Doctoral Thesis in
Artifical Intelligence

Toward an Efficient Deep Image Restoration Method

Graduate School of Ajou University

Department of Artifical Intelligence

Namhyuk Ahn

Toward an Efficient Deep Image Restoration Method

Kyung-Ah Sohn, Advisor

I submit this thesis as the
Doctoral thesis in Artifical Intelligence

August, 2021

Graduate School of Ajou University

Department of Artifical Intelligence

Namhyuk Ahn

The Doctoral thesis of Namhyuk Ahn in
Artifical Intelligence is hereby approved.

Thesis Defense

Committee Chair	Tae-Sun Chung	Seal [illegible]
Member	Kyung-Ah Sohn	Seal [illegible]
Member	Wonjun Hwang	Seal [illegible]
Member	Kyungsik Han	Seal [illegible]
Member	Byungkon Kang	Seal [illegible]

Graduate School of Ajou University

June 21, 2021

Abstract

Image restoration is a classic but challenging problem in the computer vision field. In recent, deep learning-based methods have achieved superior performance by the huge capacity of neural networks and the large volume of the dataset. However, are these heavy models applicable in real-world applications? If not suitable, what are the directions the deep methods would take? To answer these questions, the thesis explore three aspects: model efficiency, data efficiency, and multi-modal distortion.

In model efficiency, we define the “efficiency” as both network size and the computation cost to run the network. Many studies have focused on the former alone, but in reality, the latter one is the key ingredient because of the runtime latency and the battery consumption issues. To tackle this, we devise the network structure and rethinking the training strategy to maintain the performance as much as possible while effectively advance both efficiency aspects: network size, and the number of the operations.

For data efficiency, we investigate the data augmentation and the unsupervised training in the image restoration task. The data augmentation method is fruitful when the training dataset is small or the network capacity is large without any computation cost in runtime. The unsupervised training assumes the scenario where only low-quality images exist, much challenging compared to the supervised regime. These two concepts have been well analyzed in the high-level vision field, but not many in the image restoration community. With both training strategies, we achieve the huge performance leap to the recent image restoration methods in many real-world scenarios and datasets.

Last but not least, we tackle the multi-modal distortion, in particular, when multiple distortions corrupt the different regions of image. The single distortion restoration network or the distortion recognition-restoration pipeline system are not satisfactory in terms of both the performance and the efficiency when serving a model. In contrast, the proposed multi-expert network based on the multi-task learning and the analysis of the multi-modal distribution performs superior restoration accuracy with reasonable computation cost and good efficiency in model serving perspective.

Keywords: Low-level vision, Image restoration, Deep neural network, Model efficiency, Data efficiency, Multi-modal distortion

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The image restoration task aims to reconstruct the high-quality (HQ) image $\mathbf{x}$ from the low-quality (LQ) image $\mathbf{y}$. In general, degraded observation $\mathbf{y}$ is generated as $\mathbf{y} = \mathbf{H}(\mathbf{x}) + \mathbf{v}$, where $\mathbf{H}$ is a degradation function and $\mathbf{v}$ is noise. With this formula, we can define many image restoration tasks: image denoising if $\mathbf{H}$ is an identity function and $\mathbf{v}$ is Gaussian noise of standard deviation σ, image deblurring when $\mathbf{H}$ is a function of a blur kernel, and image super-resolution when $\mathbf{H}$ is a composition of a blur kernel and subsampling operation. Since the LQ image $\mathbf{y}$ and HQ image $\mathbf{x}$ are one-to-many mapping, solving this formula is an ill-posed problem, thus developing the effective restoration algorithm is very challenging. Despite the difficulty, this task is a long-standing problem in the computer vision community and has deeply been investigated due to its highly practical value in various potential applications. For example, we can apply the restoration method to surveillance systems to reduce the streaming latency or we can use it to enhance the low-quality and old media into clean ones. Besides, many modern smartphones adopt the restoration algorithm to the camera module for better user experience.

Recently, the use of a deep learning-based approach shows outstanding improvements in image restoration performance. The major factors of the success of deep methods are the large dataset and the huge capacity of the convolutional neural networks. From the pioneer deep restoration model, SRCNN [1], which has three layers, the capacity of the networks rapidly has grown as over time. For example, one of the state-of-the-art methods, RCAN [2], is composed of four hundred convolutional layers. Thanks to the high capacity of the model, we have witnessed the powerful performance when the input dataset is adequately enough to train such heavy network. However, although the modern deep-based approaches have shown outstanding performance, there are strong drawbacks to this paradigm— the necessity of a massive dataset and the computation resources.

Beyond improving the method on the performance only, now it is the time to rethink the deep restoration models as to how to apply in real-world applications. Is high performance is the only optimal solution in the real world? If this is not true, what are the major issues? Here, we present three topics below and we investigate these throughout the thesis.

Large computation and memory. Can real-world devices manage the deep models? Modern deep restoration networks use more than hundreds of layers to increase the capacity, but paradoxically, these trends decrease the usability of such models. Not all the applications are computed in the state-of-the-art GPU, rather most of them are running in CPU or even small-computation edge devices. In this circumstance, latency and battery consumption are the major factors where the modern deep methods cannot be fulfilled easily. To achieve the efficiency criteria, it is necessary to view the "efficiency" as computation cost, while most of the studies have overlooked this.

Limits of the data-hungry training. Most of the deep restoration networks are based on the supervised training scheme where they require massive HQ and LQ pairs. Unfortunately, a large-scale and high-quality dataset (HQ and LQ pairs) is not always accessible, especially in real-world scenarios. Even worse, it is impossible to collect HQ images for some cases such as the medical imagining field. In the data limitation environment, how we adequately train the deep networks? In the high-level vision (e.g., classification) they have applied the data augmentation [3, 4] when the volume of the training dataset is small. For the case where no labels are given, some studies have used the unsupervised or self-supervised to make the model learns the data representation and fine-tune this pre-trained network with an additional small dataset [5, 6, 7]. However, none of both approaches have been deeply investigated in low-level vision such as the image restoration field.

Single distortion is all we need? In real-world applications, an image could be corrupted with multiple distortions while most of the studies have focused on restoring a single distortion only. However, it is well known that the deep networks are fragile to the disparity distribution between the training and test dataset. To detour this, recent studies have proposed mixed distortion [8], but the multi-modal nature of image distortion should be profoundly analyzed from many points of view.

1.1 Thesis Outline

We provide an outline of the chapters according to the three parts of the thesis.

Part I: Model Perspective Efficiency. We first study the efficiency of the deep networks for the image restoration task. We define the model efficiency as not only the model parameters but also the number of the operations, which is the key ingredient of the inference latency and memory consumption when serving a deep method.

In this part, we tackle the efficiency on two aspects, the number of the parameters and the operations. The simplest strategy is naively reducing the number of channels or layers, but this is not an appropriate solution since the performance degradation is rapid compared to the improved efficiency. In Chapter 2, we devise the network structure mechanism that maintains the performance as much as possible while effectively advance the efficiency. With a novel architectural design, we achieve the best efficiency-performance compared to the state-of-the-art methods (at time of publication). Besides, we control the efficiency with the techniques from the high-level vision models and provide some guidance to squeeze the efficiency of the network. With the study in Chapter 2, we extend the model to catch both the performance and the efficiency of the extreme image restoration task (Chapter 3). In extreme situations, many deep methods fall into training instability because of the lack of input information and the sudden increase of the feature resolution. To alleviate these issues, we combine the core concept of the lightweight and the heavy networks so to make a golden balance point between the restoration accuracy and the model efficiency.

Part II: Data Perspective Efficiency. As mentioned before, data efficiency is another key feature in the application perspective, yet still is the limitation of many deep image restoration methods. To alleviate this issue, we first deeply investigate the data augmentation strategy in the image restoration tasks. Data augmentation is the widely used training technique in high-level vision tasks while not many studies have a focus on this in the low-level vision (including the image restoration). In Chapter 4, we conduct a comprehensive analysis of the data augmentation on the diverse image restoration tasks and propose a new augmentation method specifically designed for the image restoration purpose. With this strategy, the restoration accuracy is increased without an extra cost in inference, and furthermore, the data augmentation enhances the generalization ability of the model. One interesting observation is that the effects of the augmentation are enlarged when the datasets are closer to the real environment. This is because images from the realistic dataset are corrupted with different but subtle distortion kernels or noise so that the generalization capability is particularly beneficial.

Then, we study the unsupervised image restoration task in Chapter 5. This scenario assumes that only low-quality images can be acquired, and this is a very common in the medical imagining field. Previously, ZSSR [9] has tackled this problem by seeing given single input image only with a runtime optimization approach. However, the performance of ZSSR is inferior to the supervised models and the latency is not manageable (even in modern GPU) due to the runtime training. Instead, we relax the assumption of ZSSR and reformulate the unsupervised image restoration problem to the supervised regime. With this approach, the performance gap between the supervised and the proposed is drastically reduced with the identical runtime of the supervised-based networks.

Part III: Toward Multi-modal Distortion. In Part III, the thesis focuses on studying the multi-modality in distortion. Here, we narrow the definition of the multi-modality as the scenario where multiple distortions are represented in various locations (e.g., Gaussian noise on the left while blur on the right of the image). Unlike the single distortion, directly applying a restoration method does not guarantee satisfactory results. We hypothesize that this is similar to the performance trends in realistic distortion— the diversity implied in multi-modality requires the ability on generalization. In addition, because distortions are applied in different locations, the restoration models should know both "what" and "where" the distortions are in this region. To analyze these in-depth, we first begin this part with a deep investigation of the multi-modality (Chapter 6), especially on whether the deep model can recognize the type or location of the distortion very accurately.

In Chapter 7, we study the effective and efficient restoration model on this scenario. The very naive approach is the pipeline system that detects the corruptions and reconstructs them with distortion-specific restoration networks. This framework, however, is composed of the distortion recognizer and multiple restoration modules. When serving this framework in a real-world application, this is particularly inefficient since we manage all the sub-modules simultaneously, and even this framework requires extra latency on recognizing the distortions. Besides, each restoration module is trained independently so that the shared representation of both the image and distortion are learned in their way, inefficient in training perspective. On the other hand, a single-branch network design (as in the single distortion restoration) is ineffective since the network handles multiple distortions. If we increase the network capacity, then the efficiency significantly deteriorates. Instead of the two approaches, we build the network as a mixture-of-experts concept following the multi-task learning literature [10] with an unsupervised distortion recognition sub-network. This framework has several advantages: 1) Effective since each expert learns both shared and distortion-specific representation. 2) Efficient in model serving perspective since neither additional distortion recognizer nor distortion-specific networks are required. 3) Interpretable due to the attention-based expert fusing mechanism.

Part I

Model Perspective Efficiency

Chapter 2

Lightweight Image Restoration Model

Recent progress in the deep learning-based models has improved the image restoration performance significantly. However, despite their powerful performance, many methods are difficult to apply to real-world applications because of the heavy computational requirements. To facilitate the use of a deep model under such demands, we focus on keeping the network efficient while maintaining its performance. In detail, we design an architecture that implements a cascading mechanism on a residual network to boost the performance with limited resources via multi-level feature fusion. In addition, our proposed model adopts group convolution and recursive schemes to achieve extreme efficiency. We further improve the perceptual quality of the output by employing the adversarial learning paradigm and a multi-scale discriminator approach. To benchmark the superiority of our method, we mainly evaluate the networks to the super-resolution task. We conduct extensive internal experiments and compare the model to the recent methods in the various datasets. Our results show that proposed models outperform the other methods with similar complexity, for both traditional pixel-based and perception-based tasks.

