# Multi-Curve Translator for High-Resolution Photorealistic Image Translation

Yuda Song ⓘ, Hui Qian ⓘ, and Xin Du ✉ ⓘ

Zhejiang University, Hangzhou, China
{syd,qianhui,duxin}@zju.edu.cn

**Abstract.** The dominant image-to-image translation methods are based on fully convolutional networks, which extract and translate an image's features and then reconstruct the image. However, they have unacceptable computational costs when working with high-resolution images. To this end, we present the Multi-Curve Translator (MCT), which not only predicts the translated pixels for the corresponding input pixels but also for their neighboring pixels. And if a high-resolution image is downsampled to its low-resolution version, the lost pixels are the remaining pixels' neighboring pixels. So MCT makes it possible to feed the network only the downsampled image to perform the mapping for the full-resolution image, which can dramatically lower the computational cost. Besides, MCT is a plug-in approach that utilizes existing base models and requires only replacing their output layers. Experiments demonstrate that the MCT variants can process 4K images in real-time and achieve comparable or even better performance than the base models on various photorealistic image-to-image translation tasks.

**Keywords:** high-resolution, real-time, image-to-image translation

## 1 Introduction

Image-to-image (I2I) translation aims to translate images from a source domain to a target domain. Many computer vision tasks, such as image denoising [64], dehazing [4], colorization [65], attribute editing [8], and style transfer [11], can be posed as I2I translation problems. Some approaches [23,68,39,21,48] use a universal framework to handle various I2I translation problems. No matter the training scheme and the addressed problem, their network architectures are generally based on fully convolutional networks (FCNs) [40]. However, the computational cost of FCN is proportional to the input image pixels, making high-resolution (HR) images be a considerable obstacle to employing these methods. For example, CycleGAN [68] requires 56.8G multiply-accumulate operations (MACs) to process a $256 \times 256$ image and requires 7.2T MACs when working with a 4K ($3840 \times 2160$) image, which is unacceptable even for high-performance GPUs.

To this end, some researchers design lightweight networks [2,35] or employ model compression [52,32] to save computational cost. However, designing and training a new lightweight FCN is not easy since it involves a trade-off between efficiency and effectiveness. And repeating this procedure for every I2I translation
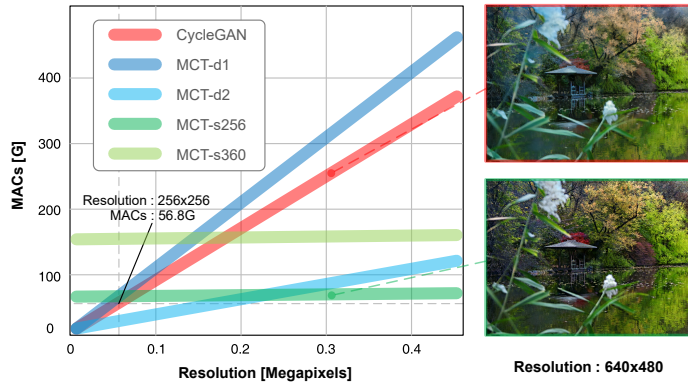
**Fig. 1.** An example comparing CycleGAN and its MCT variant on `autumn2summer`. The only difference between two models is the output layer. `d2` means the input image of the backbone network is downsampled by a factor of 2, and `s256` means the input image of the backbone network is downsampled to $256 \times 256$.

task can be highly time-consuming and power-consuming. Therefore, we prefer to propose a more flexible approach to the problem. We found that some photo-realistic I2I translation methods [28,41,34] apply post-processing techniques that constrain the mapping to be spatially smooth to preserve the image's structure information. So why not just predict a spatially smooth mapping to approximate this translation process in the image space? We can downsample HR images and use the downsampled images to predict the mappings for the original images. In this way, we can feed low-resolution (LR) images to the backbone networks, which exponentially reduces the computational cost. Besides, we can try to reuse the existing FCN architectures.

At this point, the key to the problem lies in designing a mapping that can provide sufficient I2I translation capability but has a much lower computational cost than the backbone network. To address the above challenges, we propose an I2I translator, dubbed **M**ulti-**C**urve **T**ranslator (MCT). Specifically, we take an existing FCN as the backbone network and find that it only predicts the output pixels for their corresponding input pixels. So we increase its last layer's output channels to make each output pixel indicate a set of mapping functions in the form of curves. We quantize these curves as look-up tables (LUTs) [24,37], then given the output pixel (*i.e.*, curves' parameters) responding to the input pixel of the downsampled image, we can derive the output pixels for all pixels in the full-resolution image's corresponding region. Besides reducing the computational cost, MCT has additional advantages. Firstly, an FCN's receptive field is limited, so it may not extract meaningful semantic information when processing HR images. But for MCT, we can adjust the downsampling ratio to change the backbone network's receptive field size dynamically. Secondly, since MCT only requires increasing the output channels of the FCN, it is easy to employ it on another I2I translation task without designing a new network architecture.

We extended some I2I translation models to their MCT variants and found that they have significant advantages in saving computational cost and preserving details. Fig. 1 illustrates the performance comparison between CycleGAN and its MCT variant. Because we can increase the downsampling ratio to reduce the computational cost, the MCT-CycleGAN can always be less computationally intensive than CycleGAN. In practice, the input images of the MCT's backbone network are downsampled to $256 \times 256$ (MCT-s256 in Fig. 1), consistent with the training set's image size to minimize the gap between inference and training. In this case, the gap between CycleGAN and MCT-CycleGAN becomes increasingly large as the input image size grows. Specifically, when processing 4K images, the computational cost of MCT-CycleGAN is only 0.8% of that of CycleGAN, leading to the former being $40 \times$ faster than the latter on GPUs. Finally, MCT enables the input image's high-frequency information to flow easily to the output image, making the trees sharper to improve image realism. While MCT looks appealing, it is to be noted that MCT focuses on photorealistic image-to-image translation and does not work well on more general image-to-image translation tasks, which is our primary future work.

