Path-Restore: Learning Network Path Selection for Image Restoration

Ke Yu, Xintao Wang, Chao Dong, Xiaoou Tang, *Fellow, IEEE*, and Chen Change Loy, *Senior Member, IEEE*

Abstract—Very deep Convolutional Neural Networks (CNNs) have greatly improved the performance on various image restoration tasks. However, this comes at a price of increasing computational burden, hence limiting their practical usages. We observe that some corrupted image regions are inherently easier to restore than others since the distortion and content vary within an image. To leverage this, we propose Path-Restore, a multi-path CNN with a pathfinder that can dynamically select an appropriate route for each image region. We train the pathfinder using reinforcement learning with a difficulty-regulated reward. This reward is related to the performance, complexity and "the difficulty of restoring a region". A policy mask is further investigated to jointly process all the image regions. We conduct experiments on denoising and mixed restoration tasks. The results show that our method achieves comparable or superior performance to existing approaches with less computational cost. In particular, Path-Restore is effective for real-world denoising, where the noise distribution varies across different regions on a single image. Compared to the state-of-the-art RIDNet [1], our method achieves comparable performance and runs 2.7x faster on the realistic Darmstadt Noise Dataset [2]. Models and codes are available on the project page: https://www.mmlab-ntu.com/project/pathrestore/.

Index Terms—Image restoration, denoising, dynamic network, deep reinforcement learning

1 Introduction

IMAGE restoration aims at estimating a clean image from its distorted observation. Very deep Convolutional Neural Networks (CNNs) have achieved great success on various restoration tasks. For example, in the NTIRE 2018 Challenge [3], the top-ranking super-resolution methods are built on very deep models such as EDSR [4] and DenseNet [5], achieving impressive performance even when the input images are corrupted with noise and blur. However, as the network becomes deeper and more complex, the increasing computational cost makes it less practical for real-world applications.

Do we really need a very deep CNN to process all the regions of a distorted image? In reality, a large variation of image content and distortion may exist in a single image, and some regions are inherently easier to process than others. An example is shown in Fig. 1, where we use several denoising CNNs with different depths to restore a noisy image. It is observed, in the red box, that a smooth region with mild noise is reasonably recovered with a 3-layer CNN; further increasing the network depth brings only incremental improvement. In the green box, a similar smooth region yet with severe noise requires an 8-layer CNN to process, whereas a texture region in the blue box still has some artifacts (highlighted by the white arrow) after being processed by a 20-layer CNN. This observation motivates the possibility of saving computations by selecting an optimal network path for each region based on its content and distortion.

In this study, we propose Path-Restore, a novel framework that can dynamically select a path for each image region with specific content and distortion. In particular, a pathfinder is trained together with a multi-path CNN to find the optimal dispatch policy for each image region. Since path selection is non-differentiable, the pathfinder is trained in a reinforcement learning (RL) framework driven by a reward that encourages both high performance and few computations. There are mainly two challenges. First, the multi-path CNN should be able to handle a large variety of distortions with limited computations. Second, the pathfinder requires an informative reward to learn a good policy for different regions with diverse contents and distortions.

We make two contributions to address these challenges:

(1) We devise a dynamic block as the basic unit of our multi-path CNN. A dynamic block contains a shared path followed by several dynamic paths with different complexity. The proposed framework allows flexible design on the number of paths and path complexity according to the specific task at hand.

(2) We devise a difficulty-regulated reward. This reward values the performance gain of hard examples more than simple ones, unlike conventional reward

- *Ke Yu and Xiaoou Tang are with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong. E-mail: {yk017, xtang}@ie.cuhk.edu.hk.*
- *Xintao Wang is with the Applied Research Center (ARC), Tencent PCG, Shenzhen, Guangdong 518054, China. E-mail: xintao.wang@outlook.com.*
- *Chao Dong is with the Guangdong-Hong Kong-Macao Joint Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong 518055, China. E-mail: chao.dong@siat.ac.cn.*
- *Chen Change Loy is with the S-Lab, Nanyang Technological University, Singapore 639798, Singapore. E-mail: ccloy@ntu.edu.sg.*

Manuscript received 12 Oct. 2019; revised 5 May 2021; accepted 2 July 2021. Date of publication 13 July 2021; date of current version 9 Sept. 2022. (Corresponding author: Chen Change Loy.) Recommended for acceptance by E. Shechtman. Digital Object Identifier no. 10.1109/TPAMI.2021.3096255

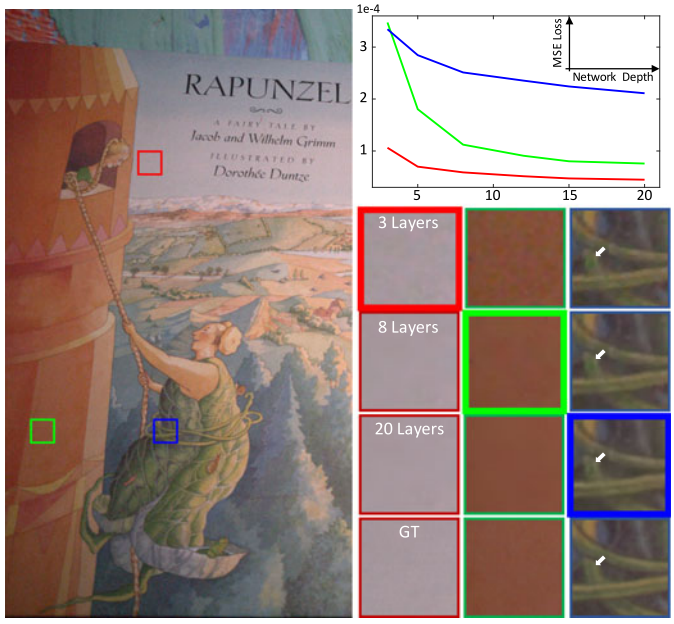


Fig. 1. The relation between performance and network depth for different image regions. The noisy image is shown on the left. The restored outputs and the ground-truth image regions are presented on the right. Each curve represents the MSE loss of the corresponding region. The bold boxes denote a good trade-off between performance and complexity for each region. Zoom in for best view.

functions. Specifically, the difficulty-regulated reward scales the performance gain of different regions by an adaptive coefficient that represents the difficulty of restoring this region. The difficulty is made proportional to a loss function (e.g., MSE loss), since a large loss suggests that the region is difficult to restore. For example, in Fig. 1, the loss of the green region decreases by a larger quantity than that of the blue region, resulting in a higher performance reward for the green region compared to the blue one. However, after regulated by a higher difficulty, the final reward of the blue region becomes larger, encouraging more computations for the restoration of hard regions. We experimentally find that this reward helps the pathfinder learn a more effective dispatch policy.

We further investigate two mechanisms to process different image regions. The first way is to treat all regions independently. During inference, an input image is first split into overlapping regions (patches of the same size). Then these regions are restored separately and merged together at last. This strategy is straightforward and easy to implement, thus is adopted in the basic version of Path-Restore. The second mechanism is to jointly process all regions at image level. The pathfinder first predicts a low-resolution policy mask, where each pixel represents a policy for a region in the input image. Then the features of different regions are sent to specific paths according to the corresponding policy on the predicted mask. With the policy mask, we can jointly process different regions with the guidance of adjacent features. Our method with the second strategy, namely Path-Restore-Mask, can achieve a better performance-complexity trade-off.

We summarize the merits of Path-Restore as follows: 1) Thanks to the dynamic path-selection mechanism, our method achieves comparable or superior performance to existing approaches with faster speed on various restoration tasks. 2) Our method is particularly effective in real-world denoising, where noise distribution is spatially variant within an image. We achieve state-of-the-art performance on the realistic Darmstadt Noise Dataset (DND) [2] benchmark[1]. 3) Our method is capable of balancing the performance and complexity trade-off by adjusting a hyper-parameter in the proposed reward function during training.

2 Related Work

Advances in Image Restoration. Image restoration has been extensively studied for decades. CNN-based methods have taken the leading role in several restoration tasks including denoising [6], [7], [8], [9], deblurring [10], [11], [12], deblocking [13], [14], [15], [16] and super-resolution [17], [18], [19], [20], [21], [22].

Although most of the previous papers focus on addressing a specific degradation, several recent works aim at dealing with a large variety of distortions simultaneously. For example, Zhang *et al.* [23] propose DnCNN that employs a single CNN to address different levels of Gaussian noise. Guo *et al.* [9] develop CBDNet that estimates a noise map as a condition to handle diverse real noise. Plotz *et al.* [24] devise a continuous and differentiable neural nearest neighbor block that works effectively on several restoration tasks. State-of-the-art denoising methods RIDNet [1] and PRIDNet [25] adopt feature attention and pyramid architectures to enlarge network capacity, respectively. However, processing all image regions using a single path makes these methods less efficient. Smooth regions with mild distortions do not need such a deep network to restore.

The idea of region-wise processing has been explored in the literature. Anwar *et al.* [26] propose a category-specific denoising algorithm. They apply internal denoising for smooth regions and category-specific external denoising for textured regions. We offer a thorough discussion about the differences between this method and our approach in Section 4.5.

Our earlier work [27] proposes RL-Restore to address mixed distortions through applying deep reinforcement learning. The method adaptively selects a sequence of CNNs to process each distorted image, achieving comparable performance with a single deep network while using fewer computational resources. Unlike RL-Restore that requires manual design for a dozen CNNs, we only use one network to achieve dynamic processing. Our method can thus be more easily migrated and generalized to various tasks. As Path-Restore does not need to switch among different models to process various regions, our method runs faster and produces more spatially consistent results than RL-Restore. Moreover, our new framework enjoys the flexibility to balance the trade-off between performance and complexity by merely adjusting the reward function, a feature that is not possible in RL-Restore [27].