2.1 Overview

Image super-resolution (SR) is a longstanding computer vision task that can be widely used in many applications. This task focuses on recovering a high-resolution (HR) image from low-resolution (LR) images. In particular, single-image super-resolution (SISR) performs SR using a single LR image. Since the SISR problem is a one-to-many mapping, constructing an effective SISR algorithm is challenging. Despite the difficulties, SISR has been actively studied since it can be applied to a variety of scenarios (*i.e.* enhancing human face [11] or biometrics [12]). Recently, deep learning-based methods have shown prominent performance on SR task [13]. The major trend of deep models is not only stacking layers to their networks [14] but also designing and assembling internal blocks and network topologies [15] to achieve more accurate results.

Although deep learning-based networks significantly increase the quality of the SR outputs, applying such models to real-world scenarios is another challenge. Many cases require not only quality but also efficiencies such as streaming services or mobile applications. However, the recent state-of-the-art methods [16, 17, 2] use very deep networks, which can be computationally heavy. From this perspective, it is obvious that designing a lightweight SR network is very crucial.

Several works [18, 19, 20] make efforts to design a *lightweight* SR model by reducing the number of parameters. One of the most simple and effective approaches is to construct the model in a recursive manner [18]. However, even though such studies show good SR performance using a small number of parameters, they have some downsides: These works increase the depth or width of the network to compensate for the performance loss caused by the use of the recursive scheme, making the inference very slow. Moreover, their *early-upsample* design, which upsamples the input image before inputting it to the network, results in high computational cost.

However, as mentioned earlier, the number of operations is also an important factor to consider in real-world demands. For the SR systems that operate on mobile devices, the execution speed also plays an important role from a user-experience perspective. Especially the battery capacity, which is heavily dependent on the amount of computation performed, becomes a major problem. In this respect, reducing the number of operations is a challenging and necessary step that has largely been ignored until now. A relevant practical scenario can be found in video streaming services. The demand for streaming media has skyrocketed, and hence large storage for massive multimedia data is required. It is therefore imperative to compress data using lossy compression techniques before storing. Then, an SR technique can be applied to restore the data to the original resolution. However, because latency is the most critical factor in such services, the decompression process has to be performed in near-real time. To do so, it is essential to make the SR methods lightweight in terms of the number of operations to satisfy timing constraints.

To handle these requirements and improve the recent models, we propose a Cascading residual network (CARN) and its variant CARN-Mobile (CARN-M). We first build our CARN model to increase the performance and extend it to CARN-M to optimize it for speed and the number of operations. Following the FSRCNN [21], the CARN family takes the LR images and computes the HR counterparts as the output of the network. The middle parts of our models are designed based on the ResNet [22]. The ResNet architecture has been widely used in deep learning-based SR methods [23, 14] because of the ease of training and superior performance. In addition to the ResNet architecture, CARN uses a *cascading mechanism* at both the local and the global level to incorporate the features from multiple layers. This has the effect of reflecting various levels of input representations to receive more information. In addition, CARN-M allows a user to tune the trade-off between the performance and the *heaviness* of the model. It does so through the efficient residual block (residual-E) and recursive network architecture.

Besides, we further improve the CARN and CARN-M to being photo-realistic SR methods. Even though SR performance continues to be enhanced, there still exists a gap between the quantitative scores and human-perceived judgment. Various methods including the CARN family adopt pixel-based (or distortion-based) error functions (e.g., mean squared error or L1 loss), to train the network. Minimizing such objectives leads to a high peak signal-to-noise ratio (PSNR) score, which is a commonly used quality measure in the SR community. However, the ability to restore the high-frequency details in such cases is limited, since pixel-based error functions only capture the difference between two images pixel-wise. Moreover, they often result in blurry output images, thus usually disagreeing with the subjective evaluation scores given by human judges. To address such shortcomings, several deep learning-based methods perceptually optimize their network to improve human-visual quality. Starting with SRGAN [24], most of the models that aim for good perceptual quality employ the generative adversarial network (GAN) [25] paradigm and perceptual loss [26]. Enhanced SRGAN (ESRGAN) [16] achieves the best perceptual quality by improving both the generator and the discriminator simultaneously.

To overcome such an issue, we adopt adversarial training to build photo-realistic CARN and CARN-M (PCARN, PCARN-M). More specifically, we set the CARN(-M) as a generator and attach an additional discriminator network that distinguishes whether the input images are from the HR or the SR set (Figure 2.1). Additionally, we also enhance the discriminator by using a multi-scale discriminator strategy instead of using a single discriminator to make the model produce images with high perceptual quality. The multi-scale discriminator consists of multiple networks, where each network is in charge of handling a certain scale. It improves the ability of the generator and the discriminator to preserve the details by taking into account both the coarse and fine textures.

2.2 Background

Deep super-resolution. The performance of the SR has been greatly improved with the powerful capabilities of the deep learning-based methods. As a pioneer work, SRCNN [27] surpasses the traditional approaches by firstly using a deep learning-based model. However, SRCNN requires large computation resources compared to its depth, since the model takes upsampled images as an input. On the other hand, ESPCN [28] takes an LR image as an input, after which it upsamples the image at the end of the network. This strategy reduces the computation substantially compared to the *early-upsample* scheme.

A shortcoming of the aforementioned methods is that they only use a few convolutional layers because of training instability. To tackle this issue, VDSR [29] introduces global residual learning and shows significant improvement over the previous methods by stacking more layers. The global residual learning maps the LR image $\mathbf{x}$ to its residual image $\mathbf{r}$. Then, it produces the SR image $\tilde{\mathbf{y}}$ by adding the residual back to the original, *i.e.*, $\tilde{\mathbf{y}} = \mathbf{x} + \mathbf{r}$. The ESCN [30] uses an ensemble technique to overcome the training instability and to increase the representation power. All the methods mentioned directly super-resolve to the desired spatial resolution, resulting in unsatisfying quality when the input is severely downsampled. To tackle this issue, recent studies use a progressive upsampling [31], which upsamples the intermediary features periodically to restore the image gradually.

One possible disadvantage of applying a deep SR method is the efficiency of the network. That is, there is a problematic increase in the size of the model. To address this concern, most previous studies [18, 19, 20] aim to build a lightweight model in terms of the number of parameters. DRCN [18] and MemNet [19] use a recursive layer to boost the SR quality without additional parameters. Similarly, MSLapSRN [20] ties the parameters of each scale-wise block and takes advantage of both the recursive scheme and progressive approach, resulting in superior SR performance in terms of both SR quality and efficiency. However, many of the parameter-efficient methods use very deep networks to compensate for degraded SR performance caused by the use of the recursive scheme and thus require heavy computing resources. On the other hand, we aim to build a model that is lightweight in both size and computational aspects.

Photo-realistic super-resolution. Generally, deep learning-based SR networks are trained using pixel-based (or distortion-based) loss functions (*e.g.,* MSE or L1 loss). The network with these objectives can be optimized easily, but it tends to create blurry artifacts and fails to recover the structural details. This characteristic can be problematic since a human can judge the absence of high-frequency information effortlessly [24]. Hence, to overcome the inherent issue of using pixel-based losses, a generative adversarial network (GAN) [25] has been adopted to the SR field [24]. By doing so, GAN-based methods show promising results in preserving human-perceptive quality. However, since using only an adversarial loss makes the training process unstable, most of the GAN-based models are trained with the addition of pixel losses [24, 16]. To overcome the inherent problems of using pixel-based losses, Johnson et.al., [26] introduces the perceptual loss that calculates the distance between the embedded features of two output images.

To increase the perceptual quality, EnhanceNet [32] and TSRN [33] adopt texture matching loss [34] in combination with adversarial training and perceptual losses. By providing texture information, the model can produce more realistic textures and reduce artifacts. ESRGAN [16] improves the SRGAN by replacing the standard residual unit [14] with the residual-in-residual dense block (RRDB) inspired by the SRDenseNet [15]. In addition, this model uses the relative discriminator loss [35]. However, the aforementioned models are not suitable for real-world applications despite the great visual quality of the SR output, because of the heavy computational requirements. On the contrary, our proposed models create photo-realistic images with a reasonable amount of computation.

In the photo-realistic SR task, measuring the quality of the resulting image is a major issue. There are many studies that propose distortion-based metrics for image quality assessment such as structural similarity [36]. But these approaches do not always reflect the human's perception of visual quality, and some metrics often contradict human judgment [24]. Considering the trade-off, we mainly use NIMA [37] and LPIPS [38] perceptual quality metrics as our benchmark test. NIMA predicts the distribution of human opinion scores using a deep network. It makes the assessment non-referentially, where all the evaluation is done without ground-truth images. LPIPS measures the perceptual quality by using the distance between the features generated by a pretrained network. Upon the pretrained network, they add an extra linear layer and fine-tune it to a human-perceptual dataset. In our experiments, we use a fine-tuned AlexNet [39] as the pretrained network needed to compute these measures.

Efficient deep neural network. There has been a rising interest in building a small and efficient network [40]. These approaches can be categorized into three groups: **1)** Compressing pretrained networks using pruning or quantizing techniques, **2)** transferring knowledge of a deep model to a shallow one, and **3)** designing small but efficient models. In this section, we summarize the latter category, which aims to build a lean neural network in terms of design engineering, as it matches our approach most closely.

Iandola et.al., [41] introduces SqueezeNet to build a parameter-efficient architecture based on AlexNet [39]. By doing so, they achieve comparable classification accuracy with 50× fewer parameters than the baseline model. Unlike SqueezeNet, MobileNet [40] aims to decrease the number of operations to reduce the inference runtime. This model

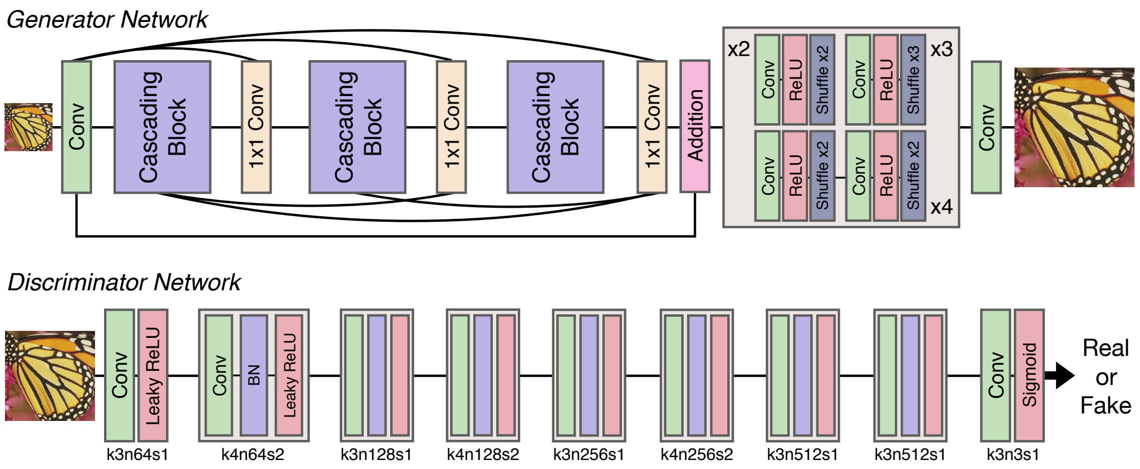
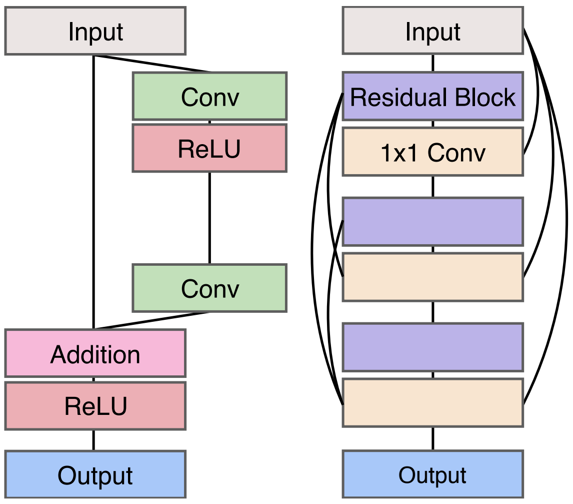


Figure 2.1: **Network architecture of CARN and PCARN.** (**Top**) Generator network. This network consists of cascading blocks and upsample blocks. (**Bottom**) Discriminator network with corresponding kernel size (k), number of feature map (n), and stride (s) indicated for each convolution layer.

decomposes the standard convolution to 1×1 and depthwise separable convolutions. While the MobileNet effectively cuts down the computational cost, 1×1 convolution becomes the new bottleneck and thus can be the limitation to pushing down the overall cost. To mitigate this issue, ShuffleNet variants [42] use the channel shuffle unit following the 1×1 group convolution. Referring to the recent literature [40, 42], we apply a depthwise separable convolution technique in residual blocks to build a fast and lightweight SR model. Instead of using depthwise separable convolution, however, we use group convolution to make the efficiency of the network tunable.

