Crafting a Toolchain for Image Restoration by Deep Reinforcement Learning

Ke Yu [1] Chao Dong[2] Liang Lin[2,3] Chen Change Loy[1]
[1]CUHK - SenseTime Joint Lab, The Chinese University of Hong Kong
[2]SenseTime Research [3]Sun Yat-sen University
{yk017, ccloy}@ie.cuhk.edu.hk {dongchao, linliang}@sensetime.com

Abstract

We investigate a novel approach for image restoration by reinforcement learning. Unlike existing studies that mostly train a single large network for a specialized task, we prepare a toolbox consisting of small-scale convolutional networks of different complexities and specialized in different tasks. Our method, RL-Restore, then learns a policy to select appropriate tools from the toolbox to progressively restore the quality of a corrupted image. We formulate a step-wise reward function proportional to how well the image is restored at each step to learn the action policy. We also devise a joint learning scheme to train the agent and tools for better performance in handling uncertainty. In comparison to conventional human-designed networks, RL-Restore is capable of restoring images corrupted with complex and unknown distortions in a more parameter-efficient manner using the dynamically formed toolchain[1].

1. Introduction

Deep convolutional neural network (CNN) has achieved immense success, not only in high-level vision tasks, but also low-level vision tasks such as deblurring [31, 35, 42], denoising [6, 24], JPEG artifacts reduction [7, 9, 41] and super-resolution [8, 19, 17, 37, 39]. In particular, good performance and fast testing speed are demonstrated over conventional model-based optimization methods.

Owing to the discriminative nature of CNN, most of these models are trained to handle a specialized low-level vision task. In JPEG artifacts reduction [7], for instance, different networks for different compression qualities have been designed to achieve satisfactory restoration. In the case of super-resolution [8], it is common to have different networks to handle different scaling factors. Some recent studies [10, 37] have shown the possibility of handling multiple distortion types or coping with different levels of degradation at once using CNN. Nevertheless, this usually comes with the expenses of using much deeper networks. In addition, such networks process all images with the same structure, despite some of which are inherently less difficult and can be restored in a cheaper way.

In this paper, we explore the possibility of having some smaller-scale but specialized CNNs to solve a harder restoration task collaboratively. Our idea departs from the current philosophy that one would need a large-capacity CNN to solve a complex restoration task. Instead, we wish to have a set of tools (based on small CNNs) and learn to use them adaptively for solving the task at hand. The aforementioned idea could provide new insights how CNN can be used for solving real-world restoration tasks, of which images are potentially contaminated with a mix of distortions, *e.g.*, blurring, noise and blockiness after several stages of processing. Moreover, the new approach may lead to parameter-efficient restoration in comparison to existing CNN-based models. In particular, tools of different complexities can be selected based on the severity of distortion.

Towards this goal, we present a framework that treats image restoration as a decision making process by which an agent would adaptively select a sequence of tools to progressively refine an image, and the agent may choose to stop if the restored quality is deemed satisfactory. In our framework, we prepare a number of light-weight CNNs with different complexities. They are task-specific aiming to handle different types of restoration assignments including deblurring, denoising, or JPEG artifacts reduction. Choosing the order of tools is formulated in a reinforcement learning (RL) framework. An agent learns to decide the next best tool to select by analyzing the content of the restored image in the current step and observing the last action chosen. Rewards are accumulated when the agent improves the quality of the input image.

We refer to the proposed framework as *RL-Restore*. We summarize our **contributions** as follows:

1) We present a new attempt to address image restoration in a reinforcement learning framework. Unlike existing methods that deploy a single and potentially large network structure, *RL-Restore* enjoys the flexibility of using tools of dif-

[1]Codes and data are available at http://mmlab.ie.cuhk.edu.hk/projects/RL-Restore/

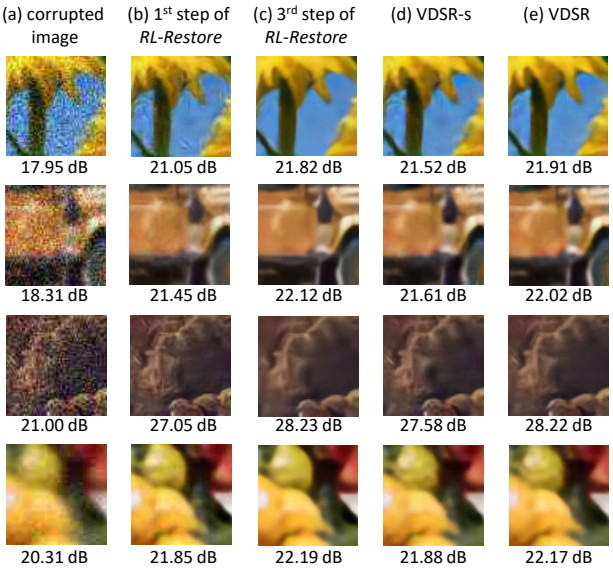

Figure 1. (a) shows images corrupted by complex distortions. (b-c) depict some chosen steps of the decision process to restore an image by *RL-Restore*. At each step, a specific tool is selected by the agent to improve the image quality. (d-e) are CNN-based results, where (d) has comparable parameters to *RL-Restore* while (e) has twice more. PSNR values are presented for better comparison.

ferent capacities to achieve the desired restoration.

2) We propose a joint learning scheme to train the agent and tools simultaneously so that the framework possesses better capability in coping with new and unknown artifacts emerged in the mid of processing.

3) We show that the dynamically formed toolchain performs competitively against strong human-designed networks with less computational complexity. Our approach can cope with unseen distortions to certain extent. Interestingly, our approach is more transparent than existing methods as it can reveal how complicated distortions could be removed step by step using different tools.