## 2   Methodology

### 2.1   Problem Formulation & Prior Work

Let $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the goal of I2I translation is to learn a mapping $G: \mathcal{X} \to \mathcal{Y}$ such that the distribution $p(G(x))$ is as close as possible to the distribution $p(y)$. Although models for different I2I translation tasks are trained in different manners, they commonly use FCN-based models [23,68,43,48]. We assume that $G$ is an FCN with weight $\theta$, then the translated image $\tilde{y}$ can be formulated as:

$$\tilde{y} = G(x; \theta). \tag{1}$$

However, the FCN's computational cost is proportional to the image pixels [18], and our goal is to break it. We divide the mapping into two components: the translator $G$ that translates the images from $\mathcal{X}$ to $\mathcal{Y}$ and the encoder $E$ that predicts the translator's parameters. Assuming that the encoder $E$ with weight $\theta$ encodes the parameters of $G$ from the condition $z$, the translation is:

$$\tilde{y} = G(x; E(z; \theta)). \tag{2}$$

There have been several works based on similar ideas. Since conventional image processing methods commonly employ filters [56,15,61], KPN [44] predicts the convolutional filter $G$ with parameter $\theta_i = E(x; \theta)_i$ for each pixel $x_i$ and applies it to its spatial support $\Omega(x_i)$ to obtain $\tilde{y}_i$ for burst image denoising. It can be formulated as:

$$\tilde{y}_i = G(\Omega(x_i); E(x; \theta)_i). \tag{3}$$

However, it still requires to perform the FCN on the HR image, leading to no reduction in its computational cost. Besides, the convolutional filter is still a linear mapping, which has limited translation capability.

Another feasible solution is HyperNetwork [14], which uses a network to generate the weights of another network and is originally designed for neural network compression. Following some HyperNetwork-based works [10,27,47,51] on image processing tasks, we can use a encoder $E$ to predict the weights of a lightweight FCN $G$ on the downsampled image $x\downarrow$, which can be formulated as:

$$\tilde{y} = G(x; E(x\downarrow; \theta)). \tag{4}$$

If $E$ is fed with only fixed-size images $x\downarrow$, the total computational cost of the model grows slowly with the size of the input images [55]. In our experiments, this plain idea works well for photo retouching but not other tasks.

We try to combine the two approaches above to overcome their respective shortcomings. We expect the encoder $E$ to encode the parameter maps $F = E(x\downarrow; \theta)$, in which each cell $F_j$ contains a set of translator's parameters:

$$\tilde{y}_i = G(x_i; F_j). \tag{5}$$

Since the parameter maps $F$ cannot be aligned with $x$, we also need to define the relation between $i$ and $j$. We will detail our MCT in the next subsection.

Similar works to MCT consist of bilateral learning [12,59,67], curve mapping [25,31,46] and 3D LUTs [63,60], as they are all based on slicing operation. In contrast to bilateral learning-based methods, MCT predicts pixel values rather than affine transformations, which allows us to directly constrain the output of MCT to prevent falling into poor solutions when training on unpaired datasets. Curve-based methods usually predict a global transformation, which prevents them from working on more challenging I2I tasks such as daytime translation. 3D LUTs predict global transformations like curve mappings, but they have a stronger translation capability. Ideally, we could introduce spatial coordinates to extend 3D LUTs to 5D LUTs, but this would lead to an unacceptable computational cost and memory consumption. From the implementation perspective, MCT extends the curve-based methods by introducing spatial coordinates and channel interactions to improve translation capability, which can be implemented using 3D LUTs. More importantly, MCT is a plug-in module that does not rely on fancy backbone networks and loss functions, and it can be trained directly on small-sized images, dramatically reducing the effort to modify the methods.

## 2.2   Multi-Curve Translator

Recalling Eq.(5), our goals are to 1) design the encoder $E$; 2) design the translator $G$; and 3) define the relation between $i$ and $j$. We desire our approach to be plug-in for the existing I2I translation models. Since the models of I2I translation tasks are often based on FCNs, we directly use these networks as our base models (*i.e.*, backbone networks) to eliminate the effort involved in designing the encoder $E$. Then the only modification needed is to increase their last layer's output channels to match the parameters of the translator $G$. Given that $x \in \mathcal{R}^{H \times W \times 3}$ and $x\downarrow \in \mathcal{R}^{H_d \times W_d \times 3}$, then $F \in \mathcal{R}^{H_d \times W_d \times C}$, where $C$ is the number of parameters of $G$.
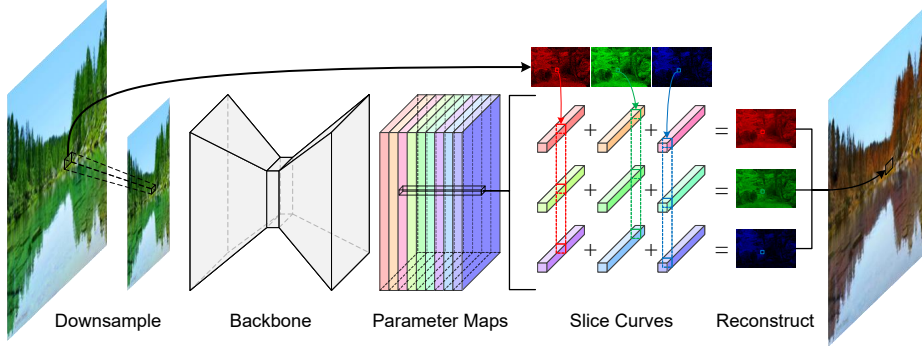
**Fig. 2.** Inference workflow of MCT. The backbone network receives the downsampled image and predicts the curve parameter maps with the same spatial size as the downsampled image. A cell of curve parameter maps consists of 9 sets of curves in the form of 1D LUTs, responsible for translating the corresponding region in the HR image.

Reviewing the idea of HyperNetwork, a simple idea is to employ an FCN as the translator $G$. However, since convolutional layers lead to a large $C$, we seek another expressive nonlinear mapping with fewer parameters to replace the FCN. We found that some curve-based methods [25,31,46] achieved better performance than FCN-based methods [7,22,58] on photo retouching. These curve-based methods make the network regress the knot points of the curve to mimic the color adjustment curve tool. Although these methods implement knot points in different ways, they are all equivalent to 1D LUTs [24,37]. We illustrate the transformation function using the curve in the form of a 1D LUT for a grayscale image. Given a 1D LUT $\mathbf{T} = \{t_{(k)}\}_{k=0,...,M-1}$ (*i.e.*, $M$ knot points), pixel $x_{(i,j)}$ can find its location z in the LUT via a lookup operation:

$$\text{z} = x_{(i,j)} \cdot \frac{M-1}{C_{max}}, \tag{6}$$

where $C_{max}$ is the maximum pixel value. Since z may not be an integer, we should derive the output pixel value via interpolation. Let $d_{\text{z}} = \text{z} - \lfloor \text{z} \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function. Given that $\lceil \cdot \rceil$ is the ceil function, we derive output pixel value $y_{(i,j)}$ via linear interpolation:

$$\tilde{y}_{(i,j)} = (1 - d_{\text{z}}) \cdot t_{(\lfloor \text{z} \rfloor)} + d_{\text{z}} \cdot t_{(\lceil \text{z} \rceil)}. \tag{7}$$

