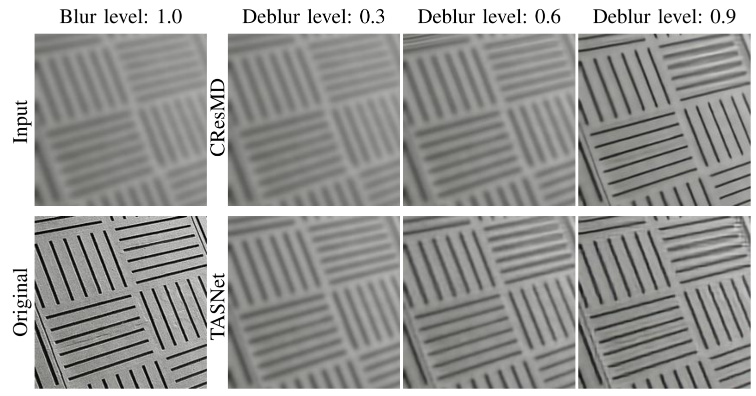


Figure 3.8: Blind setting (uneven modulation). CResMD restores blurred images negligibly or drastically by deblur level changes, while TASNet gradually modulates outputs.

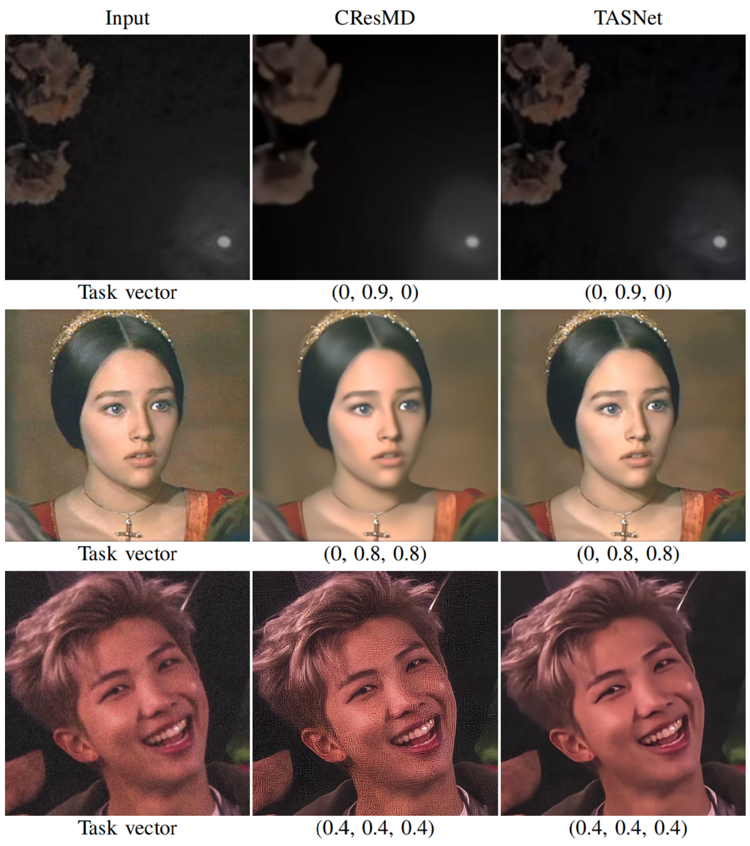


Figure 3.9: Restoration from real images. A task vector $(\cdot,\cdot,\cdot)$ denotes the levels of (deblur, denoise, dejpeg). Compared to TASNet, CResMD generates more artifacts or over-smoothed images.

stored by algorithms that modulate an input image using different levels of deblurring when the input image is corrupted with a mixture of blur, noise, and compression degradation. Images restored by TASNet are shown to be less blurry (in fact, the outputs become sharper as deblurring level becomes higher), compared to CResMD that generates critical artifacts. An interesting observation is that other degradations (noise and JPEG artifact) still remain in images deblurred by TASNet, implying that our algorithm manages to learn a disentangled restoration process for each restoration type and level. As observed in Figure 3.7, denoising by CResMD

Table 3.3: Ablation study of naïve shared networks. We modify CResMD by sharing its early layers. TASNet-A achieves 9 times FLOPs reduction than CResMD with 62% shared layers.

Method	#Shared Layer	PSNR$_\uparrow$	NIQE$_\downarrow$	FLOPs$_\downarrow$
CResMD	0 %	25.86 dB	6.7165	189.1 G
	31 %	25.82 dB	6.8035	132.0 G
	62 %	25.78 dB	6.8205	69.5 G
	99 %	25.34 dB	6.9109	7.0 G
TASNet-A	62 %	25.75 dB	6.7982	7.5 G

results in over-smooth images while TASNet maintains the overall contents and structures of input images. Figure 3.8 illustrates the restored images when varying the restoration (deblurring in this case) levels for identical degradation and restoration types. With the same amount of change in restoration levels, CResMD restores the degradation unevenly (the restoration quality changes negligibly or drastically), whereas TASNet generates images with smoothly-varying restoration quality.

Restoration on real images. Figure 3.9 displays the output images restored from real images with unknown degradation (downloaded from the internet). Similar to the synthetic examples, CResMD generates images with over-smooth effect or significant artifacts while TASNet successfully reduces the degradation of input images. For more restored output images, please refer to Section 3.6.

3.4.4 Ablation Study

Comparison to naïve shared networks. Table 3.3 shows the comparisons between the variations of CResMD with a different number of early shared layers that is manually determined. For fair comparisons in terms of performance, all models in the table are trained with absolute GT. TASNet-A has the same network architecture as TASNet, but is trained with absolute GT. TASNet-A reduces the computation cost of CResMD to $\frac{1}{9}$ by sharing 62% of the layers with a similar PSNR.

Table 3.4: Ablation study for the effectiveness of shared layers. TA and TS, respectively, denote task-agnostic layers and task-specific layers. TSNet is our searched model without forcibly sharing layers. Image quality is measured on CBSD68 using the non-blind setting.

Method	TA	TS	PSNR$_\uparrow$	NIQE$_\downarrow$	FLOPs$_\downarrow$
CResMD	-	-	**25.86 dB**	6.7165	189.1 G
TSNet	-	✓	25.59 dB	6.6332	39.6 G
TASNet	✓	✓	25.64 dB	**6.6301**	**7.5 G**

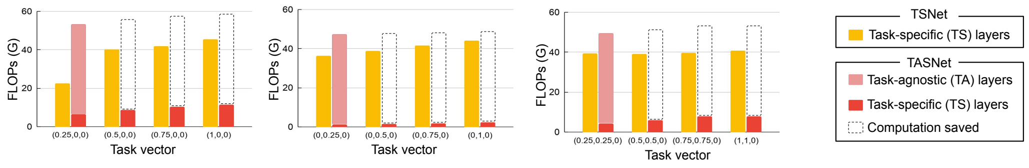
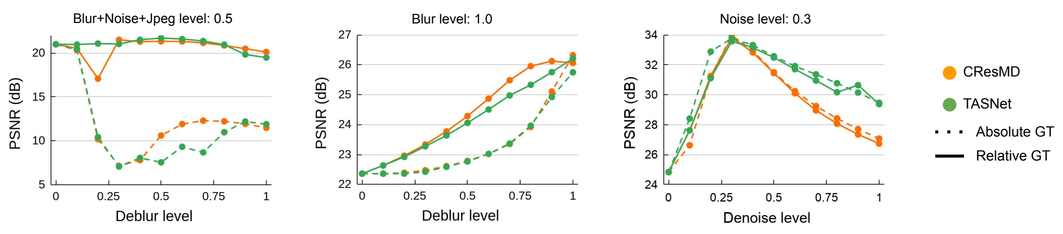


Figure 3.10: Computation cost comparisons across restoration levels. The higher restoration levels demand more computation costs for TS layers. TASNet is efficient for multiple inferences, as a result of reusing the feature map of TA layers across tasks. Each graph presents computation costs of removing blur (left), noise (middle), and joint degradation of blur and noise (right). A task vector (·,·,·) denotes the levels of (deblur, denoise, dejpeg).

The effectiveness of sharing layers in TASNet. Table 3.4 studies the importance of task-agnostic layers in TASNet. In particular, we examine how the performance and computation cost change after disabling layer sharing, the resulting network from which is denoted as TSNet in the table. TASNet is observed to save the computation costs of TSNet by more than 5 times, owing to its shared layers that allow feature reuse and thus reducing the redundant computation for multiple inferences. Regardless, TSNet greatly reduces the computation cost of CResMD, suggesting the effectiveness of the task-specific layers that adaptively select important channels. In stark contrast, conventional channel pruning approaches do not change their architectures *w.r.t.* the task.

To further emphasize the effectiveness of TASNet, Figure 3.10 shows the computation cost of TSNet and TASNet across various tasks with multiple numbers (4) of inferences for each image of HVGA resolution (481×321). Task-specific layers tend to require more channels for higher restoration levels, which translate to more



(a) Artifact generation (b) Uneven modulation (c) Over-smoothing outputs

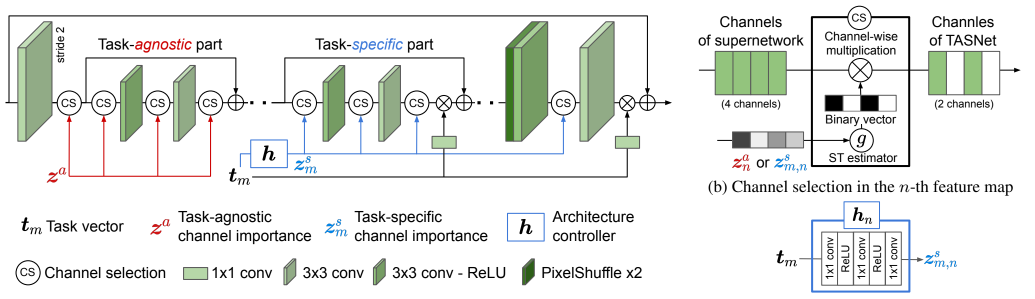
Figure 3.11: Ablation study for three image modulation problems in the blind setting. Models trained by relative GT reduce (a) network artifact generation when deblurring images corrupted by a mixture of degradation types and (b) uneven image modulation across deblurring levels. Further, task-agnostic feature maps from TASNet prevent (c) over-smoothing outputs.

difficult restoration problems. Task-agnostic layers are computed only once for each input image and hence require a substantially smaller amount of computation from the second pass. Although TASNet requires a higher computation cost than TSNet for the first inference, the overhead becomes negligible during multiple inferences.

Image restoration quality analysis. To study the effectiveness of the proposed data sampling strategy and the TASNet architecture, Figure 3.11 presents the quantitative restoration performance of three major failure cases in CIR when the controlled restoration levels differ from the actual types and levels of degradation in an input image. CResMD trained with relative GT generates less artifacts and evenly modulated outputs, validating the capability of the proposed data sampling to improve image restoration quality. Also, TASNet achieves higher PSNR (over 3 dB compared to CResMD) in denoising without over-smoothing (Figure 3.7 and 3.11(c)), alluding to the effectiveness of the shared layers in providing better image quality.

3.5 Implementation Details

Supernetwork architecture. We use the network architecture of CResMD as the supernetwork in the proposed search algorithm. Our supernetwork consists of



(a) Overview of TASNet architecture

(b) Channel selection in the n-th feature map

(c) Architecture controller in the n-th feature map

Figure 3.12: TASNet architecture. (a) During searching for the number of shared early layers (task-*agnostic* part) in the supernetwork, z^a determines where to prune in the task-*agnostic* part. By contrast, z^s_m selects channels in the remaining layers (task-*specific* part) specialized in the m-th task-vector t_m (the control factor of restoration levels). We omit the notations for the feature map index n and the channel index c for simplicity. (b) In the network training phase, channel-wise multiplication between a binary vector and a feature map operates as virtual channel selection (CS) for the differentiable neural architecture search process. (c) Architecture controller consists of fully connected layers and predicts task-*specific* channel importance from the task vector.

32 enhanced residual blocks which have a ReLU activation layer between two convolution layers with 64 filters of the kernel size 3×3. The first convolution layer with a stride of 2 downscales the input images, and the last upsampling module consists of PixelShuffle layer, two convolution layers, and a ReLU activation layer. Global skip connection adds the input image to the output of the upscaling module. A task vector scales the residual feature map in the location of 32 local connections and 1 global connection by a 1×1 convolution layer with channel-wise multiplication.

TASNet architecture. The proposed algorithm determines the number of shared layers and selects the channels of each shared or non-shared layer. Figure 3.12(a) illustrates the TASNet architecture. For the shared layers (task-agnostic part), the channels that are not selected at the end of training are pruned in the final model. On the other hand, the non-shared layers (task-specific part) adaptively select their channels *w.r.t* the input task vector. During the training, the channels are virtually selected by channel-wise multiplication to the binary vectors, as described in Figure 3.12(b). Our channel selection modules are located at all feature maps after the initial PixelShuffle layer of CResMD. The architecture controller consists of 3 fully-connected layers with ReLU activation function, as described in Figure 3.12(c). In the task-agnostic part of the supernetwork, the residual scaling modules are removed to make the feature maps independent to specific tasks.

Hyperparameters for the search algorithm. TASNet sets the hyperparameters α, γ, M, λ_1, and λ_2 as 0.9, 0.9, 64, $5 \times e^{-11}$, and $1 \times e^{-2}$, respectively. The mini-batch consists of 64 image patches with 64×64 resolution. The initial learning rate is 1×10^{-4}. TASNet is trained for 1×10^{6} iterations using Adam optimizer [64] with the learning rate decay of ×0.5 after the first half of training.

Image quality measure. In this work, we utilize three widely used image quality measures, PSNR, SSIM, NIQE [7], and BRISQUE [86] to evaluate the quality of images produced by models. PSNR and SSIM are full-reference measures in that the restored images are compared with the original clean images. On the other hand, NIQE and BRISQUE are no-reference evaluation metrics, in which the restored image quality is measured without referring to the original image. Images with higher PSNR, higher SSIM, lower NIQE, and lower BRISQUE scores are considered to have better quality. However, measuring image quality during adjusting restoration levels has not been studied thoroughly. Thus, we visualize extensive qualitative results in Section 3.4.3 and 3.6.

Degradation in non-blind test images. For fair comparisons in a non-blind setting, we construct CBSD68 with the combinations of three levels and three types of degradation; Gaussian blur with $r \in \{0, 2, 4\}$, Gaussian noise with $\sigma \in \{0, 25, 50\}$,

Table 3.5: Ablation study of hyperparameters λ_1 and λ_2 on CBSD68.

Ex.#	λ_1	λ_2	#Shared layer	PSNR	FLOPs	
					Single inference	Multiple inferences
①		1×10^{-3}	18 %	25.67 dB	35.2 G	23.1 G
②	5×10^{-11}	1×10^{-2}	62 %	25.75 dB	52.9 G	7.5 G
③		1×10^{-1}	99 %	25.48 dB	125.5 G	4.8 G
④	5×10^{-12}		99 %	25.46 dB	154.6 G	6.0 G
⑤	5×10^{-11}	1×10^{-2}	62 %	25.75 dB	52.9 G	7.5 G
⑥	5×10^{-10}		16 %	25.50 dB	15.4 G	1.9 G

and JPEG compression with $q \in \{None, 60, 10\}$. Among the 27 combinations of degradation, we omit $(r, \sigma, q) = (0, 0, None)$ which generates identical images to the original. PSNR, SSIM, NIQE, and BRISQUE in all tables of this paper report the average scores on CBSD68 with the 26 combinations of degradation.

Computation cost metric. We measure the computation costs of the networks in FLOPs and latency. FLOPs is a classical device-agnostic metric and exponentially increases by image resolution. Since latency is device-dependent, we measure latency on CPU with single-core (CPU latency (single)), CPU with multi-core (CPU latency (multi)), and GPU (GPU latency). We use Intel i7-5960X CPU which has 16 cores and GeForce RTX 2080 Ti GPU. The computation costs reported in this paper are average scores to generate images with 27 restoration levels unless otherwise mentioned, where $t_d \in \{0, 0.5, 1\}$ for $\boldsymbol{t} \in \mathbb{R}^3$.

3.6 Additional Experiments

Balancing the hyperparameters. Table 3.5 presents the ablation study of hyperparameters λ_1 and λ_2 which balance the trade-off between the network computation cost and the number of shared layers while minimizing Eq. (3.8) of the main paper. The models trained with small λ_1 and large λ_2 have large portions of shared

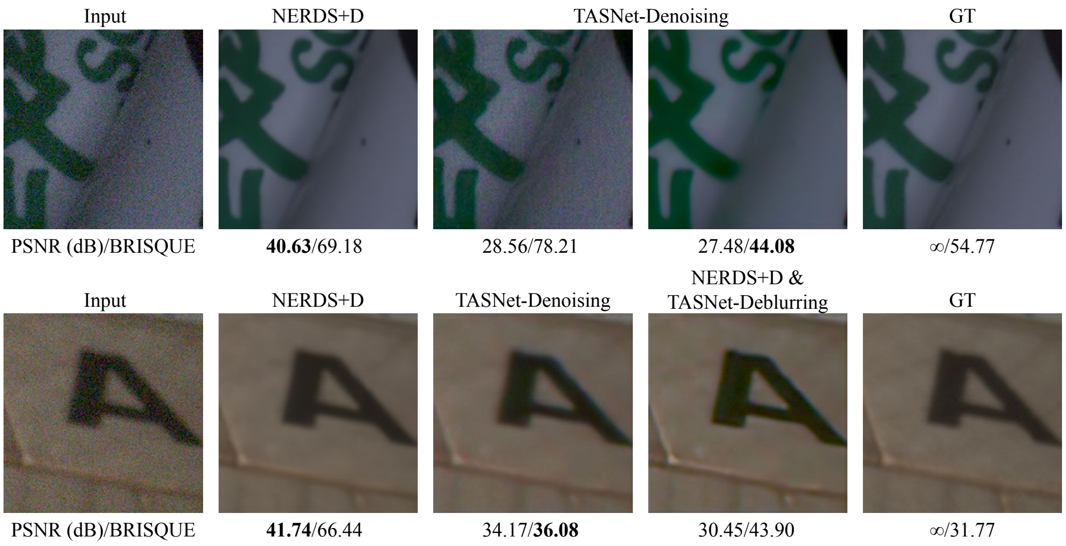


Figure 3.13: Comparisons between NERDS+D and TASNet.

layers, and thus they are efficient in generating multiple (27) images (② *vs.* ③ and ⑤ *vs.* ④). In contrast, the models trained with the opposite balance between λ_1 and λ_2 are efficient for a single inference (② *vs.* ① and ⑤ *vs.* ⑥).