Dynamic Networks. Dynamic networks have been investigated to achieve a better trade-off between speed and performance in different tasks. Bengio *et al.* [28] use conditional

1. https://noise.visinf.tu-darmstadt.de/benchmark/#results_srgb

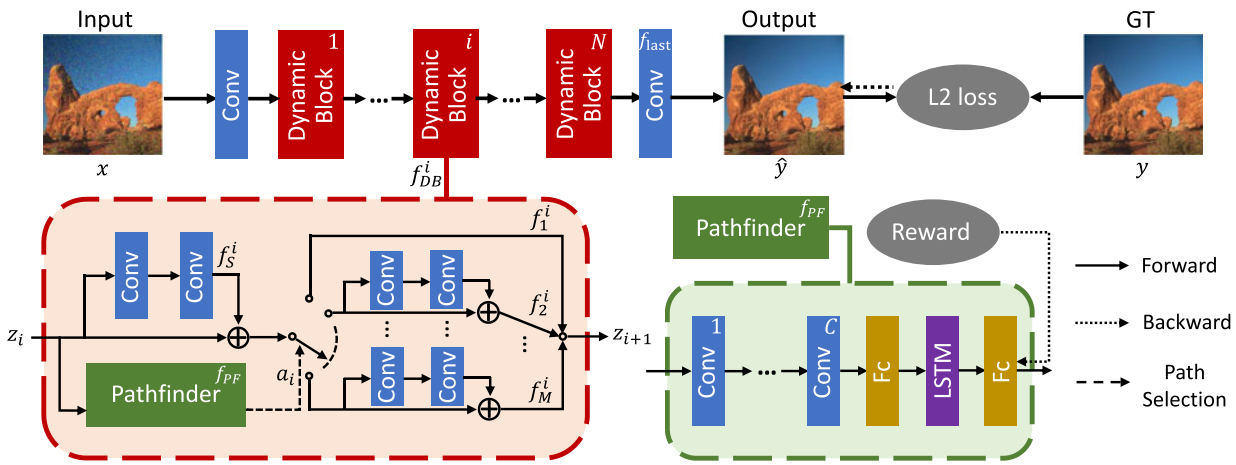


Fig. 2. Framework Overview. Path-Restore is composed of a multi-path CNN and a pathfinder. The multi-path CNN contains N dynamic blocks, each of which has M optional paths. The number of paths is made proportional to the number of distortion types we aim to address. The pathfinder is able to dynamically select paths for different image regions according to their contents and distortions.

computation to selectively activate one part of the network at a time. Recently, several approaches [29], [30], [31] are proposed to save the computational cost in ResNet [32] by dynamically skipping some residual blocks. While the aforementioned methods mainly address one single task, Rosenbaum *et al.* [33] develop a routing network to facilitate multi-task learning, and their method seeks an appropriate network path for each specific task. However, the path selection is irrelevant to the image content in each task.

Existing dynamic networks successfully explore a routing policy that depends on either the whole image content or the task to address. However, we aim at selecting dynamic paths to process different regions of a distorted image. In this case, both the content (e.g., smooth regions or rich textures) and the restoration task (e.g., distortion type and severity) have a weighty influence on the performance, and they should be both considered in the path selection. To address this specific challenge, we propose a dynamic block to offer diverse paths and adopt a difficulty-regulated reward to effectively train the pathfinder. The contributions are new in the literature.

Handling Non-Differentiable Operations. Deep reinforcement learning is comprehensively studied to address non-differentiable action selection. One of the most popular applications is decision making in games [34], [35], [36]. A tremendous success is also achieved in neural architecture search [37], [38], [39], where the selection of a layer or block is not differentiable. Hu *et al.* [40] develop an RL-based framework to learn a sequence of filters for image retouching.

Gumbel Softmax [41] is a widely-used technique to make a categorical distribution differentiable. For instance, Veit *et al.* [42] adopt Gumbel Softmax to achieve adaptive inference in a network. Xie *et al.* [43] and Wu *et al.* [44] leverage this technique for differentiable neural architecture search. In this work, we use an RL-based approach to train the pathfinder, as RL offers more flexibility to devise the reward. The reward function is not necessarily differentiable. On the contrary, as a differentiable estimator, Gumbel Softmax requires a differentiable target function to optimize. Nevertheless, a differentiable estimator is worth trying in future work given its high training efficiency.

3 Methodology

Our task is to recover a clear image y from its distorted observation x. We propose Path-Restore that can process each image region according to its content and distortion through a specific network path. We first provide an overview of the architecture of Path-Restore in Section 3.1. In Section 3.2, we then introduce the structure of pathfinder and the reward to evaluate its policy. In Section 3.3, we describe Path-Restore-Mask that can jointly process all image regions. Finally in Section 3.4, we detail the training algorithm.

3.1 Architecture of Path-Restore

We aim to design a framework that can offer different options of complexity. To this end, as shown in Fig. 2, Path-Restore is composed of a convolutional layer at the start- and end-point, and N dynamic blocks in the middle. In the ith dynamic block f^i_{DB}, there is a shared path f^i_S that every image region should pass through. Paralleling to the shared path, a pathfinder f_{PF} generates a probabilistic distribution of plausible paths for selection. Following the shared path, there are M dynamic paths denoted by $f^i_1, f^i_2, \ldots, f^i_M$. Each dynamic path contains a residual block except that the first path is a bypass connection. According to the output of pathfinder, the dynamic path with the highest probability is activated, where the path index is denoted by a_i. For instance, if there are two dynamic paths in total, then $a_i \in \{1, 2\}$. $a_i = 1$ and $a_i = 2$ represent that the first and the second path is chosen, respectively. The ith dynamic block can be formulated as:

$$z_{i+1} = f^i_{a_i}(f^i_S(z_i)), \tag{1}$$

where z_i and z_{i+1} denote the input and output of the ith dynamic block, respectively. Note that in two different dynamic blocks, the parameters of each corresponding path are different, while the parameters of pathfinder are shared. We use ReLU as the activation function between different layers. The last fully-connected layer of the pathfinder is activated by the Softmax function.

As can be seen from Eq. (1), the image features go through the shared path and one dynamic path. The shared

path is designed to capture the similarity among different tasks and images. The dynamic paths offer different options of complexity. Intuitively, simple samples should be led to the bypass path to save computations, while hard samples might be guided to another path according to the content and distortion.

3.2 Pathfinder

The pathfinder is the key component to achieve path selection. As shown in Fig. 2, the pathfinder contains C convolutional layers, followed by two fully-connected layers with a Long Short-Term Memory (LSTM) module in the middle. The number of convolutional layers depends on the specific task that will be specified in Section 4. The LSTM module is used to capture the correlations of path selection in different dynamic blocks. The pathfinder accounts for less than 3 percent of the overall computations.

Since path selection is non-differentiable, we formulate the sequential path selection as a Markov Decision Process (MDP) and adopt reinforcement learning to train the pathfinder. We will first clarify the state and action of this MDP, and then illustrate the proposed difficulty-regulated reward to train the pathfinder.

State and Action. In the ith dynamic block, the state s_i consists of the input features z_i and the hidden state of LSTM h_i, denoted by $s_i = \{z_i, h_i\}$. Given the state s_i, the pathfinder f_{PF} generates a distribution of path selection that can be formulated as $f_{PF}(s_i) = \pi(a|s_i)$. In the training phase, the action (path index) is sampled from this probabilistic distribution, denoted by $a_i \sim \pi(a|s_i)$. While in the testing phase, the action is determined by the highest probability, i.e., $a_i = \arg\max_a \pi(a|s_i)$.

Difficulty-Regulated Reward. In an RL framework, the pathfinder is trained to maximize a cumulative reward, and thus a proper design of reward function is critical. Existing dynamic models for classification [30], [31] usually use a trade-off between accuracy and computations as the reward. However, in our task, a more effective reward is required to learn a reasonable dispatch policy for different image regions that have diverse contents and distortions. Therefore, we propose a difficulty-regulated reward that not only considers performance and complexity, but also depends on "the difficulty of restoring an image region". In particular, the reward at the ith dynamic block is formulated as:

$$r_i = \begin{cases} -p \times (1 - \mathbf{1}_{\{1\}}(a_i)), & 1 \le i < N, \\ -p \times (1 - \mathbf{1}_{\{1\}}(a_i)) + d \times (-\Delta L_2), & i = N, \end{cases} \tag{2}$$

where $-p$ is the reward penalty for choosing a complex path in one dynamic block. $\mathbf{1}_{\{1\}}(\cdot)$ represents an indicator function, i.e., $\mathbf{1}_{\{1\}}(a_i) = 1$ only if $a_i = 1$, otherwise $\mathbf{1}_{\{1\}}(a_i) = 0$. a_i is the selected path index in the ith dynamic block. For example, $a_i = 1$ denotes that the first path (bypass connection) is selected. In this case the reward is not penalized, i.e., the reward penalty term is multiplied by zero. The performance and difficulty are only considered in the Nth dynamic block. In particular, $-\Delta L_2$ represents the performance gain in terms of L2 loss. The difficulty is denoted by d in the following formula:

$$d = \begin{cases} L_d / L_0, & 0 \le L_d < L_0, \\ 1, & L_d \ge L_0, \end{cases} \tag{3}$$

where L_d is the loss function of the output image and L_0 is a threshold. As L_d approaches zero, the difficulty d decreases, indicating the input region is easy to restore. In this case, the performance gain is regulated by $d < 1$ and the reward also becomes smaller, so that the proposed difficulty-regulated reward will penalize the pathfinder for wasting too much computation on easy regions. There are multiple choices for L_d such as L2 loss and VGG loss [46], and different loss functions may lead to different policy of path selection.

3.3 Path-Restore-Mask

In Path-Restore, the pathfinder selects paths for each region without considering the selection results of adjacent regions. This may decrease the accuracy of path selection since the observed region is small. Moreover, as different regions are processed independently, adjacent regions must overlap to reduce the artifacts along borders. Overlapping regions will inevitably introduce additional computation cost. Therefore, we propose Path-Restore-Mask to process a high-resolution input at image level with higher efficiency.