Finally we need to define the relation between $i$ and $j$. We can upsample the parameter maps to make their resolution the same as $x$ (*i.e.*, $F \uparrow \in \mathcal{R}^{H \times W \times C}$). Unfortunately, while the computational cost of this operation is acceptable, it produces larger parameter maps, which may consume a lot of memory. Inspired by bilateral grid [6], we employ a 3D LUT $\mathbf{T} \in \mathcal{R}^{H_d \times W_d \times M}$. Given a grayscale pixel $x_{(i,j)}$, its location $(\text{x}, \text{y}, \text{z})$ in the 3D LUT lattice is:

$$\text{x} = \text{i} \cdot \frac{H_d - 1}{H - 1}, \text{ y} = \text{j} \cdot \frac{W_d - 1}{W - 1}, \text{ z} = x_{(i,j)} \cdot \frac{M-1}{C_{max}}. \tag{8}$$

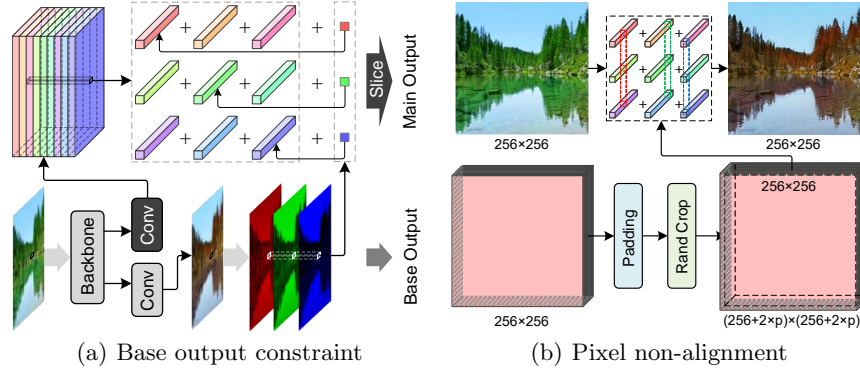(a) Base output constraint               (b) Pixel non-alignment

**Fig. 3.** Two strategies to train MCT variants in a stable manner. (a) Use the translated LR image as the base of the parameter maps. We can constrain the base output in the training phase to ensure that the backbone network does not degrade. (b) Train the MCT using only LR images. We use padding and random cropping to obtain parameter maps that are not pixel-aligned with the LR images.

Let $d_{\mathrm{x}} = \mathrm{x} - \lfloor \mathrm{x} \rfloor$ and $d_{\mathrm{y}} = \mathrm{y} - \lfloor \mathrm{y} \rfloor$, we extend Eq.(7) to trilinear interpolation to slice the output pixel:

$$
\begin{aligned}
\tilde{y}_{(\mathrm{i,j})} = {} & (1-d_{\mathrm{x}})(1-d_{\mathrm{y}})(1-d_{\mathrm{z}})t_{(\lfloor \bar{\mathrm{x}} \rfloor, \lfloor \mathrm{y} \rfloor, \lfloor \mathrm{z} \rfloor)} + d_{\mathrm{x}}d_{\mathrm{y}}d_{\mathrm{z}}t_{(\lceil \bar{\mathrm{x}} \rceil, \lceil \bar{\mathrm{y}} \rceil, \lceil \bar{\mathrm{z}} \rceil)} \\
& + (1-d_{\mathrm{x}})d_{\mathrm{y}}(1-d_{\mathrm{z}})t_{(\lfloor \bar{\mathrm{x}} \rfloor, \lceil \bar{\mathrm{y}} \rceil, \lfloor \mathrm{z} \rfloor)} + d_{\mathrm{x}}(1-d_{\mathrm{y}})d_{\mathrm{z}}t_{(\lceil \bar{\mathrm{x}} \rceil, \lfloor \mathrm{y} \rfloor, \lceil \bar{\mathrm{z}} \rceil)} \\
& + d_{\mathrm{x}}d_{\mathrm{y}}(1-d_{\mathrm{z}})t_{(\lceil \bar{\mathrm{x}} \rceil, \lceil \bar{\mathrm{y}} \rceil, \lfloor \mathrm{z} \rfloor)} + (1-d_{\mathrm{x}})(1-d_{\mathrm{y}})d_{\mathrm{z}}t_{(\lfloor \bar{\mathrm{x}} \rfloor, \lfloor \mathrm{y} \rfloor, \lceil \bar{\mathrm{z}} \rceil)} \\
& + d_{\mathrm{x}}(1-d_{\mathrm{y}})(1-d_{\mathrm{z}})t_{(\lceil \bar{\mathrm{x}} \rceil, \lfloor \mathrm{y} \rfloor, \lfloor \mathrm{z} \rfloor)} + (1-d_{\mathrm{x}})d_{\mathrm{y}}d_{\mathrm{z}}t_{(\lfloor \bar{\mathrm{x}} \rfloor, \lceil \bar{\mathrm{y}} \rceil, \lceil \bar{\mathrm{z}} \rceil)}.
\end{aligned}
\tag{9}
$$

For RGB color images, we employ the channel-crossing strategy [54]. Specifically, 9 curves should be learned, corresponding to $\{\mathbf{T}^{p \to q}\}_{p,q \in \{R,G,B\}}$ respectively ($C = 9M$). Let $\mathbf{T}(\cdot)$ denote Eq.(8-9), we derive output pixel via:

$$
\tilde{y}_{(\mathrm{i,j})}^{q} = \mathbf{T}^{R \to q}(x_{(\mathrm{i,j})}^{R}) + \mathbf{T}^{G \to q}(x_{(\mathrm{i,j})}^{G}) + \mathbf{T}^{B \to q}(x_{(\mathrm{i,j})}^{B}).
\tag{10}
$$

Since the translator consists of a large number of curves, we call it Multi-Curve Translator (MCT). Fig. 2 shows how MCT processes a HR image.

### 2.3   Training Strategy

Although MCT appears to be more complex than reconstructing images using the convolutional layer, its last layer still outputs pixel values. As a special case, when only the LUTs $\{\mathbf{T}^{p \to p}\}_{p \in \{R,G,B\}}$ are included and $M = 1$, MCT is equivalent to upsampling the output image of the base model. However, we found that the MCT variant is more like to fall into poor solutions than the base model. We review the MCT and find that the input image's information flows into the output image through the backbone network and slicing operation. We

suppose that the cause of the performance degradation is that the network has difficulty balancing the information flowing through the two routes. Therefore, we add constraints to the MCT in the training phase to drive the information from both routes to flow adequately into the output image.