Comparisons to NERDS+D. Chapter 2 presents a camera denoiser (NERDS+D) for Poisson-Gaussian noise while TASNet is able to remove Gaussian noise, Gaussian blur, and JPEG compression artifact. Figure 3.13 visualizes two examples for comparisons between NERDS+D and TASNet on real noisy images from the SIDD [2] validation. In the first row, NERDS+D achieves a high PSNR score of which the result is closely similar to the reference image (GT). Instead, TASNet-Denoising suffers low PSNR scores due to the inaccurate noise model (Gaussian). However, some outputs from TASNet-Denoising have better BRISQUE scores than the output from NERDS+D, where BRISQUE measures image naturalness without reference images. The second row demonstrates another example of the joint restoration of NERDS+D and TASNet-Deblurring. Despite the input image being assumed to contain only noise, the alphabet 'A' in GT looks blurry. While NERDS+D and TASNet-Denoising remove noise with better scores in different image quality measures, NERDS+D&TASNet-Deblurring outputs sharper images than GT. TASNet allows users to choose the preferred output for their preferences in such cases.

Extra qualitative results. We present more qualitative comparisons between CResMD and TASNet in the blind setting where users have to generate diverse restored images by controlling the restoration levels (task vectors) for unknown degradation of an input image. Recall that CResMD incurs three problems in this scenario: artifacts in the generated images, over-smoothed outputs, and uneven modulation across the task vectors. Figure 3.14 and 3.15 show that CResMD produces output images with undesired and visually unpleasing artifacts. Figure 3.16 presents less artifacts in the outputs of CResMD, but the outputs are over-smoothed compared to the outputs of TASNet even for the true task vector. Figure 3.17 also shows over-smoothed outputs for CResMD when restoring the input images with high restoration levels for denoising and dejpeg. By contrast, TASNet maintains the sharp textural details of the input image and removes visually unpleasing noise and compression artifacts of the input. Figure 3.18 exemplifies the problem of uneven modulation for CResMD. While CResMD produces images with negligible changes for lower values of deblurring level, it exhibits drastic changes for higher levels. In contrast to CResMD, TASNet demonstrates more even modulation across the different task vectors and generates smoothly-varying images. Figure 3.19, 3.20, and 3.21 presents modulation scenarios for a *real-word image* with unknown degradation, in which modulations with various task vectors are inevitable to find the visually pleasing images. These results demonstrate that CResMD sometimes generates severely destructive artifacts (especially in Figure 3.19) and overly-smooth outputs (especially in Figure 3.20) during the modulation process whereas TASNet generates plausible images for various task vectors.

3.7 Conclusion

We propose a novel neural architecture search algorithm to find efficient networks for controllable image restoration (or image modulation). In particular, the proposed algorithm searches for a network with task-agnostic and task-specific layers, referred to as TASNet, by determining the number of layers and channels to share across tasks and adaptively selecting channels in non-shared feature maps. We formulate all learning objectives in a differentiable manner and perform the architecture search in an end-to-end manner. The shared layers facilitate feature reuse that pushes the network efficiency further for controllable image restoration that requires a several number of inferences. Together with the proposed new data sampling strategy, not only does TASNet reduce the network computation costs of the state-of-the-art network greatly but also provides the better image quality.

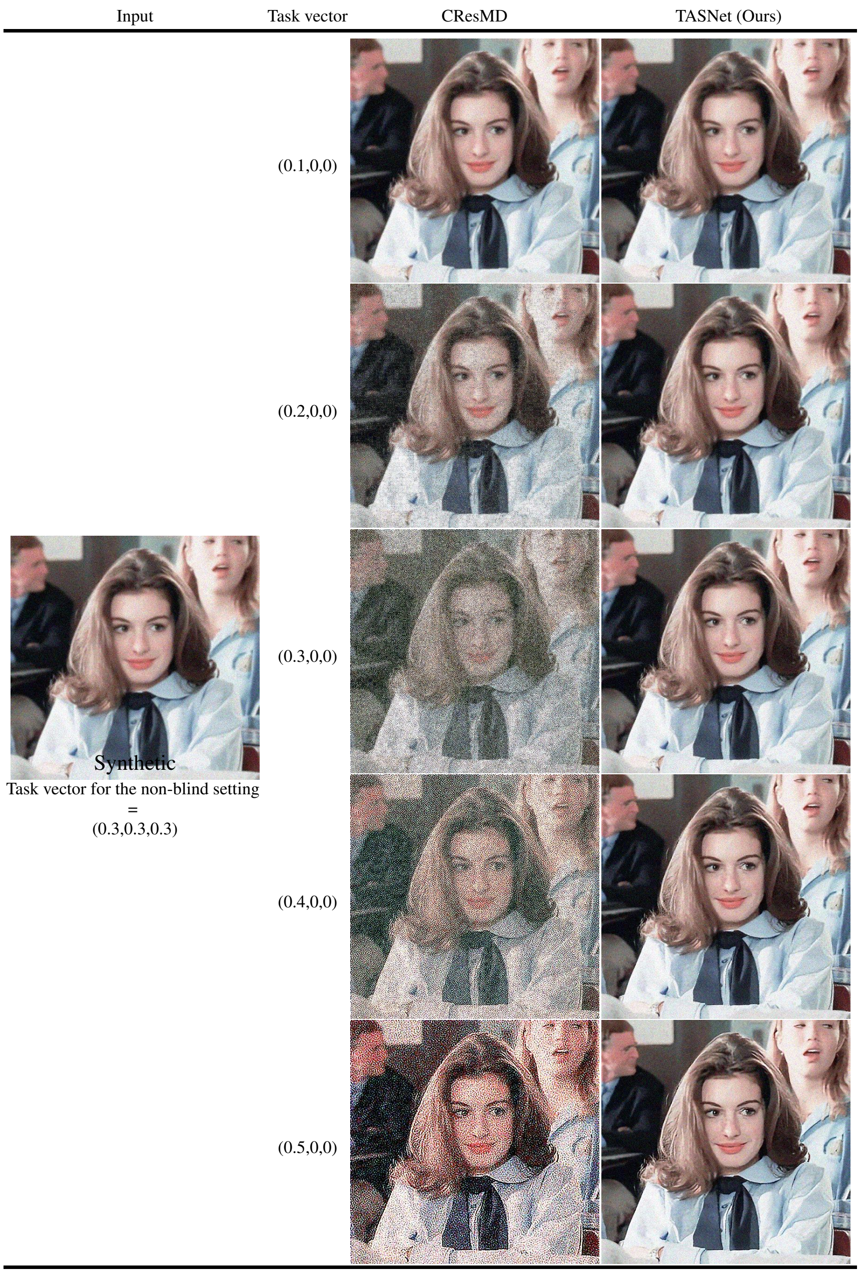


Figure 3.14: Deblur modulation examples to the image with blur, noise, and JPEG compression. Our TASNet generates diverse images with respect to the given restoration levels (task vectors). TASNet generates less auxiliary visual artifacts. The values of task vector denote restoration levels of (deblur, denoise, dejpeg), respectively.

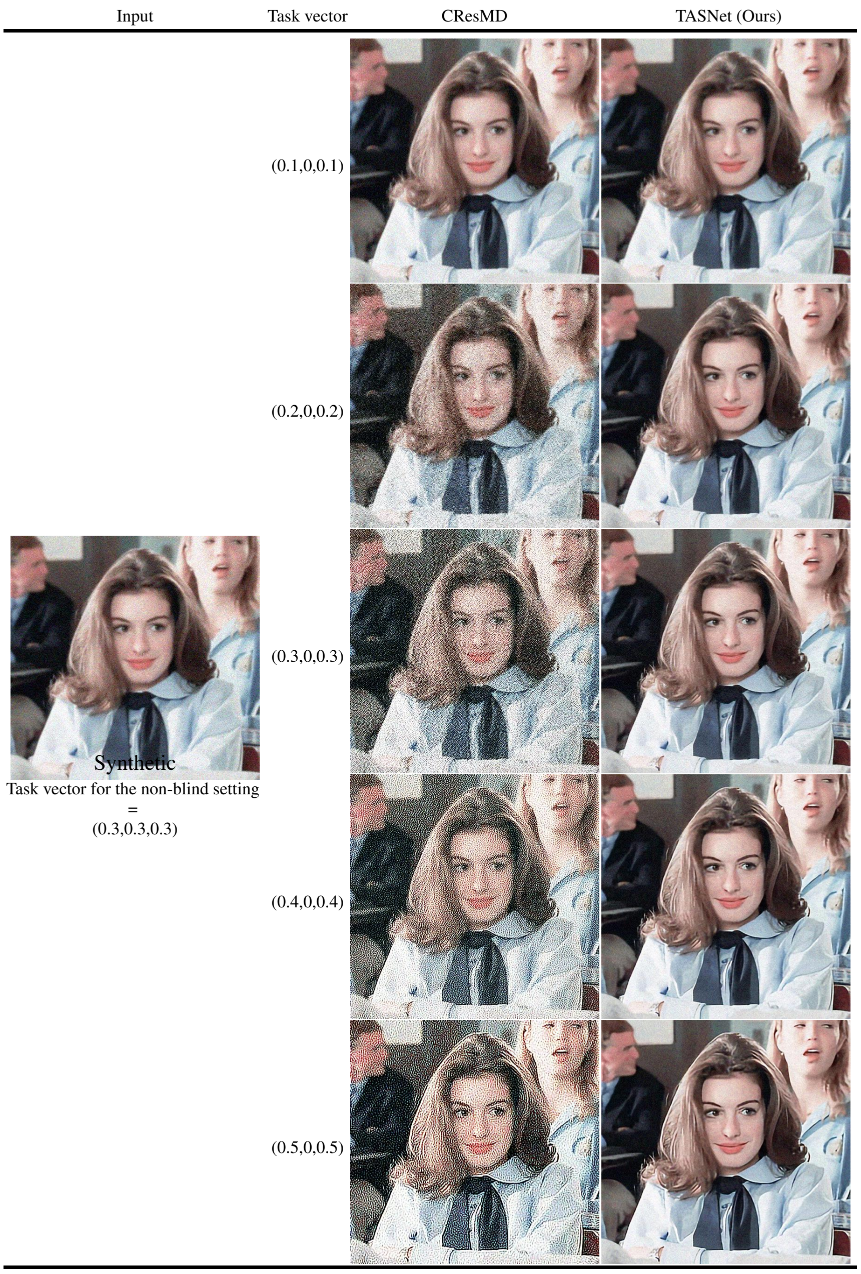


Figure 3.15: Deblur and dejpeg modulation examples to the image with blur, noise, and JPEG compression. Our TASNet generates diverse images with respect to the given restoration levels (task vectors). TASNet generates less auxiliary visual artifacts. The values of task vector denote restoration levels of (deblur, denoise, dejpeg), respectively.

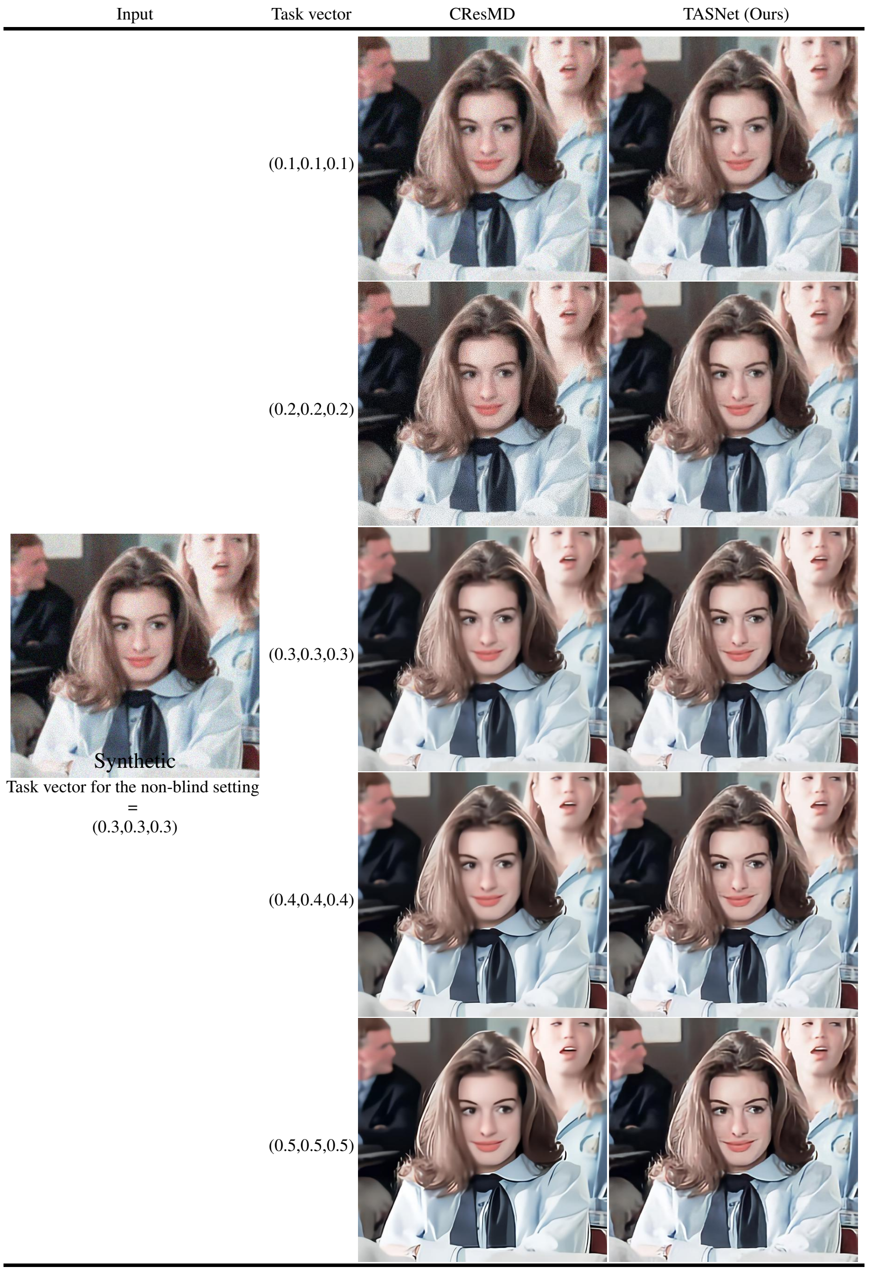


Figure 3.16: Deblur, denoise, and dejpeg modulation examples to the image with blur, noise, and JPEG compression. Our TASNet generates diverse images with respect to the given restoration levels (task vectors). TASNet generates less auxiliary visual artifacts and over-smoothed textures. The values of task vector denote restoration levels of (deblur, denoise, dejpeg), respectively.

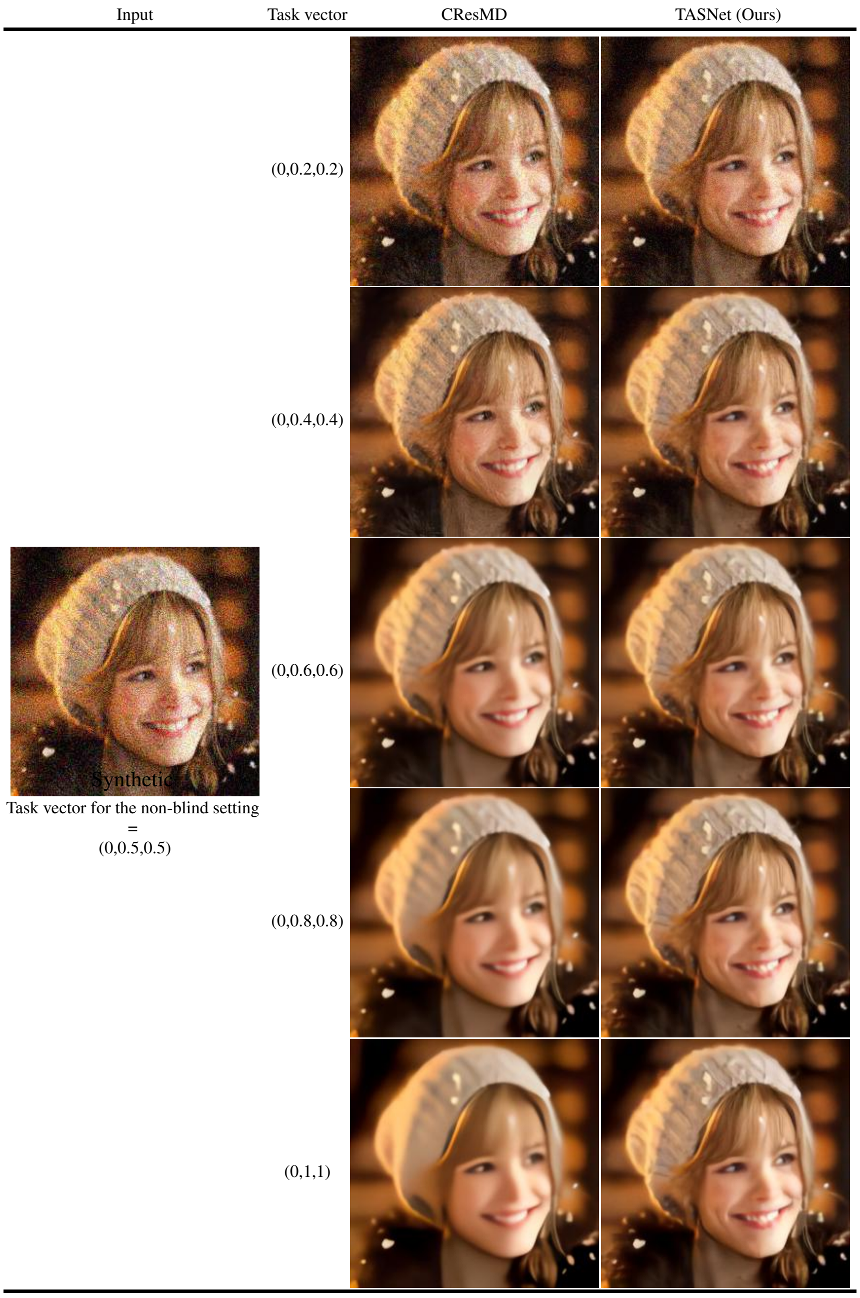


Figure 3.17: Denoise and dejpeg modulation examples to the image with noise and JPEG compression. Our TASNet generates less over-smoothed textures. The values of task vector denote restoration levels of (deblur, denoise, dejpeg), respectively.