Figure 1(b-c) illustrate a learned policy to restore an image corrupted by multiple distortions, where image quality is refined step-by-step. The results of two baseline CNN models are depicted in Figure 1(d-e), where (d) has similar number of parameters as ours (agent + tools applied), while (e) has twice more. As we will further present in the experimental section, *RL-Restore* is superior to CNN approaches given similar complexity and it requires 82.2% fewer computations to achieve the same performance as a single large CNN.

2. Related Work

CNN for Image Restoration. Image restoration is an extensively studied topic that aims at estimating the clear/original image from a corrupted/noisy observation. Convolutional neural networks (CNN) based methods have demonstrated outstanding performance in various image restoration tasks. Most of these studies train a single network specializing on the task at hand, *e.g.*, deblurring [31, 35, 42], denoising [6, 24], JPEG artifacts reduction [7, 9, 41] and super-resolution [8, 17, 19, 20, 22, 36, 37, 39]. Our work offers an alternative that is more parameter efficient yet adaptive to the form of distortions.

There are several pioneering studies that deal with multiple degradations simultaneously. By developing a 20-layer deep CNN, Kim *et al*. [19] use a single model to handle multi-scale image super-resolution. Guo *et al*. [10] build a one-to-many network that can handle images with different levels of compression artifacts. Zhang *et al*. [44] propose a 20-layer deep CNN to address multiple restoration tasks simultaneously, including image denoising, JPEG artifacts reduction and super-resolution. None of these studies considers mixed distortion, where a single image is affected by multiple distortions. Different from the aforementioned works, we are interested to explore if smaller-scale CNNs of 3 to 8 layers could be used to jointly restore images that are contaminated with mixed distortions.

There exist approaches [5, 11, 14] that can be used to compress a large network to a smaller one for computational efficiency. In the domain of image restoration, recursive neural networks [20, 36, 37] are investigated to reduce network parameters. However, the computational cost is still high due to the large number of recursions. The objective of our work is orthogonal to the aforementioned studies – our framework saves parameters and computation through learning a policy to make decision in selecting appropriate CNNs for a task rather than compressing an existing one.

Deep Reinforcement Learning. Reinforcement learning is a powerful tool for learning an agent making sequential decisions to maximize accumulative rewards. Early works of RL mainly focus on robotic control [27, 38]. Recently traditional RL algorithms are incorporated in deep learning frameworks and are successfully applied in various domains such as game agents [26, 30, 33, 34] and neural network architecture design [3, 45]. Attention is also drawn to deep RL in the field of computer vision [2, 4, 13, 16, 25, 28, 29, 32, 43]. For instance, Huang *et al*. [16] use RL to learn an early decision policy for speeding up object tracking by CNN. Cao *et al*. [4] explore deep RL algorithms in low-level vision and apply attention mechanism [29] to face hallucination. In this study, we investigate restoration tool selection in a RL framework. The problem is new in the literature.

3. Learning a Restoration Toolchain

Problem Definition. Given a distorted image $\mathbf{I}_{dis}$, our goal is to restore a clear image $\mathbf{I}_{res}$ that is close to the ground truth image $\mathbf{I}_{gt}$. The distortion process can be formulated

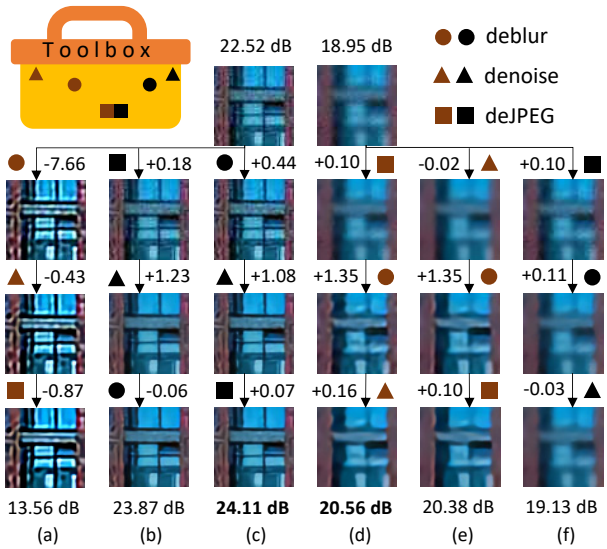

Figure 2. **Different toolchains for image restoration**. We perform a preliminary test here. Given two distorted images and the corresponding appropriate toolchains as (c) and (d), we construct other toolchains by rearranging the order (represented by shape) or adjusting the level (represented by color) of the selected tools. The restored results indicate that such minor changes of a toolchain could lead to very different performance.

as:

$$\mathbf{I}_{dis} = D(\mathbf{I}_{gt}); \quad D = D_n \circ \cdots \circ D_1, \tag{1}$$

where $\circ$ denotes function composition and each of $D_1, \ldots, D_n$ represents a specific type of distortion. In contrast to existing methods [6, 8, 31, 37, 41] that concentrate on a single type of distortion, we intend to handle a mix of multiple distortions (*i.e.*, $n > 1$). For example, the final output image may be sequentially affected by out-of-focus blur, exposure noise and JPEG compression. In such a case, the number of distortions n is 3, and D_1, D_2, D_3 represent blur, noise and compression, respectively. To address mixed distortions, we propose to restore the corrupted image step by step with a sequence of restoration tools.

Challenges. The task of tool selection is non-trivial and presents unique challenges to RL. First, *the choice of the restoration type, level and the processing order all influence the final performance*. An example is shown in Figure 2, where the images are corrupted by two different combinations of distortions. With an appropriate toolchain, as in Figure 2 (c, d), the image quality and the Peak Signal-to-Noise Ratio (PSNR) values are improved sequentially. Then we slightly re-arrange the tools order as in Figure 2(b, e) or adjust the restoration level of the tools as in Figure 2(a, f). The results indicate that minor changes in a toolchain can severely impact the restoration performance. Specifically, using improper tools may lead to unnatural outputs, such as over-sharpening in Figure 2(a) and blurring in Figure 2(f). Even the tools are well chosen, an inappropriate order could decrease the performance (Figure 2(b, e)). Since the sequence of toolchain dramatically influences the results, selecting which tool to use at each step becomes crucial.