2.3 Approach

In this section, we first introduce cascading residual network (CARN) in Section 2.3.1. Then, in Section 2.3.2, we describe photo-realistic CARN (PCARN) which is the improved version in terms of the perceptual quality. Finally, we discuss on how to make more efficient SR networks by showing CARN-M and PCARN-M (Mobile) in Section 2.3.3.



(a) Residual Block (b) Cascading Block

Figure 2.2: **Structures of local blocks in CARN and PCARN.** (a) Residual block. (b) Cascading block composed of residual blocks and local cascading connections.

2.3.1 Cascading Residual Network

The main architecture of our generator (CARN) is based on the EDSR [14]. The prime difference between EDSR-like networks and ours is the presence of local and global cascading modules. Figure 2.1 (top) graphically depicts how global cascading occurs. The outputs of intermediary features are cascaded into the higher blocks and finally converge on a single 1×1 convolution layer. Note that the intermediary modules are implemented as cascading blocks, which also host cascading connections themselves in a local way. Such local cascading operations (Figure 2.2b) is identical to a global one, except that the backbone units are the residual blocks.

To express how cascading works formally, we first define the standard residual block (Figure 2.2a) as $R_i(H^{i-1}; W_R^i)$, where H^{i-1} is the input feature of the i-th residual block and W_R^i is the parameter set of the convolution layers inside of each residual blocks. Then, we replace the residual block with the local cascading block (Figure 2.2b). To formulate the local cascading as well, we denote $B^{i,j}$ as the output of the j-th residual block in the i-th cascading block, and W_{local}^i as the set of parameters of the i-th local cascading block.

Then, the i-th local cascading block B^i_{local} is defined as in Equation 2.1.

$$B^i_{local}\left(H^{i-1}; W^i_{local}\right) \equiv B^{i,U}, \tag{2.1}$$

where $B^{i,U}$ is defined recursively from the $B^{i,u}$'s as:

$$\begin{aligned} B^{i,0} &= H^{i-1} \\ B^{i,u} &= g\left(\left[B^{i,0}, \ldots, B^{i,u-1}, R^u\left(B^{i,u-1}; W^u_R\right)\right]\right) \quad \text{for } u = 1, \ldots, U, \end{aligned} \tag{2.2}$$

where, g is a 1×1 convolution layer. Finally, we define the output of the final cascading block H^B by combining all H^i's for $i = 0, \cdots, b-1$.

$$\begin{aligned} H^0 &= f\left(\boldsymbol{X}; W_c\right) \\ H^B &= g\left(\left[H^0, \ldots, H^{b-1}, B^B_{local}\left(H^{b-1}; W^b_B \right)\right]\right) \quad \text{for } b = 1, \ldots, B, \end{aligned} \tag{2.3}$$

where $\mathbf{X}$ is the input LR image, f is the first convolution layer (with parameter W_c) of the network, and W^b_B is the parameter set of each cascading block. By applying the cascading mechanism on the local and global levels, we can get two advantages:

1. The model incorporates features from multiple layers, which allows learning multi-level feature representations.

2. The multi-level cascading connection operates as a multi-level shortcut connection that easily propagates information from lower to higher layers (and vice-versa, in the case of back-propagation). Hence, the network can reconstruct the LR image based on multi-level features, and the upsampling unit also upsamples images by taking diverse features (from multiple layers) into account.

Thus, our design helps the model to boost SR performance. We will show how such modules effectively work in Section 2.4.3. Inspired by VDSR [29] and EDSR [14], we apply the multi-scale learning by embedding all up-sample blocks to a single network (Figure

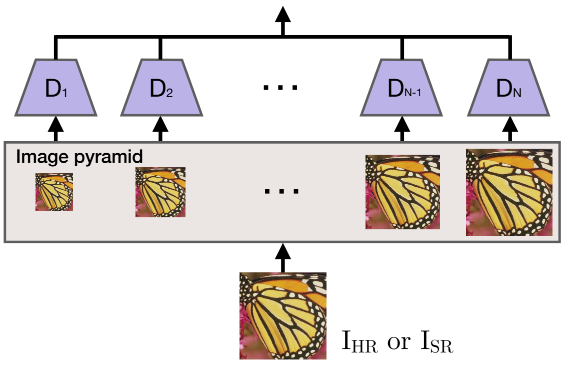


Figure 2.3: **Illustration of the multi-scale discriminator.** Input image is resized using average pooling and each downsampled image is taken by the corresponding scale discriminator. Total discriminative loss is calculated by summing over all the scale.

2.1). The benefit of using such a strategy is that it can process multiple scales using a single trained model. It also helps us alleviate the burden of multiple model sizes when deploying the SR application on small devices since our (P)CARN family only needs a single network for multiple scales.

In addition to the cascading scheme, we use the following design choices to advance the network performance: **1)** Inspired by the VDSR [29], we adopt the global residual learning to our framework. To do that, we aggregate the output of the entry layers and the final 1×1 convolution layer right before the upsampling block. Formally, it can be written as $O = H^b + H^0$, where the final feature map O becomes the input to the upsampling block. The effect of this final addition might appear redundant since the output of the first convolution is already added to the 1×1 before being added again in the next step. Nonetheless, we found that this duplicate addition is beneficial to the overall SR performance with little computational overhead. **2)** We find that nonlinearities (ReLU in this case) following the 1×1 convolution layer marginally decreased the performance so that we remove these. On the other hand, we add nonlinearities in the upsampling unit to increase the expressive power of the network.

2.3.2 Improving the Perceptual Quality

Following the GAN formulation [25], we define a discriminator network D, which we optimize in an alternative procedure along with the CARN generator G. Using the discriminator and the generator, we denote the adversarial loss as:

$$L_{GAN}(G, D) = \mathbb{E}_{I_{HR}} [\log D(I_{HR})] + \mathbb{E}_{I_{LR}} [\log(1 - D(G(I_{LR})))] \,, \tag{2.4}$$

where I_{HR} and I_{LR} denote the HR and LR images, respectively. The idea of adversarial loss is that it trains the generative model G to fool the discriminator D, whereas the discriminator is trained to distinguish whether the images are from the SR or the HR sets. This formulation encourages the generator to create perceptually superior images compared to the pixel-based (distortion-based) losses.

Many previous works have mixed the adversarial loss with a traditional pixel-based loss to stabilize the training process [24, 16]. In this case, the task of a generator is not only to fool the discriminator but also to create an SR image similar to the HR. We also take this option but use the VGG loss [26] instead of the pixel-based loss to avoid blurriness. The VGG loss is defined as the distance between the outputs of the ReLU layers of the pre-trained VGG-19 network [43]. Formally, we denote the output feature map of the j-th ReLU following a convolutional layer before the i-th pooling layer as $\phi_{i,j}$. Then, we define the VGG loss as the L2 distance between the feature representation of the HR image I_{HR}, and the super-resoluted image $G(I_{LR})$:

$$L_{VGG}(G) = \frac{1}{W_{i,j}H_{i,j}} \sum_{x}^{W_{i,j}} \sum_{y}^{H_{i,j}} \left[\phi_{i,j}(I_{HR})_{x,y} - \phi_{i,j}(G(I_{LR}))_{x,y}\right]^2 . \tag{2.5}$$

Here, $W_{i,j}$ and $H_{i,j}$ are the spatial resolutions of the feature map. We use $i = j = 5$.

To enhance the fine details of the computed outputs, we adopt the multi-scale discriminator strategy (Figure 2.3). The idea is to use multiple discriminators instead of a single one to make each discriminator handle a specific scale. Thus, it allows the model to gather information across coarse- to fine-resolution images. To do so, we first downsample the input image (SR or HR) to make an image pyramid. Then, the scaled images are fed into the corresponding discriminators, and finally the multi-scale discriminator loss L_D^{MS} is calculated by collecting each of the losses as in the equation below.

$$L_D^{MS} = \sum_i^S D_i(F_i(\mathbf{I})), \tag{2.6}$$

where $\mathbf{I}$ is the input image and $F(.)$ is the scale-specific downsample function. In all our experiments, we use average pooling as the downsampling module and set S as three.

The total loss for the generator is computed by summing the GAN and VGG losses as:

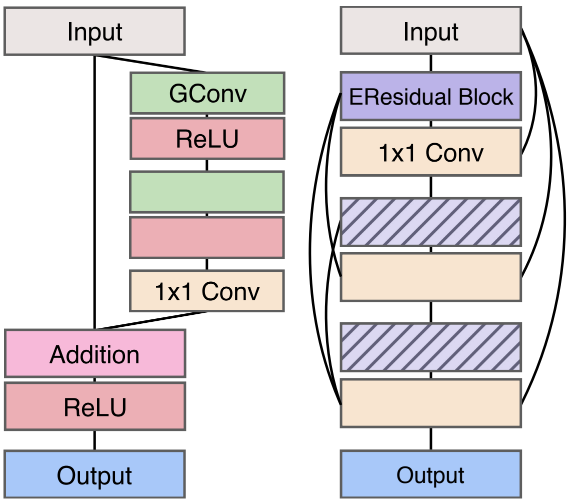
$$L_G = L_{GAN}^{MS} + \lambda L_{VGG}, \tag{2.7}$$

where L_{GAN}^{MS} denotes the adversarial loss in terms of the generator with multi-scale discriminator and λ is the hyperparameter to balance the two losses.

2.3.3 Improving the Efficiency

To improve the efficiency of CARN and PCARN, we propose an efficient residual and cascading block. This approach is analogous to the MobileNet [40], but we use group convolution instead of depthwise separable convolution. Our efficient residual (EResidual) block is composed of two consecutive 3×3 group convolutions and a single pointwise convolution (Figure 2.4a). The advantage of using group convolution over the depthwise separable convolution is that it makes the efficiency of the model manually tunable. Thus, the user can choose the appropriate group count for the desired performance, since the number of groups and the performance are in a trade-off relationship.