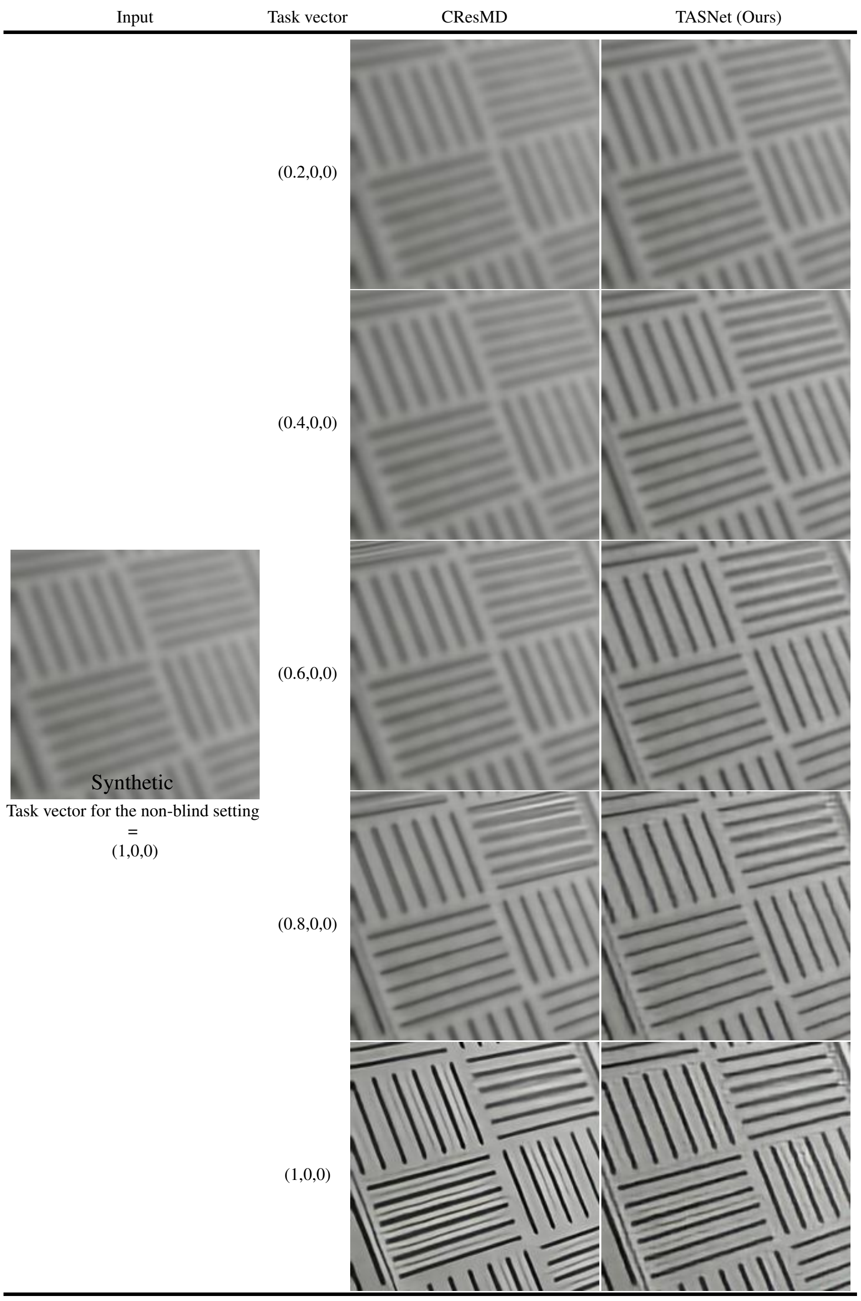


Figure 3.18: Our TASNet generates evenly modulated images with respect to the given restoration level changes. The values of task vector denote restoration levels of (deblur, denoise, dejpeg), respectively.

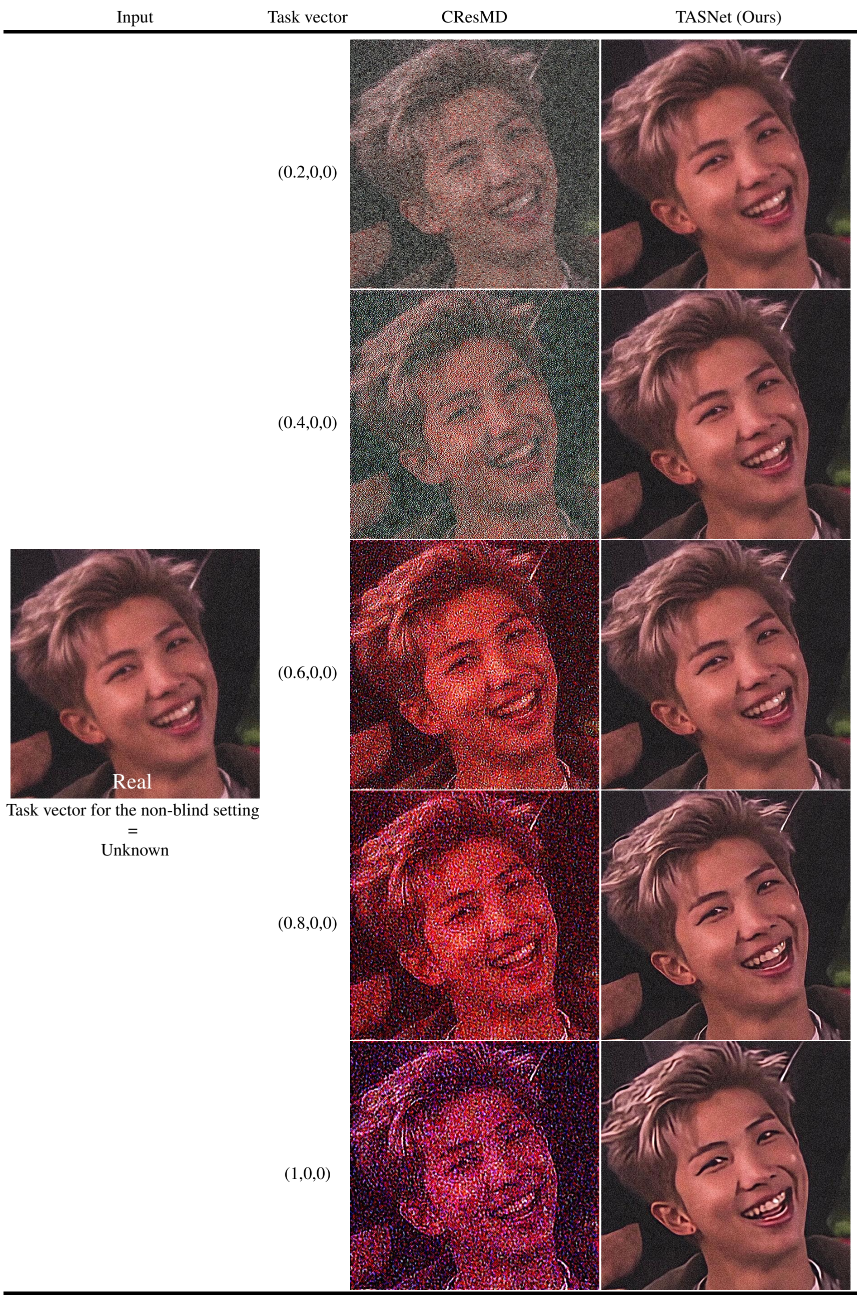


Figure 3.19: Deblur modulation examples to the real world image on the Internet. Our TASNet generates diverse images with respect to the given restoration levels (task vectors). TASNet generates less auxiliary visual artifacts. The values of task vector denote restoration levels of (deblur, denoise, dejpeg), respectively.

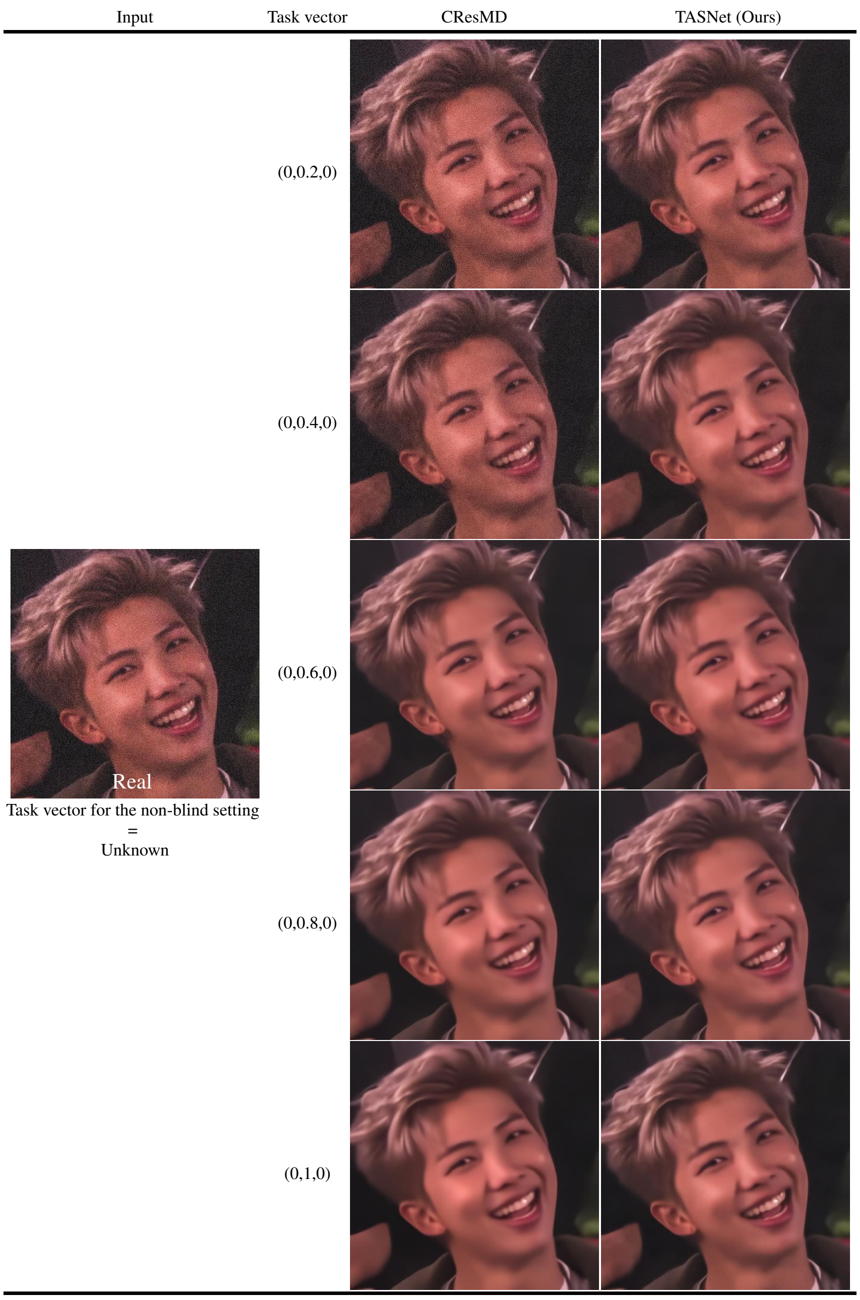


Figure 3.20: Denoise modulation examples to the real world image on the Internet. Our TASNet generates diverse images with respect to the given restoration levels (task vectors). TASNet generates less over-smoothed textures. The values of task vector denote restoration levels of (deblur, denoise, dejpeg), respectively.

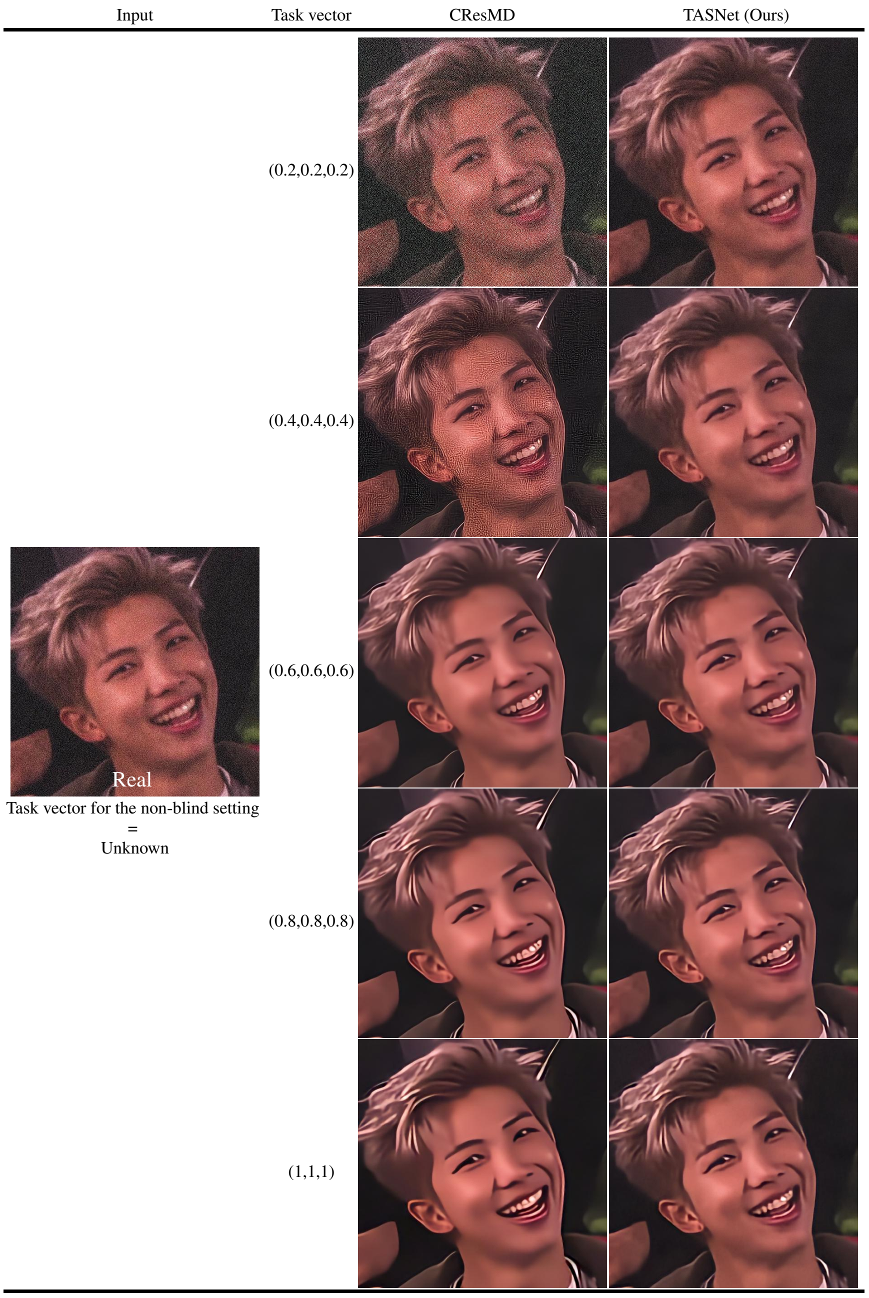


Figure 3.21: Deblur, denoise, and dejpeg modulation examples to the real world image on the Internet. Our TASNet generates diverse images with respect to the given restoration levels (task vectors). TASNet generates less auxiliary visual artifacts and over-smoothed textures. The values of task vector denote restoration levels of (deblur, denoise, dejpeg), respectively.

Chapter 4

Adaptive ISP Parameter Estimation for Controllable Image Enhancement

4.1 Introduction

Most people enjoy taking pictures of precious moments with smartphones. However, turning a flat-looking mobile photo into a stunning image requires skill and time. Even adjusting contrast and brightness is a challenge given that it depends on the image contents. To leverage supervised machine learning for this problem, Bychkovsky *et al.* provided a large-scale, high-quality dataset [15]. They hired trained photographers to retouch each photo manually. The photographers used professional image editing tools to achieve attractive expressions, including tone, color, contrast, and brightness. The result is a collection of 5,000 low-quality (LQ) images with high-quality (HQ) versions retouched by photographers. On this dataset, image enhancement aims to convert LQ images into HQ images (LQ2HQ image transformation).

Many machine learning approaches for image enhancement have focused on learning retouching preferences. The works in [24, 104, 88, 105, 75] learned the prefer-

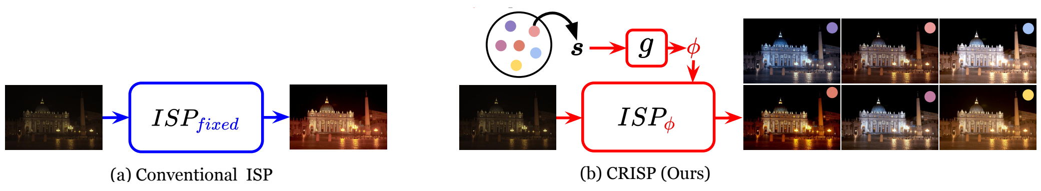


Figure 4.1: Overview of the Image Signal Processor (ISP) for image enhancement. (a) Conventional ISPs have predetermined functions with fixed parameters, generating a single output for an image input. (b) Our ContRollable ISP (CRISP) produces diverse styles of high-quality images by adjusting the parameters of the ISP (ϕ). CRISP represents high-quality image styles into low-dimensional codes (style representation) through an autoencoder and decodes a selected code ($\boldsymbol{s}$) to the ISP parameters using a neural network (g).

ences of a particular photographer. They trained neural networks with the images retouched by a particular photographer to estimate the photographer's adjustments for arbitrary images. More works [56, 18, 61, 99] learned the retouching preferences that can change on the basis of each user. The user selects preferred images from the dataset, and the neural networks aim to transform a test image into a style similar to the selected images. However, the concept of *retouching preference learning* confuses machines about what to learn because the retouching preferences of users can vary depending on many factors; the preferred styles change every so often depending on the user's mood, environment, or personal experience.

Recently, a few works have alleviated the preference-learning problem by providing multiple outputs. Zero-DCE [40] adjusts the brightness of low-light images with multiple levels. CSRNet [44] generates intermediate styles between two different image styles. Users can select the desired image from the multiple outputs or adjust some factors to explore the style of the output images. However, all these methods are limited in generating diverse expressions or styles that can satisfy users.

Another issue of deep-learning-based image enhancement methods is the model efficiency. When a neural network performs pixel-to-pixel mapping [24] or pixel-wise transformation from input to output [36, 116, 104], it requires many computations

(FLOPs) for high-resolution images. Thus, instead of using original input images, existing methods often utilize downscaled images with predefined efficient functions. For instance, reinforcement-learning-based methods [91, 48, 66] take mapping curves for global enhancement [15, 115], and 3DLUT [125] utilizes trilinear interpolation with 3D look-up tables. The neural networks adapt the mapping curves or 3D look-up tables to the input downscaled image. However, these works still have limitations in real-world applications. Reinforcement learning-based methods sacrifice efficiency by interactively enhancing the input images. 3DLUT also uses many parameters for 3D look-up tables, and efficient trilinear interpolation must be realized with custom implementation for each computing device.