When the tools are trained on specific tasks, we encounter another problem that *none of the tools can perfectly handle the 'middle state'*, which refers to the intermediate result after several steps of processing. As most distortions are irreversible, the restoration of their mixture is not a simple composition of the corresponding restorers. New artifacts could be introduced in the middle states. For example, the deblurring operation will also enhance the noises, causing the following denoisers fail in removing the newly introduced artifacts. The challenge is unique to our task.

To address the first challenge, we treat the sequential tool selection problem as a Markov Decision Process (MDP) and solve it in a deep reinforcement learning manner. To address the second challenge, we propose a training scheme to refine the agent and tools jointly so that the tools are more well-informed with the middle states observable by the agent. We first provide an overview of the proposed framework as follows.

Overview of *RL-Restore*. The proposed framework aims at discovering a toolchain given a corrupted input image. As shown in Figure 3, *RL-Restore* consists of two components: 1) a toolbox that contains various tools for image restoration and 2) an agent with a recurrent structure that dynamically chooses a tool at each step or an early stopping action. We cast the tool selection process as a reinforcement learning procedure – a sequence of decision on tool selection is made to maximize a reward proportional to the quality of the restored image. Next, we first describe a plausible setting of toolbox and then explain the details of the agent.

3.1. Toolbox

The toolbox contains a set of tools that might be applied to the corrupted image. Our goal is to design a powerful and light-weight toolbox, we thus restrict each tool to be proficient in a specific task. That is, each tool is trained only on a narrow range of distortions. To further reduce the overall complexity, we use smaller networks for easier tasks. For the purpose of our research, we prepare 12 tools as shown in Table 1, where each tool is assigned to address a certain level of Gaussian blur, Gaussian noise or JPEG compression. We apply a three-layer CNN (as in [8]) for slight distortions and a deeper eight-layer CNN for severe distortions. Note that the tools need not be restricted to solve the aforementioned distortions. We made these selections since they are typically considered in the literature of image restoration. In practice, one could design their tools with appropriate complexity based on the task at hand.

As discussed at the beginning of Sec. 3, a finite set of tools is not perfect to handle new artifacts emerged in middle states. To address this issue, we propose two strategies :

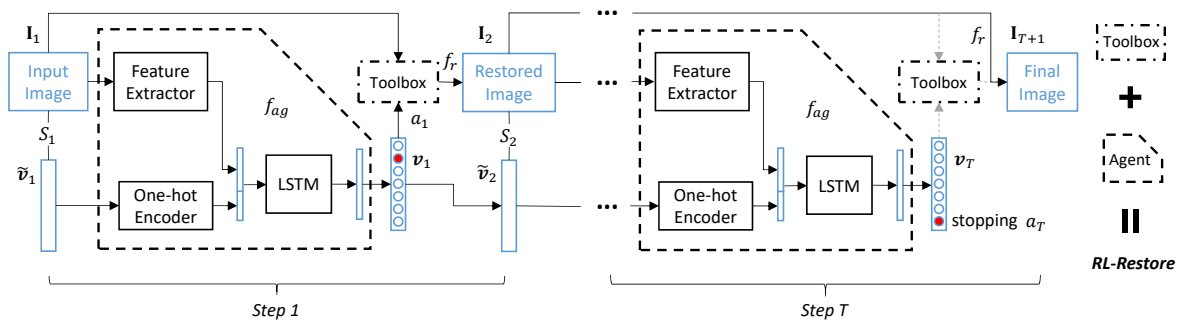

Figure 3. **Illustration of our *RL-Restore* framework.** At each step t, the agent f_{ag} observes the current state S_t, including the current restored image $\mathbf{I}_t$ and input value vector $\tilde{\boldsymbol{v}}_t$, which is the output of the agent at the previous step. Note that $\mathbf{I}_1$ represents the input image and $\tilde{\boldsymbol{v}}_1$ is a zero vector. Based on the maximum value of the agent's output $\boldsymbol{v}_t$, an action a_t is selected and the corresponding tool is used to restore the current image. After restoration process f_r, with the newly restored image $\mathbf{I}_{t+1}$ and value vector $\tilde{\boldsymbol{v}}_{t+1} = \boldsymbol{v}_t$, *RL-Restore* conducts another step of restoration iteratively until the stopping action is selected.

Table 1. Tools in the toolbox. We consider three types of distortion and various degradation levels. Each tool is either a 3-layer CNN or an 8-layer CNN according to the distortion it targets to solve.

Distortion Type (Parameters)	Distortion Level Interval	CNN Depth
Gaussian Blur (σ)	[0, 1.25], [1.25, 2.5]	3
	[2.5, 3.75], [3.75, 5]	8
Gaussian Noise (σ)	[0, 12.5], [12.5, 25]	3
	[25, 37.5], [37.5, 50]	8
JPEG Compression (Q)	[60, 100], [35, 60]	3
	[20, 35], [10, 20]	8

1) To increase robustness of the tools, we add slight Gaussian noises and JPEG compression to all the training data. 2) After training the agent, all tools are jointly fine-tuned on the basis of the well-trained toolchains. Then the tools will be more adaptive to the agent task, and be able to deal with middle states more robustly. We discuss the training steps in Sec. 3.3. Experiments in Sec. 4 validate the effectiveness of the proposed strategies.