The analysis of the efficiency of the EResidual block usage is as follows. Let K be the kernel size and C_{in}, C_{out} be the number of input and output channels. Since we retain the spatial resolution of the feature map by the padding, we can denote F to be both the



(a) EResidual Block (b) ECascading Block

Figure 2.4: **Simplified structures of efficient cascading blocks.** (a) Efficient residual block, and (b) is the efficient cascading block. Hatched boxes in (b) denote the residual block with a parameter tying.

input and output feature size. Then, the cost of a standard residual block is

$$2 \times \left(K^2 \cdot C_{in} \cdot C_{out} \cdot F^2\right). \tag{2.8}$$

Note that we exclude the cost of addition or nonlinearity, and consider only the convolution layers. This is because both the standard and the efficient blocks have the same number of such modules and these occupy a negligible portion of the entire computational cost.

Let G be the number of groups. Then, the cost of an EResidual block, which consists of two group convolutions and one 1×1 convolution, is as given in Equation 2.9.

$$2 \times \left(K^2 \cdot C_{in} \cdot \frac{C_{out}}{G} \cdot F^2\right) + C_{in} \cdot C_{out} \cdot F^2 \tag{2.9}$$

Hence, by changing a standard residual block to our efficient block, we can reduce the computation by the ratio of

$$\frac{2 \times \left(K^2 \cdot C_{in} \cdot \frac{C_{out}}{G} \cdot F^2\right) + C_{in} \cdot C_{out} \cdot F^2}{2 \times (K^2 \cdot C_{in} \cdot C_{out} \cdot F^2)} = \frac{1}{G} + \frac{1}{2K^2}. \tag{2.10}$$

Because we use a 3×3 kernel size and the number of the channels is constant except the entry, exit, and upsampling block, the EResidual block reduces the computation from 1.8 up to 14 times depending on the number of groups. To find the best trade-off between SR quality and computation cost, we perform an extensive case study (Section 2.4.5).

To further reduce the parameters, we apply a technique that is used by the recursive network [18]. In other words, we force the EResidual blocks to be shared in the cascading block, so only one-third of the parameters are needed compared to the standard block. Figure 2.4b shows our efficient cascading (ECascading) block after applying such a scheme. Unlike the previous studies that adopt the recursive scheme [18], we do not increase the depth or the width of the network, so the number of operations is kept the same.

2.3.4 Differences with Prior Works

Difference with MemNet. MemNet [19] and ours have a similar motivation, but there are two main distinctions from our schemes. **1)** Feature fusion is done in a different location and manner. For instance, MemNet fuses the output features of each recursive unit at the end of the memory blocks. On the other hand, we gather the information at every possible site in the local block, thus can boost up the representation power via additional layers. **2)** MemNet takes an *early-upsample* approach which upsamples the image before giving it to the model. Although it becomes easier to implement residual learning, it worsens the model efficiency substantially. In contrast, our model gets LR images and intermediate features are upsampled at the end of the network, which enables us to accomplish a good balance between the SR quality and efficiency.

Difference with DenseNet. SRDenseNet [15] uses a densely connected block and skip connection. Although the overall design concept can be similar, our model has two main advantages. **1)** In our models, the output of each block is associated with a global cascading connection which is a generalized form of the skip connection. In SRDenseNet, all levels of features are combined after the final dense block, but our global cascading scheme connects all blocks, which behaves as a multi-level skip connection. **2)** The connectivity schemes that we use are economical for both memory and speed. n a densely connected block [15], concatenated features are distilled to reduce the number of channels only at the end of the block. Such a block design can allow fluent information flow (since no channel reducing operation exists), but it requires a high amount of computation because blocks have to carry all the intermediate features. In contrast, we incorporate features using an additional 1×1 convolution layer at each concatenation point, which facilitates composing more lightweight models.

2.4 Experiment

2.4.1 Experimental Setting

Datasets. We use the DIV2K dataset [44], which consists of 800 training and 100 validation images in 2K resolution. Because of the richness of this dataset, recent SR models [14, 16] use DIV2K as well. To prepare the training input, we randomly crop images to the 48×48 LR patches and augment to horizontal flip or rotation. To enable the multi-scale training, we first randomly select the scale from [2, 4]. Then we construct the training batch using a chosen scale since our model can process only a single scale for each batch. For the test and benchmark, Set5 [45], Set14 [46], B100 [47] and Urban100 [48] datasets are used.

Implementation details. For the CARN generator, we set $B = U = 3$ and the number of channels in all convolutional layers to 64 except the first, last layer, and upsample block. For the upsampling unit, we use the *pixelshuffle* layer following the convolutional layer that is proposed in ESPCN [28]. Our discriminator (of PCARN) network has 9 convolutional layers (Figure 2.1). We train our models with ADAM by setting $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ in 6×10^5 steps. The minibatch and patch sizes are 64 and 48×48, respectively.

We use the initial learning rate as 10^{-4} and halved every 4×10^5 steps. All the weights and biases are initialized by $\theta \sim U(-k,\ k)$ with $k = 1/\sqrt{c_{in}}$ where, c_{in} is the number of channels of input feature map. In Section 2.4.4, we will describe the detailed analysis of the weight initialization strategy.

To train our model in a multi-scale manner, we first set the scaling factor to one of $\times 2$, $\times 3$, and $\times 4$ because our model can only process a single scale for each batch. We use the L1 loss as our loss function instead of the L2. The L2 loss is widely used in the image restoration task due to its relationship with the peak signal-to-noise ratio (PSNR). However, in our experiments, L1 provides better convergence and performance. The downside of the L1 loss is that the convergence speed is relatively slower than that of L2 without the residual block. However, this drawback could be mitigated by using a ResNet-style model. With trained CARN in pixel-based loss, we set CARN as a generator and train both the generator and discriminator alternatively to make the perceptually improved CARN (PCARN). When train in an adversarial manner, we fine-tune the generator for 2×10^5 steps following the training strategy of SRGAN [24].

2.4.2 Evaluation Metric

To evaluate the performance of the image restoration model, including SR, we use following metrics throughout the thesis: peak-signal-to-noise ratio (PSNR), structural similarity index (SSIM) [49], learned perceptual image patch similarity (LPIPS) [50], and neural image assesment (NIMA) [37]. PSNR is defined using the maximum pixel value and mean-squared error between the two images in the log-space. SSIM [49] measures the structural similarity between two images based on the luminance, contrast and structure. Note that we use Y channel only (by first converting the RGB image into YCbCR colorspace) when calculating PSNR and SSIM unless otherwise specified.

Although high PSNR and high SSIM of an image are generally interpreted as a good image restoration quality, it is well known that these metrics cannot represent human visual perception very well [50]. LPIPS [50] has been recently proposed to address this mismatch. It measures the perceptual distance of the restored images using the L1 dis-

Table 2.1: **Analysis of the effect of model design choices.** `Local`, `Global` and `L/G` means the models with local, global, and both cascading connection respectively.

Model	**Params.**	**PSNR**	**SSIM**
Baseline	963K	28.42±0.02	0.7773±3e-4
+ Local	1,074K	28.45±0.01	0.7780±3e-4
+ Global	1,000K	28.47±0.02	0.7787±4e-4
+ L/G	1,111K	28.49±0.02	0.7788±3e-4
+ Residual	1,111K	**28.50±0.01**	**0.7792±3e-4**

tance between features extracted from the pre-trained AlexNet [39] on distortion dataset specially designed for the quality assessment so that this gives a better perceptual score between two images than the traditional pixel-based metrics. Similarly, NIMA [37] is designed of purpose measuring the human-perception score of the image. However, unlike LPIPS, NIMA is a non-reference image assessment metric; no reference image is needed to calculate the score. To do that, this metric uses the pre-trained network on the image and its paired aesthetic score based on the human rating. This concept leads the NIMA score to reflect the human opinion score of the given image.

To compare the efficiency, we mainly use the number of the network parameters and the number of the operations. The latter measurement is based on MultAdds, which is the number of composite multiply-accumulate operations for a single image. We assume the HR image to be 720p (1280×720) to calculate MultAdds. MultAdds is calculated by the multiplication of the parameters of the layer with the output resolution of the feature, thus this is increased when the network is enlarged or the feature map size is expanded.

2.4.3 Model Design Analysis

To investigate the performance of the proposed methods, we analyze our models via ablation study. We select the baseline to be the SRResNet [24]. Other models have the same topology except for the inherent modules (e.g., additional 1×1 convolution) that are needed for each particular architecture. Thus, the overall number of parameters is increased by up to 15% from the baseline.

Table 2.2: **Model analysis study.** `MSD` indicates the multi-scale discriminator.

Model	LPIPS	PSNR	SSIM
CARN (L1)	0.289±1e-3	28.50±0.01	0.7792±3e-4
+ GAN	0.162±5e-3	26.36±0.25	0.7120±6e-3
+ MSD	**0.155±2e-3**	26.10±0.17	0.6980±8e-3

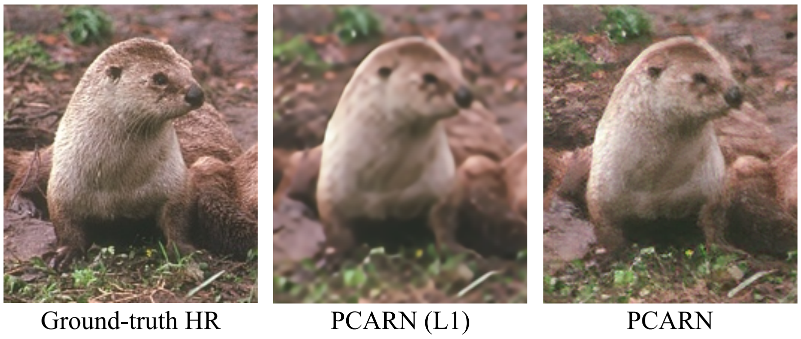


Figure 2.5: **Visual comparison of adversarial training.** We compare the results trained with PCARN and without the adversarial training, PCARN (L1).

Table 2.1 shows the model analysis on the effect of cascading modules and the global residual learning scheme. A model with local cascading improves the baseline SR performance. We conjecture that this is because the cascading module passes not only the inputs but also the mixture of intermediate features to the next block, thus leveraging multi-level representations. By incorporating multi-level representations, the model can consider a variety of information from many different receptive fields when reconstructing the image. We observed higher performance gain with the global cascading scheme. This is because the advantages of the local scheme are limited to each block, which lessens the model's ability to exploit the cascading effect. One major benefit of the global cascading is that it allows information integration from lower layers, and this information shortcut provides useful clues for reconstructing the HR image in the upsampling and final reconstructing processes. Besides, we employ the global residual learning scheme shown in many recent SR methods [29]. The benefit of using it can be minor, since the roles of the global cascading and residual learning overlap. However, we choose to embed the global residual learning into our framework since it does improve performance with negligible overhead.

To build a photo-realistic SR model, it is essential to design a well-functioning discriminator. To examine how the choice of discriminator affects the SR performance, we conducted a series of comparisons across various types of discriminator losses by using the LPIPS [38] metric. As shown in Table 2.2, PCARN with adversarial training (`+GAN`) outperforms the baseline by a large margin. Figure 2.5 also shows the advantage of using GAN, where it successfully recovers the fine details and generates more photo-realistic images. However, because the model is not directly optimized using pixel-based loss, the performance of pixel-based metrics (PSNR and SSIM) is degraded. Moreover, the overall training process is substantially unstable and shows a high variance in all metrics. Using the multi-scale discriminator (`+MSD`) also gives an additional gain to the LPIPS. The main reason is that all the generators can distinguish between fake and real more easily since each generator covers different receptive fields. This gives the generator more useful information to restore both fine- and coarse-level structures.