To address the limitations of existing methods on style diversity and model efficiency, we introduce a new method on the basis of the Image Signal Processor (ISP). Our method is motivated by camera image pipeline [95], where some functions in ISPs are specialized in adjusting image expressions, *i.e.*, enhancement, such as color correction, tone mapping, and gamma correction. ISPs have been widely used in real-world digital cameras or smartphones given that they have a few parameters and computations. Camera manufacturers carefully define the functions of ISPs and tune their parameters based on the principles of an image sensor and human perception of colors. Once manually tuned, the ISP is fixed and generates a single style of output for a given input image (Fig. 4.1(a)). As a result, conventional ISPs suffer from limited styles of output images.

Thus, we propose a *ContRollable* Image Signal Processor (CRISP) that users can easily control to generate various HQ image styles for a given image while leveraging the practical benefits of ISPs. CRISP employs an efficient ISP backbone for global image enhancement (style generation) and predicts its desired parameters via a neural network (Fig. 4.1(b)). Users can select the output styles from the style representation (style vectors), which an autoencoder learns into a low-dimensional latent code for easy and fast adjustments. The ISP comprises differentiable functions so that the autoencoder can still be trained end-to-end through gradient-descent-based optimization. In testing, for style adjustment, users can find the desired style representation in several ways, including 1) a style predicted from downscaled LQ images, as done in previous works, 2) a style frequently used in the training datasets, and 3) a style determined by user interaction. We *do not focus on a specific scenario for style adjustments* but on how easy it is to generate diverse image styles. Thus, we evaluate CRISP with several practical scenarios for real-world applications.

We present comprehensive experiments on MIT-Adobe FiveK [15] to show the diversity and controllability of output styles. We show that: 1) CRISP generates diverse image styles for users that are learned from different photographers, whereas the counterpart conventional *fixed* ISP often fails to generate the desired results. 2) The latent style vector in CRISP encodes insensitive and distinct trends in each dimension for ISP parameters and output image styles. 3) CRISP achieves better image quality with less computational costs and parameters than previous works.

Our contributions are summarized as follows,

- To the best of our knowledge, we are the first to learn a controllable ISP using an autoencoder for image enhancement. More importantly, our proposed model (CRISP) easily produces various high-quality image styles by adjusting the parameters of the ISP.
- CRISP incorporates a highly efficient plug-and-play ISP for practical applications, comprising 19 parameters and requiring less than 100 FLOPs per pixel.

- CRISP easily controls output image styles with several practical scenarios for style adjustments by encoding the styles into low-dimensional latent codes.
- CRISP outperforms previous works for image enhancement in image quality (MOS, PSNR) and model efficiency (FLOPs, parameters).

4.2 Related Works

4.2.1 Image Enhancement

Image enhancement has a long and rich research history. Professional image editing tools (*e.g.*, Photoshop) offer a wide range of control and flexibility to adjust image expressions, such as tone, color, contrast, and brightness, but they require skill and time. Research in this area aims to automate the adjustment of image expressions.

We categorize image enhancement algorithms on the basis of the degree of automation. The first category includes early rule-based approaches for one-size-fits-all enhancement. These approaches produce a single result for everyone to adjust image contrast using histogram equalization [92], gamma correction [49], or retinex algorithms [70]. However, they often generate unacceptable results given that the style adjustment must consider image contents.

Thereafter, MIT-Adobe FiveK [15] proposed a large-scale dataset for image enhancement. In the dataset, five human experts (or photographers) retouched flat-looking (*i.e.*, low-quality) images into the images with different stunning expressions (*i.e.*, high-quality) using a professional image editing tool (Lightroom). This dataset not only allows data-driven approaches to consider image contents for enhancement but also derives the second category, learning the retouching preferences of a particular expert. Early approaches [51, 15, 115] and deep-learning-based approaches [24, 88, 104, 58, 60] train their models using the LQ-HQ image pairs from a particular expert and estimate the images retouched by the expert. Recent works [75, 105, 44] focused on improving the efficiency of neural networks due to their high computational costs. However, given that they produce the results of a particular expert's preference, they do not satisfy the needs of general users.

Third, the line of works [56, 55, 46, 18, 61, 99] aim to provide personalized enhancement. They allow users to select preferred images first and then retouch the input image with the styles similar to those of the selected ones. For instance, PieNet [61] identifies the user preferences with 512 parameters which are generated by 20 images selected in 500 images. Similarly, StarEnhancer [99] fine-tunes output styles on the basis of the selected images. Recently, Google proposed Real Tone which more accurately highlights personalized skin tones on smartphones. However, satisfying users with a single image remains a challenge.

The fourth and last category allows users to adjust image outputs interactively. Although works in this area also reduce the time and effort of using image editing tools, they have attracted less attention. Lischinski *et al.* [77] presented a local tone adjustment for the regions selected by users. Xiao *et al.* [110] proposed histogram-based algorithms to control brightness and contrast. Zero-DCE [40] estimates light-enhancement curves that allow users to adjust the brightness of low-light images. LTMNet [133] enables editing tone curves for local image enhancement. However, these approaches cover only a fraction of image expressions.

Recently, CSRNet [44] presented a deep network for the second category and an image interpolation method for user interaction. CSRNet generates an image output that estimates the style of an expert and then produces intermediate styles between the input and output images using image interpolation. Users can adjust the interpolation coefficient to obtain the desired results. However, the interpolated images have limited expression diversity. In contrast, the proposed algorithm (CRISP) generates high-quality (HQ) images with various styles easily and efficiently.

4.3 Proposed Method

4.3.1 Overview

We propose a novel algorithm, CRISP (ContRollable ISP) that aims to generate multiple high-quality (HQ) images with various attractive expressions (styles) for a given low-quality (LQ) input image. We learn CRISP using an autoencoder, including an ISP as a decoder as visualized in Fig. 4.2. The autoencoder represents the HQ image style in a low dimension (3 in this chapter), allowing fast and easy exploration of the style space. During testing, various scenarios are possible for users to provide the latent code (or style vector) of the style they want without using the encoder. For instance, users can adjust the style vectors interactively or use a predetermined set of style vectors.

Formally, our CRISP (denoted by a function $\mathcal{CRISP}$) is a style-autoencoder with a *style-adaptive* ISP (denoted by a function $\mathcal{ISP}$) as a controllable decoder with parameters $\boldsymbol{\phi}$. The encoder (f) encodes the HQ image styles in a D-dimensional latent code using the LQ image ($\boldsymbol{x}$) and the paired HQ image ($\hat{\boldsymbol{x}}$) as inputs. The style decoder (g) predicts the parameters ($\boldsymbol{\phi}$) of the $\mathcal{ISP}$ from the latent code. The $\mathcal{ISP}$ predicts the estimated HQ image ($\tilde{\boldsymbol{x}}$) from the LQ image ($\boldsymbol{x}$) as follows,

$$\tilde{\boldsymbol{x}} = \mathcal{CRISP}(\boldsymbol{x}; \boldsymbol{\phi}) = \begin{cases} \mathcal{ISP}(\boldsymbol{x}; g(f(\boldsymbol{x}, \hat{\boldsymbol{x}})) & \text{if training} \\ \mathcal{ISP}(\boldsymbol{x}; g(\boldsymbol{s})) & \text{if testing,} \end{cases} \tag{4.1}$$

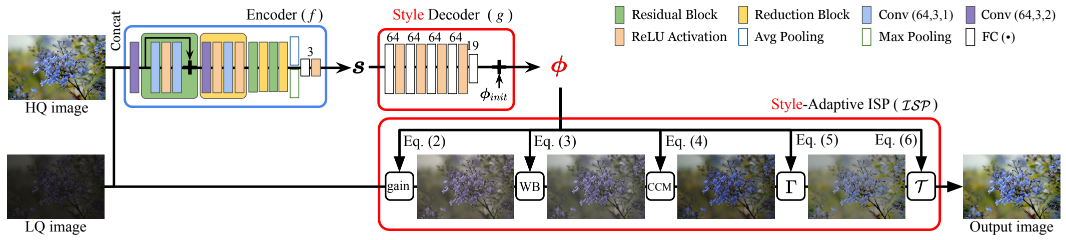


Figure 4.2: Proposed learning framework for CRISP. CRISP learns the representation of HQ image styles through an autoencoder. Our encoder (f) accepts concatenated LQ/HQ images as input and represents the styles in latent codes named style vectors ($\boldsymbol{s}$). Our decoder consists of two parts: the style decoder (g) and the style-adaptive ISP ($\mathcal{ISP}$). g produces the ISP parameters ($\boldsymbol{\phi}$) from $\boldsymbol{s}$ and $\mathcal{ISP}$ with $\boldsymbol{\phi}$ estimates the HQ image from the paired LQ image. Eq. (4.2)$\sim$(4.6) are differentiable functions that enable end-to-end training by gradient-descent methods. Conv is a convolution layer, where ($\cdot$,$\cdot$,$\cdot$) denote the number of filters, the kernel size, and the stride size, respectively. Avg pooling and Max pooling reduce the feature resolution to 1×1 and concatenate both output features. FC ($\cdot$) is a fully connected layer, where ($\cdot$) is the number of filters denoted at the top of the boxes.

where the style vector $\boldsymbol{s} \in \mathbb{R}^D$ denotes the latent codes (or style representation), which determines the styles of output images. The style vector ($\boldsymbol{s}$) in Eq. (4.1)(if testing) enables multiple style generation and can easily obtain the desired results through k-means clustering or greedy search algorithms (Fig. 4.12).

4.3.2 Learning framework

Conventional autoencoders, such as VAE [65], encode the image input into a latent code by reconstructing the input from the code. Here, we propose a *style-autoencoder* for image style encoding by removing image content information in the autoencoder framework (Eq. (4.1)(if training)). The style-autoencoder uses an HQ image as input and reconstructs it similarly to the VAE, but *the encoder and the decoder also take the paired LQ image as input.* Our decoder comprises two steps, estimating the parameters of $\mathcal{ISP}$ and transforming LQ images to HQ images using the $\mathcal{ISP}$. $\mathcal{ISP}$ is specialized in image expression and cannot change image contents. Thus, the style-autoencoder is forced to encode image styles *only.*

Fig. 4.2 presents the architecture of the style-autoencoder. A resnet-style architecture is used for the encoder (f), comprising 12 convolution layers with 64 channels and concatenated LQ and HQ images as input. The style vector ($\boldsymbol{s}$) is the D-dimensional non-negative vector of which values determine the styles of final output images. The style decoder (g) comprises 5 fully connected layers with 64 channels that predict the residuals to the initial $\mathcal{ISP}$ parameters ($\boldsymbol{\phi}_{init}$). The residual learning enables stable training by estimating the parameters of $\mathcal{ISP}$ within effective ranges to adjust image expressions. The style decoder (g) is computationally efficient given that it has a spatial resolution of 1×1 and can be computed independently to the test image. The next section describes the functions constituting $\mathcal{ISP}$.

4.3.3 ISP Functions

Our $\mathcal{ISP}$ module performs a *global style adjustment* with digital gain, white balance, color correction, gamma correction, and tone mapping (see Fig. 4.2 ($\mathcal{ISP}$)). The style-adaptive parameters ($\boldsymbol{\phi}$) determine an exact function of image transformation applied to all pixels of an LQ image. The constraint of global adjustment has the advantage of preventing the image content from being distorted by the transformation. The functions of $\mathcal{ISP}$ are differentiable for end-to-end training. This section describes each function of our $\mathcal{ISP}$ module with exact formulations and output styles (or expressions).

4.3.3.1 Digital Gain

Digital cameras commonly apply a global scaling to all pixel values for intensity adjustment, where the exposure time determines the scaling factor. Similarly, photographers brighten or darken an image for different styles. We set the global scaling factor ϕ_{dg} as an $\mathcal{ISP}$ parameter,

$$\mathrm{gain}(x; \boldsymbol{\phi}) = \phi_{dg} \cdot x, \tag{4.2}$$

where $x \in [0, 1]$ is a normalized pixel value of each red, green, and blue channel for an input image. The range of ϕ_{dg} is [0.85, 2.17] in test images.

4.3.3.2 White Balance

The illumination color changes the color of the objects captured by a camera. White balance conventionally aims to visualize the 'true' color of an object or adjust it to different light conditions. Conversely, photographers often change the light conditions to capture visually pleasing colors. We reinterpret white balance as illumination color control by per-channel scaling functions for red and blue colors,

$$\mathrm{WB}\left(\begin{bmatrix} x_r \\ x_g \\ x_b \end{bmatrix}; \boldsymbol{\phi}\right) = \begin{bmatrix} \phi_r \cdot x_r \\ x_g \\ \phi_b \cdot x_b \end{bmatrix}, \tag{4.3}$$

where x_r, x_g, and x_b represent the pixel values of red, green, and blue, respectively, of which ranges are [0,1]. In the test dataset, ϕ_r has a smaller variation of [0.73, 1.07] than ϕ_b of [0.80, 2.41]. This result is reasonable since the human visual system is more sensitive to red than blue.

4.3.3.3 Color Correction

In general, the color filters of an image sensor have their own RGB spectra. Using a color correction matrix (CCM), an ISP converts the 'camera space' RGB color to

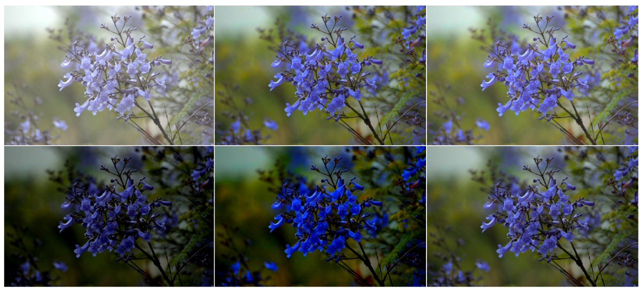


Figure 4.3: Example results of different CCMs in Eq. (4.4).

the standard sRGB color. We assume that the photographer's adjustment includes the color space conversion. We use the CCM which consists of 3×4 ISP parameters:

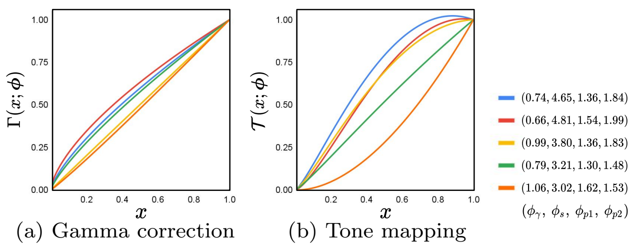
$$\mathrm{CCM}\left(\begin{bmatrix} x_r \\ x_g \\ x_b \end{bmatrix}; \boldsymbol{\phi}\right) = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} \begin{bmatrix} x_r \\ x_g \\ x_b \end{bmatrix} + \begin{bmatrix} \phi_{o1} \\ \phi_{o2} \\ \phi_{o3} \end{bmatrix}, \tag{4.4}$$

where ϕ_{oi} denotes the color offset for the i-th row in the matrix. We follow a general constraint of CCM as $\sum_j \phi_{ij} = 1$ where $i \in \{1, 2, 3\}$. Fig. 4.3 presents the color space conversion by the CCM used for the test dataset.

4.3.3.4 Gamma Correction

Human eyes perceive the gradations of color in the dark area better. Gamma correction is a function that displays the low-intensity pixels with more bits. While a conventional ISP has fixed parameters, we adapt the parameters for each style in the following gamma correction function:

$$\Gamma(x; \boldsymbol{\phi}) = \max(x, \epsilon)^{\phi_\gamma}, \tag{4.5}$$



(a) Gamma correction (b) Tone mapping

Figure 4.4: Example results of (a) gamma correction curves in Eq. (4.5), and (b) tone mapping curves in Eq. (4.6).

where $\epsilon = 10^{-8}$ to prevent negative values in output pixels for the stable training and $x \in [0, 1]$ is a pixel value of each red, green, and blue channel for an input image. Fig. 4.4(a) displays the gamma curves in the test dataset. Most values of ϕ_γ are smaller than 1 to increase low-intensity values.

4.3.3.5 Tone Mapping

When visualizing high dynamic range images (or high-bit images) in 24-bit RGB format, tone mapping curves often adopt S-shape that allocates more bits for mid-intensity values. Similar to gamma correction, we adopt a simple function that can shape S-curves with parameter tuning,

$$\mathcal{T}(x;\boldsymbol{\phi}) = \phi_s \cdot \max(x, \epsilon)^{\phi_{p1}} - (\phi_s - 1) \cdot \max(x, \epsilon)^{\phi_{p2}}, \tag{4.6}$$

where $\epsilon = 10^{-8}$ and x is the pixel value of each red, green, and blue channel for an input image, scaled to [0,1] for a general expression for various bit-widths. ϕ_s, ϕ_{p1}, and ϕ_{p2} determine the shapes of S-curves, passing through (0,0) and (1,1) regardless of the parameter values. Fig. 4.4(b) presents the examples of our tone mapping curves with various S-shapes.

4.3.3.6 Parameter Initialization

We initialize the parameters of $\mathcal{ISP}$, $\boldsymbol{\phi}_{init}$ to avoid indirect solutions of image transformation (*e.g.*, digital gain with a negative value). Specifically, ϕ_{dg} is set to 1.2, the parameters of WB and CCM are set to ensure the identity mapping that generates the same output image as the input, ϕ_{γ} is set to $\frac{1}{2.2}$, and ϕ_{s}, ϕ_{p1}, and ϕ_{p2} are set to 3, 2 and 3, respectively. The style decoder (g) learns the residual parameters of $\mathcal{ISP}$, denoted as $\boldsymbol{\phi} - \boldsymbol{\phi}_{init}$.

4.4 Experiments

4.4.1 Dataset

We train and evaluate our method on the MIT-Adobe FiveK dataset [15] that consists of 5,000 camera raw images and paired sRGB images retouched by five professional photographers, denoted as Expert-A, Expert-B, Expert-C, Expert-D, and Expert-E. We use images retouched by Expert-C unless otherwise specified. State-of-the-art methods on this dataset commonly downscale the raw and sRGB images and preprocess the raw images to low-quality (LQ) sRGB images. For fair comparisons, we follow two settings for downscaling and preprocessing in CSRNet [44] and 3DLUT [125] by using Lightroom[1] and downloading the released dataset[2], respectively. Specifically, CSRNet downscales images to 500 pixels for the long edge, and 3DLUT downscales the images to 480 pixels for the short edge. The numbers of test images are 500 and 498 for CSRNet and 3DLUT. We use the remaining for training.

[1]https://github.com/yuanming-hu/exposure/wiki/Preparing-data-for-the-MITAdobe-FiveK-Dataset-with-Lightroom

[2]https://github.com/HuiZeng/Image-Adaptive-3DLUT

4.4.2 Implementation Details

Training. We set the latent style code $\boldsymbol{s}$ as a non-negative 3D vector ($D = 3$). We adopt an MSE loss between the outputs and the ground truth HQ images to train the style-autoencoder with Adam optimizer [64]. We randomly crop images to 200×200 pixels and flip and rotate the patches for augmentation. We set the batch size to 16 and the initial learning rate to 1×10^{-4}. During 1.6×10^{5} iterations, we halve the learning rate for every quarter of training.