The architecture of Path-Restore-Mask is illustrated in Fig. 3. The overall structure is the same as Path-Restore (see Fig. 2), so we present the distinct parts – dynamic block and the pathfinder in Fig. 3. Let z_i and z_{i+1} denote the input and output features of a large image, respectively. The shared path is denoted by f_S^i. For simplicity, we consider a special case with two dynamic paths ($M = 2$), where the first path f_1^i is a bypass connection and the second path f_2^i is a residual block. The pathfinder is composed of three strided convolutional layers followed by a convolutional LSTM (ConvLSTM) [45]. The action a_i is a two-channel mask rather than a scalar. The first channel m_1^i implies the regions that would go through the first path, and similarly the second mask corresponds to the second path. As depicted by the orange dash line, the two masks are then up-sampled by nearest-neighbor interpolation to the spatial resolution of z_i. Finally, the processed features of different paths are combined to generate the output features z_{i+1}. The whole process can be formulated as:

$$z_{i+1} = \sum_{j=1}^{M} f_j^i(f_S^i(z_i) \odot \bar{m}_j^i), \tag{4}$$

where $\bar{m}_j^i$ represents the up-sampled mask for the jth path. The $\odot$ denotes element-wise multiplication, and each channel of features $f_S^i(z_i)$ performs the same element-wise multiplication with the mask $\bar{m}_j^i$. Note that the regions with zero mask are ignored in each path, and hence the features of each region go through only one path. We illustrate the architecture details and demonstrate the effectiveness of Path-Restore-Mask in Section 4.4.

3.4 Training Algorithm

The training process is composed of two stages. In the first stage, we train the multi-path CNN with random policy of path selection as a good initialization. In the second stage, we train the pathfinder and the multi-path CNN

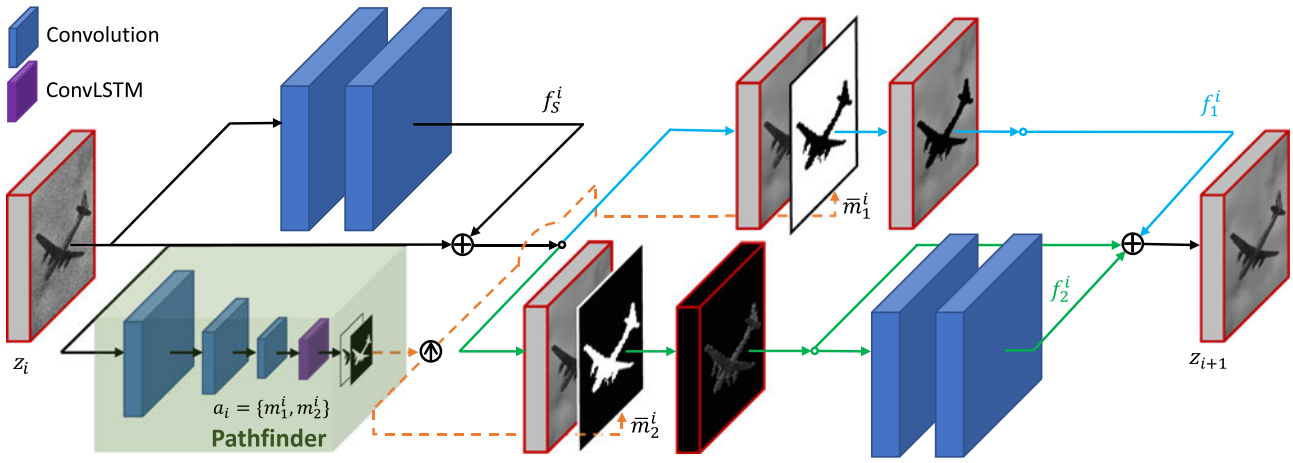


Fig. 3. The architecture of dynamic block and pathfinder in Path-Restore-Mask. For simplicity, we show a special case with two paths. The pathfinder, depicted at the bottom left corner, contains three strided convolutions and a ConvLSTM [45]. It predicts a policy mask with two channels, both of which are then up-sampled by nearest-neighbor interpolation to select eligible regions for the corresponding paths.

simultaneously, so that the two components are better associated and optimized.

Algorithm 1. REINFORCE (1 update)

Get a batch of K image pairs, $\{x^{(1)}, y^{(1)}\}, \ldots, \{x^{(K)}, y^{(K)}\}$
With policy π, derive $(s_1^{(k)}, a_1^{(k)}, r_1^{(k)}, \ldots, s_N^{(k)}, a_N^{(k)}, r_N^{(k)})$
Compute gradients using REINFORCE

$$\Delta\theta = \frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{N}\nabla_\theta \log \pi(a_i^{(k)}|s_i^{(k)};\theta)\left(\sum_{j=i}^{N} r_j^{(k)} - b^{(k)}\right)$$

Update parameters $\theta \leftarrow \theta + \beta\Delta\theta$

The First Stage. We train the multi-path CNN with randomly selected routes. The path-selection policy is a uniform distribution if not specified. The loss function consists of two parts. First, a final L2 loss constrains the output images to have a reasonable quality. Second, an intermediate loss enforces the states observed by the pathfinder at each dynamic block to be consistent. Specifically, the loss function is formulated as:

$$L = ||y - \hat{y}||_2^2 + \alpha \sum_{i=1}^{N} ||y - f_{\text{last}}(z_i)||_2^2, \tag{5}$$

where $\hat{y}$ and y denote the output image and the ground truth, respectively. The N represents the number of dynamic blocks. The last convolutional layer of Path-Restore is denoted by f_{last}, and α is the weight of the intermediate loss. If no intermediate loss is adopted, the features observed at different dynamic blocks may vary a lot from each other, hence making it more difficult for the pathfinder to learn a good dispatch policy.

The Second Stage. We train the pathfinder together with the multi-path CNN. In this stage, the multi-path CNN is trained using the loss function in Eq. (5) with $\alpha = 0$, i.e., intermediate loss does not take effect. Meanwhile, we train the pathfinder using the REINFORCE algorithm [47], as shown in Algorithm 1. The parameters of pathfinder and learning rate are denoted by θ and β, respectively. We adopt an individual baseline reward $b^{(k)}$ to reduce the variance of gradients and stabilize training. In particular, $b^{(k)}$ is the reward of the kth image when the bypass connection is selected in all dynamic blocks.

Implementation Details. As shown in Table 1, the threshold of difficulty L_0 and the reward penalty p are set based on the specific task. Generally it requires larger threshold and reward penalty when the distortions are more severe. For all tasks, the difficulty d is measured by L2 loss. In the first training stage, the coefficient of intermediate loss α is set as 0.1. Both training stages take 800k iterations. Training images are cropped into 63×63 patches and the batch size is 32. The learning rate is initially 2×10^{-4} and decayed by a half after every quarter of the training process. We use Adam [48] as the optimizer and implement our method on TensorFlow [49].

4 Experiments

In this section, we first clarify the general settings of network architecture and evaluation details. In Section 4.1, we present the results of real-world denoising. In Section 4.2, we demonstrate that our method performs well on blind Gaussian denoising. In Section 4.3, we further show the capability of Path-Restore to address a mixture of complex distortions. In Section 4.4, we verify the effectiveness of Path-Restore-Mask on real-world denoising. Finally, in Section 4.5, we offer discussions about the architecture of dynamic block, the difficulty-regulated reward, the effects of pathfinder, the number of network parameters and more comparisons.

Network Architecture. As shown in Table 1, we select different architectures for different restoration tasks. Specifically, we employ $M = 2$ (1 bypass, 1 denoising) paths for the

TABLE 1
Settings of Architecture and Reward for Different Tasks

Task	Architecture			Reward	
	N	M	C	Threshold(L_0)	Penalty(p)
Denoising	6	2	2	5×10^{-4}	8×10^{-6}
Mixed	5	4	4	0.01	4×10^{-5}

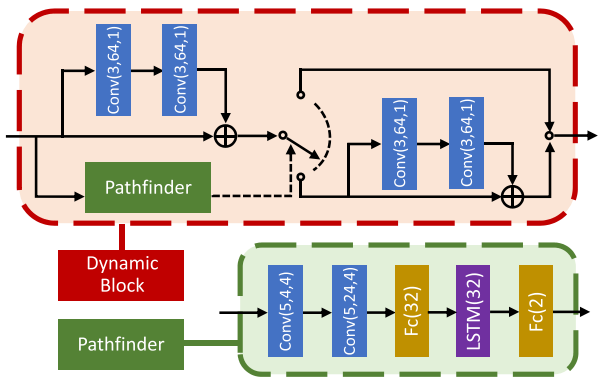


Fig. 4. The architecture of dynamic block and pathfinder for all the denoising tasks.

denoising tasks while $M = 4$ (1 bypass, 3 distortions) paths for addressing a mixed of three distortions. When the restoration task is more complex, we adopt more convolutional layers in the pathfinder. In particular, the pathfinder has $C = 2$ and $C = 4$ convolutional layers for denoising and addressing mixed distortions, respectively. As for the number of dynamic blocks, we use $N = 6$ for denoising and $N = 5$ for mixed distortions so that the network complexity is appropriate for a fair comparison with the baseline methods.

Evaluation Details. While testing Path-Restore, each image is split into 63×63 regions with a stride 53. After being processed by the multi-path CNN, all the regions are merged into a large image with overlapping pixels averaged. Following RL-Restore [27], we report the number of floating point operations (FLOPs) that are required to process a 63×63 region on average. Each multiplication or addition contributes one FLOP. We also record the GPU[2] or CPU[3] runtime. We report the time to process an image of size 512×512 if not specified.

4.1 Evaluation on Real-World Denoising

Architecture Details. We first conduct experiments on real-word denoising. The detailed architectures of dynamic block and the pathfinder are shown in Fig. 4. Conv(k, n, m) denotes a convolutional layer with a $k \times k$ kernel size, n filters and a stride of m. A fully-connected layer with output size n_o and an LSTM module with hidden size n_h are denoted by Fc(n_o) and LSTM(n_h), respectively.

Training and Testing Details. We evaluate our method on the Darmstadt Noise Dataset (DND) [2] and the Smartphone Image Denoising Dataset (SIDD) [52]. The DND contains 50 realistic high-resolution paired noisy and noise-free images for evaluation, yet it does not offer data for training. For DND benchmark, we adopt the same training data as CBDNet [9] for a fair comparison. In particular, we use BSD500 [54] and Waterloo [55] datasets to synthesize noisy images, and use the same noise model as CBDNet. A real dataset RENOIR [56] is also adopted for training as CBDNet. As for SIDD benchmark, we use SIDD Medium dataset for training. For non-blind denoising methods like BM3D [50] and FFDNet [51], we follow [25] to exploit Neat Image, a plug-in of Photoshop, for noise estimation.