2.4.4 Initialization Strategy

Appropriately initializing the network is the key component for boosting performance. To verify the optimal initialization scheme for our model, we experimented by comparing the benchmark to six common initialization schemes: uniform, and normal distribution with various settings, as shown in Table 2.3. Note that we conduct this experiment using the PCARN with L1 loss since when training a model with GAN loss, we use the pre-trained network (with L1 loss) as the starting point. Interestingly, the MSRA initialization [51] (4th row) and the high-range uniform (5th row) were inferior to the other methods. We argue that a narrow 1×1 convolution affects the quality of initialization since the high-variance initial values tend to result in high-variance activations. Multiplying the initial random values by 0.1 (1st$\sim$3rd rows) degrades the performance as well.

We hypothesize that the degraded SR performance of other initialization schemes is mainly due to the saturated activations of the network. If the activation generated from the deep layer saturates, useful information can be lost and the gradient signal can vanish, which results in poor performance of the model [52]. As shown in Figure 2.7, initializing

Table 2.3: **Effect of the weight initialization.** F denotes the number of the input channels (fan-in). **std/range** behaves as a standard deviation for the zero-mean normal distribution ($N(0, \text{std})$), or as a range for uniform distribution ($\text{Unif}(-\text{range}, \text{range})$).

Distribution	std/range	PSNR	SSIM
Normal	$0.1 \times \sqrt{2/F}$	28.46±0.01	0.7781±3e-4
Uniform	$0.1 \times \sqrt{6/F}$	28.47±0.01	0.7782±3e-4
Uniform	$0.1 \times \sqrt{1/F}$	28.45±0.02	0.7775±4e-4
Normal	$1.0 \times \sqrt{2/F}$	28.45±0.02	0.7777±3e-4
Uniform	$1.0 \times \sqrt{6/F}$	28.44±0.01	0.7778±2e-4
Uniform	$1.0 \times \sqrt{1/F}$	**28.50±0.01**	**0.7791±3e-4**

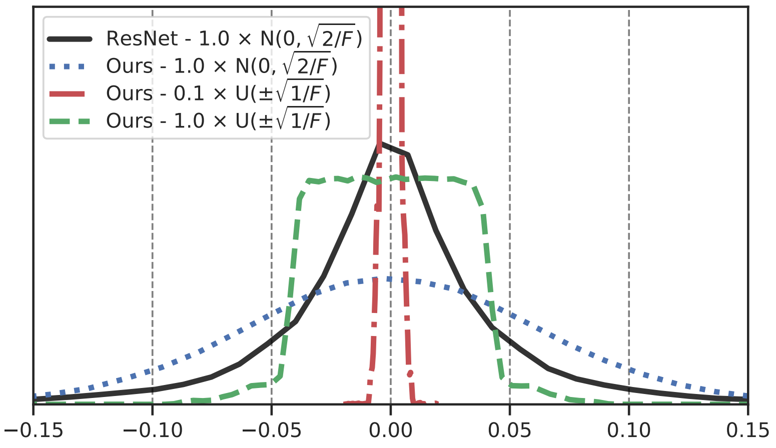
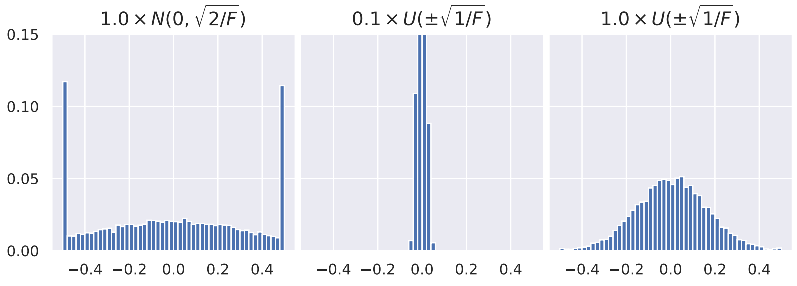
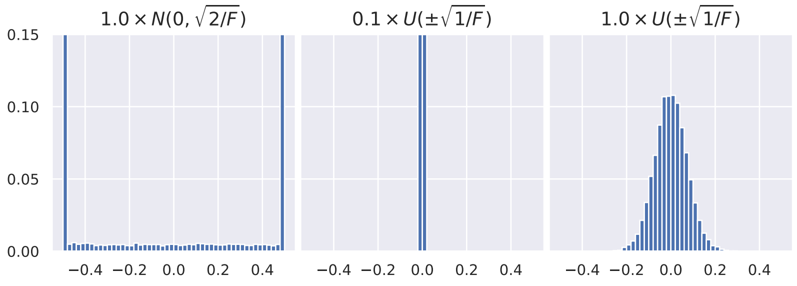


Figure 2.6: **Distribution for the randomly initialized networks.** We plot 100K parameters for each model using 1000 bins. Each of the parameter distributions is from diverse sources because of the variation in the number of input channels.

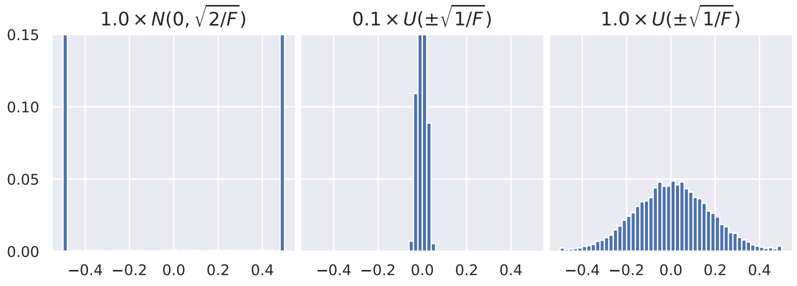
with $1.0 \times U(\pm\sqrt{1/F})$ does not suffer the saturation behavior, while others drive the activations toward zero or infinity. Figure 2.6 explains why such initialization strategies suffer saturation. Unlike the ResNet [53] results (black solid), the weights of our network initialized with $1.0 \times N(\sqrt{2/F})$ (blue dots) have high-variance due to the narrow 1×1 convolutions, so the output activations can be large. On the other hand, initializing with $0.1\times U(\pm\sqrt{1/F})$ (red dots) makes the range of the parameters too narrow, thus saturating activations in deep layers toward zero.



(a) Activation values normalized histogram from the first layer.



(b) Activation values normalized histogram from the second cascading block.



(c) Activation values normalized histogram from the last cascading block.

Figure 2.7: **Activation values normalized histogram from different initializations.** We plot the activation (before ReLU) from the first layer (a), the middle block (b), and the final block (c). A model initialized with $1.0 \times U(\pm\sqrt{1/F})$ (right) effectively carries the signal to the last block, while others tend to saturate activations. Note that we clip the values to [-0.5, 0.5] and normalize the histogram.

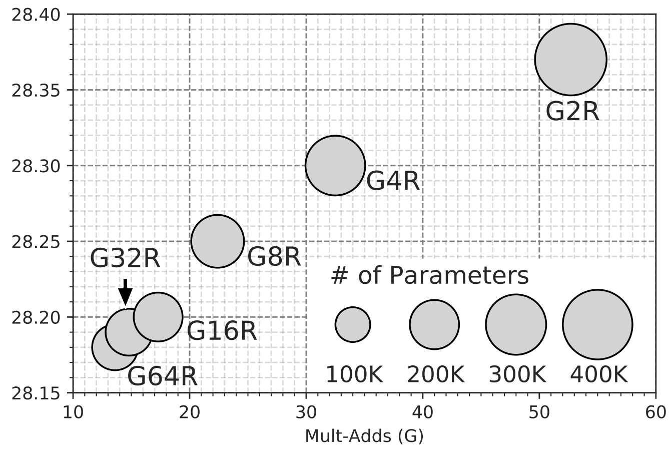


Figure 2.8: **Efficiency analysis of efficient models.** We evaluate all models on Set14 with $\times 4$ scale. `G` represents the number of groups of the group convolution, and `R` means the model with the recursive network scheme.

2.4.5 Efficiency Trade-off

In this section, we dissect the efficiency by varying the group size and the existence of the recursive scheme. Here, for a concise explanation, we confine the SR network as CARN only (which is trained by L1 loss), however, the insight of this analysis beneficial to the photo-realistic version of our method as well. Figure 2.8 depicts the trade-off analysis between the SR performance and efficiency of the efficient CARN that uses convolution and a recursive scheme. We use the model with L1 loss and evaluate using pixel-based metrics, PSNR and SSIM. Although all efficient models perform worse than the CARN, the number of parameters and operations are decreased dramatically. We choose `G4R` as the best-balanced model, which we denote as CARN-M (mobile) since the effect of compressing the model is reduced when the number of groups is larger than four. As a result, CARN-M reduces the number of parameters by four times and the number of operations by nearly three times with a 0.20 dB loss in PSNR and 0.0053 in SSIM, compared to the CARN.

In addition to our trade-off analysis, we also observed that depthwise separable convolution (`G64R`) extremely degrades the performance (-0.32 dB). There can be many explanations why such an observation occurs, but we suspect that this is because the image

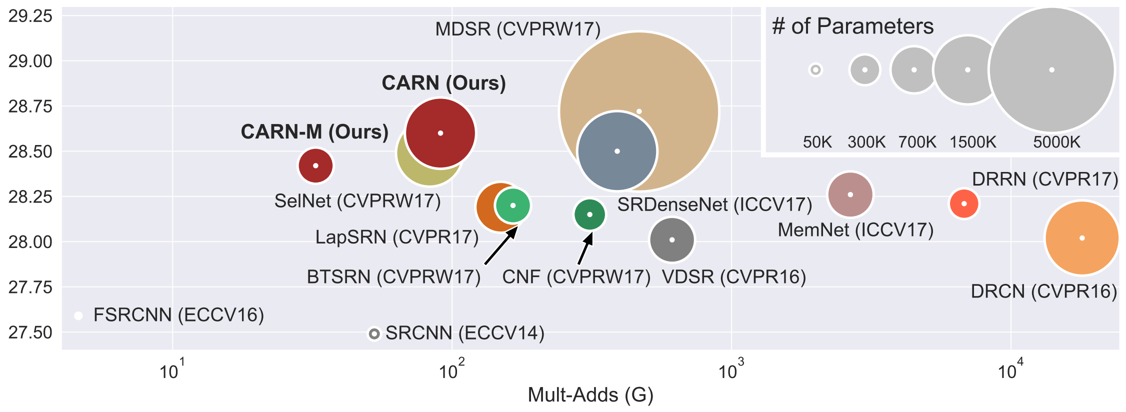


Figure 2.9: **Trade-off between performance vs. number of operations and parameters on Set14 ×4 dataset.** The x-axis and the y-axis denote the Multi-Adds and PSNR, and the size of the circle represents the number of parameters. The Mult-Adds is computed by assuming that the resolution of HR image is 720p.

recognition and generation tasks are entirely different, so applying group and depthwise convolution, which are mainly used in recognition fields, has to be done very carefully. Therefore, creating an efficient SR model (or image generation model in general) needs more investigation with plenty of room to improve performance.

2.4.6 Comparison with Pixel-based Methods

In this section, we compare CARN and CARN-M with previous methods that are trained with pixel-based losses using PSNR and the structural similarity index (SSIM) [36] metrics. In Figure 2.9, we benchmark CARN family against the various benchmark algorithms in terms of the Mult-Adds and the number of the parameters on the Set14 ×4 dataset. Here, CARN outperforms all previous models that have less than 5M parameters. Especially, CARN has a similar number of parameters to that of DRCN [54], SelNet [55] and SRDenseNet [15], but we outperform all three models. The MDSR [14] achieves better performance than ours, which is not surprising because MDSR has 8M parameters which are nearly six times more parameters than ours. The CARN-M model also outperforms most of the benchmark methods and shows comparable results against the heavy models.