3.2. Agent

The processing pipeline of *RL-Restore* is shown in Figure 3. Given an input image, the agent first selects a tool from the toolbox and uses it to restore the image, then the agent chooses another tool according to the previous result and repeats the restoration process until it decides to stop. We will first clarify some terminologies such as action, state and reward, and then go into the details of the agent structure and restoration procedure.

Action. The action space, denoted as A, is a set of all possible actions that the agent could take. At each step t, an action a_t is selected and applied to the current input image. Each action represents a tool in the toolbox and there is one additional action that represents stopping. If there are N tools in the toolbox, then the cardinality of A is $N + 1$. Hence, the output, $\boldsymbol{v}_t$, of the agent is an $(N + 1)$-dimensional vector that implicates the value of each action. Once the stopping action is chosen, the restoration procedure will be terminated and the current input image will become the final result.

State. The state contains information that the agent could observe. In our formulation, the state is formulated as $S_t = \{\mathbf{I}_t, \tilde{\boldsymbol{v}}_t\}$, where $\mathbf{I}_t$ is the current input image, and $\tilde{\boldsymbol{v}}_t$ is the past historical action vector. At step 1, $\mathbf{I}_1$ is the input image and $\tilde{\boldsymbol{v}}_1$ is a zero vector. The state provides rich contextual knowledge to the agent. 1) The current input image $\mathbf{I}_t$ is essential because the selected action will be directly applied to this image to derive a better restored result. 2) the information of previous action vector $\tilde{\boldsymbol{v}}_t$, which is the output value vector of the agent at $t-1$ step, *i.e.*, $\tilde{\boldsymbol{v}}_t = \boldsymbol{v}_{t-1}$, is important too. The knowledge of the previous decision could help the action selection at the current step. This is found to work better empirically than using $\mathbf{I}_t$ only.

Reward. The reward drives the training of the agent as it learns to maximize the cumulative reward. The agent is supposed to learn a good policy so that the final restored image is satisfactory. We wish to ensure that the image quality is enhanced at each step, therefore a stepwise reward is designed as follows:

$$r_t = P_{t+1} - P_t, \tag{2}$$

where r_t is the reward function at step t, P_{t+1} denotes the PSNR between $\mathbf{I}_{t+1}$ and the reference image $\mathbf{I}_{gt}$ at the end of the t-th step restoration, and P_t represents the input PSNR at step t. The cumulative reward can be written as $R = \sum_{t=1}^{T} r_t = P_{T+1} - P_1$, which is the overall PSNR gain during the restoration procedure, and it is maximized to achieve optimal enhancement. Note that it is flexible to use other image quality metrics (*e.g.*, perceptual loss [18], GAN loss [23]) as the reward in our framework. The investigation is beyond the focus of this paper.

Structure. At each step t, the agent assesses the value of each action given the input state S_t, which can be formu-

lated as follows:

$$\boldsymbol{v}_t = f_{ag}(S_t; W_{ag}), \tag{3}$$

where f_{ag} indicates the agent network and W_{ag} denotes its parameters. The vector $\boldsymbol{v}_t$ represents the value of actions. The action with the maximum value is selected as a_t, *i.e.*, $a_t = \text{argmax}_a v_{t,a}$, where $v_{t,a}$ indicates the element of value vector $\boldsymbol{v}_t$ corresponding to action a.

The agent is composed of three modules as depicted in Figure 3. The first module, named feature extractor, is a four-layer CNN followed by a fully-connected (fc) layer that outputs a 32-dimensional feature. The second module is a one-hot encoder with $N+1$ dimensional input and N dimensional output, preserving the information of the previous chosen action. Note that the output is one dimension lower than the input, because the stopping action cannot be adopted at the previous step, and thus we simply drop the last dimension. The outputs of the first two modules are concatenated into the input of the third module, which is a Long Short-Term Memory (LSTM) [15]. The LSTM not only observes the input state, but also stores historical states in its memory, which offers contextual information of historical restored images and actions. Finally, with another fc layer following LSTM, a value vector $\boldsymbol{v}_t$ is derived for tool selection.

Restoration. Once an action a_t is obtained based on the maximum value in $\boldsymbol{v}_t$, the corresponding tool will be applied to the input image $\mathbf{I}_t$ to get a new restored image:

$$\mathbf{I}_{t+1} = f_r(\mathbf{I}_t, a_t; W_r), \tag{4}$$

where f_r denotes the restoration fucntion and W_r indicates the parameters of a tool in the toolbox. If a stopping action is selected, f_r represents an identity mapping. By denoting $\mathbf{I}_{dis}$ and $\mathbf{I}_{res}$ as the input distorted image and final restored output respectively, the overall procedure of restoration can be expressed as:

$$\begin{cases} \mathbf{I}_1 = \mathbf{I}_{dis} \\ \mathbf{I}_{t+1} = f(\mathbf{I}_t; W) \quad 1 \le t \le T \\ \mathbf{I}_{res} = \mathbf{I}_{T+1}, \end{cases} \tag{5}$$

where $f = [f_{ag}; f_r]$ and $W = [W_{ag}; W_r]$. T is the step when the stopping action is chosen. We also set a maximum step $T_{\max}$ to prevent excessive restoration. When $t = T_{\max}$ and the stopping action is not selected, we will terminate the restoration process after the current step. In other words, we add a constraint that $T \le T_{\max}$.