Firstly, we should make the MCT leverage the information of downsampled images. MCT makes high-frequency information from the input image be easily retained in the output image, but we found that MCT may learn a simple color transformation. The problem arises because the information flows too easily from the input to the output, leading to a "short circuit" phenomenon that traps the network in a poor local optimum solution. Recalling the special case of MCT, we find that $\{\mathbf{T}^{p \to p}\}_{p \in \{R,G,B\}}$ can be decomposed into LUTs and biases, as shown in Fig. 3(a). Specifically, we use the last layer of the base model to predict the reconstructed image and add its pixel values as biases to the corresponding LUTs. So we can obtain the base output $\tilde{y}_b$ and the main output $\tilde{y}_m$ by $[\tilde{y}_b, \tilde{y}_m] = [G_b(x), G_m(x)]$ and constrain $\tilde{y}_b$ at training phase to ensure that the backbone network does not degrade. This strategy has a bonus that the pre-trained base model's weight can be fully utilized, including the output layer. Therefore, we employ this strategy even if we do not need to constrain the base output. In extreme cases, we can fine-tune only the added output convolutional layers to make the training faster.

Secondly, we should make the MCT leverage the information of HR images. MCT allows us to perform the backbone network on LR images, dramatically reducing the computational cost of translating HR images. However, we are still unable to use HR images as training data during training directly. The reasons are threefold: 1) loading and preprocessing HR images takes a lot of time, resulting in inefficient training; 2) The discriminator's computational cost remains proportional to the input image pixels. 3) Existing datasets often provide low-resolution images. For this reason, we still use LR images to train the MCT. As shown in Fig. 3(b), we first pad each side of the parameter maps by size $p$ with duplication and randomly crop them to the size before padding, then the image and the parameter maps are not pixel-wise aligned, forcing MCT to extract high-frequency information from the image. We can also achieve more complex pixel misalignment by adding small random noise to x and y, but this does not visibly improve performance in our experiments.

## 3   Applications

We apply MCT to extend some representative I2I translation methods. Unless otherwise noted, we set $H_d = W_d = 256$ and $M = 8$, and employ pixel unaligned training strategy with $p = 1$ but do not constrain the base output.

### 3.1   Photorealistic I2I Translation

We refer here to the I2I translation tasks done with GANs [13]. We perform the daytime translation (`day2dusk`) and season translation (`summer2autumn`)

for experiments. To extend to HR scenes, we collected new unpaired datasets from Flickr[1] with image resolutions ranging from 480p to 8K. Each domain of the datasets contains 2200 images, of which 2000 LR images are downsampled for training and the remaining 200 HR images for testing.

We employ CycleGAN [68] and UNIT [39] as the base models, which use different training procedures. When training the MCT variants, we add constraints to the base output $y_b$ when updating the generator. Let the conventional generator's loss function be $\mathcal{L}_{base}$, then the loss function of the MCT variant is $\mathcal{L} = \mathcal{L}_{base} + \lambda \mathcal{L}_{reg}$, where $\lambda = 1$ for CycleGAN and $\lambda = 10$ for UNIT. $\mathcal{L}_{reg}$ is a cycle-consistency loss [68,26] constrainting the base output:

$$\mathcal{L}_{reg} = \left\| G_b^{y \to x}(G_m^{x \to y}(x)) - x \right\|_1 . \tag{11}$$

### 3.2 Style Transfer

Style transfer aims at transferring the style from a reference image to a content image and is divided into two types: artistic style transfer and photorealistic style transfer. We only study the photorealistic style transfer since it fits our motivation. We use the Microsoft COCO dataset [38] to train the base models and their MCT variants, and the test set consists of the examples provided by DPST [41] with image resolutions ranging from 720p to 4K.

We use AdaIN [20] and WCT$^2$ [62] as the base models since they employ different training schemes. AdaIN is designed for artistic style transfer, with few constraints on preserving high-frequency information. It employs a weighted combination of the content loss $\mathcal{L}_c$ and the style loss $\mathcal{L}_s$ with the weight $\lambda$, *i.e.* $\mathcal{L} = \mathcal{L}_c + \lambda \mathcal{L}_s$. Both $\mathcal{L}_c$ and $\mathcal{L}_s$ use pre-trained VGG-19 [53] to compute the loss function without constraining the pixels of the images. So we add a gradient loss $\mathcal{L}_g$ to prompt the preservation of the geometric structure:

$$\mathcal{L}_g = \left\| \nabla_h G(x) - \nabla_h x \right\|_2^2 + \left\| \nabla_v G(x) - \nabla_v x \right\|_2^2 , \tag{12}$$

where $\nabla_h$ ($\nabla_v$) denotes the gradient operator along the horizontal (vertical) direction. The modified AdaIN's full objective is $\mathcal{L} = \mathcal{L}_c + \lambda_1 \mathcal{L}_s + \lambda_2 \mathcal{L}_g$, where $\lambda_1 = 1$ and $\lambda_2 = 100$. The WCT$^2$'s scheme is special because it only requires the output image to reconstruct the input during training and performs WCT [33] sequentially at each scale to achieve stylization during testing. Let the reconstruction loss function of WCT$^2$ be $\mathcal{L}_{rec}(G(x), x)$, and its MCT variant's loss function is:

$$\mathcal{L} = \mathcal{L}_{rec}(G_b(x), x) + \mathcal{L}_{rec}(G_m(x), x). \tag{13}$$

WCT$^2$'s training scheme makes the HR image's low-frequency information flow easily to the output image, so we perform the grayscale operation on the HR image. When performing stylization, we further match the HR image's brightness with the reference image's brightness to prevent the brightness of the HR image from being retained in the output image.

---

[1] https://www.flickr.com/

### 3.3   Image Dehazing

Image dehazing aims to recover clean images from hazy images, which is essential for subsequent high-level tasks. We choose 6000 synthetic image pairs from RESIDE dataset [30] for training, 3000 from the ITS subset, and 3000 from the OTS subset. We use two datasets, named SOTS [30] and HazeRD [66], to evaluate the performance of the methods. Note that the SOTS has more image pairs while the HazeRD has 4K-resolution image pairs.

We take GCANet [5] and MSBDN [9] as the base models. GCANet expands the receptive field by dilated convolution [36], while MSBDN uses upsampling and downsampling operations. For simplicity, we use only the $\mathcal{L}_1$ loss function to train the models instead of using the original training scheme of the base models. Note that for supervised training, the base output constraint is optional.

### 3.4   Photo Retouching

Photo retouching aims to adjust an image's brightness, contrast, and so on to make the image fit people's aesthetics. We choose 4500 image pairs from the MIT-Adobe-5K dataset [3] for training and the remaining 500 image pairs for testing with image resolutions ranging from 2K to 6K. We also employ an unpaired training scheme, with the 2250 images as domain $\mathcal{X}$ and the remaining 2250 images as domain $\mathcal{Y}$ in the training set.