Testing. We measure the quality of enhanced images and the model efficiency. We assess the image quality with non-reference-based measures (qualitative results and MOS) and reference-based measures (PSNR and SSIM). We use images retouched by Expert-C as the reference images for PSNR and SSIM unless otherwise specified. We compare model parameters, floating-point operations (FLOPs), and the number of inferences to find final outputs for model efficiency. The number of inferences indicates how easily and fast users can obtain the results they prefer by an algorithm. We select the style vectors with multiple scenarios using random sampling, an image-adaptive encoder, k-means clustering, and greedy search algorithms.

4.4.3 Models for Comparisons

We compare the enhanced results of our method (CRISP) with 9 state-of-the-art (SotA) methods including DAR [91], White-Box [48], Pix2Pix [53], HDRNet [36], DPED [52], DeepLPF [88], 3DLUT [125], SA3DLUT [105], and CSRNet [44]. For a given LQ image, most existing methods produce a single enhanced image, while CRISP can generate multiple output images with different styles. The 'Expert (A-E)' represents HQ images enhanced by 5 human experts in the MIT-Adobe FiveK dataset [15]. We summarize the main characteristics and difference between our CRISP and the most recent models, CSRNet, 3DLUT, and SA3DLUT as follows:

- **CSRNet [44]** adopts fully connected layers for image processing and convolution layers for feature modulation. The output of convolution layers transforms the features of fully connected layers adaptively to the image contents.
- **3DLUT [125]** uses image-adaptive trilinear interpolation for image processing. Trilinear interpolation allows FLOPs-efficient inference but requires a relatively large number of parameters for lookup tables.
- **SA3DLUT [105]** adopts spatially adaptive 3DLUT that requires more FLOPs and parameters for spatially varying image adjustment.
- **CRISP (Ours)** uses an ISP for image processing and fully connected layers to adjust the parameters of the ISP.

4.4.4 Results of Multiple Style Generation

Visual comparisons. Our CRISP generates diverse realistic styles with natural and vivid colors like the human experts (Expert (A-E)) as visualized in Fig. 4.5. We randomly select the output images of CRISP on each LQ image for style diversity. The LQ images from the MIT-Adobe FiveK dataset generally have low pixel values to avoid saturation. DAR, White-Box, and HDRNet brighten the input LQ image but change its natural color. Pix2Pix generates a natural-colored image but it contains checkerboard artifacts. DPED, DeepLPF, 3DLUT, and SA3DLUT output realistic images but their colors are generally dark. CSRNet generates the output images (the right-most images in Fig. 4.5) and the intermediate styles between the LQ image and the output. CSRNet often produces too bright images, and the intermediate results

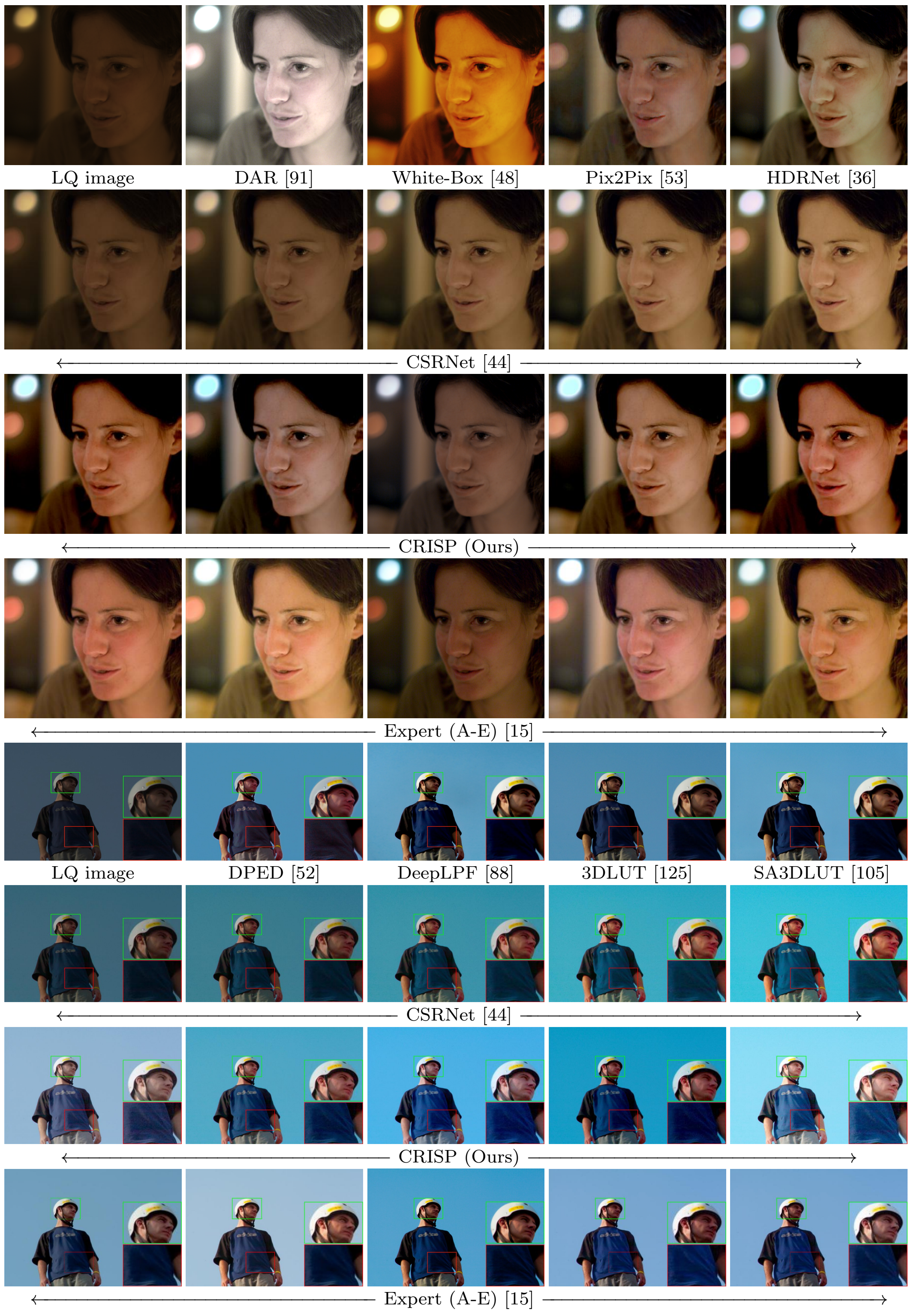
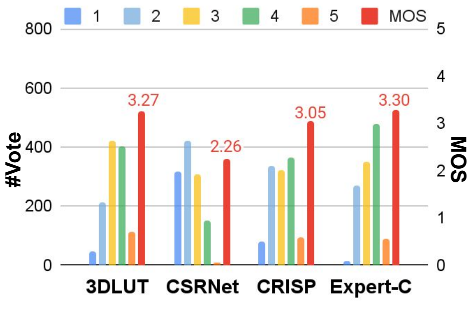
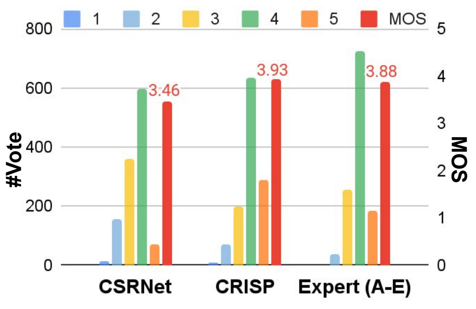


Figure 4.5: Qualitative style comparison with SotA models. We visualize five enhanced images with different styles for our method (CRISP), CSRNet, and the five Experts (A-E) [15]. The single enhanced image for each LQ image is visualized for the other methods.



(a) Single-style MOS



(b) Multiple-style MOS

Figure 4.6: User study results of mean opinion score (MOS) on MIT-Adobe FiveK. For each method 1200 samples (40 images × 30 participants) were assessed. MOS scores (red blocks) are along the right axis and the numbers of votes for each rating (the other color blocks) are along the left axis. (a) Participants rate the image quality of a single enhanced output for each LQ image. (b) Participants rate the image quality of the output they prefer the most out of five styles for each LQ image. CRISP in (b) outperforms all other methods including Expert (A-E) (3.93 *vs.* 3.88).

adjust image intensity where style diversity is limited. Fig. 4.15 and 4.16 provide more empirical comparisons.

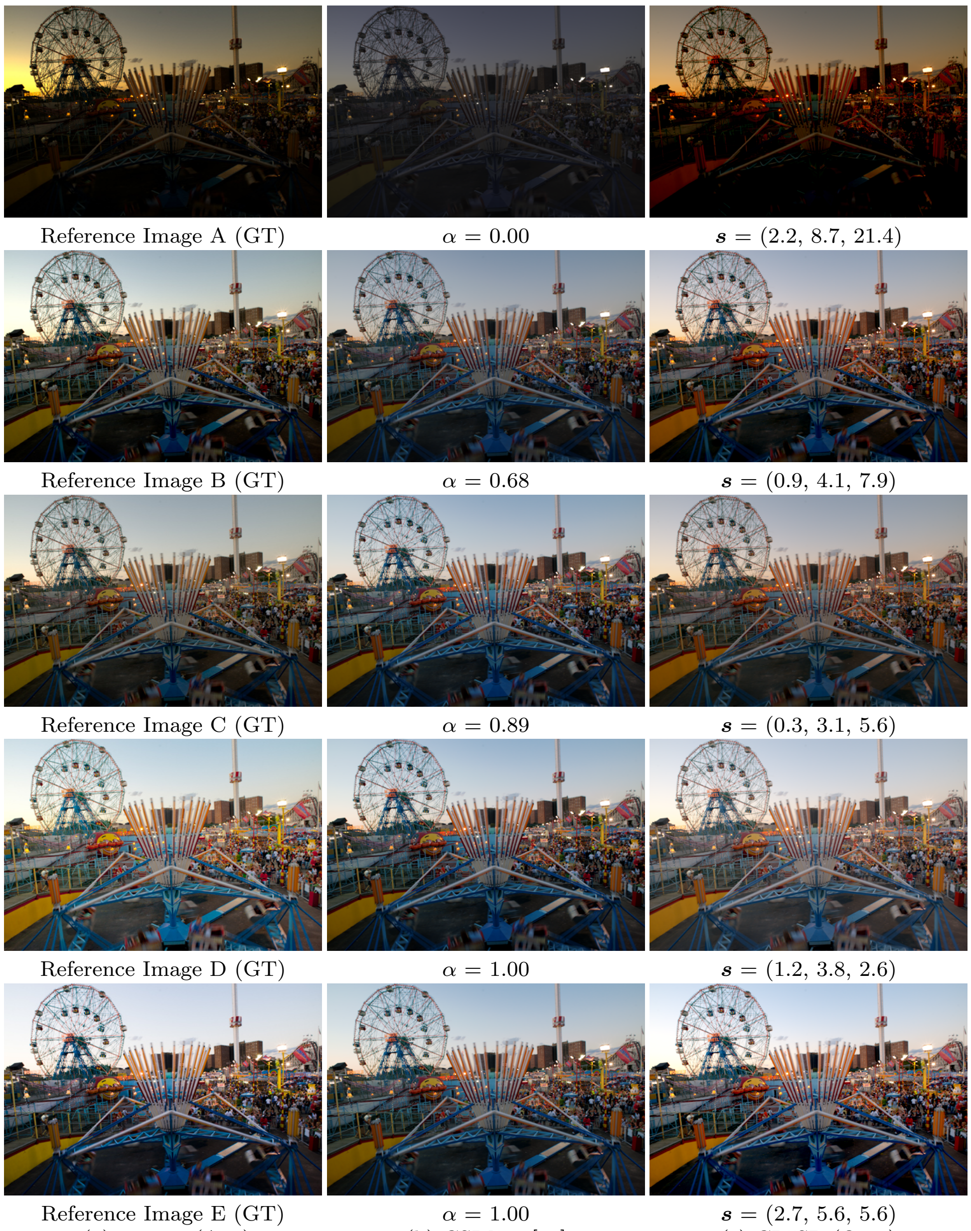
User study. To quantify the effectiveness of multiple style generation, we conducted two types of user study with 30 participants and 40 LQ images. The participants are asked to rate the image quality from 1 (bad quality) to 5 (excellent quality) to the enhanced images. In the first type of user study, called single-style MOS (Mean Opinion Score), the participants rate the quality of the single-style outputs from 3DLUT, CSRNet, CRISP, and an Expert. We regard a style as the outputs of 3DLUT and CSRNet, a randomly selected result of CRISP, and the image of Expert-C. In the second type of user study, called multiple-style MOS, the participants first select the most preferred image among five different styles from CSRNet, CRISP, and the Expert (A-E) and then rate the quality of the selected images. We use intermediate images between the input and the output for CSRNet, randomly selected images for CRISP, and images enhanced by five experts for Expert (A-E) (see Fig. 4.5) as the five styles. Fig. 4.6 visualizes the results of the user studies. In

Table 4.1: Reference-based image quality (PSNR) comparisons for multiple style generation on MIT-Adobe FiveK. Each row presents PSNR (dB) with the images retouched by each expert as the reference images. We find the most similar output of each model to the reference image. Each column presents the model performances with specified training data. CSRNet can train the model with a single HQ image (*e.g.*, the image retouched by Expert-C) for an LQ image. CRISP allows model training with multiple HQ images for an LQ image so that Expert (A-E) for training data denotes the model training using the images retouched by five experts for an LQ image.

Method	CSRNet [44]	CRISP (Ours)	
Training data	Expert-C	Expert-C	Expert (A-E)
Expert-A	24.15	28.29	**29.98**
Expert-B	24.95	30.60	**32.06**
Expert-C	24.98	**29.62**	29.11
Expert-D	22.73	29.28	**30.73**
Expert-E	21.74	**29.69**	29.65

single-style MOS, CSRNet has the lowest score since it often generates too bright images. All methods including Expert-C often fail to satisfy users as rating scores of 1 and 2. By contrast, multiple-style MOS results of CSRNet, CRISP, and Expert (A-E) outperform their scores for single-style MOS by a large margin. CRISP has the most significant number of votes for 5 (excellent quality) and the highest MOS compared to all other methods, including Expert (A-E). Fig. 4.15 and 4.16 provide more images for user studies.

Reference-based image quality comparisons. Each expert retouched an image in different styles (see Fig. 4.7(a)). To measure the ability to generate diverse styles, we choose the most similar output to each expert (or each reference image) for comparison. While CSRNet changes only image intensity for different styles (Fig. 4.7(b)), CRISP adjusts diverse image expressions such as colors, tones, contrast, and brightness, to match the reference images (Fig. 4.7(c)). Table 4.1 presents PSNR between the reference image retouched by each expert and the most similar model output. CRISP achieves 4$\sim$7 dB higher PSNR than CSRNet for all experts. Moreover, training CRISP using multiple HQ images for a single LQ image is possible. Thus, by training CRISP with multiple HQ images of all five experts, we can



(a) Expert (A-E) (b) CSRNet [44] (c) CRISP (Ours)

Figure 4.7: Qualitative comparisons of output images which are most similar to the images retouched by Expert (A-E).

improve the performance of CRISP further as presented in Table 4.1 (Expert-All). For CSRNet, we vary interpolation coefficient α with the step size of 0.01 between 0.00 and 1.00 and select the image with the highest PSNR for each expert. For CRISP, we use a GSA (Algorithm 1) to adjust the non-negative values of style vectors ($\boldsymbol{s}$) and select the output with the lowest MSE. The GSA is a proxy of user behavior in obtaining desired results with quantitative measures. Initial style vector ($\boldsymbol{s}_{init}$), step size (t), and stop condition (K) represent the propensity of user behaviors (meticulous or hasty) that affect the output image quality and the number of inferences. We use $\boldsymbol{s}_{init}$, t, and K are (0.0,0.0,0.0), 0.1, and 100, representing a meticulous user to obtain HQ images, unless otherwise specified. s_d denotes the d-th element of $\boldsymbol{s}$ and $\boldsymbol{s}+t$ denotes adding t to all elements of $\boldsymbol{s}$. The trend of the output style along each dimension of the style vector is visualized in Fig. 4.11.

Algorithm 1 Greedy Search Algorithm (GSA)

```
Require: Initial style vector $\boldsymbol{s}_{init} \in \mathbb{R}^D$, Step size $t$, Stop condition $K$
Require: Input LQ image $\boldsymbol{x}$, Reference image $\hat{\boldsymbol{x}}$
  $e \leftarrow \text{inf},\ \boldsymbol{s} \leftarrow \boldsymbol{s}_{init},\ d \leftarrow 1,\ i \leftarrow 0,\ k \leftarrow 0$
  while $k \le K$ do
      $\phi \leftarrow g(\boldsymbol{s})$
      if $e < \text{MSE}(\mathcal{CRISP}(\boldsymbol{x};\phi), \hat{x})$ then
          $s_d \leftarrow s_d - t,\ d \leftarrow (d \mod D) + 1,\ i \leftarrow i + 1,\ k \leftarrow k + 1$
          if $i = D$ then
              $\boldsymbol{s} \leftarrow \boldsymbol{s} + t,\ d \leftarrow 1,\ i \leftarrow 0$
      else
          $e \leftarrow \text{MSE}(\mathcal{CRISP}(\boldsymbol{x};\phi), \hat{x}),\ i \leftarrow 0,\ k \leftarrow 0$
      $s_d \leftarrow s_d + t$
```

4.4.5 Model Efficiency Comparisons

We compare CRISP with SotA methods in PSNR, SSIM, model parameters, FLOPs, and the number of inferences to find the desired results. We set the desired results (or reference images) to the images retouched by Expert-C as in the previous works.

Table 4.2: Model efficiency comparisons on MIT-Adobe FiveK.