3.3. Training

The training of tools follows a standard setting in [19], where a mean square error (MSE) $\frac{1}{2}\|\mathbf{y} - h(\mathbf{x})\|_2^2$ is minimized. The ground truth image, input image and the tool are denoted as $\mathbf{y}, \mathbf{x}$ and h, respectively. As for the agent, the training is addressed by deep Q-learning [30] since we do not have *a priori* knowledge about the correct action to choose. In the proposed framework, each element of $\boldsymbol{v}_t$ is an action value as defined in [30], so the loss function can be written as $L = (y_t - v_{t,a_t})^2$ where

$$y_t = \begin{cases} r_t + \gamma \max_{a'} v_{t+1,a'} & 1 \le t < T \\ r_T & t = T, \end{cases} \tag{6}$$

and $\gamma = 0.99$ is a discount factor. We also employ a target network f'_{ag} to stabilize training, which is a clone of f_{ag} and updates its parameters every C steps while training. In the above formula, $v_{t+1,a'}$ is derived from f'_{ag} and v_{t,a_t} is from f_{ag}. While training, episodes are randomly selected from a replay memory, and there are two updating strategies as proposed in [12], where 'random updates' refer to updating from a random point of each episode and proceeding a fixed number of steps, and 'sequential updates' indicate that all the updates begin at the beginning of the episode and proceed to its endpoint. In [12], it is claimed that both updating strategies have similar performance. Since our toolchain is not too long, we simply adopt 'sequential updates' where each training sequence contains an entire toolchain.

Joint Training. As discussed in Section 3.1, none of the tools can perfectly handle the middle state, where new and complex artifacts may be introduced in the previous steps of restoration. In order to address this issue, we propose a joint training algorithm, as shown in Algorithm 1, to train the tools in an end-to-end manner so that all the tools can learn to deal with the middle state. Specifically, for each toolchain in a batch, the distorted image $\mathbf{I}_1$ is forwarded to get a restored result $\mathbf{I}_{T+1}$. Given a final MSE loss, the gradients then pass backward along the same toolchain. Meanwhile, the gradients of each tool are accumulated within a batch, and finally an average of gradient is used to update the corresponding tool. The above updating process is repeatedly conducted for a few iterations.

Implementation Details. In our implementation, the training of tools is similar to [19], where all experiments run over 80 epochs (3.2×10^5 iterations) with a batch size of 64. The initial learning rate is 0.1 and it decreases by a factor of 0.1 every 20 epochs. For joint training, we set $M = 64, \alpha = 0.0001$ in Algorithm 1, denoting the batch size and learning rate respectively. The joint training runs over 2×10^5 iterations. While training the agent, we use Adam [21] optimizer and a batch size of 32. The maximum step $T_{\max}$ is set to be 3 empirically and the size of replay memory is chosen as 5×10^5. The updating frequency $C = 2,500$ so that the target network f'_{ag} is copied from the latest agent network f_{ag} every $2,500$ iterations. The learning rate is decayed exponentially from 2.5×10^{-4} to 2.5×10^{-5} within 5×10^5 iterations.

Algorithm 1 Joint training algorithm (1 iteration)

Initialize counters $c_1, c_2, \ldots, c_N = 0$
Initialize gradients $G_1, G_2, \ldots, G_N = 0$
for $m = 1, M$ **do** ▷ For each toolchain
  $\mathbf{I}_1 \leftarrow$ Input image
  for $t = 1, T$ **do** ▷ Forward paths
    $a_t \leftarrow f_{ag}(S_t)$
    $\mathbf{I}_{t+1} \leftarrow f_r(\mathbf{I}_t, a_t)$
  end for
  $L \leftarrow \frac{1}{2}\|\mathbf{I}_{gt} - \mathbf{I}_{T+1}\|_2^2$
  for $t = T$ **to** 1 **step** -1 **do** ▷ Backward paths
    $c_{a_t} \leftarrow c_{a_t} + 1$
    $G_{a_t} \leftarrow G_{a_t} + \partial L / \partial W_{a_t}$
    $L \leftarrow \mathbf{I}_t \cdot \partial L / \partial \mathbf{I}_t$
  end for
end for
for $i = 1, N$ **do** ▷ Update tools
  if $c_i > 0$ **then**
    $W_i \leftarrow W_i - \alpha G_i / c_i$
  end if
end for

4. Experiments

Datasets and Evaluation Metrics. We perform experiments on the DIV2K dataset [1], which is the most recent large-scale and high-quality dataset for image restoration. The 800 DIV2K training images are divided into two parts: 1) the first 750 images for training and 2) the rest 50 images for testing. The DIV2K validation images are used for validation. Training images are augmented by down-scaling with factors of 2, 3 and 4. The images are then cropped into 63×63 sub-images, forming our training set and testing set with 249,344 and 3,584 sub-images, respectively.

We employ mixed distortions for agent training and testing. Specifically, a sequence of Gaussian blur, Gaussian noise and JPEG compression is added to the training images with random levels. The standard deviations of Gaussian blur and Gaussian noise are uniformly distributed in [0, 5] and [0, 50], respectively, while the quality of JPEG compression is subjected to a uniform distribution in [10, 100]. All mixed distortions are categorized into five groups, as shown in Figure 4, from extremely mild to extremely severe. We discard two extreme cases that are either too easy or too hard for restoration. Training and testing are performed on the moderate group. To further test the generalization ablity, we also perform testing on mild and severe groups that are not included in the training data.

Comparisons. We compare *RL-Restore* with DnCNN [44] and VDSR [19], which are the state-of-the-art models for image restoration and super-resolution, and both of them are capable of handling multiple degradations. DnCNN and VDSR share similar structure with 20 convolutional layers while batch normalization is adopted in DnCNN. Their parameters are over 0.6 million (shown in Table 2). In contrast, the complexity of *RL-Restore* (including the agent and the selected tools[2]) is only about a third of those for DnCNN and VDSR, with 0.19 million parameters in total. A much larger gap can be observed on computations when we refer to the number of multiplications on a 63×63 input image. For a fair comparison with *RL-Restore*, we shrink VDSR from 20 to 15 layers (42 filters in each layer) to form a new baseline, named VDSR-s, which bares similar complexity as *RL-Restore*. Following the same training strategy in [19, 44], we first train the baselines with the agent training set. Then we fine-tune the models with both the agent and tools training sets till convergence.