Moreover, our models are most efficient in terms of the computation cost: CARN shows second-best results with 90.9G Mult-Adds, which is on par with SelNet [55]. This efficiency mainly comes from the *late-upsample* approach that many recent models [15, 56, 21] used. In addition, our novel cascading mechanism shows increased performance compared to other similar approaches. For example, CARN outperforms its most similar model SelNet by a margin of 0.11 PSNR using a similar number of operations. Also, the CARN-M model obtains comparable results against computationally expensive models, while only requiring a similar number of operations to SRCNN.

Table 2.4 also shows the quantitative comparisons of the performances over the benchmark datasets. Note that MDSR is excluded from this table because we only compare models that have a roughly similar number of parameters as ours; MDSR has a parameter set whose size is four times larger than that of the second-largest model. Our CARN exceeds all the previous methods on numerous benchmark datasets. CARN-M model achieves comparable results using very few operations. We would also like to emphasize that although CARN-M has more parameters than SRCNN or DRRN, it is tolerable in real-world scenarios. The sizes of SRCNN and CARN-M are 200KB and 1.6MB, respectively, all of which are acceptable on recent mobile devices.

To make our models even more lightweight, we apply the multi-scale learning approach. The benefit of using multi-scale learning is that it can process multiple scales using a single trained model. This helps us alleviate the burden of heavy-weight model size when deploying the SR application on mobile devices; CARN(-M) only needs a single fixed model for multiple scales, whereas even the state-of-the-art algorithms require to train separate models for each supported scale. This property is well-suited for real-world products because the size of the applications has to be fixed while the scale of given LR images could vary. Using multi-scale learning in our models increases the number of parameters since the network has to contain possible upsampling layers. On the other hand, VDSR and DRRN do not require this extra burden, even if multi-scale learning is performed, because they upsample the image before processing it.

In Figure 2.10, we visually illustrate the qualitative comparisons over three datasets

Table 2.4: **Quantitative comparison of pixel-based SR methods.** Red/blue text indicates the best/second-best.

Scale	Model	Params	MultAdds	Set5 PSNR/SSIM	Set14 PSNR/SSIM	B100 PSNR/SSIM	Urban100 PSNR/SSIM
2	SRCNN[13]	57K	52.7G	36.66/0.9542	32.42/0.9063	31.36/0.8879	29.50/0.8946
	FSRCNN [21]	12K	6.0G	37.00/0.9558	32.63/0.9088	31.53/0.8920	29.88/0.9020
	VDSR [29]	665K	612.6G	37.53/0.9587	33.03/0.9124	31.90/0.8960	30.76/0.9140
	DRCN [54]	1,774K	17,974.3G	37.63/0.9588	33.04/0.9118	31.85/0.8942	30.75/0.9133
	CNF [57]	337K	311.0G	37.66/0.9590	33.38/0.9136	31.91/0.8962	-
	LapSRN [56]	813K	29.9G	37.52/0.9590	33.08/0.9130	31.80/0.8950	30.41/0.9100
	DRRN [23]	297K	6,796.9G	37.74/0.9591	33.23/0.9136	32.05/0.8973	31.23/0.9188
	BTSRN [58]	410K	207.7G	37.75/-	33.20/-	32.05/-	31.63/-
	MemNet [19]	677K	2,662.4G	37.78/0.9597	33.28/0.9142	32.08/0.8978	31.31/0.9195
	SelNet [55]	974K	225.7G	37.89/0.9598	33.61/0.9160	32.08/0.8984	-
	CARN (ours)	1,592K	222.8G	37.76/0.9590	33.52/0.9166	32.09/0.8978	31.92/0.9256
	CARN-M (ours)	412K	91.2G	37.53/0.9583	33.26/0.9141	31.92/0.8960	31.23/0.9193
3	SRCNN [13]	57K	52.7G	32.75/0.9090	29.28/0.8209	28.41/0.7863	26.24/0.7989
	FSRCNN [21]	12K	5.0G	33.16/0.9140	29.43/0.8242	28.53/0.7910	26.43/0.8080
	VDSR [29]	665K	612.6G	33.66/0.9213	29.77/0.8314	28.82/0.7976	27.14/0.8279
	DRCN [54]	1,774K	17,974.3G	33.82/0.9226	29.76/0.8311	28.80/0.7963	27.15/0.8276
	CNF [57]	337K	311.0G	33.74/0.9226	29.90/0.8322	28.82/0.7980	-
	DRRN [23]	297K	6,796.9G	34.03/0.9244	29.96/0.8349	28.95/0.8004	27.53/0.8378
	BTSRN [58]	410K	176.2G	34.03/-	29.90/-	28.97/-	27.75/-
	MemNet [19]	677K	2,662.4G	34.09/0.9248	30.00/0.8350	28.96/0.8001	27.56/0.8376
	SelNet [55]	1,159K	120.0G	34.27/0.9257	30.30/0.8399	28.97/0.8025	-
	CARN (ours)	1,592K	118.8G	34.29/0.9255	30.29/0.8407	29.06/0.8034	28.06/0.8493
	CARN-M (ours)	412K	46.1G	33.99/0.9236	30.08/0.8367	28.91/0.8000	27.55/0.8385
4	SRCNN [13]	57K	52.7G	30.48/0.8628	27.49/0.7503	26.90/0.7101	24.52/0.7221
	FSRCNN [21]	12K	4.6G	30.71/0.8657	27.59/0.7535	26.98/0.7150	24.62/0.7280
	VDSR [29]	665K	612.6G	31.35/0.8838	28.01/0.7674	27.29/0.7251	25.18/0.7524
	DRCN [54]	1,774K	17,974.3G	31.53/0.8854	28.02/0.7670	27.23/0.7233	25.14/0.7510
	CNF [57]	337K	311.0G	31.55/0.8856	28.15/0.7680	27.32/0.7253	-
	LapSRN [56]	813K	149.4G	31.54/0.8850	28.19/0.7720	27.32/0.7280	25.21/0.7560
	DRRN [23]	297K	6,796.9G	31.68/0.8888	28.21/0.7720	27.38/0.7284	25.44/0.7638
	BTSRN [58]	410K	165.2G	31.85/-	28.20/-	27.47/-	25.74/-
	MemNet [19]	677K	2,662.4G	31.74/0.8893	28.26/0.7723	27.40/0.7281	25.50/0.7630
	SelNet [55]	1,417K	83.1G	32.00/0.8931	28.49/0.7783	27.44/0.7325	-
	SRDenseNet [15]	2,015K	389.9G	32.02/0.8934	28.50/0.7782	27.53/0.7337	26.05/0.7819
	CARN (ours)	1,592K	90.9G	32.13/0.8937	28.60/0.7806	27.58/0.7349	26.07/0.7837
	CARN-M (ours)	412K	32.5G	31.92/0.8903	28.42/0.7762	27.44/0.7304	25.62/0.7694

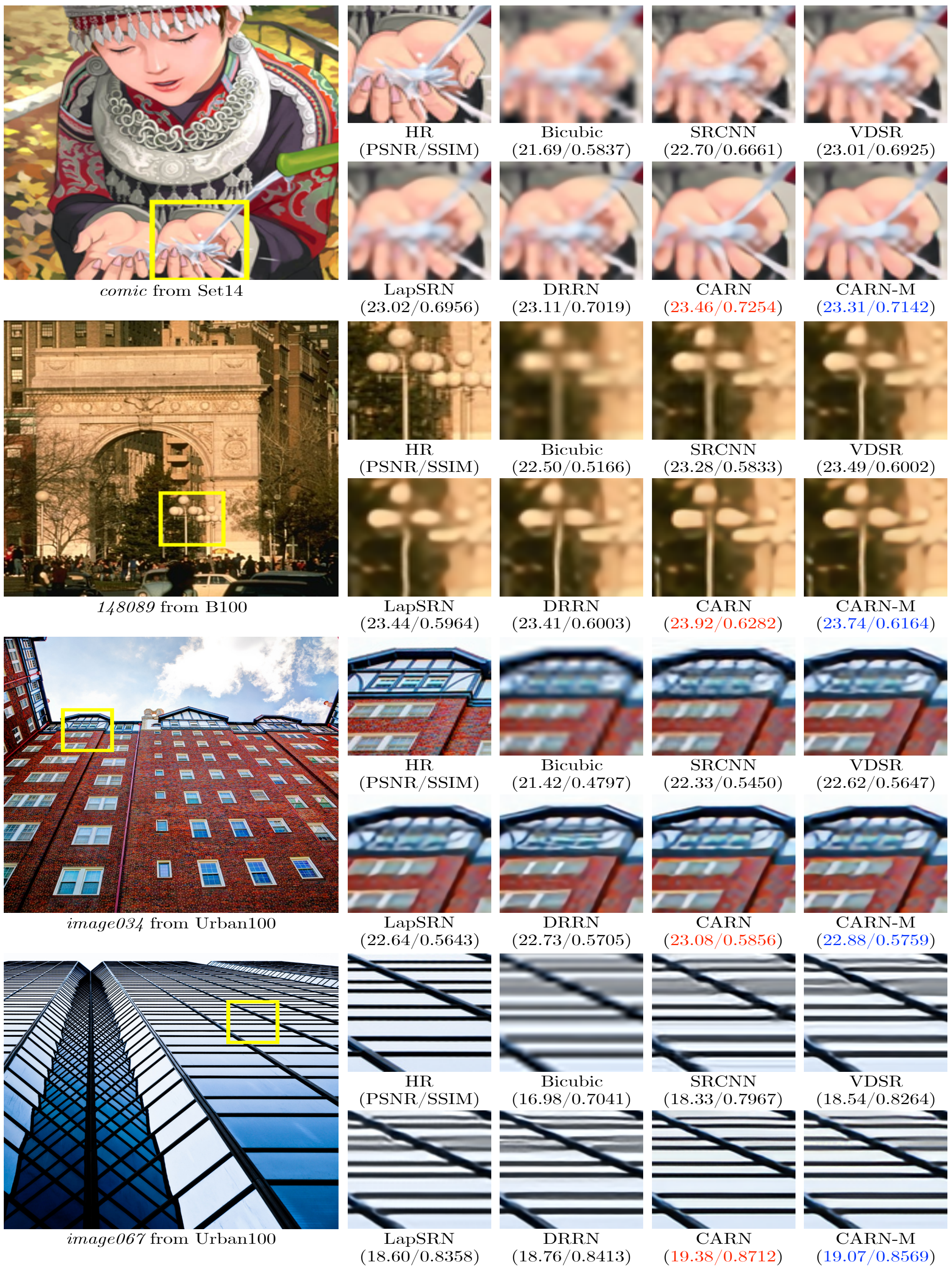


Figure 2.10: **Qualitative comparison of pixel-based SR methods.**

Table 2.5: **Quantitative comparison (LPIPS) of perception-based SR methods.** Lower score is better. `MAs` denotes MultAdds. (**Above**) Pixel-based SR methods. (**Below**) Perception-based SR methods including PCARN variants.

Model	Params	MAs	Set5	Set14	B100	Urban100
SRCNN [13]	57K	52.7G	0.201	0.315	0.410	0.316
MSLapSRN [20]	222K	435.9G	0.178	0.299	0.389	0.252
SRResNet [24]	1,543K	127.8G	0.173	0.284	0.375	0.226
CARN	1,589K	90.9G	0.177	0.290	0.381	0.237
SRGAN [24]	1,543K	127.8G	0.088	0.174	0.203	0.156
ENet [32]	1,073K	120.6G	0.099	0.162	0.210	0.171
TSRN-G [33]	1,073K	120.6G	0.088	0.155	0.196	0.154
PCARN	1,589K	90.9G	**0.075**	**0.145**	**0.188**	**0.152**
PCARN-M	412K	32.5G	0.080	0.150	0.198	0.167

(Set14, B100 and Urban100) for ×4 scale. It can be seen that our model works better than others and accurately reconstructs not only stripes and line patterns, but also complex objects such as hand and street lamps.