Method	PSNR	SSIM	Params	FLOPs	Inferences
White-Box [48]	18.59	0.797	8.56×10^{6}	-	-
DAR [91]	19.54	0.800	2.59×10^{8}	-	-
HDRNet [36]	22.65	0.880	4.82×10^{5}	-	**1**
Pix2Pix [53]	22.05	0.788	1.14×10^{7}	5.68×10^{10}	**1**
CSRNet [44]	23.69	0.895	3.65×10^{4}	2.17×10^{9}	**1**
CRISP w/IA (Ours)	23.34	0.881	4.58×10^{5}	1.39×10^{8}	**1**
CRISP w/KMC (Ours)	25.70	0.887	$\mathbf{1.37\times10^{4}}$	$\mathbf{1.29\times10^{7}}$	27
CRISP w/GSA (Ours)	**29.62**	**0.920**	$\mathbf{1.37\times10^{4}}$	$\mathbf{1.29\times10^{7}}$	311
DPE [24]	23.76	0.881	3.34×10^{6}	-	**1**
DPED [52]	24.06	0.856	-	-	**1**
LPTN [75]	22.14	0.854	4.03×10^{5}	5.08×10^{9}	**1**
CSRNet [44]	24.23	0.900	3.65×10^{4}	4.49×10^{9}	**1**
DeepLPF [88]	25.29	0.899	8.00×10^{8}	-	**1**
3DLUT [125]	25.24	0.886	5.93×10^{5}	2.06×10^{8}	**1**
SA3DLUT [105]	25.50	0.890	4.52×10^{6}	1.11×10^{9}	**1**
CRISP w/IA (Ours)	24.38	0.880	4.58×10^{5}	1.52×10^{8}	**1**
CRISP w/KMC (Ours)	28.12	0.902	$\mathbf{1.37\times10^{4}}$	$\mathbf{2.66\times10^{7}}$	27
CRISP w/GSA (Ours)	**30.99**	**0.924**	$\mathbf{1.37\times10^{4}}$	$\mathbf{2.66\times10^{7}}$	654

Table 4.2 presents the results on the MIT-Adobe FiveK dataset with two different settings described in Section 4.4.1. We replicate PSNR and SSIM scores of the compared methods from CSRNet [44] and SA3DLUT [105]. We reproduce missing scores in two papers for fair comparisons. We measure FLOPs with the average of a single inference on each dataset. White-Box and DAR are reinforcement-based methods where the agents determine the number of adjustment steps (inferences) for each LQ image, whereas the other compared methods produce the output image with a single inference. CRISP adjusts image styles through three scenarios using image-adaptation (IA), k-means clustering (KMC), and a greedy search algorithm (GSA). CRISP w/IA redefines an encoder that takes an LQ image *only* as the input for the test setup; HQ images are not accessible. The encoder downsamples the LQ images as 64×64 and predicts 64D style vectors. CRISP w/IA requires a single inference for each LQ image like other methods and achieves similar PSNR and SSIM with fewer parameters and FLOPs. CRISP w/KMC uses k-means clustering of style vectors on the training dataset. The centers of the 27 clusters are used as a predetermined set of style vectors. Thus, in this case, the number of inferences

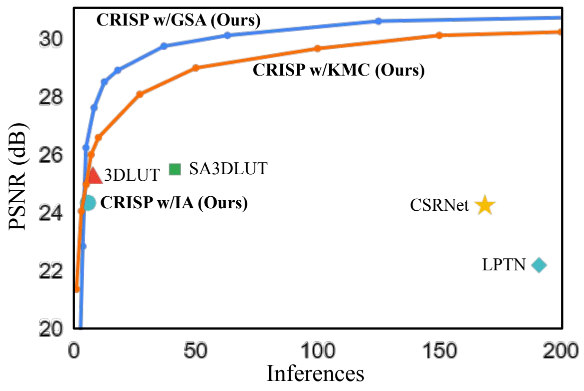


Figure 4.8: PSNR *vs.* Inferences. CRISP w/KMC and CRISP w/GSA generate multiple output styles with different scenarios for style adjustments. They estimate closer styles to the references (high PSNR) by increasing the number of output styles (high inferences). The inferences for the other methods denote the FLOPs multiplier to CRISP for visualization.

indicates the number of clusters visited. CRISP w/KMC achieves higher PSNR and SSIM than the existing methods with the smallest parameters and FLOPs by removing the encoder. Meanwhile, 27 inferences of CRISP w/KMC still require smaller FLOPs than CSRNet given that CRISP w/KMC has 160 times reduced FLOPs than CSRNet. CRISP w/GSA finds the most similar output images to the reference images using the greedy search algorithm. CRISP w/GSA achieves over 5 dB higher PSNR than the existing methods. CRISP w/GSA requires 311 inferences to obtain the desired result, but the FLOPs for all inferences are only twice that of CSRNet. CRISP w/GSA gives higher PSNR (29.62 dB *vs.* 29.61 dB) than CRISP with the encoder (f), using the reference (GT) images as the input.

For more comparisons between the multiple-style generation models (CRISP w/KMC and CRISP w/GSA) and the single-style generation models, Fig. 4.8 exhibits the trade-off between PSNR and the number of inferences. We use different numbers of clusters for CRISP w/KMC and different hyper-parameters of Algorithm 1 for CRISP w/GSA to change the number of inferences. Given that GSA is a proxy

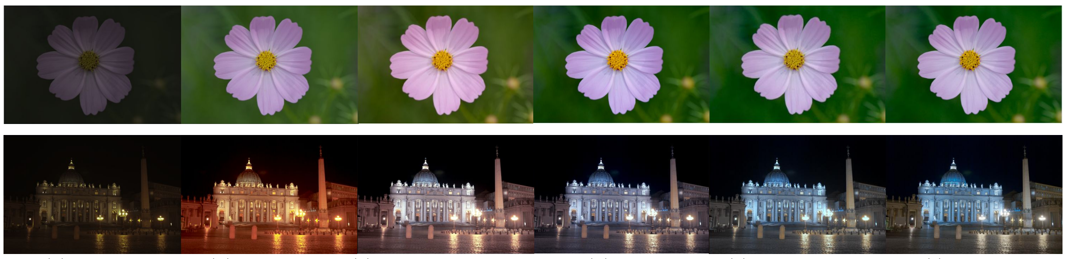


Figure 4.9: Qualitative style comparison with different types of the ISP.

Table 4.3: Quantitative style comparison with different types of the ISP.

Type	PSNR	SSIM
Fixed ISP	19.98	0.826
Image-adaptive ISP	22.72	0.866
CRISP	29.61	0.920
Upper-bound	34.95	0.950

of the user behavior, each blue dot in Fig. 4.8 represents a distinct user behavior. CRISP w/KMC outperforms the PSNR of 3DLUT with seven inferences, whereas CRISP w/GSA outperforms it with five inferences with $s_{init} = (3, 3, 3)$, $t = 3$, and $K = 4$. Both models require fewer FLOPs than a single inference of 3DLUT. This efficiency comes from the fact that CRISP does not employ any convolution or fully connected layers. This leads to less than 100 FLOPs for individual pixels, which is much less than that of trilinear interpolation in 3DLUT and SA3DLUT. CRISP w/GSA performs lower than that CRISP w/KMC when the number of inferences is small, but outperforms significantly as the number of inferences increases.

4.4.6 Analysis

CRISP requires multiple inferences to generate multiple styles for better image quality. We analyze the image quality improvement and the controllability of CRISP.

Style-adaptation. To evaluate the effectiveness of the style-adaptation, we set the following types of models which share the ISP functions described in Section 4.3.3:

- **Fixed ISP.** Without using the encoder and the style decoder, the ISP parameters ϕ are directly optimized in training and fixed during testing.
- **Image-adaptive ISP.** We modify the encoder to use only LQ images as the input for training and testing. The input LQ images adapt the ISP parameters.
- **CRISP.** We use the training encoder in testing to present the effectiveness of the style-adaptation. The only difference from the image-adaptive ISP is that CRISP takes HQ images as the input of the encoder.
- **Upper-bound.** We directly optimize the ISP parameters for *each test image* to maximize the enhancing performance of the style-adaptive ISP.

Fig. 4.9 and Table 4.3 show the qualitative and quantitative results of the test models. The upper bound achieves outstanding performance on all reference-based image quality measures and qualitative results. This indicates that our style-adaptive ISP model has huge potential for image enhancement while consisting of only 19 parameters and performing the global adjustment. By contrast, the fixed ISP performs a 15 dB lower in PSNR than the upper-bound with the wrong white balance under the colored illumination (see Fig. 4.9(b)). The poor performances are mainly because the fixed ISP produces a single style for all test images. The image-adaptive ISP improves the expression for the illumination and some objects (*e.g.*, the outer wall in Fig. 4.9(c)) by the styles that consider the image contents. However, these outputs are far from the styles of the reference images, resulting in a significant low PSNR. The results of CRISP are qualitatively (Fig. 4.9(d)) and quantitatively (Table 4.3) closer to the upper-bound than others. The performance improvement indicates that a style should consider the target HQ image given that there exist various versions of HQ images for an LQ image. CRISP uses HQ images as input of the encoder for

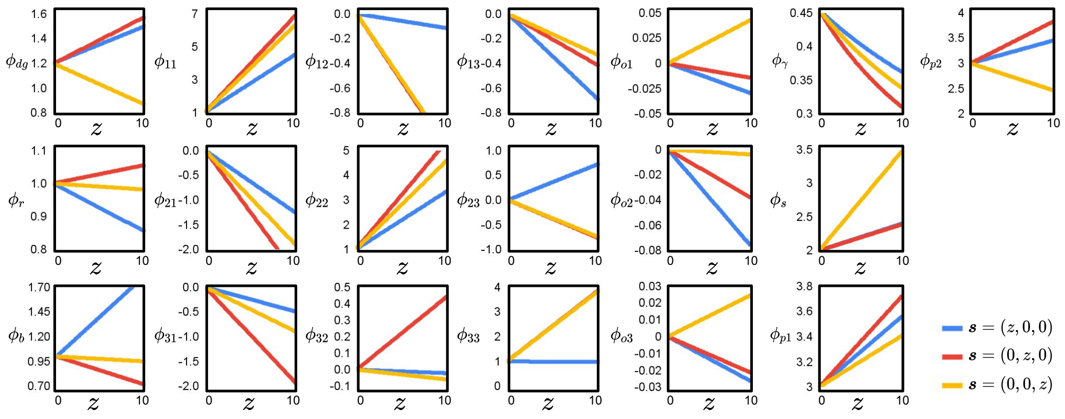
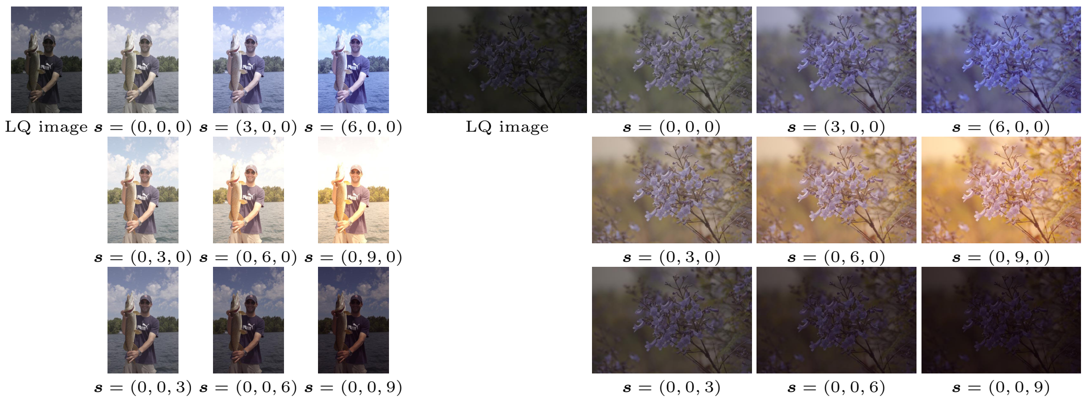


Figure 4.10: Graphs of the style-adaptive ISP parameters ($\boldsymbol{\phi}$) in CRISP. Each graph visualizes each ISP parameter *vs.* the style vector. Each colored line changes the style vector in different dimensions denoted by z and exhibits distinct and insensitive trends in ISP parameters.



LQ image
$\boldsymbol{s} = (0, 0, 0)$
$\boldsymbol{s} = (3, 0, 0)$
$\boldsymbol{s} = (6, 0, 0)$
LQ image
$\boldsymbol{s} = (0, 0, 0)$
$\boldsymbol{s} = (3, 0, 0)$
$\boldsymbol{s} = (6, 0, 0)$
$\boldsymbol{s} = (0, 3, 0)$
$\boldsymbol{s} = (0, 6, 0)$
$\boldsymbol{s} = (0, 9, 0)$
$\boldsymbol{s} = (0, 3, 0)$
$\boldsymbol{s} = (0, 6, 0)$
$\boldsymbol{s} = (0, 9, 0)$
$\boldsymbol{s} = (0, 0, 3)$
$\boldsymbol{s} = (0, 0, 6)$
$\boldsymbol{s} = (0, 0, 9)$
$\boldsymbol{s} = (0, 0, 3)$
$\boldsymbol{s} = (0, 0, 6)$
$\boldsymbol{s} = (0, 0, 9)$

Figure 4.11: Examples for consistent style generation. CRISP generates consistent image styles over LQ images for the same style vectors.

analysis, but in practice, the predetermined style vectors and user interaction allow to obtain the desired results without HQ images.

ISP parameters. Although CRISP automatically encodes the image styles (or the ISP parameters) into the style vectors, each dimension of the style vector has a reasonable trend to change image expressions. Fig. 4.10 visualizes the values of ISP parameters for style vectors. Each colored line presents an *insensitive and distinct trend*, where each color indicates a different dimension in style vectors. The visualized

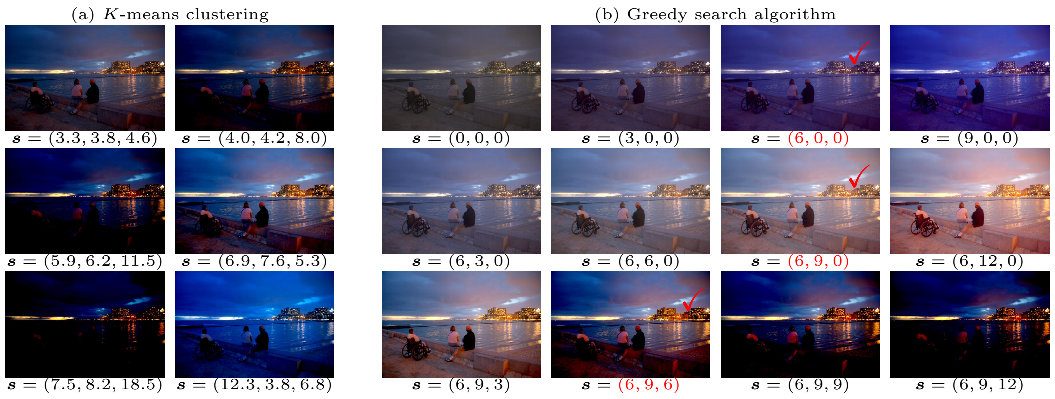


Figure 4.12: Example scenarios for style adjustments. (a) CRISP can generate images with the style vectors ($\boldsymbol{s}$) predetermined as cluster centers in k-means clustering. This example visualizes the enhanced test images using 6 cluster centers of style vectors on the training dataset. (b) CRISP can also generate images with user interaction to adjust style vectors ($\boldsymbol{s}$). In this example, the greedy search algorithm indicates that users increase the value of style vectors to obtain the desired results in each dimension from the first (first row) to the last (third row). The check mark (✓) denotes the most preferred image in each row.

range of z (0$\sim$10) is appropriate to present the trends, because most style vectors from the training data are in this range.

Generalization test. CRISP determines the output image styles *independently* to the input LQ image. Fig. 4.11 illustrates the results from the same style vectors for two different LQ images. Although the LQ images have different contents and pixel values, the output styles are similar for the same style vectors. The first dimension emphasizes blue colors, and the second dimension highlights the yellow and red colors while brightening images. The last dimension darkens images while increasing contrast. The image styles that change predictably according to style vectors ease to obtain desired results.

Control examples. CRISP can have various scenarios for style adjustments, which is how to select the style vector ($\boldsymbol{s}$) for the desired results. Fig. 4.12 demonstrates the example of the k-means clustering and the greedy search algorithm (or

Table 4.4: Quantitative style comparison with CRISP trained by unpaired images.

Type	PSNR	SSIM
CRISP	29.61	0.920
CRISP w/UnpairedImage	29.74	0.921

user interaction). In this example, we do not use the reference image to measure image quality and assume that users can choose any images they prefer in each stage. In Fig. 4.12(a), the k-means clustering provides 6 cluster centers of style vectors on the training dataset, and we visualize the results from the cluster centers. All results have different styles, which some users may prefer. We can set the higher number of clusters to generate more styles. In Fig. 4.12(b), we visualize the style adjustment with the user interaction on the basis of a greedy search algorithm. A user adjusts the value of each dimension of the style vector with a step size of 3 until finding the preferred image in a sequential manner. Once the preferred value of the current dimension is determined, the user fixes it and then adjusts the values of the next dimension of the style vector. The user repeats this adjustment process until obtaining a satisfying result. In practice, the step size and the number of dimension switching depend on user behaviors. Algorithm 1 approximates the user interaction with hyper-parameters.

Network training with unpaired images. Unpaired images for style encoder can alleviate overfitting to the training LQ-HQ image pairs. However, CRISP can not use a random image as a style image due to the style mismatch between the random and HQ images. To this end, CRISP w/UnpairedImage flips and rotates the input HQ images of the style encoder. The augmented HQ images share the styles of the originals but have spatially different contents. Table 4.4 presents that CRISP w/UnpairedImage slightly outperforms our original model (CRISP).

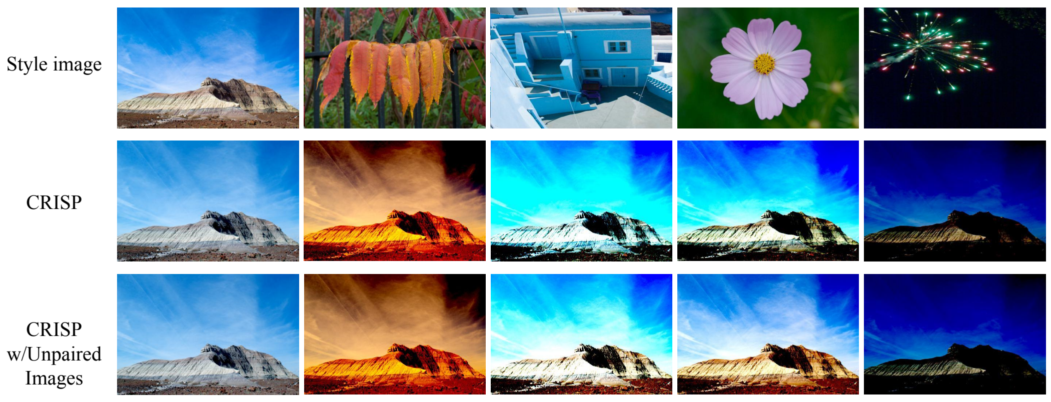


Figure 4.13: Style generation results with style images.

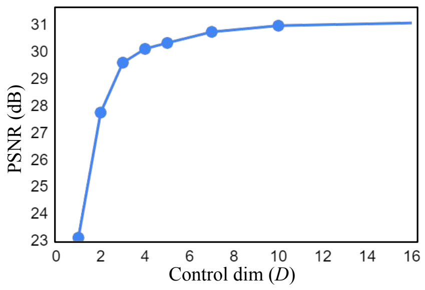


Figure 4.14: Ablation study on the number of control dimensions (D).