We use DPED [22] and DPE [7] as the base models. Specifically, DPED uses a residual network [16] without downsampling and upsampling, leading to a small receptive field. But large scale context is critical for photo retouching, which is used to sense the illumination and contrast of an image [12,7]. So we set $H_d = W_d = 32$ for DPED to enlarge the receptive field without modifying the network architecture. Since color mapping is critical for photo retouching, we set $p = 8$ for DPE. For paired training, we use the $\mathcal{L}_1$ loss function to train the models. We employ the CycleGAN's training scheme for unpaired training rather than the base models' training scheme for comparison purposes.

## 4   Experiments

### 4.1   Runtime

We have shown the advantages of the MCT in terms of computational cost in Fig. 1, but MACs are indirect metrics of speed [42], which is an unconvincing indicator. So we test the runtime of the base models and their MCT variants on multiple hardware platforms. Specifically, we use the PyTorch framework to test each method's frames per second (FPS) in float32 data format and set the mini-batch size to 1. Given the size of the input image, we randomly generate 200 images and compute the FPS for a single experiment by recording the total time to process the 200 images. Then we repeated each experiment 10 times and took the median of the 10 results as the final result. Fig. 4 illustrates the FPS of base models and their variants running on 3 models of GPUs.
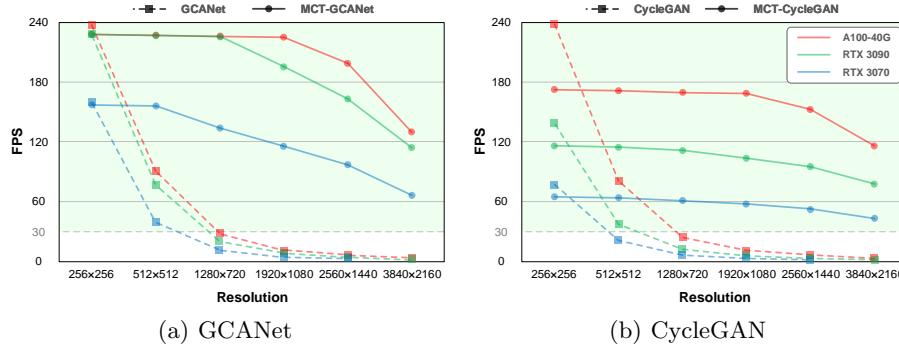
**Fig. 4.** Runtime comparison of the base models and their MCT variants on GPUs. No data for base models means that they runs with an OOM at that resolution.

If we feed a $256 \times 256$ image to the models, the MCT variants do not have any advantage over the base models since we set $H_d = W_d = 256$ in the experiment. On the other hand, the large $7 \times 7$ convolution kernels for the output layer introduce an additional 25% computational cost to the backbone network since the output channels of the MCT-CycleGAN's output layer increase. Finally, the curve slicing operation contains some operations with low computational cost but high memory access cost (*e.g.* indexing), further increasing the MCT variants' runtime. Fortunately, the computational cost of MCT's backbone network does not vary with the image size, which gives it a distinct advantage when working with HR images. When the input image size is $512 \times 512$, the MCT variants are significantly faster than the base models. Moreover, the gap between the MCT variants and the base models becomes increasingly large as the input image size grows. Taking 30 FPS as the cut-off for whether a model can run in real-time, the MCT variants can process 4K images in real-time on 3 models of GPUs. As a comparison, CycleGAN takes $40\times$ longer to process a 4K image (116.0 FPS vs. 2.7 FPS on A100), even with an out-of-memory (OOM) on RTX 3070. The computational cost of the curve slicing operation is so low that it accounts for less than 1% of the overall computational cost for processing 4K images. Still, it introduces a high memory access cost, making the curve slicing operation limited by the GPU's memory bandwidth. Finally, MCT-GCANet processes $256 \times 256$ and $512 \times 512$ images at almost the same speed on the A100 and RTX 3090, probably due to the limitations of CPU performance and PyTorch runtime.

### 4.2   Qualitative Comparison

We qualitatively compare the base models with their MCT variants on four I2I translation tasks, and Fig. 5 shows some examples.

Almost all base models have a non-global and fixed-size receptive field. In contrast, the receptive field of MCT variants grows larger as the input image
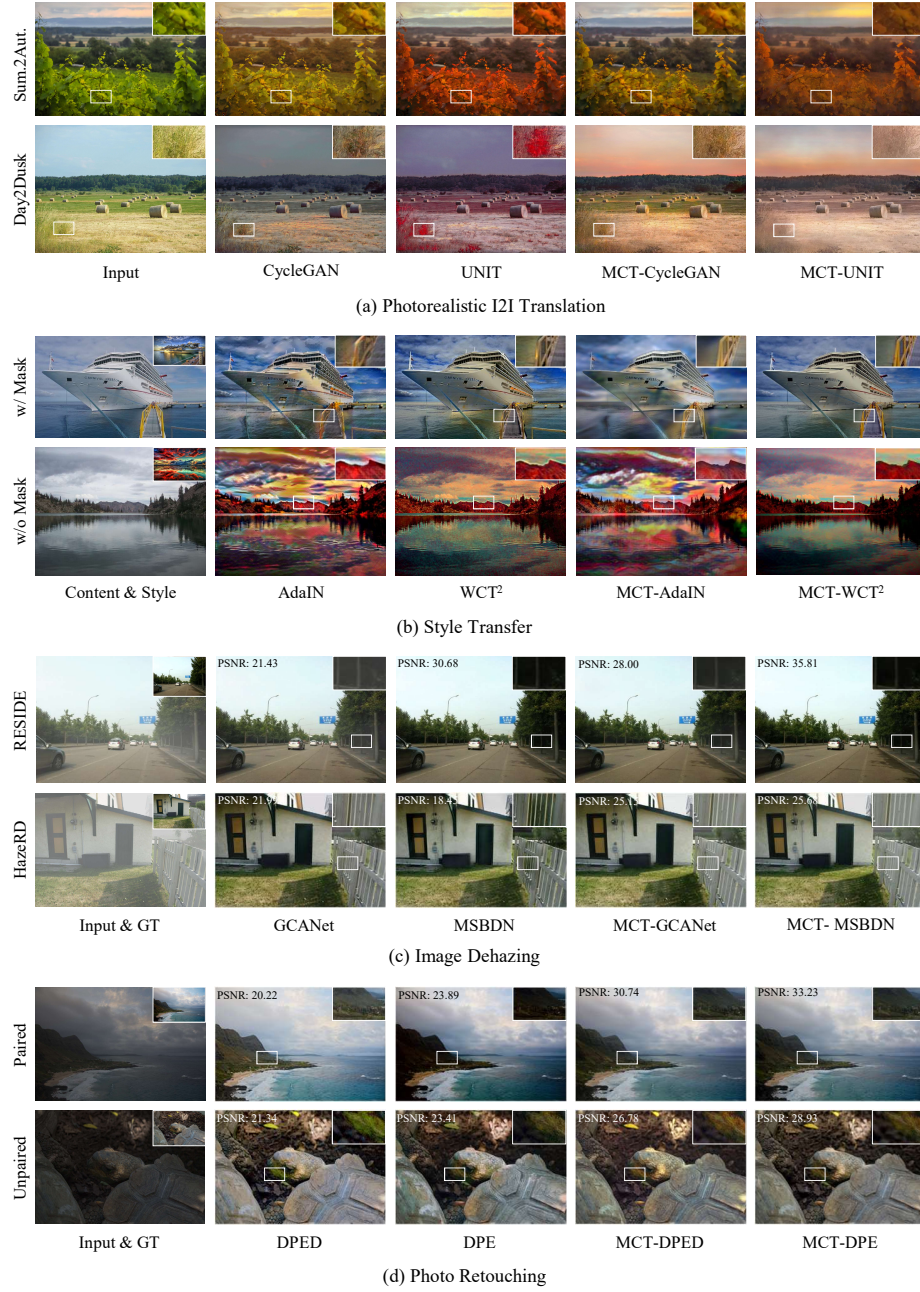
(a) Photorealistic I2I Translation