TABLE 2
Results of Real-World Denoising on the Darmstadt Noise Dataset [2]

Method	PSNR / SSIM	FLOPs (G)	Time (s)
BM3D [50]	34.51 / 0.8507	-	11
FFDNet [51]	37.61 / 0.9415	1.74	0.051
CBDNet [9]	38.06 / 0.9421	6.94	0.19
N3Net [24]	38.32 / 0.9384	∼15.4	0.67
RIDNet [1]	39.26 / 0.9528	13.0	0.41
PRIDNet [25]	39.42 / 0.9528	3.65	0.077
Path-Restore	39.00 / 0.9542	5.60	0.15
Path-Restore-Ext	**39.72 / 0.9591**	22.6	0.42

The "∼" represents an estimation of FLOPs. The time is tested on the same GPU for a 512×512 image.

TABLE 3
Results of Real-World Denoising on the SIDD Dataset [52]

Method	PSNR	SSIM	Time (s/Mpixel)
BM3D [50]	25.65	0.685	27.4
CBDNet [9]	33.28	0.868	4.48
Proxy-opt. BM3D [53]	34.34	0.911	6.69
Path-Restore	**38.21**	**0.946**	**0.89**

The runtime is directly copied from the benchmark for reference.

Quantitative Results. The quantitative results on DND [24] are presented in Table 2. Besides PSNR and SSIM metrics, we also report the FLOPs and GPU runtime. Path-Restore is significantly better than the non-blind denoising methods BM3D [50] and FFDNet [51]. Using the same training dataset as CBDNet [9], Path-Restore improves the performance by nearly 1 dB with 19 percent fewer FLOPs and 27 percent faster speed on GPU. Compared with N3Net [2] that applies denoising to the raw image, Path-Restore attains better performance and runs faster. Our method is comparable to RIDNet [1] with only about 40 percent computation and runtime. PRIDNet [25] has higher PSNR and runs faster than our method, but its number of parameters is larger than ours by two orders of magnitude (74.2 million for PRIDNet and 0.9 million for ours). Path-Restore-Ext denotes an extended deeper model trained with more data. This extended model further improves 0.7 dB and currently achieves the state-of-the-art performance on the DND benchmark. The details are specified at the end of this section.

The quantitative comparisons on SIDD are shown in Table 3. It is observed that Path-Restore surpasses the state-of-the-art proxy-optimized BM3D [50] by a large margin. We report the GPU runtime of Path-Restore for a 1024×1024 input image (1 Mega pixels). For other methods, we directly copy the time recorded on the SIDD benchmark.[4] Path-Restore is the most efficient method among all the existing works on the SIDD benchmark.

Qualitative Results. Qualitative results are shown in Fig. 5. We do not present the results of N3Net [24] because it is applied in the raw domain. In Fig. 5, the top two input images are corrupted with severe noise, while the bottom two have moderate noise. It is observed that BM3D [50]

2. NVIDIA GeForce GTX TITAN X.
3. Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz.
4. https://www.eecs.yorku.ca/∼kamel/sidd/benchmark.php

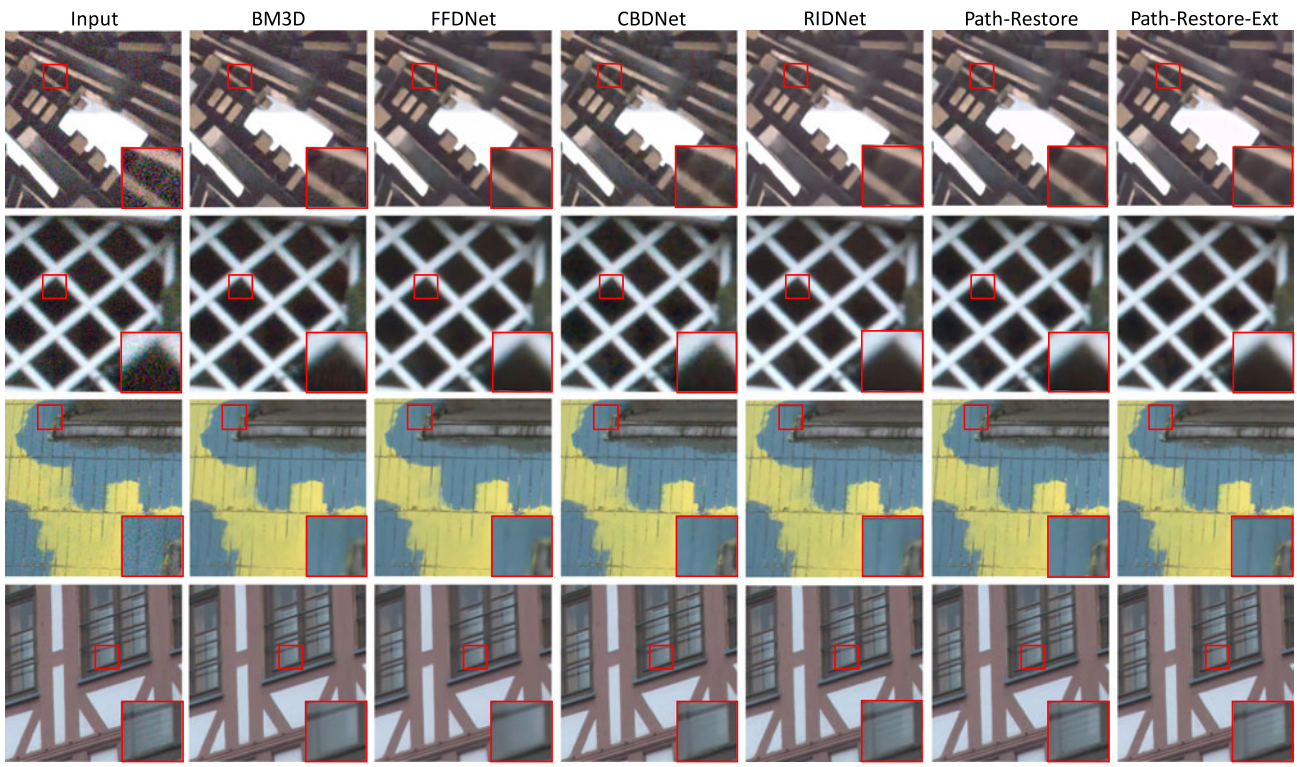


Fig. 5. Qualitative results on the Darmstadt Noise Dataset [2]. Path-Restore recovers clean results with sharp edges.

introduces artifacts for severe noise and yields over-smooth results for mild noise. FFDNet [51] could successfully remove severe noise, but the results are too smooth and some tiny edges are almost missing. CBDNet [9] fails to remove the severe noise in dim regions and tends to generate artifacts along edges. This may be caused by the incorrect noise estimation for real-world images. The results of RIDNet are comparable to ours. RIDNet [1] produces fewer artifacts while our method generates sharper edges. With a deeper architecture and more training data, Path-Restore-Ext further mitigates the artifacts along sharp edges (e.g., the highlighted region of the first image).

Path Selection. We present the policy of path selection in Fig. 6. The green color represents a short and simple path while the red color stands for a long and complex path. Path-Restore develops a reasonable policy for real-world denoising, using longer paths to process dark regions with severe noise while using shorter paths to process brighter regions with slight noise.

Path-Restore-Ext. To train the extended Path-Restore-Ext, we 1) adopt a deeper network, 2) use more training data, 3) reduce the reward penalty and 4) enlarge the patch size for training and testing. In particular, the network architecture is two times deeper than Path-Restore. We use more training data from the SIDD Small dataset [52]. Inspired by [59], we use unprocessing to get clear raw images with a similar distribution to those in DND [2]. Following [59], we add synthetic noise to the raw images. We then do raw processing for both clean and noisy raw images to get a pair of training images in sRGB domain. These images are mixed with the training data of Path-Restore, providing a larger dataset to train Path-Restore-Ext. We further reduce the reward penalty p from 8×10^{-6} to 2×10^{-7} in order to encourage longer paths and better performance. The training and testing patch

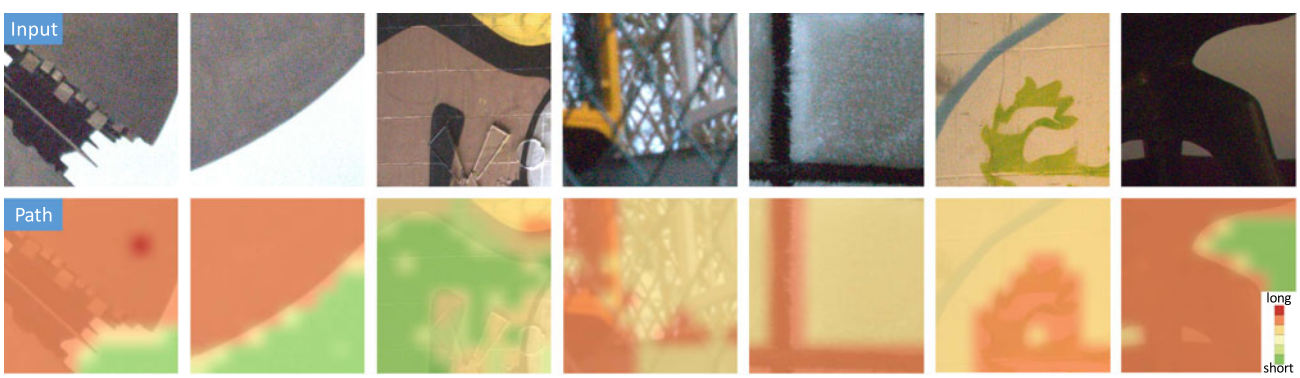


Fig. 6. The policy of path selection on DND [2]. The green color represents a short path while the red color represents a long path. Dark regions with severe noise are processed more than bright regions with slight noise.