Style generation with style images. CRISP generates various image styles by directly adjusting the values of the style vector. Here, we present a conventional approach that adjusts output styles similar to a style image. We simply use the style encoder at test time and use the style images as the input of the style encoder after image scaling. Figure 4.13 presents that CRISP and CRISP w/UnpariedImage successfully generate styles similar to the style images.

Control dimension. The style diversity of CRISP is highly dependent on the dimension of the style vector (or the latent code in the autoencoder). Table 4.14 presents an ablation study to the number of control dimensions (D) of $\boldsymbol{s}$. Although a single control dimension (D=1) improves only marginal PSNR than image-adaptive ISP (23.12 dB *vs.* 22.72 dB), the models with high numbers of D achieve better

Table 4.5: Ablation study on ISP parameter initialization with additive white Gaussian noise.

σ	0	0.1	1	2	5
PSNR (dB)	29.62	29.45	28.22	24.20	6.58

PSNR, where $D = 3$ can be Pareto efficiency. Interestingly, the model with a higher D (*e.g.*, $D = 64$) than the number of ISP parameters (19) still performs lower PSNR than the upper-bound (31.47 dB *vs.* 34.95 dB). This performance gap indicates room for improvement in our style-autoencoder to enhance the representation.

Parameter initialization. CRISP is robust to the initial ISP parameters, although the style decoder learns the residual of the initial parameters. Table 4.5 presents PSNR of CRISP with randomly generated initial parameters, where σ indicates the standard deviation of an additive white Gaussian noise. Although ϕ_γ works in the range of $0 < \phi_\gamma \leq 1$ to visualize the dark area better, CRISP with the noise with $\sigma = 2$ outperforms CSRNet (24.20 dB *vs.* 23.69 dB). CRISP does not explicitly limit the values of the parameter $\boldsymbol{\phi}$, whereas the style vector generates the values via the style decoder.

4.5 Conclusion

We present a controllable ISP, called CRISP, for image enhancement. CRISP learns the representations of image styles through a lightweight autoencoder and adaptively changes ISP parameters on the basis of the output style. Experiments demonstrate that CRISP achieves better image quality and computational efficiency than SotA methods, and even outperforms the human experts in MOS. We also provide various analyses of the effectiveness and controllability of CRISP.

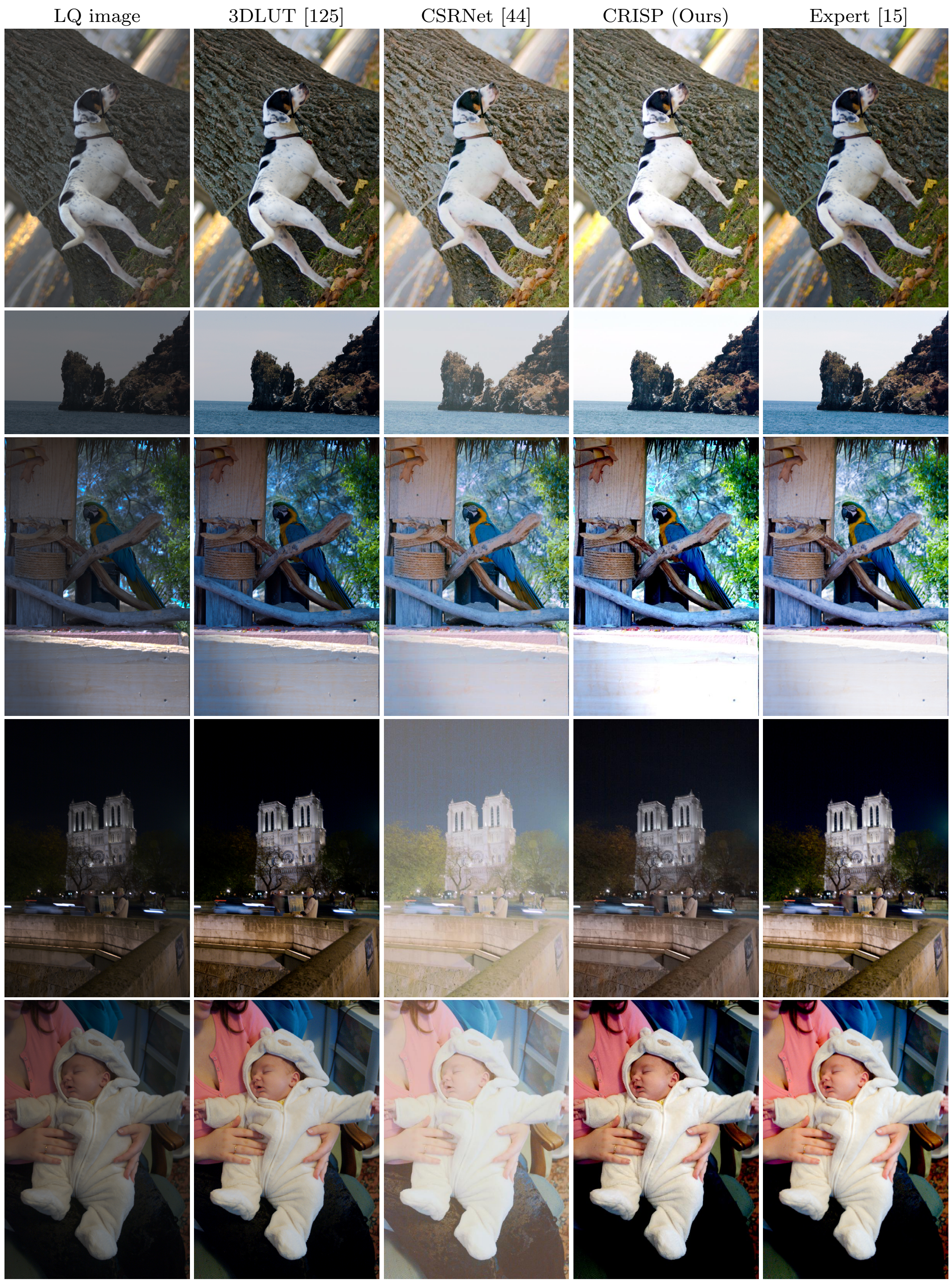


Figure 4.15: Example images for user study of single-style MOS.

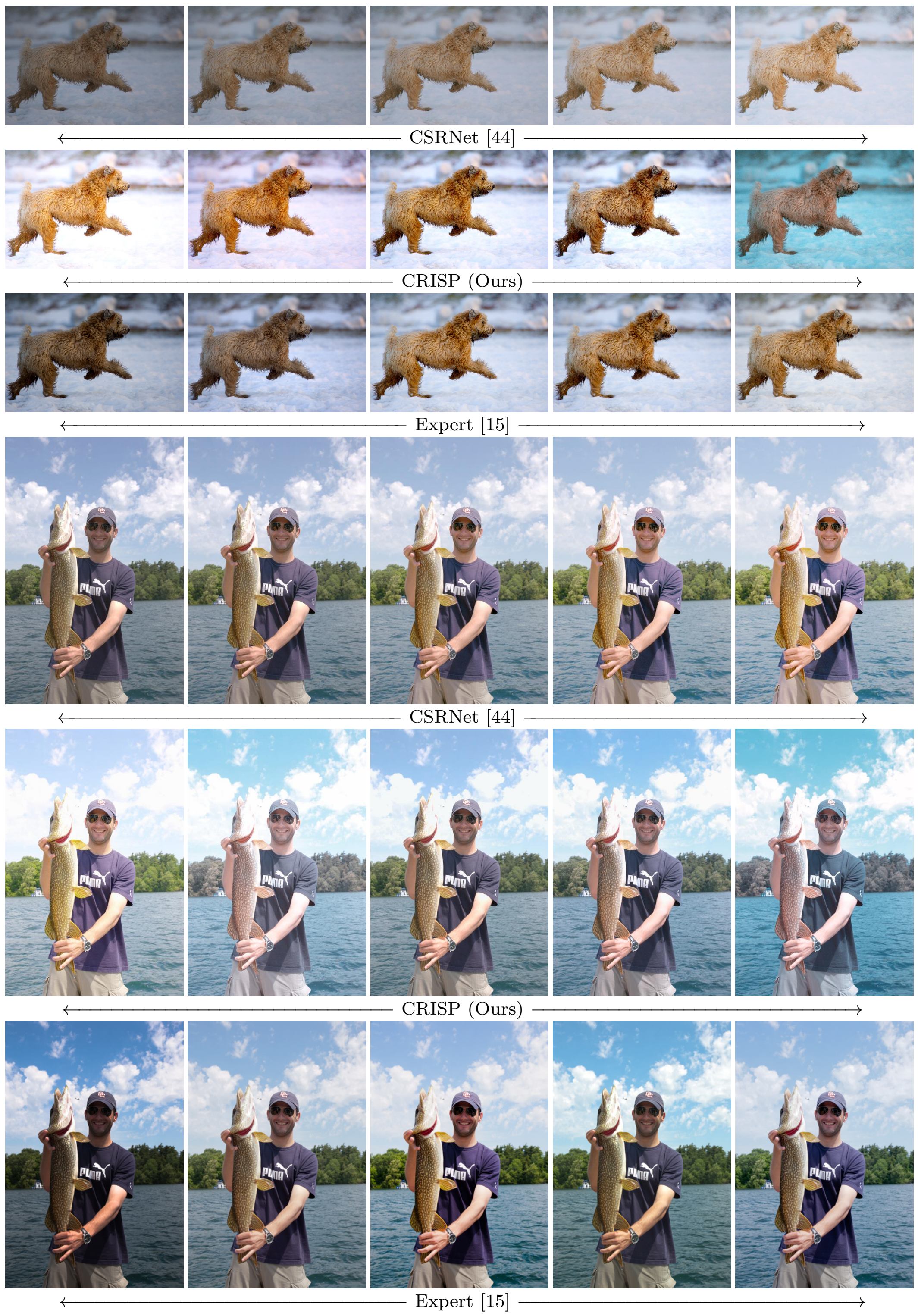


Figure 4.16: Example images for user study of multiple-style MOS.

Chapter 5

Conclusion

This dissertation presents Deep Neural Network (DNN)-based algorithms that facilitate adaptive Image Signal Processors (ISPs) by data synthesis for camera image denoising, neural architecture search for controllable image restoration, and ISP parameter estimation for controllable image enhancement. Each proposed algorithm tackles current challenges associated with low-level computer vision, taking one step further to enjoying daily life through degradation-free and high-quality photographs captured by smartphone/mobile cameras.

Chapter 2 addresses the challenge of camera image denoising, which is hard to obtain noisy-clean image pairs for denoiser training. The proposed method, named NERDS, mitigates this challenge by generating pseudo-training data from noisy images. The difference of pixel variances through downscaling a noisy raw-sensor image can measure image quality degraded by noise. Based on this finding, NERDS estimates noise parameters for Poisson-Gaussian distribution via an optimization problem and adopts a gradient-descent-based optimization through a novel reparametrization trick. Moreover, NERDS estimates the RAW2RGB conversion for a given raw-

RGB noisy image pair to generate training data in RGB space. NERDS regards the downscaled raw image as a pseudo-clean raw image and augments pseudo-training data for image contents, noise levels, and image styles. Experiments present that NERDS estimates accurate noise and image styles for data synthesis and enables accurate denoiser training for camera noise benchmarks and real noisy images.

Cahpter 3 covers improving model efficiency and output image quality for controllable image restoration. While DNN-based approaches improve the accuracy of many image restoration tasks, they require many computations and parameters, which are obstacles for resource-limited devices. Moreover, ISPs perform composite restoration tasks in which users determine the task of interest for each image with unknown degradations. The proposed method, named TASNet, has an efficient CNN architecture by sharing early layers across tasks and adopting the remaining layers for each task. When modulating restoration tasks for an image, TASNet reuses the features from task-agnostic layers to reduce computations. TASNet also introduces a data sampling approach that prevents artifact generation for modulating restoration levels. Experiments present that TASNet significantly improves the model efficiency.

Chapter 4 introduces the challenges for image enhancement: different user preferences for image styles and model efficiency for style generation. The proposed method, named CRISP, alleviates these challenges by learning style representation to improve output image style diversity and adapting ISP parameters to each style. An encoder-decoder framework enables the representation of high-quality image styles in low dimensions. CRISP uses plug-and-play ISP functions for style generation, which have a few parameters and computations. Users can find the preferred style by selecting styles in the representation through K-means clustering or user interaction. Experiments show the effectiveness of CRISP by comparing single/multiple style generation methods in various qualitative and quantitative measures.

In conclusion, this dissertation has introduced three adaptive ISP algorithms that have made substantial advancements in the capabilities of smartphone/mobile cameras. However, there is still a long way to go in order to fully achieve or surpass human-level visual perception. The proposed algorithms tackle three practical applications, but there are many advanced challenges faced by smartphone users in adverse environments such as low-light conditions, underwater scenes, camera shake, fast-moving objects, and close/far objects, which result in raw images with complex degradations that must be effectively addressed. The restoration and enhancement of videos and spatially-varying adaptation also require specialized improvement with domain expertise. As the number of applications expands, an integrated model to control the increasing number of factors is a potential research area. Finally, ISP algorithms for futuristic hardware, including space telescopes, metalenses, 360-degree cameras, and AR/VR devices, hold promise for future work.

Bibliography

[1] Abdelrahman Abdelhamed, Marcus A Brubaker, and Michael S Brown. Noise Flow: Noise Modeling with Conditional Normalizing Flows. In *ICCV*, 2019.

[2] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *CVPR*, 2018.

[3] Sergey Abramov, Victoriya Zabrodina, Vladimir Lukin, Benoit Vozel, Kacem Chehdi, and Jaakko Astola. Improved method for blind estimation of the variance of mixed noise using weighted lms line fitting algorithm. In *ISCAS*, 2010.

[4] Raanan Fattal Adam Kaufman. Deblurring using analysis-synthesis networks pair. In *CVPR*, 2020.

[5] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPRW*, 2017.

[6] Namhyuk Ahn, Byungkon Kang, and Kyung-Ah Sohn. Fast, accurate, and lightweight super-resolution with cascading residual network. In *ECCV*, 2018.

[7] Alan C. Bovik Anish Mittal, Rajiv Soundararajan. Making a completely blind image quality analyzer. *SPL*, 20(3):209–212, 2013.

[8] F. J. Anscombe. The Transformation of Poisson, Binomial, and Negative-Binomial Data. *Biometrika*, 35(3-4):246–254, 1948.

[9] Joshua Batson and Loic Royer. Noise2Self: Blind denoising by self-supervision. In *ICML*, 2019.

[10] Sefi Bell-Kligler, Assaf Shocher, and Michal Irani. Blind super-resolution kernel estimation using an internal-gan. In *NeurIPS*, 2019.

[11] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv:1308.3432, 2013.

[12] Tim Brooks, Ben Mildenhall, Tianfan Xue, Jiawen Chen, Dillon Sharlet, and Jonathan T Barron. Unprocessing images for learned raw denoising. In *CVPR*, 2019.

[13] Benoit Brummer and Christophe De Vleeschouwer. Natural image noise dataset. In *CVPRW*, 2019.

[14] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A review of image denoising algorithms, with a new one. *MMS*, 4(2):490–530, 2005.

[15] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *CVPR*, 2011.

[16] Jaeseok Byun, Sungmin Cha, and Taesup Moon. Fbi-denoiser: Fast blind image denoiser for poisson-gaussian noise. In *CVPR*, 2021.

[17] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *ICLR*, 2019.

[18] Juan C. Caicedo, Ashish Kapoor, and Sing Bing Kang. Collaborative personalization of image enhancement. In *CVPR*, 2011.

[19] Rich Caruana. Multitask learning. *ML*, 28:41–75, 1997.

[20] Sungmin Cha, Taeeon Park, and Taesup Moon. GAN2GAN: Generative noise learning for blind image denoising with single noisy images. In *ICLR*, 2021.

[21] Ke-Chi Chang, Ren Wang, Hung-Jin Lin, Yu-Lun Liu, Chia-Ping Chen, Yu-Lin Chang, and Hwann-Tzong Chen. Learning camera-aware noise models. In *ECCV*, 2020.

[22] Guangyong Chen, Fengyuan Zhu, and Pheng Ann Heng. An efficient statistical method for image noise level estimation. In *ICCV*, 2015.

[23] Jingwen Chen, Jiawei Chen, Hongyang Chao, and Ming Yang. Image blind denoising with generative adversarial network based noise modeling. In *CVPR*, 2018.

[24] Yu-Sheng Chen, Yu-Ching Wang, Man-Hsin Kao, and Yung-Yu Chuang. Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans. In *CVPR*, 2018.

[25] Shen Cheng, Yuzhi Wang, Haibin Huang, Donghao Liu, Haoqiang Fan, and Shuaicheng Liu. NBNet: Noise basis learning for image denoising with subspace projection. In *CVPR*, 2021.

[26] Xiangxiang Chu, Bo Zhang, Hailong Ma, Ruijun Xu, Jixiang Li, and Qingyuan Li. Fast, accurate and lightweight super-resolution with neural architecture search. In *ICPR*, 2020.

[27] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *NIPS*, 2016.

[28] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *TIP*, 16(8):2080–2095, 2007.

[29] Chao Dong, Yubin Deng, Chen Chang Loy, and Xiaoou Tang. Compression artifacts reduction by a deep convolutional network. In *ICCV*, 2015.

[30] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *TPAMI*, 38(2):295–307, 2016.

[31] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, 2016.

[32] David L Donoho and Iain M Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *JASA*, 90(432):1200–1224, 1995.

[33] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *TIP*, 15(12):3736–3745, 2006.

[34] Alessandro Foi, Mejdi Trimeche, Vladimir Katkovnik, and Karen Egiazarian. Practical Poissonian-Gaussian noise modeling and fitting for single-image raw-data. *TIP*, 17(10):1737–1754, 2008.

[35] Yonggan Fu, Wuyang Chen, Haotao Wang, Haoran Li, Yingyan Lin, and Zhangyang Wang. Autogan-distiller: Searching to compress generative adversarial networks. In *ICML*, 2020.

[36] Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep bilateral learning for real-time image enhancement. *TOG*, 36(4), 2017.

[37] Dong Gong, Jie Yang, Lingqiao Liu, Yanning Zhang, Ian Reid, Chunhua Shen, Anton van den Hengel, and Qinfeng Shi. From motion blur to motion flow:

A deep learning solution for removing heterogeneous motion blur. In *CVPR*, 2017.

[38] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[39] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In *CVPR*, 2014.

[40] Chunle Guo Guo, Chongyi Li, Jichang Guo, Chen Change Loy, Junhui Hou, Sam Kwong, and Runmin Cong. Zero-reference deep curve estimation for low-light image enhancement. In *CVPR*, 2020.