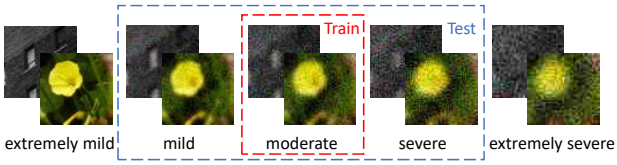

Figure 4. Different levels of distortions.

Table 2. Complexity of baselines and *RL-Restore*.

Model	DnCNN	VDSR	VDSR-s	*RL-Restore*
Parameters ($\times 10^5$)	6.69	6.67	2.09	1.96
Computations ($\times 10^9$)	2.66	2.65	0.828	0.474

Table 3. Quantitative results on DIV2K test sets.

Test Set	Mild (unseen)		Moderate		Severe (unseen)	
Metric	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
DnCNN	28.03	**0.6503**	26.42	0.5554	24.99	0.4658
VDSR	**28.04**	0.6496	26.40	0.5544	24.90	0.4629
VDSR-s	27.69	0.6383	25.99	0.5399	24.50	0.4505
RL-Restore	28.04	0.6498	**26.45**	**0.5587**	**25.20**	**0.4777**

4.1. Quantitative Evaluation on Synthetic Dataset

We present quantitative results of *RL-Restore* and baselines on different test sets in Table 3. The results on mild and moderate sets show that our approach is apparently superior to VDSR-s while comparable to DnCNN and VDSR, demonstrating that the proposed *RL-Restore* could achieve the same performance as a deep CNN with much lower complexity. It is worth noting that on severe test set *RL-Restore* surpasses DnCNN and VDSR by 0.2 dB and 0.3 dB, respectively, where the distortions are not observed in the training data. It indicates that our RL-based approach is more flexible in handling unseen distortions, while it is more difficult for a fixed CNN to generalize towards unseen cases. Visual results are shown in Figure 5.

To examine the internal behaviors of *RL-Restore* , we analyze the frequency of the tool selection at each step. Results are shown in Figure 6, where 0–12 on x-axis represent the 12 tools in Table 1 and 13 is the stopping action. As can

[2]The complexity of toolchain is calculated under the assumption that each tool is chosen with equal probabilities and the stopping action is ignored. We do not adopt batch normalization in any model.

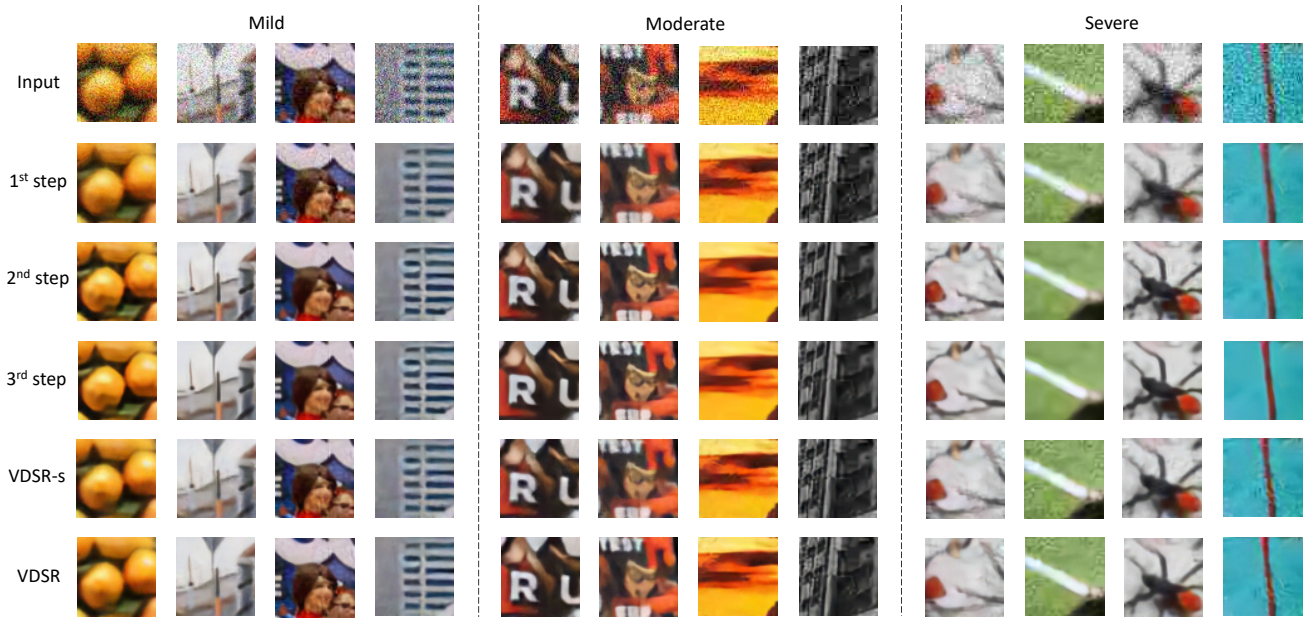

Figure 5. Qualitative comparisons with baselines on synthetic dataset.

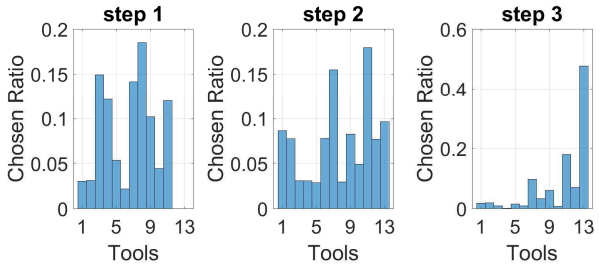

Figure 6. The chosen ratio of tool selection at each step.

be observed on the three charts, the tool selection is diverse, and all tools are utilized in a different ratio. Specifically, deblurring and denoising tools are preferred at the first step, while denoising and de-JPEG tools are frequently chosen at the second step. The last step tends to stop the agent with a large probablity – 47%. Interestingly, when testing on unseen data, the ratios of stopping action at the last step are 60% and 38% on mild and severe test sets, respectively, which indicates that more severe and complex distortions require a longer toolchain to restore.