TABLE 4
PSNR and Average FLOPs of Blind Gaussian Denoising on CBSD68 and DIV2K-T50 Datasets

Dataset	CBSD68 [57]						DIV2K-T50 [58]					
Noise	uniform			spatially variant			uniform			spatially variant		
	σ=10	σ=50	FLOPs	linear	peaks	FLOPs	σ=10	σ=50	FLOPs	linear	peaks	FLOPs
DnCNN [23]	36.07	27.96	5.31G	31.17	31.15	5.31G	37.32	29.64	5.31G	32.82	32.64	5.31G
Path-Restore	36.04	27.96	**4.22G**	31.18	31.15	**4.22G**	37.26	29.64	**4.20G**	32.83	32.64	**4.17G**

The unit of FLOPs is Giga ($\times 10^9$). Path-Restore is consistently 25 percent faster (in terms of FLOPs) than DnCNN with comparable performance on different noise settings.

Fig. 7. Qualitative results of spatially variant (type "peaks") Gaussian denoising. While most visual results are comparable, Path-Restore recovers textured regions better than DnCNN since these regions are processed with long paths.

size are increased from 63×63 to 96×96. While testing, the stride between adjacent patches is 80.

4.2 Evaluation on Blind Gaussian Denoising

Training and Testing Details. We further conduct experiments on a blind denoising task where the distortion is Gaussian noise with a wide range of standard deviation [0, 50]. The network architecture is the same as that for real-world denoising. The training dataset includes the first 750 images of DIV2K [58] training set (denoted by DIV2K-T750) and 400 images used in [60]. We choose DnCNN [23] as our baseline since it has similar number of parameters as our method (0.7 million for DnCNN and 0.9 million in total for ours). For fair comparison, we use the officially released codes to train DnCNN [23] with the same training dataset as ours. Following [23], we test our model on CBSD68[5] and we further do evaluation on the remaining 50 images in the DIV2K training set (denoted by DIV2K-T50). In addition to uniform Gaussian noise, we also conduct experiments on spatially variant Gaussian noise using the settings in FFDNet [51]. Specifically, "linear" denotes that the Gaussian noise level gradually increases from 0 to 50 from the left to the right side of an image. Another spatially variant noise, denoted by "peaks", is obtained by translating and scaling Gaussian distributions as in [51], and the noise level also ranges from 0 to 50.

Quantitative and Qualitative Results. We report the PSNR performance and average FLOPs in Table 4. For both uniform and spatially variant denoising on different datasets, Path-Restore is able to achieve about 25 percent speed-up (in terms of FLOPs) than DnCNN [23] while achieving comparable performance. The acceleration on DIV2K-T50 is slightly larger than that on CBSD68, because high-resolution images in DIV2K-T50 tend to have more smooth regions that can be restored in short network paths. When processing a 512×512 image, the CPU runtime of DnCNN is 3.16s, while the runtime of Path-Restore is 2.37s and 2.99s given $\sigma = 10$ and $\sigma = 50$, respectively. In practice, Path-Restore is about 20 percent faster than DnCNN on average.

Several qualitative results of spatially variant (type "peaks") Gaussian denoising are shown in Fig. 7. Although most of the restored images are visually comparable, Path-Restore achieves better results on some regions with fine-grained textures, since Path-Restore could select more complex paths to process hard examples with detailed textures.

Path Selection. We first visualize the path selection for uniform Gaussian denoising in Fig. 8. The input noisy images are shown in the first row, and the policy under $p = 8 \times 10^{-6}$ and $p = 5 \times 10^{-6}$ are presented in the second and third rows, respectively.

It is observed that a larger reward penalty p leads to a shorter path, as the pathfinder learns to avoid choosing overly complex paths that have very high reward penalty. Although the noise level is uniform within each image, the pathfinder learns a dispatch policy that depends on the image content. For example, the pathfinder focuses on processing the subject of an image (e.g., the airplane and penguin in the left two images) while assigning the smooth background to simple paths.

The dispatch policy for spatially variant (type "linear") Gaussian denoising is shown in Fig. 9. As the noise level gradually increases from left to right, the chosen paths become more and more complex. This demonstrates that the pathfinder learns a dispatch policy based on distortion level. Moreover, on each blue dash line, the noise level remains the same but the path selection is quite diverse, again demonstrating the pathfinder could select short paths

5. Color images of BSD68 dataset [57].

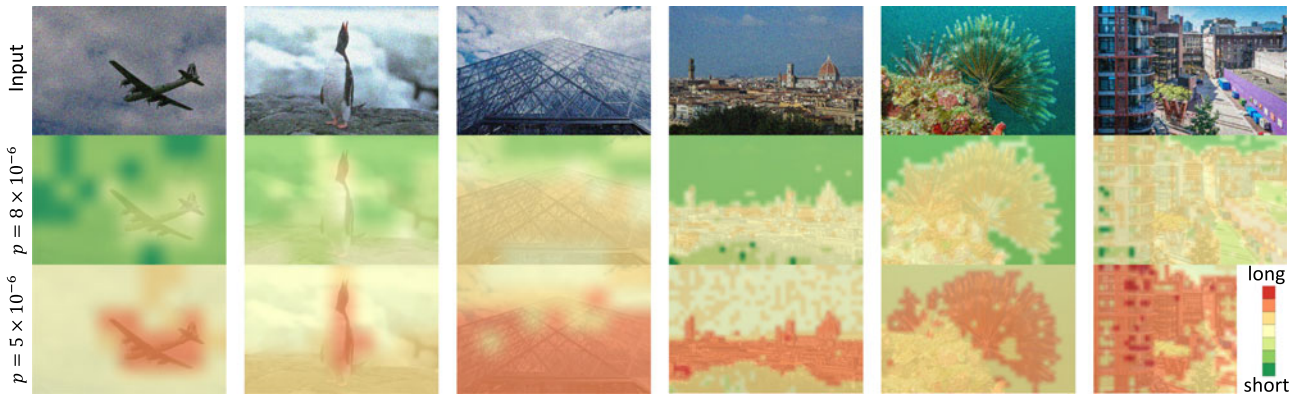


Fig. 8. The policy of path selection for uniform Gaussian denoising $\sigma = 30$. The pathfinder learns to select long paths for the objects with detailed textures while process the smooth background with short paths.

Fig. 9. The policy of path selection for spatially variant (type "linear") Gaussian denoising. The pathfinder learns a dispatch policy based on both the content and the distortion, i.e., short paths for smooth and clean regions while long paths for textured and noisy regions.

for smooth regions while choose long paths for textured regions.

4.3 Evaluation on Mixed Distortions

Training and Testing Details. We further evaluate our method on a complex restoration task as that in RL-Restore [27], where an image is corrupted by different levels of Gaussian blur, Gaussian noise and JPEG compression simultaneously. The network architecture is presented in Fig. 10, where there are four paths in each dynamic block. Following [27], we use DIV2K-T750 for training and DIV2K-T50 for testing. We report results not only on 63×63 sub-images as in [27] but also on the whole images with 2K resolution. To conduct a thorough comparison, we test RL-Restore in two ways for large images: 1) each 63×63 region is processed using a specific toolchain (denoted by RL-Restore-re), and 2) each 63×63 region votes for a toolchain, and the whole image is processed with a unified toolchain that wins the most votes (denoted by RL-Restore-im).

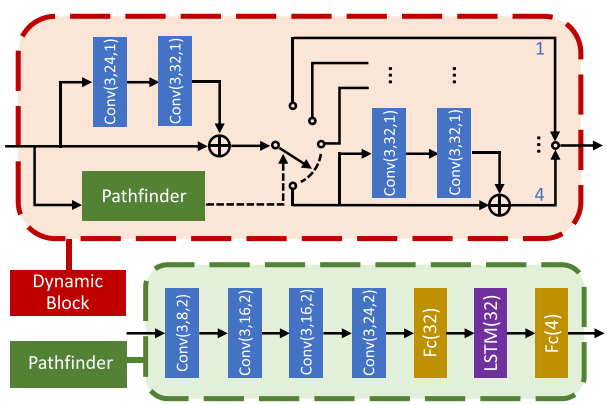


Fig. 10. The architecture of dynamic block and pathfinder for addressing mixed distortions.

In the first training stage, the random policy (initialization of path section) is a categorical distribution related to the specific combination of distortions. In particular, we classify each type of distortion (i.e., Gaussian blur, Gaussian noise and JPEG compression) into 10 levels (from 1 to 10) in the same way as RL-Restore [27]. The degradation levels of blur, noise and JPEG are denoted by l_b, l_n and l_j, respectively. The parameters of categorical distribution can be written as $\sigma(\boldsymbol{l})$, where $\sigma(\cdot)$ is the Softmax function and $\boldsymbol{l} = (1, 0.2l_b - 0.1, 0.2l_n - 0.1, 0.2l_j - 0.1)$. Given this prior distribution, if an image is mainly corrupted by one type of distortion, it will be assigned to the corresponding path with a high probability. If none of these three types of distortion is severe, the input image will be dispatched to the first path (bypass connection).