[41] Shi Guo, Zifei Yan, Kai Zhang, Wangmeng Zuo, and Lei Zhang. Toward convolutional blind denoising of real photographs. In *CVPR*, 2019.

[42] Jingwen He, Chao Dong, and Yu Qiao. Modulating image restoration with continual levels via adaptive feature modification layers. In *CVPR*, 2019.

[43] Jingwen He, Chao Dong, and Yu Qiao. Interactive multi-dimension modulation with dynamic controllable residual learning for image restoration. In *ECCV*, 2020.

[44] Jingwen He, Yihao Liu, Yu Qiao, and Chao Dong. Conditional sequential modulation for efficient global image retouching. In *ECCV*, 2020.

[45] Zhiwei Hong, Xiaocheng Fan, Tao Jiang, and Jianxing Feng. End-to-end unpaired image denoising with conditional adversarial networks. In *AAAI*, 2020.

[46] Eugene Hsu, Tom Mertens, Sylvain Paris, Shai Avidan, and Fredo Durand. Light mixture estimation for spatially varying white balance. *TOG*, 27(3), 2008.

[47] Xiaowan Hu, Ruijun Ma, Zhihong Liu, Yuanhao Cai, Xiaole Zhao, Yulun Zhang, and Haoqian Wang. Pseudo 3D auto-correlation network for real image denoising. In *CVPR*, 2021.

[48] Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. Exposure: A white-box photo post-processing framework. *TOG*, 37(2), 2018.

[49] Shih-Chia Huang, Fan-Chieh Cheng, and Yi-Sheng Chiu. Efficient contrast enhancement using adaptive gamma correction with weighting distribution. *TIP*, 22(3):1032–1041, 2013.

[50] Tao Huang, Songjiang Li, Xu Jia, Huchuan Lu, and Jianzhuang Liu. Neighbor2Neighbor: Self-supervised denoising from single noisy images. In *CVPR*, 2021.

[51] Sung Ju Hwang, Ashish Kapoor, and Sing Bing Kang. Context-based automatic local image enhancement. In *ECCV*, 2012.

[52] Andrey Ignatov, Nikolay Kobyshev, Radu Timofte, Kenneth Vanhoey, and Luc Van Gool. Dslr-quality photos on mobile devices with deep convolutional networks. In *ICCV*, 2017.

[53] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[54] Geonwoon Jang, Wooseok Lee, Sanghyun Son, and Kyoung Mu Lee. C2N: Practical generative noise modeling for real-world denoising. In *ICCV*, 2021.

[55] Neel Joshi, Wojciech Matusik, Edward H. Adelson, and David J. Kriegman. Personal photo enhancement using example images. *TOG*, 29(2), 2010.

[56] Sing Bing Kang, Ashish Kapoor, and Dani Lischinski. Personalization of image enhancement. In *CVPR*, 2010.

[57] Heewon Kim, Myungsub Choi, Bee Lim, and Kyoung Mu Lee. Task-aware image downscaling. In *ECCV*, 2018.

[58] Hanul Kim, Su-Min Choi, Chang-Su Kim, and Yeong Jun Koh. Representative color transform for image enhancement. In *ICCV*, 2021.

[59] Heewon Kim, Seokil Hong, Bohyung Han, Heesoo Myeong, and Kyoung Mu Lee. Fine-grained neural architecture search for image super-resolution. *JVCI*, 89:103654, 2022.

[60] Han-Ul Kim, Young Jun Koh, and Chang-Su Kim. Global and local enhancement network for paired and unpaired image enhancement. In *ECCV*, 2020.

[61] Han-Ul Kim, Young Jun Koh, and Chang-Su Kim. Pienet: Personalized image enhancement. In *ECCV*, 2020.

[62] Jiwon Kim, Jungkwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016.

[63] Yoonsik Kim, Jae Woong Soh, Gu Yong Park, and Nam Ik Cho. Transfer learning from synthetic to real-noise denoising with adaptive instance normalization. In *CVPR*, 2020.

[64] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[65] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[66] Satoshi Kosugi and Toshihiko Yamasaki. Unpaired image enhancement featuring reinforcement-learning-controlled image editing software. In *AAAI*, 2020.

[67] Shayan Kousha, Ali Maleky, Michael S. Brown, and Marcus A. Brubaker. Modeling srgb camera noise with normalizing flows. In *CVPR*, 2022.

[68] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2Void-learning denoising from single noisy images. In *CVPR*, 2019.

[69] Samuli Laine, Tero Karras, Jaakko Lehtinen, and Timo Aila. High-quality self-supervised deep image denoising. In *NeurIPS*, 2019.

[70] Edwin H. Land and John J. McCann. Lightness and retinex theory. *OSA*, 61(1):1–11, 1971.

[71] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[72] Wooseok Lee, Sanghyun Son, and Kyoung Mu Lee. Ap-bsn: Self-supervised denoising for real-world images via asymmetric pd and blind-spot network. In *CVPR*, 2022.

[73] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2Noise: Learning image restoration without clean data. In *ICML*, 2018.

[74] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. Dhp: Differentiable meta pruning via hypernetworks. In *ECCV*, 2020.

[75] Jie Liang, Hui Zeng, and Lei Zhang. High-resolution photorealistic image translation in real-time: A laplacian pyramid translation network. In *CVPR*, 2021.

[76] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPRW*, 2017.

[77] Dani Lischinski, Zeev Farbman, Matt Uyttendaele, and Richard Szeliski. Interactive local adjustment of tonal values. *TOG*, 25(3):646–653, 2006.

[78] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *ICLR*, 2019.

[79] Ming Liu, Zhilu Zhang, Liya Hou, Wangmeng Zuo, and Lei Zhang. Deep adaptive inference networks for single image super-resolution. In *ECCVW*, 2020.

[80] Xinhao Liu, Masayuki Tanaka, and Masatoshi Okutomi. Single-image noise level estimation for blind denoising. *TIP*, 22(12):5226–5237, 2013.

[81] Xinhao Liu, Masayuki Tanaka, and Masatoshi Okutomi. Practical signal-dependent noise parameter estimation from a single noisy image. *TIP*, 23(10):4361–4371, 2014.

[82] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017.

[83] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *ICCV*, 2009.

[84] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *NIPS*, 2016.

[85] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001.

[86] Anish Mittal, Anush K. Moorthy, and Alan C. Bovik. Blind/referenceless image spatial quality evaluator. In *ASILOMAR*, 2011.

[87] Nick Moran, Dan Schmidt, Yu Zhong, and Patrick Coady. Noisier2Noise: Learning to denoise from unpaired noisy data. In *CVPR*, 2020.

[88] Sean Moran, Pierre Marza, Steven McDonagh, Sarah Parisot, and Gregory Slabaugh. Deeplpf: Deep local parametric filters for image enhancement. In *CVPR*, 2020.

[89] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *CVPRW*, 2019.

[90] Tongyao Pang, Huan Zheng, Yuhui Quan, and Hui Ji. Recorrupted-to-Recorrupted: Unsupervised deep learning for image denoising. In *CVPR*, 2021.

[91] Jongchan Park, Joon-Young Lee, Donggeun Yoo, and In So Kweon. Distort-and-recover: Color enhancement using deep reinforcement learning. In *CVPR*, 2018.

[92] Stephen M. Pizer, E. Philip Amburn, John D. Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bartter Haar Romeny, John B. Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variationse. *CVGIP*, 39(3):355–368, 1987.

[93] Tobias Plotz and Stefan Roth. Benchmarking denoising algorithms with real photographs. In *CVPR*, 2017.

[94] Stanislav Pyatykh, Jürgen Hesser, and Lei Zheng. Image noise level estimation by principal component analysis. *TIP*, 22(2):687–699, 2013.

[95] Rajeev Ramanath, Wesley E. Snyder, Youngjun Yoo, and Mark S. Drew. Color image processing pipeline. *SPM*, 22(1):34–43, 2005.

[96] Assaf Shocher, Nadav Cohen, and Michal Irani. ”zero-shot” super-resolution using deep internal learning. In *CVPR*, 2018.

[97] Alon Shoshan, Roey Mechrez, and Lihi Zelnik-Manor. Dynamic-net: Tuning the objective without re-training for synthesis tasks. In *ICCV*, 2019.

[98] Dehua Song, Chang Xu, Xu Jia, Yiyi Chen, Chunjing Xu, and Yunhe Wang. Efficient residual dense block search for image super-resolution. In *AAAI*, 2020.

[99] Yuda Song, Hui Qian, and Xin Du. Starenhancer: Learning real-time and style-aware image enhancement. In *ICCV*, 2021.

[100] Masanori Suganuma, Xing Liu, and Takayuki Okatani. Attention-based adaptive selection of operations for image restoration in the presence of unknown combined distortions. In *CVPR*, 2019.

[101] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. MemNet: A persistent memory network for image restoration. In *ICCV*, 2017.

[102] Mikhail L Uss, Benoit Vozel, Vladimir V Lukin, and Kacem Chehdi. Image informative maps for component-wise estimating parameters of signal-dependent noise. *JEI*, 22:013019, 2013.

[103] Haotao Wang, Shupeng Gui, Haichuan Yang, Ji Liu, and Zhangyang Wang1. Gan slimming: All-in-one gan compression by a unified optimization framework. In *ECCV*, 2020.

[104] Ruixing Wang, Qing Zhang, Chi-Wing Fu, Xiaoyong Shen, Wei-Shi Zheng, and Jiaya Jia. Underexposed photo enhancement using deep illumination estimation. In *CVPR*, 2019.

[105] Tao Wang, Yong Li, Jingyang Peng, Yipeng Ma, Xian Wang, Fenglong Song, and Youliang Yan. Real-time image enhancer via learnable spatial-aware 3d lookup tables. In *ICCV*, 2021.

[106] Wei Wang, Ruiming Guo, Yapeng Tian3, and Wenming Yang. Cfsnet: Toward a controllable feature space for image restoration. In *ICCV*, 2019.

[107] Xintao Wang, Ke Yu, Chao Dong, Xiaoou Tang, and Chen Change Loy. Deep network interpolation for continuous imagery effect transition. In *CVPR*, 2019.

[108] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *CVPR*, 2019.

[109] Xiaohe Wu, Ming Liu, Yue Cao, Dongwei Ren, and Wangmeng Zuo. Unpaired learning of deep image denoising. In *ECCV*, 2020.

[110] Bin Xiao, Han Tang, Yanjun Jiang, Weisheng Li, and Guoyin Wang. Brightness and contrast controllable image enhancement based on histogram specification. *Neurocomputing*, 275:2798–2809, 2018.

[111] Yaochen Xie, Zhengyang Wang, and Shuiwang Ji. Noise2Same: Optimizing a self-supervised bound for image denoising. In *NeurIPS*, 2020.

[112] Jingwei Xin, Nannan Wang, Xinrui JiangJie Li, Heng Huang, and Xinbo Gao. Binarized neural network for single image super resolution. In *ECCV*, 2020.

[113] Jun Xu, Yuan Huang, Ming-Ming Cheng, Li Liu, Fan Zhu, Zhou Xu, and Ling Shao. Noisy-As-Clean: Learning self-supervised denoising from corrupted image. *TIP*, 29:9316–9329, 2020.

[114] Xiangyu Xu, Jinshan Pan, Yujin Zhang, and Ming-Hsuan Yang. Motion blur kernel estimation via deep learning. *TIP*, 27(1):194–205, 2018.

[115] Jianzhou Yan, Stephen Lin, Sing Bing Kang, and Xiaoou Tang. A learning-to-rank approach for image color enhancement. In *CVPR*, 2014.

[116] Zhicheng Yan, Hao Zhang, Baoyuan Wang, Sylvain Paris, and Yizhou Yu. Automatic photo adjustment using deep neural networks. *TOG*, 35(2), 2017.

[117] Ke Yu, Chao Dong, Liang Lin, and Chen Change Loy. Crafting a toolchain for image restoration by deep reinforcement learning. In *CVPR*, 2018.

[118] Ke Yu, Xintao Wang, Chao Dong, Xiaoou Tang, and Chen Change Loy. Path-restore: Learning network path selection for image restoration. *TPAMI*, 44(10):7078–7092, 2022.

[119] Songhyun Yu, Bumjun Park, and Jechang Jeong. Deep iterative down-up CNN for image denoising. In *CVPRW*, 2019.

[120] Yuan Yuan, Wei Su, and Dandan Ma. Efficient dynamic scene deblurring using spatially variant deconvolution network with optical flow guided training. In *CVPR*, 2020.

[121] Zongsheng Yue, Hongwei Yong, Qian Zhao, Lei Zhang, and Deyu Meng. Variational denoising network: Toward blind noise modeling and removal. In *NeurIPS*, 2019.

[122] Zongsheng Yue, Qian Zhao, Lei Zhang, and Deyu Meng. Dual adversarial network: Toward real-world noise removal and noise generation. In *ECCV*, 2020.

[123] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, Ming-Hsuan Yang, and Ling Shao. Cycleisp: Real image restoration via improved data synthesis. In *CVPR*, 2020.

[124] Sun-Yuan Kung Zejiang Hou. Efficient image super resolution via channel discriminative deep neural network pruning. In *ICASSP*, 2020.

[125] Hui Zeng, Jianrui Cai, Lida Li, Zisheng Cao, and Lei Zhang. Learning image-adaptive 3d lookup tables for high performance photo enhancement in real-time. *TPAMI*, 44(4):2058–2073, 2020.

[126] Kai Zhang, Yawei Li, Jingyun Liang, Jiezhang Cao, Yulun Zhang, Hao Tang, Radu Timofte, and Luc Van Gool. Practical blind denoising via swin-conv-unet and data synthesis. arXiv:2203.13278, 2022.

[127] 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.

[128] 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.

[129] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for CNN based image denoising. *TIP*, 27(9):4608–4622, 2018.

[130] Yi Zhang, Hongwei Qin, Xiaogang Wang, and Hongsheng Li. Rethinking noise synthesis and modeling in raw denoising. In *ICCV*, 2021.

[131] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, 2018.

[132] Yu Zhang and Qiang Yang. A survey on multi-task learning. *TKDE*, 34(12):5586–5609, 2022.

[133] Luxi Zhao, Abdelrahman Abdelhamed, and Michael S. Brown. Learning tone curves for local image enhancement. *IEEE Access*, 10:60099–60113, 2022.

[134] Daniel Zoran and Yair Weiss. From learning models of natural image patches to whole image restoration. In *ICCV*, 2011.

국문초록

현대인에게 카메라는 삶을 기록할 뿐 아니라 소셜 네트워크 서비스와 화상 회의, 개인 방송을 위해 없어서는 안 될 존재가 되었다. 스마트폰 제조사(예: 삼성, 애플)는 스마트폰 시장 점유율에서 카메라의 성능이 중요한 역할을 수행함에 따라, 차세대 스마트폰에 더 많고 더 큰 카메라 랜즈를 사용하고 있다. 디지털 카메라의 이미지 파이프라인에서 이미지 센서는 빛을 원시 이미지라고 하는 디지털 신호로 변환하고 이미지 시그널 프로세서(ISP)는 원시 이미지를 사람이 읽을 수 있는 RGB 이미지로 변환한다. 이러한 ISP는 카메라 하드웨어의 제약과 소프트웨어적 변형으로 인해 손상된 이미지를 복구하는 과정(이미지 복원)과 이미지를 매력적인 스타일로 보정하는 과정(이미지 개선)에 관련된 다양한 영상 처리 과제를 수행한다.

ISP의 도전 과제는 모든 이미지 손상 또는 모든 이미지 스타일에 대한 일반적인 모델을 설계하고 구현하는 것이다. 특히 이미지는 노이즈, 블러, 압축과 같은 다양한 요인으로 인해 이미지의 세부 정보가 손실된다. 모든 손상을 복원하는 일반적인 모델은 충분히 많은 수의 모델 파라미터를 필요하며 각 손상에 대해 최적의 결과도 얻지 못한다. 그뿐만 아니라, 매력적인 이미지 스타일은 주관적이어서, 개인적 경험이나 분위기, 기분과 같은 다양한 요소에 의해 달라진다. 대부분의 최신 인공신경망 모델들은 입력 이미지에 대해 단일 스타일 이미지를 생성하는데, 이는 사용자를 만족시키기에 제한적이다.

전술한 도전 과제를 해결하기 위해 본 학위 논문은 심층 신경망을 사용하여 실용적인 애플리케이션에 대한 적응형 ISP를 제안한다. 구체적으로, 제안하는 세 가지 알

고리즘과 애플리케이션은 각각: (1) 카메라 이미지의 잡음 제거를 위한 적응형 데이터 합성, (2) 조작 가능한 이미지 복원을 위한 적응형 신경망 구조 탐색, (3) 조작 가능한 이미지 개선을 위한 적응형 ISP 매개변수 추정이다. 첫 번째로, 카메라 이미지 잡음 제거에서는 지도 학습을 위해서 잡음이 있는 이미지와 깨끗한 이미지의 쌍을 얻기가 어렵다. 제안 방법은 잡음이 있는 시험 이미지만으로 지도 학습을 가능하게 하는 모조 학습 데이터를 생성하여 일반적인 CNN 기반 잡음 제거기를 시험 이미지에 적합하게 학습시킨다. 두 번째로, 조작 가능한 이미지 복원은 알 수 없는 손상에 대해 미리 정해진 복수의 복원 작업의 결과를 생성하고 사용자가 원하는 결과를 선택하는 새로운 이미지 복원 애플리케이션이다. 제안 방법은 이러한 복수의 이미지 복원 결과를 생성하는데 효율적인 CNN 구조를 자동으로 찾는다. 찾아진 CNN은 앞선 계층을 공유하고, 남은 계층을 과제에 맞게 조정한다. 세 번째로, 제안하는 방법은 주관적인 사용자 선호도를 만족시키기 위해 다수의 고품질 이미지 스타일 생성 방법을 학습한다. 사용자는 학습된 잠재 표현에서 스타일을 선택하고 신경망은 스타일을 ISP 매개변수로 변환한다. 이미지 스타일에 대한 주관적인 평가를 위해 제안 방법은 고품질의 다양한 스타일을 생성한다. 스타일 생성은 플러그 앤드 플레이 ISP를 사용하여서 효율적이다.

제안 방법들은 각각의 컴퓨터 비전 과제에서 유의미한 성능 향상을 얻을 수 있었으며, 면밀한 실험적 분석과 구성 요소별 분석을 통해 유효성을 검증하였다. 또한 각 과제에서 널리 사용되는 벤치마크에서 탁월한 화질 개선 능력과 모델의 효율성을 보였으며, 실제 영상에서도 유의미한 성능을 확인했다.

주요어: 이미지 시그널 프로세서, 이미지 복원, 이미지 개선, 심층 학습, 데이터 합성, 신경망 구조 탐색, ISP 매개변수 추정

학번: 2017-27265