2.4.7 Comparison with Perception-based Methods

Here, we benchmark the performance using the perception-based metrics, LPIPS [38] and NIMA [37]. We compare with SRGAN [24], EnhanceNet [32] (shortly ENet), and TSRN [33] (we choose TSRN-G since it is better in LPIPS and NIMA). In addition, we also present the performance of the pixel-based SR networks (SRCNN [13], MSLapSRN [20], SRResNet [24], and CARN) to show the effect of using perceptual-oriented losses.

Table 2.5 and 2.6 depict the quantitative comparisons for the ×4 scale datasets. Among all the methods, our PCARN and PCARN-M have the least MultAdds and a similar number of parameters with the other models. With the limited resources, our models outperform all the competitors in the LPIPS metric and show comparable results on the NIMA score. All the pixel-based methods show lower perceptual scores although the PSNR and SSIM are higher. This is because the pixel loss is not aligned with human perception. For those who fall into the perception-based category, SRGAN, ENet, and TSRN-G have an analogous number of parameters and MultAdds since they design the generator similar

Table 2.6: **Quantitative comparison (NIMA) of perception-based SR methods.** Higher score is better. `MAs` denotes MultAdds. (**Above**) Pixel-based SR methods. (**Below**) Perception-based SR methods including PCARN variants.

Model	**Params**	**MAs**	**Set5**	**Set14**	**B100**	**Urban100**
SRCNN [13]	57K	52.7G	4.32±1.63	4.37±1.69	4.34±1.70	4.62±1.64
MSLapSRN [20]	222K	435.9G	4.67±1.58	4.80±1.66	4.61±1.66	5.04±1.60
SRResNet [24]	1,543K	127.8G	4.74±1.57	4.91±1.64	4.69±1.64	5.14±1.60
CARN	1,589K	90.9G	4.73±1.56	4.91±1.64	4.69±1.64	5.15±1.60
SRGAN [24]	1,543K	127.8G	4.87±1.56	5.00±1.62	4.94±1.64	5.19±1.59
ENet [32]	1,073K	120.6G	4.86±1.58	5.13±1.64	5.07±1.65	5.20±1.59
TSRN-G [33]	1,073K	120.6G	**5.05±1.57**	**5.22±1.61**	**5.17±1.61**	**5.27±1.58**
PCARN	1,589K	90.9G	4.93±1.53	5.06±1.60	5.08±1.61	5.24±1.58
PCARN-M	412K	32.5G	4.87±1.53	4.98±1.61	5.01±1.62	5.18±1.58

to SRResNet. In contrast, our method shows the best performance on the LPIPS metric using a small number of MultAdds (120.G vs. 90.9G). Furthermore, PCARN-M shows comparable results using only one-fourth of both parameters and operations. For the NIMA metric, our PCARN shows comparable results with the TSRN-G, and PCARN-M achieves akin performance to the other competitors.

We also compare the models using pixel-based metrics such as PSNR and SSIM (Table 2.7). All the models trained using the pixel loss outperform perception-based SR methods since the pixel loss is directly related to the pixel-based metrics. Among the perception-based SR networks (below rows), the proposed PCARN family shows the best performance for all benchmark datasets. Such observation shows that our models achieve a good balance between perception and distortion, which is the ultimate goal of the SR algorithm [59].

We also report the perception scores on the various scale factors (Table 2.8). Note that our models are capable of processing multiple scale factors with a single network. Unlike ours, a network without multi-scale training can only restore a specific scale factor, which limits the generalization ability to unseen degradation. However, our methods enjoy such generalization capability by using the multi-scale training strategy.

In Figure 2.13, we illustrate the qualitative comparisons of our methods for the various $\times 4$ scale datasets. It can be seen that our models work better than others and accurately

Table 2.7: **Quantitative comparison (PSNR/SSIM) of perception-based SR methods.** Higher score is better. `MAs` denotes MultAdds. (**Above**) Pixel-based SR methods. (**Below**) Perception-based SR methods including PCARN variants.

Model	Params	MAs	Set5	Set14	B100	Urban100
			PSNR/SSIM	PSNR/SSIM	PSNR/SSIM	PSNR/SSIM
SRCNN [13]	57K	52.7G	30.48/0.8628	27.49/0.7503	26.90/0.7101	24.52/0.7221
MSLapSRN [20]	222K	435.9G	31.74/0.8890	28.26/0.7740	27.43/0.7310	25.51/0.7680
SRResNet [24]	1,543K	127.8G	32.05/0.8910	28.53/0.7804	27.57/**0.7354**	**26.07/0.7839**
CARN	1,589K	90.9G	**32.13/0.8937**	**28.60/0.7806**	**27.58**/0.7349	**26.07**/0.7837
SRGAN [24]	1,543K	127.8G	28.70/0.8281	25.51/0.6763	24.37/0.6190	23.75/0.7063
ENet [32]	1,073K	120.6G	28.85/0.8160	26.03/0.6858	25.26/0.6353	23.62/0.6925
TSRN-G [33]	1,073K	120.6G	28.98/0.8288	25.87/0.6890	25.19/0.6492	23.68/0.7049
PCARN	1,589K	90.9G	**29.75/0.8393**	**26.57/0.7097**	**25.56/0.6534**	**24.17/0.7192**
PCARN-M	412K	32.5G	29.38/0.8369	26.31/0.7087	25.48/0.6570	23.73/0.7043

Table 2.8: **Records of the PCARN variants in diverse scales on LPIPS/NIMA.**

Scale	Model	Set5	Set14	B100	Urban100
		LPIPS/NIMA	LPIPS/NIMA	LPIPS/NIMA	LPIPS/NIMA
2	PCARN	0.019/4.87$\pm$1.55	0.045/5.12$\pm$1.62	0.060/4.98$\pm$1.63	0.040/5.24$\pm$1.60
	PCARN-M	0.023/4.85$\pm$1.58	0.049/5.04$\pm$1.63	0.064/4.94$\pm$1.64	0.047/5.20$\pm$1.60
3	PCARN	0.044/4.87$\pm$1.54	0.096/5.12$\pm$1.60	0.129/5.04$\pm$1.62	0.095/5.21$\pm$1.58
	PCARN-M	0.053/4.79$\pm$1.56	0.106/5.00$\pm$1.60	0.139/4.95$\pm$1.63	0.108/5.17$\pm$1.58
4	PCARN	0.075/4.93$\pm$1.53	0.145/5.06$\pm$1.60	0.188/5.08$\pm$1.61	0.152/5.24$\pm$1.58
	PCARN-M	0.080/4.87$\pm$1.53	0.150/4.98$\pm$1.61	0.198/5.01$\pm$1.62	0.167/5.18$\pm$1.58

reconstruct not only the linear patterns but produce more photo-realistic textures, such as the mane of the lion and pebbles on the ground. Moreover, the proposed networks also generate cleaner outputs while other perception-based methods suffer visual artifacts.

To investigate how our models can generate SR images from different domains, we examine the visual comparison on text images using manga109 [60] dataset (Figure 2.14). Since humans can easily distinguish high-frequency details, it is important to adequately recover the edge region in this task. Here, our method effectively restores various texts, even those with very small fonts that are barely recognizable on the bicubic results. Our GAN-based PCARN can produce sharp and realistic images. However, for some dense structures (Figure 2.11), it generates undesirable artifacts, unlike the PCARN (L1). We suspect that this is a common limitation shared by most, if not all, GAN-based methods.

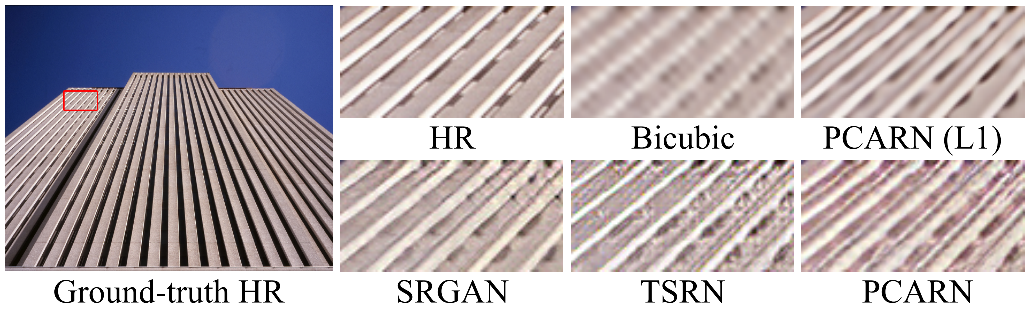


Figure 2.11: **A failure case of PCARN.** Our perception-based model is not able to reconstruct the details without sufficient information when constructing a dense structure.

2.4.8 Execution Time

While MultAdds can reflect the heaviness of the model well, there still exists a misalignment between the true execution times, especially when the models are run on GPUs [42]. To investigate the efficiency in the real devices, we evaluate the inference runtime with other prior networks (Figure 2.12). In this benchmark, we compare computationally-heavy models as well (ESRGAN [16], G-MGBP [61] and EPSR [62]). For fair comparison, we perform inference on the same machine (Intel i5 CPU @3.3 GHz, 32GB RAM, and NVIDIA TITAN X GPU). To calculate the inference time, we use a resolution of 320×240 for the LR input so that the network generates a 720p (1280×720) SR image.

For CPU execution (top row in Figure 2.12), the speed of our PCARN is faster than the other methods such as SRGAN and SRResNet, while it produces a better result and comparable with the EPSR and G-MGBP. Our PCARN-M network is the fastest, while on a par with the heavy models. Such illustration is also reflected by the NIMA metric. The PCARN and PCARN-M can obtain good results at a relatively low computational cost. However, unlike to the results on CPU, our methods do not show such improvement on the GPU (bottom row in Figure 2.12). In fact, our models show slightly worse execution time than the ENet and TSRN-G. The reason is mainly due to the distinct characteristic of CPU and GPU environments. For example, as empirically proved in recent study [42], memory fragmentation reduces the parallelism which worsens the GPU speed a lot. In our case, the cascading mechanism hinders GPU parallelism so that both PCARN and