Quantitative Results. The quantitative results are shown in Table 5. The results on 63×63 sub-images show that Path-Restore achieves a slightly better performance compared with RL-Restore. While testing on large images with 2K resolution, our method shows more apparent gain (0.2 dB higher) than RL-Restore-re and RL-Restore-im with just a tiny increase in computational complexity (FLOPs). The CPU runtime of RL-Restore-re, RL-Restore-im and Path-Restore are 1.34s, 1.09s and 0.954s, respectively. These

TABLE 5
Results of Addressing Mixed Distortions on DIV2K-T50 [58] Compared With Two Variants of RL-Restore [27]

Image size	63x63 image	2K image	
Metric	PSNR / SSIM	PSNR / SSIM	FLOPs(G)
RL-Restore-re	-	25.61 / 0.8264	1.34
RL-Restore-im	26.45 / 0.5587	25.55 / 0.8251	**0.948**
Path-Restore	**26.48 / 0.5667**	**25.81 / 0.8327**	1.38

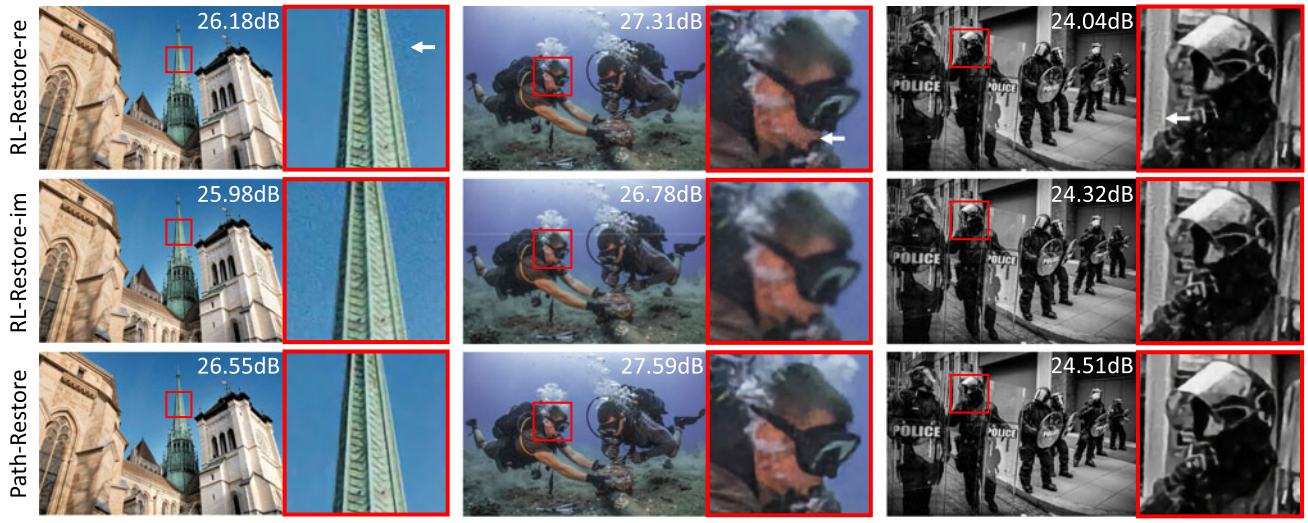


Fig. 11. Qualitative results of addressing mixed distortions. RL-Restore [27] tends to generate artifacts across different regions (see the white arrow), while Path-Restore is able to handle diverse distortions and produce more spatially consistent results (zoom in for best view).

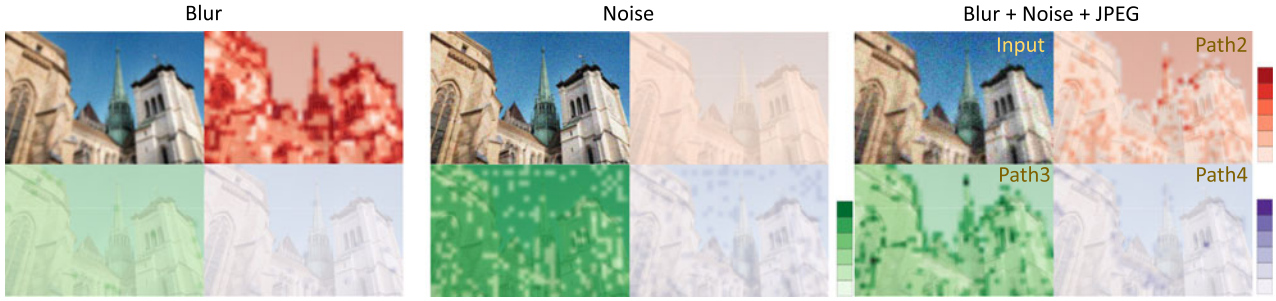


Fig. 12. The policy of path selection for mixed distortions (4 paths including 1 bypass connection in each dynamic block). The light color represents that a small number of the corresponding paths are selected, while the dark color shows that a large number of the corresponding paths are chosen.

results show that Path-Restore runs faster than RL-Restore in practice. The reason is that RL-Restore needs extra time to switch among different models while this overhead is not needed in the unified framework – Path-Restore.

Qualitative Results. As shown in Fig. 11, RL-Restore-re tends to produce inconsistent boundaries and appearance between adjacent 63×63 regions, as each region may be processed by entirely different models. RL-Restore-im fails to address severe noise or blur in some regions because only a single toolchain can be selected for the whole image. On the contrary, thanks to the dynamic blocks in a unified network, Path-Restore not only removes the distortions in different regions but also yields spatially consistent results.

Path Selection. We show the results of path selection in Fig. 12. The number of the 2nd, 3rd and 4th path selected are shown in the red, green and purple heat maps, respectively. For a blurry image with slight noise and compression, the pathfinder dispatches almost all image regions to the 2nd path, namely, more blurry images are assigned to the 2nd path compared with other paths, during the first training stage. Smooth regions are routed to the bypass path even when the blur is severe, because blur (without other types of distortion) does not affect the looking of smooth regions. Given an image with severe noise, the pathfinder routes most regions to the 3rd path, and even smooth regions require long paths to restore when the noise level is high. When multiple distortions co-exist, the path selection depends on both the image content and the combination of distortions. Interestingly, the 4th path that has been assigned many JPEG images in the first training stage, is frequently selected at the first dynamic block yet seldom chosen at successive blocks. Perhaps the pathfinder learns that addressing JPEG compression at the beginning is a good restoration policy.

4.4 Evaluation of Path-Restore-Mask

Implementation Details. In Path-Restore-Mask, the architectures of the shared path and dynamic paths are the same as those of Path-Restore (in Section 4.1). As for the pathfinder, the convolutional layers are "Conv(5,16,4)–Conv(5,8,4)–Conv(3,2,2)", where the parameters denote kernel size, number of filters and stride, respectively. The hidden state of ConvLSTM has two channels. Note that one pixel on the mask represents a 32×32 region on the features. The training data is the same as Path-Restore, as specified in Section 4.1. We evaluate the performance on DND [2] and SIDD [52] benchmarks for real-world denoising.

Quantitative Results. Quantitative comparisons between Path-Restore-Mask and Path-Restore are shown in Table 6. On both DND and SIDD benchmarks, Path-Restore-Mask consistently outperforms Path-Restore by more than 0.2 dB. The better performance does not come at a price of higher complexity. It is observed that the FLOPs of Path-Restore-Mask are even 10 percent fewer than Path-Restore on both benchmarks, and the GPU runtime of Path-Restore-Mask is

TABLE 6
Quantitative Evaluation of Path-Restore-Mask on the Darmstadt Noise Dataset [2] and the Smartphone Image Denoising Dataset [52]

Dataset	DND [2]				SIDD [52]			
Metric	PSNR	SSIM	FLOPs (G)	Time (s)	PSNR	SSIM	FLOPs (G)	Time (s/Mpixel)
Path-Restore	39.00	0.9542	5.60	0.149	38.21	0.946	7.13	0.89
Path-Restore-Mask	**39.18**	**0.9563**	**4.77**	**0.146**	**38.42**	**0.949**	**6.34**	**0.76**

also shorter. Processing a large image at image level is more efficient than at patch level. When treating different regions jointly, the features of adjacent regions can be exploited, and computation on overlapping regions is no longer required.

Qualitative Results and Path Selection. Qualitative comparisons are shown in Fig. 13. We observe that most of the visual results are comparable, but Path-Restore-Mask yields fewer artifacts in the regions with severe noise (see the highlighted regions in the red box). The path-selection results indicate that Path-Restore-Mask selects more diverse paths with spatially smoother policy. Specifically, on the path-selection map of Path-Restore, the checkerboard-like policy occasionally appears, where one region is dispatched to a different path compared with that of surrounding regions. On the contrary, such a scenario seldom occurs for Path-Restore-Mask, because the pathfinder could observe the adjacent features while selecting the current path.

4.5 Discussions

Architecture of Dynamic Block. We investigate the number of paths in each dynamic block. For the task to address mixed distortions, the number of paths is originally set as 4 since there are 3 paths for various distortions and 1 bypass connection. As shown in Table 7, we alternatively use 2 paths to address the same task. It is observed that the FLOPs increase by nearly 10 percent when achieving comparable performance with the original setting.

Difficulty-Regulated Reward. A performance-complexity trade-off can be achieved by adjusting the reward penalty p while training. The trade-off for real-world denoising on the DND benchmark [2] is shown in Fig. 14, where the red points represent Path-Restore with different architecture and reward penalty while the blue points represent other methods. Path-Restore achieves a better PSNR-FLOPs trade-off compared with other methods when tuning the reward penalty. The trade-off is reasonable because a smaller reward penalty encourages the pathfinder to select a longer path for each region, resulting in better performance. We conduct a similar study on Gaussian denoising. As shown in Fig. 15, reducing p gradually from $p = 8 \times 10^{-6}$ to 3×10^{-6}, the blue curve depicts that PSNR and FLOPs both increase as the reward penalty decreases. We also adjust the depth of DnCNN to achieve the trade-off between PSNR and FLOPs, as shown in the green curve of Fig. 15. We observe that Path-Restore is consistently better than DnCNN by 0.1 dB with the same FLOPs. When achieving the same performance, our method is nearly 30 percent faster than DnCNN.

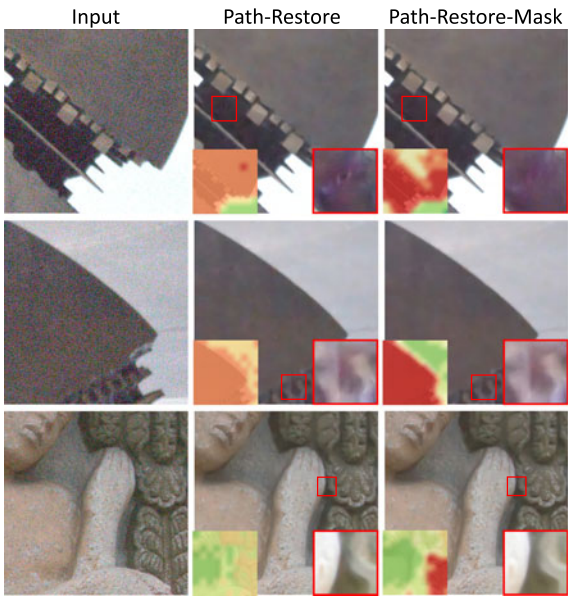


Fig. 13. Qualitative comparisons between Path-Restore-Mask and Path-Restore. The brightness and contrast of the highlighted parts are increased for better view. The path-selection map is presented in the bottom left corner of each image.