Input | CycleGAN | UNIT | MCT-CycleGAN | MCT-UNIT



(b) Style Transfer

Content & Style | AdaIN | WCT$^2$ | MCT-AdaIN | MCT-WCT$^2$



(c) Image Dehazing

Input & GT | GCANet | MSBDN | MCT-GCANet | MCT- MSBDN



(d) Photo Retouching

Input & GT | DPED | DPE | MCT-DPED | MCT-DPE

**Fig. 5.** Qualitative comparison of four I2I translation tasks, each consisting of examples in two experimental settings. From top to bottom are (a) photorealistic I2I translation, (b) style transfer, (c) image dehazing, and (d) photo retouching.
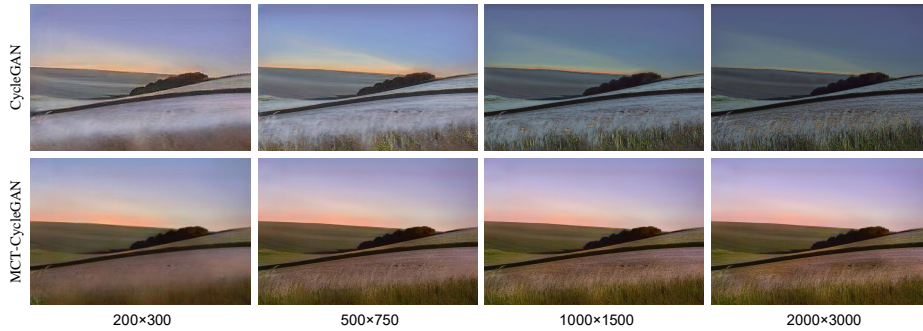
**Fig. 6.** Comparison of CycleGAN and MCT-CycleGAN when processing the same image in different resolution versions on `day2dusk`.

becomes larger. In the `day2dusk` example, the base models only change the colors of the ground and sky and do not generate sunset light. This is because the resolution of the input image is so high that the base models with limited receptive fields cannot determine the sky area. Fig. 6 futher shows the results when CycleGAN and MCT-CycleGAN process the same image in different resolution versions. CycleGAN works fine on low-resolution images but cannot translate HR images well. The same problem occurs when MSBDN processes HR images in the HazeRD dataset, where significant black artifacts appear on the white railings because the receptive field of MSBDN is not large enough to capture the railings' semantic information. The DPED's receptive field is small, so it tends to adjust the image's brightness locally to normal brightness, but the image lacks contrast globally. By lowering the resolution of the downsampled images, the MCT variant of DPED has a large receptive field so that it can better capture global illumination, leading to visually more pleasing results. In short, the MCT's receptive field is dynamic, which helps to capture the HR images' semantics.

MCT's curve slicing operation allows the backbone network to focus more on region semantics than retaining the high-frequency information. This is evident in the comparison of AdaIN with its MCT variant. The original network architecture of AdaIN does not contain any skip connection, resulting in the high-frequency information that VGG-19 loses not being recovered. Therefore, even after adjusting the weights of content loss and style loss and introducing the gradient loss, AdaIN still cannot reconstruct the input image's high-frequency information. For example, the text and railing are blurred, and the texture of the mountain is lost. In contrast, its MCT variant can preserve the high-frequency information in the input image by curve slicing operation without being limited by the network architecture. However, for network architectures like GCANet, the high-frequency information flow is only shifted from skip connections to the curve slicing operation, which does not produce visible differences in the output image details. We consider that MCT is more likely to retain high-frequency information of the input image.

**Table 1.** Quantitative comparison (PSNR & SSIM) of the image dehazing (upper) and photo retouching (lower). FPS is measured on 4K images using a single A100-40G.

| | | Compared Models | | | | Base Models | | MCT Variants | |
|---|---|---|---|---|---|---|---|---|---|
| | MS [49] | DHN [4] | AOD [29] | GFN [50] | MGBL [67] | GCANet | MSBDN | GCANet | MSBDN |
| SOTS | 20.31 | 21.02 | 20.27 | 23.52 | 24.50 | 25.09 | <u>28.56</u> | 25.71 | **28.70** |
| | 0.862 | 0.881 | 0.864 | 0.915 | 0.920 | 0.923 | **0.966** | 0.927 | <u>0.962</u> |
| HazeRD | 15.35 | 15.42 | 15.44 | 14.62 | 16.06 | 16.69 | 16.23 | **17.19** | <u>16.81</u> |
| | 0.634 | 0.622 | 0.660 | 0.580 | 0.794 | <u>0.825</u> | 0.805 | 0.810 | **0.840** |
| FPS | 13.6 | 14.8 | 41.6 | 3.0 | <u>120.8</u> | 3.1 | 2.2 | **131.1** | 35.9 |
| | WB [19] | HDR [12] | UPE [57] | LPF [45] | LPTN [35] | DPED | DPE | DPED | DPE |
| Paired | - | 23.15 | 23.24 | 24.48 | 23.86 | 24.11 | 24.14 | <u>24.73</u> | **25.10** |
| | - | 0.918 | 0.893 | 0.887 | 0.885 | 0.886 | 0.934 | <u>0.936</u> | **0.941** |
| Unpaired | 18.57 | 21.63 | 21.59 | 21.34 | 22.02 | 22.29 | 20.92 | <u>22.81</u> | **23.09** |
| | 0.701 | 0.885 | 0.884 | 0.866 | 0.879 | 0.884 | 0.854 | <u>0.902</u> | **0.905** |
| FPS | 0.13 | 14.3 | 15.9 | 2.3 | 37.9 | 2.5 | 10.8 | **181.1** | <u>162.4</u> |

### 4.3   Quantitative Comparison

We quantitatively compare the performance on image dehazing and photo retouching because the images from these two tasks have corresponding ground truth to compute PSNR and SSIM. In addition to the base models and their MCT variants, we trained some representative compared models. Table 1 shows the results. Note that HDRNet and DUPE use the open source Tensorflow [1] implementation, while the others use PyTorch framework.