4.2. Qualitative Evaluation on Real-World Images

In real-world cases, images are always distorted by a variety of complex and mixed distortions with unknown degradation kernels, making restoration tasks extremely difficult for current methods. The proposed RL-based method may shed some light on possible solutions. When real-world distortions (*e.g.*, slight out-of-focus blur, exposure noise and JPEG artifacts) are close to the training data, the proposed *RL-Restore* can be easily generalized to these problems and performs better than a single CNN model.

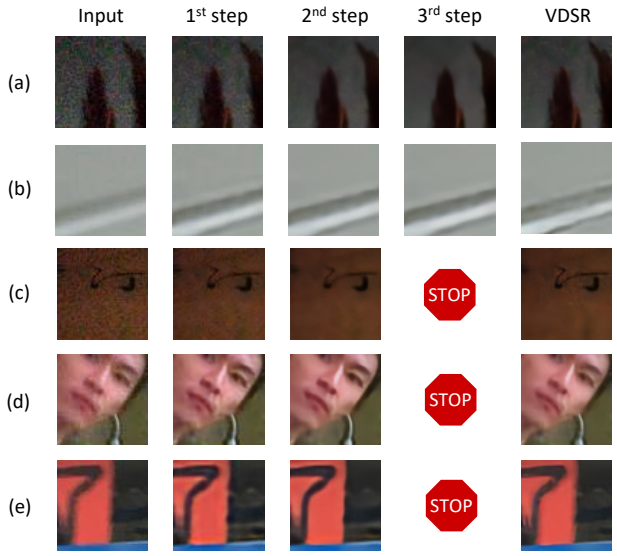

Figure 7. Results of real-world images.

Examples are shown in Figure 7, where the input images, combined with different distortions (e.g., blurring, noise, compression), are captured by smart phones. We directly apply the well-trained *RL-Restore* and VDSR on those real-world images, without further fine-tuning on the test data. It is obvious that our approach, benefiting from flexible toolchains, is more effective for restoring real-world images. Specifically, Figure 7(a, c) show that *RL-Restore* can successfully deal with severe artifacts caused by exposure and compression, while Figure 7(b, d, e) demonstrate that our approach is able to restore a mix of blur and complex noise. It is also worth noting that the stopping action is

Table 4. Ablation study on toolbox's size and toolchain's length.

Test Set		Mild (unseen)		Moderate		Severe (unseen)	
Metric		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Size	6	27.57	0.6241	25.72	0.5142	24.27	0.4291
	12	**27.78**	**0.6372**	**26.20**	**0.5441**	**24.97**	0.4643
	18	27.77	0.6361	26.17	0.5417	24.93	**0.4650**
Length	2	27.74	0.6264	25.99	0.5233	24.63	0.4444
	3	**27.78**	**0.6372**	26.20	0.5441	24.97	0.4643
	4	27.73	0.6368	**26.20**	**0.5450**	**24.98**	**0.4663**

selected by the agent when it is confident in the restored quality (Figure 7(c, d, e)). We believe that the proposed framework has the potential to deal with more complex real distortions with more powerful restoration tools.

4.3. Ablation Studies

In this section, we investigate different settings of the proposed *RL-Restore*, and give some insights on the choice of hyper-parameters. To better distinguish the effectiveness of each factor, we exclude the joint training strategy on all the experiments below.

Toolbox Size and Toolchain Length. The capacity of toolbox and the number of restoring actions dominate the restoration performance. We alternatively vary the length of toolchain and the size of toolbox. As observed in Table 4, *RL-Restore* performs well with $N = 12$ and $T_{\max} = 3$ under the current problem settings. Fewer tools and a shorter toolchain will decrease the performance. More tools and a longer toolchain achieve comparable performance. We attribute this phenomenon to the increased difficulty in learning more complex toolchains. It is worth pointing out that a toolchain with a length of two has a comparable PSNR as longer toolchains on the mild test set, indicating that slight distortions require fewer steps to restore.

Tools Training. As discussed in Sec. 3.1, we propose two training strategies for tools to eliminate the complex artifacts in middle states: 1) Add slight noise and compression in the tools training data. 2) Perform joint training with the agent. Control experiments are conducted as in Table 5, where the 'Original' setting represents the baseline, the '+Noise' adopts the first strategy and the '+Joint' uses both of them. It is obvious that adding noise to the training data successfully improves the PSNR by 0.2 dB, and joint training further pushes another 0.2 dB on all test sets, demonstrating the effectiveness of both training strategies.

Reward Function. We experimentally find that the choice of reward functions can largely influence the performance. Besides the proposed stepwise reward based on PSNR, we also investigate other reward functions: 1) stepwise SSIM [40] where the reward is the SSIM gain at each step; 2) final PSNR where the reward is the final PSNR gain given at the last step; 3) final MSE as in [4] where the reward is the negative MSE in the end. We adaptively adjust the learning rate for different rewards. As can be seen in Table 6, the stepwise SSIM, which performs the worst on PSNR metric, seems not to be a good choice for reward. The final MSE is slightly better on PSNR, but performs the worst on SSIM. The final PSNR achieves similar performance as the proposed stepwise PSNR reward. Nevertheless, we do not claim that PNSR is the best reward, and other evaluation methods are also encouraged for further comparison.

Table 5. Ablation study on tools training.