TABLE 7
Ablation Study on the Number of Paths in a Dynamic Block

Dataset	DIV2K-T50 [58]			
	63×63 sub-image		2K image	
Metric	PSNR	FLOPs (G)	PSNR	FLOPs (G)
4 paths	**26.48**	**1.00**	**25.81**	**1.38**
2 paths	26.46	1.09	25.80	1.48

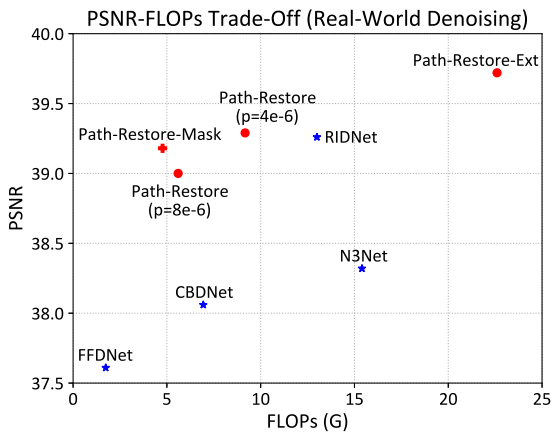


Fig. 14. Performance-complexity trade-off for real-world denoising on the DND benchmark [2].

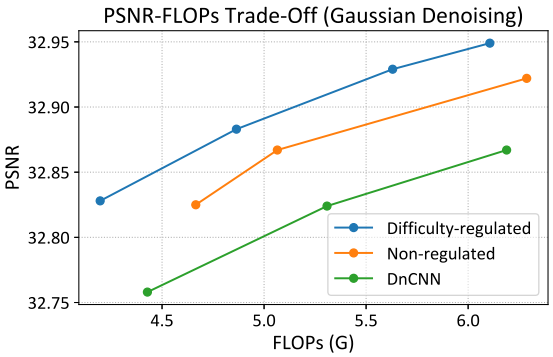


Fig. 15. Performance-complexity trade-off versus non-regulated reward and DnCNN for spatially variant (type "linear") Gaussian denoising on DIV2K-T50 [58].

Fig. 16. Path selection of regulated and non-regulated reward.

TABLE 8
Ablation Study on Training Datasets and the Pathfinder

Dataset	DND [2]		SIDD [52]	
Metric	PSNR	Time (s)	PSNR	Time (s)
Path-Restore-Mask	39.18	0.146	38.42	0.76
+ data of RIDNet [1]	39.37	0.143	38.50	0.74
- pathfinder	39.37	0.199	38.23	0.86

"+ data of RIDNet" denotes using the same training data as RIDNet. "- pathfinder" also adopts the data of RIDNet, and the pathfinder is removed.

We then study the impact of "difficulty" in the proposed reward function. In particular, we set difficulty $d \equiv 1$ in Eq. (3) to form a non-regulated reward. As shown in Fig. 15, the orange curve is consistently below the blue curve, indicating that the proposed difficulty-regulated reward helps Path-Restore achieve a better performance-complexity trade-off. We further present the path-selection policy of different rewards in Fig. 16. Compared with non-regulated reward, our difficulty-regulated reward saves more computations when processing easy regions (e.g., region in the red box), despite using a longer path to process hard regions (e.g., region in the blue box).

Training Datasets. The selection of training data is important for real-world denoising. In the aforementioned experiments, we adopt the same training data as CBDNet [9] when evaluating our method on the DND [2] benchmark, and only SIDD [52] training dataset is used for the evaluation on SIDD benchmark. To investigate the effects of training datasets, we further use the same training data as RIDNet [1] for the evaluation on both DND and SIDD. As shown in Table 8, Path-Restore-Mask represents our results with the original training data, and "+ data of RIDNet" denotes using the same training data as RIDNet. It is observed that the inference time almost remains the same. The PSNR is improved by 0.19 dB and 0.08 dB on DND and SIDD, respectively. The PSNR improvement is reasonable as the training dataset of RIDNet is larger than that of CBDNet.

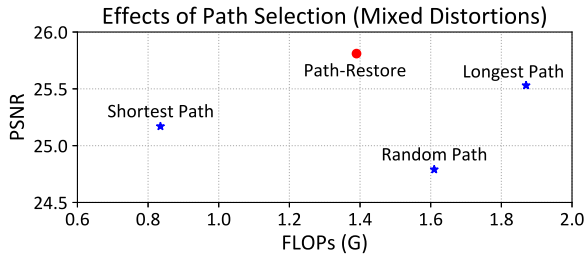


Fig. 17. Effects of path selection for addressing mixed distortions.

Fig. 18. Qualitative comparisons of path selection for addressing mixed distortions (zoom in for best view).

The Effects of Pathfinder. The pathfinder is an essential module in our framework. In Table 8, "- pathfinder" denotes a Path-Restore-Mask model without the pathfinder. In this case all image regions are processed by the same architecture. Note that we also use the training dataset of RIDNet [1] to train this model. On DND [2] benchmark, removing the pathfinder results in the same PSNR with an increase of 39 percent in runtime. On SIDD [52] benchmark, without the pathfinder, the PSNR decreases by 0.19 dB while the runtime increases by 13 percent.

We further evaluate the effects of pathfinder for addressing mixed distortions on DIV2K-T50 dataset [58]. We manually select the shortest path and one of the longest paths (the third path in each dynamic block), denoted by "Shortest Path" and "Longest Path", respectively. A "Random path" dispatch policy with uniform distribution is also used. As shown in Fig. 17, with an effective pathfinder, Path-Restore outperforms all of the above policies. A random path selection is worse than the shortest path, indicating that learning path selection is non-trivial. Qualitative results are presented in Fig. 18. Compared to manually designed dispatch policies, Path-Restore yields clearer images with fewer artifacts.

Parameters. We summarize the number of parameters of each method in Table 9. The parameters of all paths are included. On the denoising task, Path-Restore has slightly more parameters than FFDNet, yet has apparently fewer parameters than other methods. Several approaches like CBDNet and PRIDNet adopt down-

TABLE 9
Total Parameters of Each Method on Different Tasks

Task	Method	Parameters ($\times 10^5$)
Denoising	FFDNet [51]	8.52
	CBDNet [9]	67.9
	N3Net [24]	37.1
	RIDNet [1]	16.4
	PRIDNet [25]	742
	Path-Restore	9.19
Mixed Distortions	RL-Restore [27]	5.22
	Path-Restore	3.79

TABLE 10
Quantitative Comparisons to Category-Specific Denoising [26] on CBSD68 Dataset With Gaussian Noise $\sigma = 30$

Method	PSNR	SSIM	Time (s)
Anwar *et al.* [26]	25.06	0.6863	2.77×10^3
Path-Restore	30.75	0.8742	2.67

sampling for acceleration. However, it comes at a price of increasing parameters because the number of channels has to be expanded in a smaller scale. In particular, the number of parameters for PRIDNet is 140 times larger than ours. For addressing mixed distortions, Path-Restore has fewer parameters than RL-Restore. RL-Restore offers a toolbox that contains 12 task-specific CNNs. Although each CNN is light-weight, the total number of parameters is large.

Comparisons to Category-Specific Denoising. Anwar *et al.* [26] propose a denoising algorithm that separately processes different image regions. Specifically, it applies internal denoising to smooth regions while conducts category-specific external denoising for textured regions. We evaluate this method on CBSD68 dataset with Gaussian noise level $\sigma = 30$. The external dataset is the same as ours for Gaussian denoising.

Quantitative results are shown in Table 10. The metrics are calculated on grayscale images. Our performance and efficiency significantly surpass that of [26] – more than 5 dB higher in PSNR and 1,000 times faster. As Anwar *et al.* [26] leverage PatchMatch algorithm [61] for external denoising, the computational cost is very high, especially for high-resolution images. Although the category-specific method works well on images that contain regular patterns such as face and text, it is not as effective as learning-based methods for natural images where similar patches cannot be always found.

Qualitative results are presented in Fig. 19. Path-Restore produces clear and sharp results while category-specific denoising [26] yields blurry output. Interestingly, the denoising map of [26] is similar to our path-selection map. For category-specific denoising, the textured foreground is processed by external denoising while the smooth background is addressed by internal denoising. The criterion of whether a patch belongs to textured region is manually defined. As for Path-Restore, driven by a difficulty-regulated reward, the pathfinder learns to allocate textured and smooth regions to long and short paths, respectively. It demonstrates that the learned dispatch policy in our method conforms to human prior.

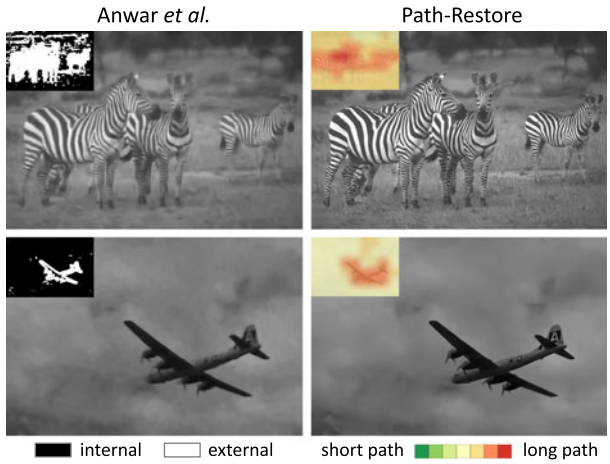


Fig. 19. Qualitative comparisons to category-specific denoising [26].