For image dehazing, it can be seen that GCANet and MSBDN are powerful models, which are significantly better than the other compared models on the SOTS dataset. And the MCT variants can achieve comparable or even better performance than the base models. MSBDN performs overfitting and shows a significant performance degradation on the HazeRD dataset. In contrast, its MCT variant has a significantly higher SSIM, which indicates that more image detail is retained. For photo retouching, DPED and DPE are not state-of-the-art methods. But their MCT variants outperform the base models and compared models since the curve-based methods are in line with the image retouching process. The DPED's low SSIM is due to the small receptive field that cannot extract image contrast information effectively. Finally, DPE performs poorly in the unpaired training setting, which may be because the CycleGAN's training strategy is not suitable for it.

In terms of runtime, FCN-based methods are significantly slower than slice-based methods, even for lightweight networks such as AODNet and LPTN. Since we set $H_d = W_d = 32$ for MCT-DPED, it runs faster than MCT-DPE. Note that DUPE and HDRNet are both much slower than MCT-DPE, which is mainly due

**Table 2.** Quantitative comparison (FID / user study) of the photorealistic I2I translation. The percentage of user study results indicates the preferred model outputs out of 95 responses. Lower is better for FID, and higher is better for the user study.

|  | day2dusk | dusk2day | summer2autumn | autumn2summer |
|---|---|---|---|---|
| CycleGAN | 89.00 / 32.6% | 94.17 / 48.4% | **101.98** / 47.4% | 100.34 / 40.0% |
| MCT-CycleGAN | **81.67 / 67.4%** | **92.14 / 51.6%** | 103.45 / **52.6%** | **94.72 / 60.0%** |
| UNIT | 92.14 / 29.5% | 96.66 / **55.8%** | 105.15 / 42.1% | 95.18 / **50.5%** |
| MCT-UNIT | **84.22 / 70.5%** | **93.14** / 44.2% | **103.43 / 57.9%** | **91.35** / 49.5% |

to the inefficient open-source implementations. In our experiments, they can all reach about 180 FPS using 3D LUTs.

Table 2 shows the quantitative comparison results of the photorealistic I2I translation. We first compare the methods using Fréchet Inception Distance (FID) [17]. Unlike the usual experimental setup, all models translate HR images and then downsample the output images to $256 \times 256$ to compute the FID. Although these tasks are not difficult I2I translation problems, since the input images are usually high-resolution, the base models are often not performing as well as their MCT variants. We also released questionnaires to colleagues who were not involved in this work to conduct a small user study. Specifically, 19 users participated in this experiment, and we provided 5 randomly selected samples for each task. Overall, most users recognized the ability of MCT to retain high-frequency information. In particular, for the day2dusk task, most users felt that the output of the MCT variants was much better than the base models.

## 5  Discussion

**Contributions.** In this paper, we propose to modify the network's output layer for the I2I translation task. The network is extended to predict the output pixels for the input pixel's neighboring pixels. Since the pixels lost during downsampling are the neighboring pixels of the pixels that remain, the modified network can receive LR images to predict the mapping for process HR images. For the adversarial training, we introduced two additional training strategies to stabilize the training. Experimental results show that it can perform comparable or even better than the conventional models but translate 4K images in real-time on various photorealistic I2I translation tasks.

**Limitations.** MCT is a trade-off between translation capability and speed, and it cannot be applied to the more difficult I2I translation tasks. For tasks that the I2I translation process greatly changes the shape and texture of the objects in the image (*i.e.*, high-frequency information), such as dog2cat, MCT is helpless. In future research, we hope to improve its capabilities further to make it be applied to more I2I translation tasks.

# References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: OSDI (2016) 13
2. Anokhin, I., Solovev, P., Korzhenkov, D., Kharlamov, A., Khakhulin, T., Silvestrov, A., Nikolenko, S., Lempitsky, V., Sterkin, G.: High-resolution daytime translation without domain labels. In: CVPR (2020) 1
3. Bychkovsky, V., Paris, S., Chan, E., Durand, F.: Learning photographic global tonal adjustment with a database of input/output image pairs. In: CVPR (2011) 9
4. Cai, B., Xu, X., Jia, K., Qing, C., Tao, D.: Dehazenet: An end-to-end system for single image haze removal. IEEE TIP (2016) 1, 13
5. Chen, D., He, M., Fan, Q., Liao, J., Zhang, L., Hou, D., Yuan, L., Hua, G.: Gated context aggregation network for image dehazing and deraining. In: WACV (2019) 9
6. Chen, J., Paris, S., Durand, F.: Real-time edge-aware image processing with the bilateral grid. ACM TOG (2007) 5
7. Chen, Y.S., Wang, Y.C., Kao, M.H., Chuang, Y.Y.: Deep photo enhancer: Unpaired learning for image enhancement from photographs with gans. In: CVPR (2018) 5, 9
8. Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: CVPR (2018) 1
9. Dong, H., Pan, J., Xiang, L., Hu, Z., Zhang, X., Wang, F., Yang, M.H.: Multi-scale boosted dehazing network with dense feature fusion. In: CVPR (2020) 9
10. Fan, Q., Chen, D., Yuan, L., Hua, G., Yu, N., Chen, B.: Decouple learning for parameterized image operators. In: ECCV (2018) 4
11. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: CVPR (2016) 1
12. Gharbi, M., Chen, J., Barron, J.T., Hasinoff, S.W., Durand, F.: Deep bilateral learning for real-time image enhancement. ACM TOG (2017) 4, 9, 13
13. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NeurIPS (2014) 7
14. Ha, D., Dai, A., Le, Q.V.: Hypernetworks. In: ICLR (2017) 4
15. He, K., Sun, J., Tang, X.: Guided image filtering. IEEE TPAMI (2012) 3
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 9
17. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. NeurIPS (2017) 14
18. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017) 3
19. Hu, Y., He, H., Xu, C., Wang, B., Lin, S.: Exposure: A white-box photo post-processing framework. ACM TOG (2018) 13
20. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: ICCV (2017) 8
21. Huang, X., Liu, M.Y., Belongie, S., Kautz, J.: Multimodal unsupervised image-to-image translation. In: ECCV (2018) 1