Test Set	Mild (unseen)		Moderate		Severe (unseen)	
Metric	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
+Joint	**28.04**	**0.6498**	**26.45**	**0.5587**	**25.20**	**0.4777**
+Noise	27.78	0.6372	26.20	0.5441	24.97	0.4643
Original	27.52	0.6027	25.91	0.5119	24.81	0.4490

Table 6. Ablation study on reward functions.

Test Set	Mild (unseen)		Moderate		Severe (unseen)	
Metric	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Step. PSNR	**27.78**	**0.6372**	**26.20**	**0.5441**	**24.97**	0.4643
Step. SSIM	26.58	0.6341	25.20	0.5368	24.18	0.4579
Final PSNR	27.71	0.6350	26.11	0.5417	24.86	**0.4656**
Final MSE	27.14	0.6009	25.66	0.5166	24.55	0.4470

Table 7. Ablation study on stopping action.

Test Set	Mild (unseen)		Moderate		Severe (unseen)	
Metric	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
w/ Stopping	**27.78**	**0.6372**	**26.20**	**0.5441**	**24.97**	**0.4643**
w/o Stopping	27.61	0.6284	26.08	0.5351	24.85	0.4589

Automatic Stopping. The stopping action gives the agent the flexibility to terminate the restoration process when it is confident about the restored results. Thanks to this flexible stopping mechanism, it can prevent the images from over restored and save much computation. To demonstrate its effectiveness, we compare the results with/without the stopping action. As can be observed in Table 7, the PSNR values drop around 0.15 dB when removing the stopping action. It is observed that the gap on mild test set is larger than that on other test sets. This is consistent with our experience that slight distortions are easily over restored if the agent does not stop in time.

5. Conclusion

We have presented a novel approach for image restoration based on reinforcement learning. Unlike most existing deep learning based methods, in our approach an agent is learned to dynamically select a toolchain to progressively restore an image that is corrupted by complex and mixed distortions. Extensive results on synthetic and real-world images validate the effectiveness of the proposed approach. With its inherent flexibility, the proposed framework can be applied to more challenging restoration tasks or other low-level vision problems by developing powerful tools and an appropriate reward.

Acknowledgement. This work is supported by SenseTime Group Limited and the General Research Fund sponsored by the Research Grants Council of the Hong Kong SAR (CUHK 14241716, 14224316. 14209217).

References

[1] E. Agustsson and R. Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPR Workshop*, 2017.

[2] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. In *ICLR*, 2015.

[3] B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcement learning. In *ICLR*, 2017.

[4] Q. Cao, L. Lin, Y. Shi, X. Liang, and G. Li. Attention-aware face hallucination via deep reinforcement learning. In *CVPR*, 2017.

[5] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In *ICML*, 2015.

[6] Y. Chen, W. Yu, and T. Pock. On learning optimized reaction diffusion processes for effective image restoration. In *CVPR*, 2015.

[7] C. Dong, Y. Deng, C. C. Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. In *ICCV*, 2015.

[8] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *TPAMI*, 38(2):295–307, 2016.

[9] J. Guo and H. Chao. Building dual-domain representations for compression artifacts reduction. In *ECCV*, 2016.

[10] J. Guo and H. Chao. One-to-many network for visually pleasing compression artifacts reduction. *arXiv preprint arXiv:1611.04994*, 2016.

[11] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

[12] M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. *CoRR, abs/1507.06527*, 2015.

[13] Y. He, K. Cao, C. Li, and C. C. Loy. Merge or not? learning to group faces via imitation learning. In *AAAI*, 2018.

[14] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS Workshop*, 2014.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[16] C. Huang, S. Lucey, and D. Ramanan. Learning policies for adaptive tracking with deep feature cascades. In *ICCV*, 2017.

[17] T.-W. Hui, C. C. Loy, and X. Tang. Depth map super-resolution by deep multi-scale guidance. In *ECCV*, 2016.

[18] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[19] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016.

[20] J. Kim, J. Kwon Lee, and K. Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, 2016.

[21] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[22] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *CVPR*, 2017.

[23] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[24] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. In *CVPR*, 2017.

[25] X. Liang, L. Lee, and E. P. Xing. Deep variation-structured reinforcement learning for visual relationship and attribute detection. In *CVPR*, 2017.

[26] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

[27] L.-J. Lin. *Reinforcement learning for robots using neural networks*. PhD thesis, Fujitsu Laboratories Ltd, 1993.

[28] B. Liu and X. He. Learning dynamic hierarchical models for anytime scene labeling. In *ECCV*, 2016.

[29] V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. In *NIPS*, 2014.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[31] S. Nah, T. H. Kim, and K. M. Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *CVPR*, 2017.

[32] Z. Ren, X. Wang, N. Zhang, X. Lv, and L.-J. Li. Deep reinforcement learning-based image captioning with embedding reward. In *CVPR*, 2017.

[33] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[34] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[35] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *CVPR*, 2015.

[36] Y. Tai, J. Yang, and X. Liu. Image super-resolution via deep recursive residual network. In *CVPR*, 2017.

[37] Y. Tai, J. Yang, X. Liu, and C. Xu. Memnet: A persistent memory network for image restoration. In *ICCV*, 2017.

[38] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *ECML*, 2005.

[39] X. Wang, K. Yu, C. Dong, and C. C. Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform. In *CVPR*, 2018.

[40] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *TIP*, 13(4):600–612, 2004.

[41] 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 *CVPR*, 2016.

[42] L. Xu, X. Tao, and J. Jia. Inverse kernels for fast spatial deconvolution. In *ECCV*, 2014.

[43] S. Yun, J. Choi, Y. Yoo, K. Yun, and J. Y. Choi. Action-decision networks for visual tracking with deep reinforcement learning. In *CVPR*, 2017.

[44] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *TIP*, 2017.

[45] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.