5 Conclusion

We have devised a new framework for image restoration with low computational cost. Combining reinforcement learning and deep learning, we propose Path-Restore that enables path selection for each image region. Specifically, Path-Restore is composed of a multi-path CNN and a pathfinder. The multi-path CNN offers several paths to address different types of distortions, and the pathfinder is able to find the optimal path to efficiently restore each region. To learn a reasonable dispatch policy, we propose a difficulty-regulated reward that encourages the pathfinder to select short paths for the regions that are easy to restore. We further investigate two mechanisms to independently and jointly process different image regions. Path-Restore achieves comparable or superior performance to existing methods on various restoration tasks with much fewer computations. Our method is especially effective for real-world denoising, where the noise distribution is diverse across different image regions. The idea of region-based path selection can also be applied to other tasks where each image region should be treated differently.

Acknowledgments

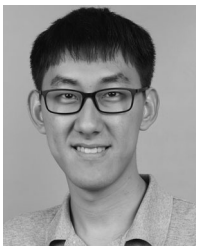
This work was partially supported by the Collaborative Research grant from SenseTime Group CUHK Agreement No. TS1610626 and No. TS1712093, Singapore MOE AcRF Tier 1 (2018-T1-002-056), NTU NAP. This study is supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s), in part by the National Natural Science Foundation of China under Grant 61906184, the Joint Lab of CAS-HK.

References

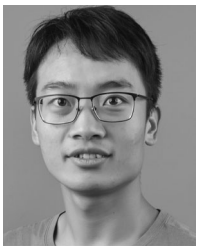
[1] S. Anwar and N. Barnes, "Real image denoising with feature attention," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 3155–3164.
[2] T. Plotz and S. Roth, "Benchmarking denoising algorithms with real photographs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1586–1595.
[3] R. Timofte, S. Gu, J. Wu, L. V. Gool, L. Zhang *et al.*, "NTIRE 2018 challenge on single image super-resolution: Methods and results," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 965–96511.

[4] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, Art. no. 4.
[5] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, Art. no. 3.
[6] S. Lefkimmiatis, "Non-local color image denoising with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.
[7] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2009, pp. 769–776.
[8] J. Chen, J. Chen, H. Chao, and M. Yang, "Image blind denoising with generative adversarial network based noise modeling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3155–3164.
[9] S. Guo, Z. Yan, K. Zhang, W. Zuo, and L. Zhang, "Toward convolutional blind denoising of real photographs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1712–1722.
[10] L. Xu, J. S. J. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Proc. Advances Neural Inf. Process. Syst.*, 2014, pp. 1790–1798.
[11] J. Sun, W. Cao, Z. Xu, and J. Ponce, "Learning a convolutional neural network for non-uniform motion blur removal," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 769–777.
[12] S. Nah, T. H. Kim, and K. M. Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 257–265.
[13] C. Dong, Y. Deng, C. Change Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 576–584.
[14] Z. Wang, D. Liu, S. Chang, Q. Ling, Y. Yang, and T. S. Huang, "D3: Deep dual-domain based fast restoration of JPEG-compressed images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2764–2772.
[15] J. Guo and H. Chao, "Building dual-domain representations for compression artifacts reduction," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 628–644.
[16] J. Guo and H. Chao, "One-to-many network for visually pleasing compression artifacts reduction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3038–3047.
[17] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.
[18] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1646–1654.
[19] J. Kim, J. K. Lee, and K. M. Lee, "Deeply-recursive convolutional network for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1637–1645.
[20] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2790–2798.
[21] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 105–114.
[22] X. Wang *et al.*, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2018, pp. 63–79.
[23] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.
[24] T. Plötz and S. Roth, "Neural nearest neighbors networks," in *Proc. Advances Neural Inf. Process. Syst.*, 2018, pp. 1087–1098.
[25] Y. Zhao, Z. Jiang, A. Men, and G. Ju, "Pyramid real image denoising network," in *Proc. IEEE Vis. Commun. Image Process.*, 2019, pp. 1–4.
[26] S. Anwar, C. P. Huynh, and F. Porikli, "Combined internal and external category-specific image denoising." in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 71.1–71.12.
[27] K. Yu, C. Dong, L. Lin, and C. C. Loy, "Crafting a toolchain for image restoration by deep reinforcement learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2443–2452.
[28] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *CoRR*, vol. abs/1308.3432, 2013.
[29] M. Figurnov *et al.*, "Spatially adaptive computation time for residual networks." in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, Art. no. 7.
[30] Z. Wu *et al.*, "BlockDrop: Dynamic inference paths in residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8817–8826.
[31] X. Wang, F. Yu, Z.-Y. Dou, and J. E. Gonzalez, "SkipNet: Learning dynamic routing in convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 409–424.
[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[33] C. Rosenbaum, T. Klinger, and M. Riemer, "Routing networks: Adaptive selection of non-linear functions for multi-task learning," in *Proc. Int. Conf. Learn. Representations*, 2018.
[34] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[35] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3215–3222.
[36] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
[37] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.
[38] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.
[39] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2423–2432.
[40] Y. Hu, H. He, C. Xu, B. Wang, and S. Lin, "Exposure: A white-box photo post-processing framework," *ACM Trans. Graphics*, vol. 37, no. 2, pp. 1–17, 2018.
[41] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proc. Int. Conf. Learn. Representations*, 2017.
[42] A. Veit and S. Belongie, "Convolutional networks with adaptive inference graphs," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 3–18.
[43] S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: stochastic neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019.
[44] B. Wu *et al.*, "FBNET: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10726–10734.
[45] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and ?re 3j?W.-C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. Advances Neural Inf. Process. Syst.*, 2015, pp. 802–810.
[46] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 694–711.
[47] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 229–256, 1992.
[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015..
[49] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symp. Operating Syst. Des. Implementation*, 2016, pp. 265–283.
[50] K. Dabov, A. Foi, V. Katkovnik, and K. O. Egiazarian, "Color image denoising via sparse 3D collaborative filtering with grouping constraint in luminance-chrominance space," in *Proc. IEEE Int. Conf. Image Process.*, 2007, pp. 313–316.
[51] K. Zhang, W. Zuo, and L. Zhang, "FFDNet: Toward a fast and flexible solution for CNN based image denoising," *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4608–4622, Sep. 2018.
[52] A. Abdelhamed, S. Lin, and M. S. Brown, "A high-quality denoising dataset for smartphone cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1692–1700.
[53] E. Tseng *et al.*, "Hyperparameter optimization in black-box image processing using differentiable proxies," *ACM Trans. Graph.*, vol. 38, no. 4, 2019, Art. no. 27.
[54] 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 *Proc. IEEE Int. Conf. Comput. Vis.*, 2001, pp. 416–423.

[55] K. Ma *et al.*, "Waterloo exploration database: New challenges for image quality assessment models," *IEEE Trans. Image Process.*, vol. 26, no. 2, pp. 1004–1016, Feb. 2017.
[56] J. Anaya and A. Barbu, "RENOIR–A dataset for real low-light image noise reduction," *J. Vis. Commun. Image Representation*, vol. 51, pp. 144–154, 2018.
[57] S. Roth and M. J. Black, "Fields of experts," *Int. J. Comput. Vis.*, vol. 82, no. 2, 2009, Art. no. 205.
[58] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, Art. no. 2.
[59] T. Brooks, B. Mildenhall, T. Xue, J. Chen, D. Sharlet, and J. T. Barron, "Unprocessing images for learned raw denoising," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1028–11037.
[60] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1256–1272, Jun. 2017.
[61] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, no. 3, 2009, Art. no. 24.



Ke Yu received the BEng degree from the Department of Electronic Engineering, Tsinghua University, in 2016. He is currently working toward the PhD degree at the Department of Information Engineering, The Chinese University of Hong Kong. He was selected as an outstanding reviewer in CVPR 2019. He won the first place in several international super-resolution challenges including NTIRE2019 and PIRM2018. His research interests include image restoration, super-resolution and reinforcement learning.



Xintao Wang received the PhD degree from the Department of Information Engineering, The Chinese University of Hong Kong, in 2020. He is currently a researcher with Applied Research Center (ARC), Tencent PCG. He was selected as an outstanding reviewer in CVPR 2019 and an outstanding reviewer (honorable mention) in BMVC 2019. He won the first place in several international super-resolution challenges including NTIRE2019, NTIRE2018, and PIRM2018. His research interests focus on low-level vision problems, including super-resolution, image and video restoration.

Chao Dong received the PhD degree from the Chinese University of Hong Kong, in 2016. He is currently an associate professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Science. In 2014, he first introduced deep learning method – SRCNN into the super-resolution field. This seminal work was chosen as one of the top ten "Most Popular Articles" of TPAMI in 2016. His team has won several championships in international challenges – NTIRE2018, PIRM2018, NTIRE2019, NTIRE2020 and AIM2020. He worked in SenseTime from 2016 to 2018, as the team leader of Super-Resolution Group. His current research interests focus on low-level vision problems, such as image/video super-resolution, denoising, and enhancement.

Xiaoou Tang (Fellow, IEEE) received the BS degree from the University of Science and Technology of China, Hefei, in 1990, the MS degree from the University of Rochester, New York, in 1991, and the PhD degree from the Massachusetts Institute of Technology, Cambridge, in 1996. He is a professor with the Department of Information Engineering, The Chinese University of Hong Kong. He worked as a group manager of the Visual Computing Group at the Microsoft Research Asia, from 2005 to 2008. His research interests include computer vision, pattern recognition and video processing. He received the Best Paper Award at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2009 and Outstanding Student Paper Award at the AAAI 2015. He was a program chair of the IEEE International Conference on Computer Vision (ICCV) 2009, a General Chair of ICCV 2019, and served as an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and editor-in-chief of the *International Journal of Computer Vision*.

Chen Change Loy (Senior Member, IEEE) received the PhD degree in computer science from the Queen Mary University of London, in 2010. He is an associate professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He is also an adjunct associate professor with the Chinese University of Hong Kong. Prior to joining NTU, he served as a research assistant professor with the MMLab of The Chinese University of Hong Kong, from 2013 to 2018. He was a postdoctoral researcher with Queen Mary University of London and Vision Semantics Limited, from 2010 to 2013. He serves as an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and the *International Journal of Computer Vision*. He also serves/served as an area chair of major conferences such as ICCV 2021, CVPR 2021, CVPR 2019, and ECCV 2018. His research interests include image/video restoration and enhancement, generative tasks, and representation learning.