22. Ignatov, A., Kobyshev, N., Timofte, R., Vanhoey, K., Van Gool, L.: Dslr-quality photos on mobile devices with deep convolutional networks. In: ICCV (2017) 5, 9
23. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: CVPR (2017) 1, 3
24. Karaimer, H.C., Brown, M.S.: A software platform for manipulating the camera imaging pipeline. In: ECCV (2016) 2, 5
25. Kim, H.U., Koh, Y.J., Kim, C.S.: Global and local enhancement networks for paired and unpaired image enhancement. In: ECCV (2020) 4, 5
26. Kim, T., Cha, M., Kim, H., Lee, J.K., Kim, J.: Learning to discover cross-domain relations with generative adversarial networks. In: ICML (2017) 8
27. Klocek, S., Maziarka, Ł., Wołczyk, M., Tabor, J., Nowak, J., Śmieja, M.: Hypernetwork functional image representation. In: ICANN (2019) 4
28. Laffont, P.Y., Ren, Z., Tao, X., Qian, C., Hays, J.: Transient attributes for high-level understanding and editing of outdoor scenes. ACM TOG (2014) 2
29. Li, B., Peng, X., Wang, Z., Xu, J., Feng, D.: Aod-net: All-in-one dehazing network. In: ICCV (2017) 13
30. Li, B., Ren, W., Fu, D., Tao, D., Feng, D., Zeng, W., Wang, Z.: Benchmarking single-image dehazing and beyond. IEEE TIP (2018) 9
31. Li, C., Guo, C., Ai, Q., Zhou, S., Loy, C.C.: Flexible piecewise curves estimation for photo enhancement. arXiv preprint arXiv:2010.13412 (2020) 4, 5
32. Li, M., Lin, J., Ding, Y., Liu, Z., Zhu, J.Y., Han, S.: Gan compression: Efficient architectures for interactive conditional gans. In: CVPR (2020) 1
33. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Universal style transfer via feature transforms. In: NeurIPS (2017) 8
34. Li, Y., Liu, M.Y., Li, X., Yang, M.H., Kautz, J.: A closed-form solution to photo-realistic image stylization. In: ECCV (2018) 2
35. Liang, J., Zeng, H., Zhang, L.: High-resolution photorealistic image translation in real-time: A laplacian pyramid translation network. In: CVPR (2021) 1, 13
36. Liang-Chieh, C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.: Semantic image segmentation with deep convolutional nets and fully connected crfs. In: ICLR (2015) 9
37. Lin, H.T., Lu, Z., Kim, S.J., Brown, M.S.: Nonuniform lattice regression for modeling the camera imaging pipeline. In: ECCV (2012) 2, 5
38. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014) 8
39. Liu, M.Y., Breuel, T., Kautz, J.: Unsupervised image-to-image translation networks. In: NeurIPS (2017) 1, 8
40. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015) 1
41. Luan, F., Paris, S., Shechtman, E., Bala, K.: Deep photo style transfer. In: CVPR (2017) 2, 8
42. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: ECCV (2018) 9
43. Mechrez, R., Talmi, I., Zelnik-Manor, L.: The contextual loss for image transformation with non-aligned data. In: ECCV (2018) 3
44. Mildenhall, B., Barron, J.T., Chen, J., Sharlet, D., Ng, R., Carroll, R.: Burst denoising with kernel prediction networks. In: CVPR (2018) 3
45. Moran, S., Marza, P., McDonagh, S., Parisot, S., Slabaugh, G.: Deeplpf: Deep local parametric filters for image enhancement. In: CVPR (2020) 13
46. Moran, S., McDonagh, S., Slabaugh, G.: Curl: Neural curve layers for global image enhancement. In: ICPR (2021) 4, 5

47. Muller, L.K.: Overparametrization of hypernetworks at fixed flop-count enables fast neural image enhancement. In: CVPRW (2021) 4
48. Park, T., Efros, A.A., Zhang, R., Zhu, J.Y.: Contrastive learning for unpaired image-to-image translation. In: ECCV (2020) 1, 3
49. Ren, W., Liu, S., Zhang, H., Pan, J., Cao, X., Yang, M.H.: Single image dehazing via multi-scale convolutional neural networks. In: ECCV (2016) 13
50. Ren, W., Ma, L., Zhang, J., Pan, J., Cao, X., Liu, W., Yang, M.H.: Gated fusion network for single image dehazing. In: CVPR (2018) 13
51. Shaham, T.R., Gharbi, M., Zhang, R., Shechtman, E., Michaeli, T.: Spatially-adaptive pixelwise networks for fast image translation. In: CVPR (2021) 4
52. Shu, H., Wang, Y., Jia, X., Han, K., Chen, H., Xu, C., Tian, Q., Xu, C.: Co-evolutionary compression for unpaired image translation. In: ICCV (2019) 1
53. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015) 8
54. Song, Y., Qian, H., Du, X.: Starenhancer: Learning real-time and style-aware image enhancement. In: ICCV (2021) 6
55. Song, Y., Zhu, Y., Du, X.: Model parameter learning for real-time high-resolution image enhancement. IEEE SPL (2020) 4
56. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: ICCV (1998) 3
57. Wang, R., Zhang, Q., Fu, C.W., Shen, X., Zheng, W.S., Jia, J.: Underexposed photo enhancement using deep illumination estimation. In: CVPR (2019) 13
58. Wei, C., Wang, W., Yang, W., Liu, J.: Deep retinex decomposition for low-light enhancement. In: BMVC (2018) 5
59. Xia, X., Zhang, M., Xue, T., Sun, Z., Fang, H., Kulis, B., Chen, J.: Joint bilateral learning for real-time universal photorealistic style transfer. In: ECCV (2020) 4
60. Yang, C., Jin, M., Jia, X., Xu, Y., Chen, Y.: Adaint: Learning adaptive intervals for 3d lookup tables on real-time image enhancement. In: CVPR (2022) 4
61. Yin, H., Gong, Y., Qiu, G.: Side window filtering. In: CVPR (2019) 3
62. Yoo, J., Uh, Y., Chun, S., Kang, B., Ha, J.W.: Photorealistic style transfer via wavelet transforms. In: ICCV (2019) 8
63. Zeng, H., Cai, J., Li, L., Cao, Z., Zhang, L.: Learning image-adaptive 3d lookup tables for high performance photo enhancement in real-time. IEEE TPAMI (2020) 4
64. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE TIP (2017) 1
65. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: ECCV (2016) 1
66. Zhang, Y., Ding, L., Sharma, G.: Hazerd: an outdoor scene dataset and benchmark for single image dehazing. In: ICIP (2017) 9
67. Zheng, Z., Ren, W., Cao, X., Hu, X., Wang, T., Song, F., Jia, X.: Ultra-high-definition image dehazing via multi-guided bilateral learning. In: CVPR (2021) 4, 13
68. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV (2017) 1, 3, 8