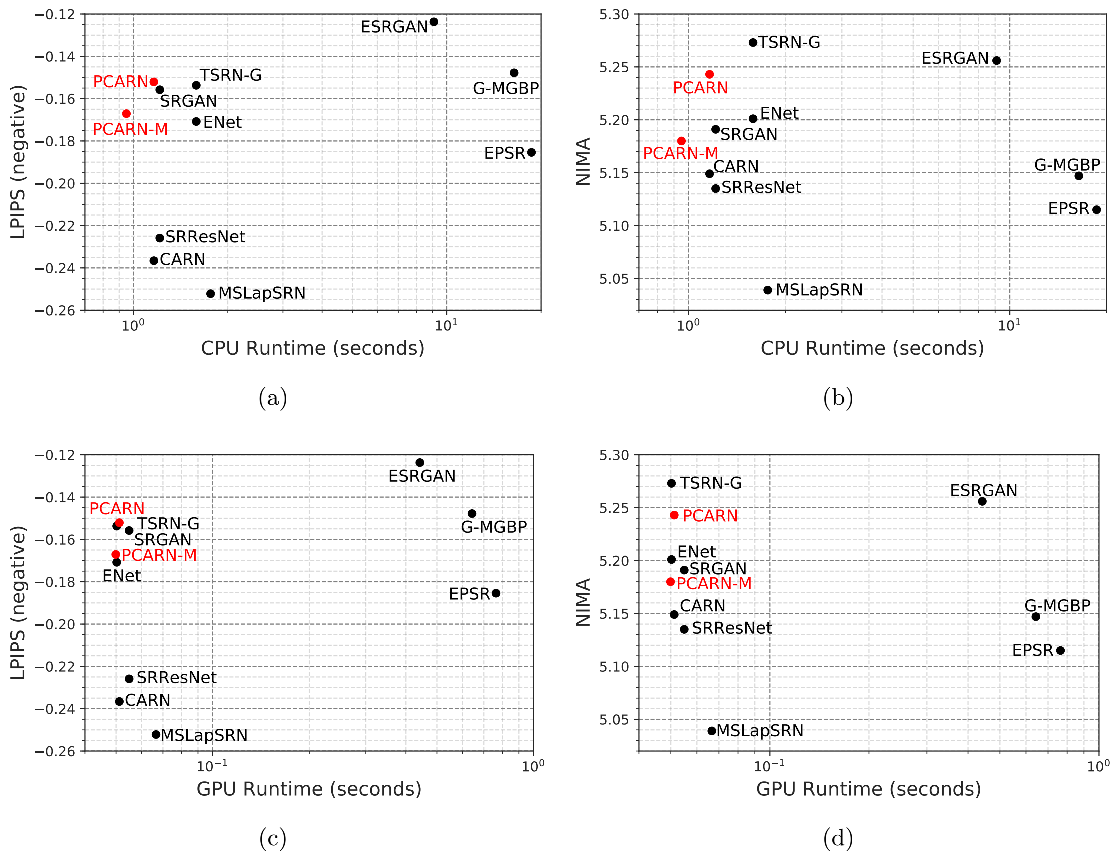


Figure 2.12: **Execution time on the CPU/GPU environments.** We measure the the runtime on CPU and GPU settings with LPIPS and NIMA score. We use negative LPIPS to match the direction of y-axis to the NIMA.

PCARN-M have less advantages on the GPU. Furthermore, group convolution used in PCARN-M is not implemented in a GPU-friendly manner, diminishing the speed gap between PCARN and PCARN-M.

2.5 Discussion

In this chapter, we introduced a deep convolutional network with a cascading scheme for fast and accurate image SR. The core idea is adding multiple cascading connections starting from each intermediary layer to the others on local and global levels. In addition, we enhance our model by using a multi-scale discriminator and achieved improved SR quality over the recent models that have complexity on par with ours.

Future directions. While our methods achieve efficiency, there are remaining issues such as improving the usability and robustness. First of all, the GPU execution time is different from the time taken under the CPU setting (Section 2.4.8), despite the decreased number of MultAdds. This phenomenon comes up because of the discrepancy between the MultAdds and the actually-measured benchmark time. While MultAdds can reflect the inference speed on the CPU, but for the GPU, there are many uncounted operations that MultAdds does not account for. For example, on a GPU, it is critical to reduce the memory access cost or to increase the parallelism level to decrease the inference time as pointed out in ShuffleNetv2 [42]. Therefore, our future goal is to improve our framework and build a GPU-friendly network by carefully modifying our modules and convolution.

Another issue is related to the limitation of the network itself, which manifests in failures such as the bad reconstruction of small textures (Figure 2.11). For the future direction, we hope to use ideas from example-based SR [63] or non-local neural networks [64] to advance the model so that it can effectively enhance severely distorted regions.

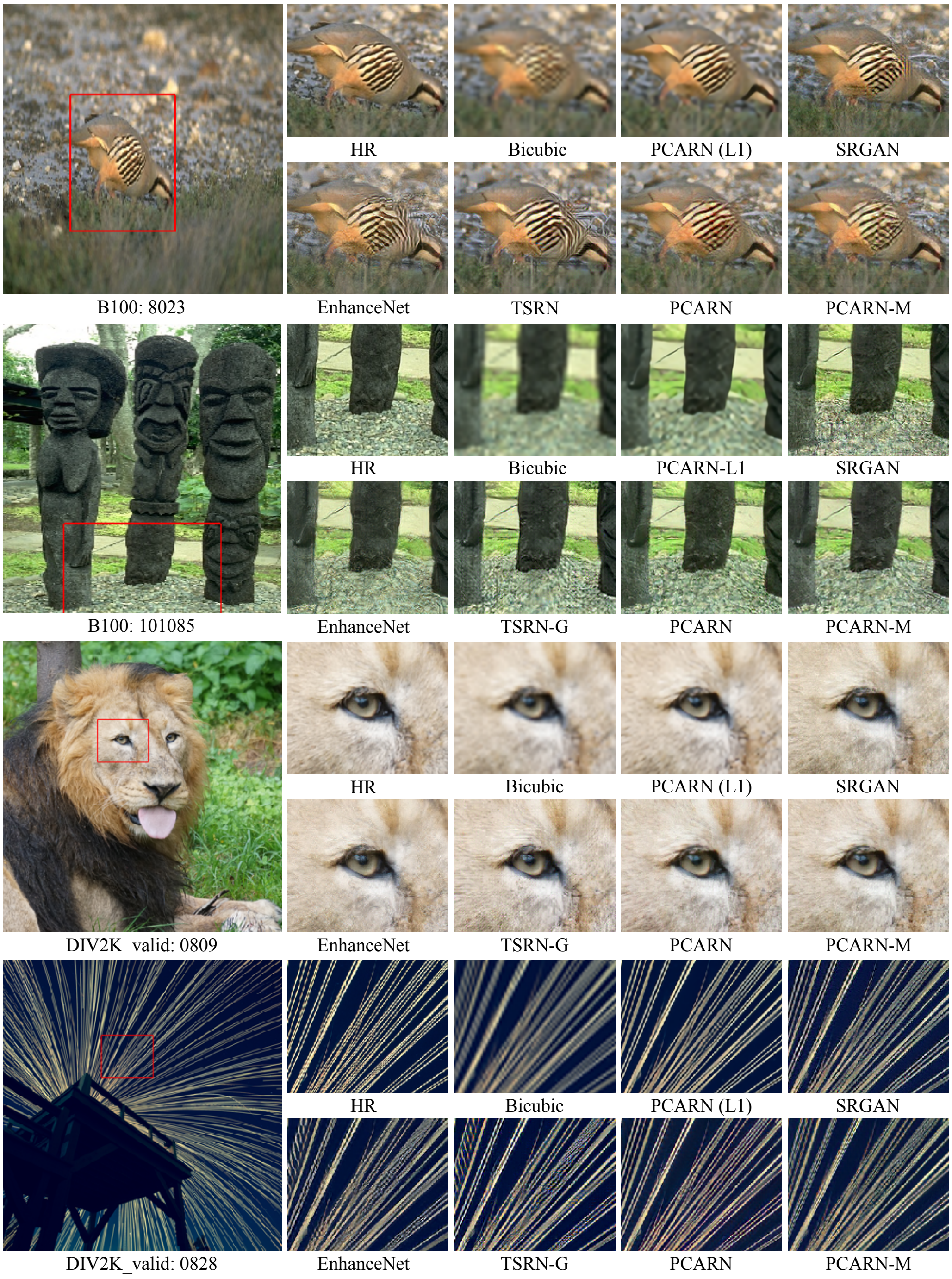


Figure 2.13: **Qualitative comparison of perception-based SR methods.**

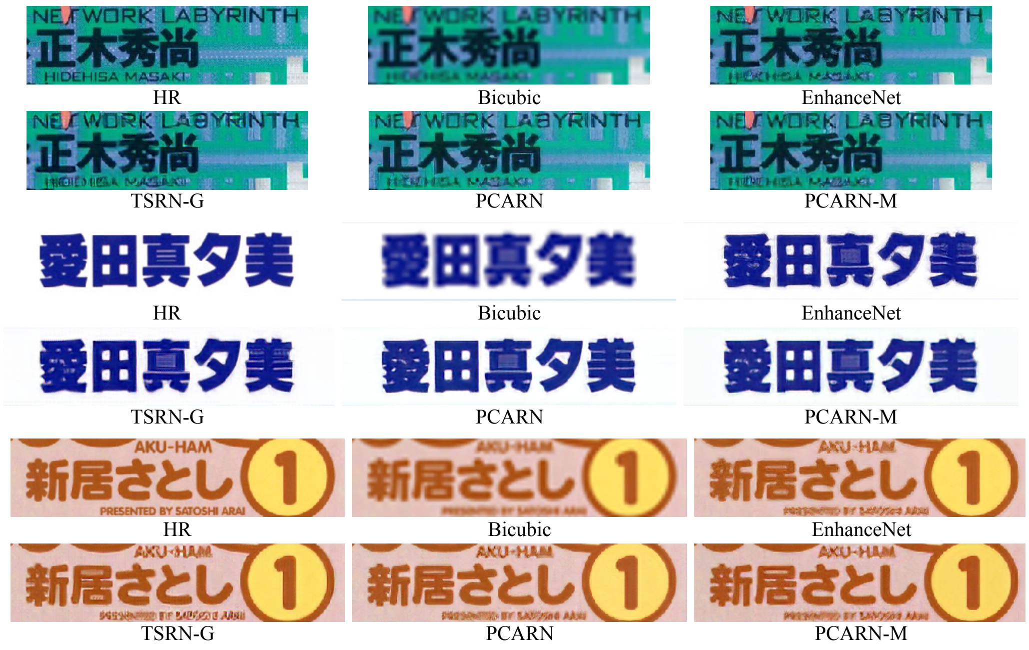


Figure 2.14: **Qualitative comparison of perception-based SR methods on text image.** We benchmark on manga109 [65] dataset (scale $\times 4$).

Chapter 3

Accurate and Lightweight Image Restoration Model

Despite the advancement of deep image restoration methods, many approaches still fail to deal with extreme cases such as $\times 8$ scale super-resolution scenarios because of the instability of training. The lightweight network presented in Chapter 2 also suffers the same issue, thus we might want to scarify some of the efficiency in this extreme condition. To address this, in this chapter, we adopt a progressive learning scheme for the deep convolutional neural network. In detail, the overall training proceeds in multiple stages so that the model gradually increases the output image resolution. In our experiments, we show that this property yields a large performance gain compared to the non-progressive learning methods. Our method also presents a good balance point between the restoration performance and the efficiency compared to the previous heavier networks.

3.1 Overview

Even though the great performance of deep learning-based methods in super-resolution task, their performance can greatly be degraded in extreme SR cases such as restoring $\times 8$ scale downsampled image. This issue is mostly because of the single upsampling step since most of the SR networks only focus on smaller scale resolutions (e.g., $\times 2, 3, 4$). There are not many studies that deal with extreme scales of more than $\times 8$, since it is too hard to infer the extra information needed to upscale an image to such a high resolution. The issue of degradation in extreme SR tasks can be more critical if there are not enough refinement layers after the upsampling process. Recent methods [28, 14] also can suffer from this problem since they upsample the input image at the end of the network.

The difficulty of training for the extreme SR cases comes from the issue of instability in training. This phenomenon occurs when the upsampling is performed at the end of the network, which is a common network design choice that most of the recent methods adopt. This gives rise to the sudden shock to the model when an image comes to the refinement block. Furthermore, the overall quality of the SR image can be degraded since not enough refining process is applied after the upsampling. Especially, the quality issue is more striking when performing extreme SR cases.

To alleviate the difficulty of training a deep convolutional network, one can train a model by a layer-wise training approach. For example, VGGNet [66] first trains a smaller convolutional network and gradually increases the depth of the network to avoid instability of the training. Another approach is adding the auxiliary classifier at the middle of the network to help the information flow of the earlier layer [67]. However, training a very deep network is still an unsolved and challenging problem.

In this chapter, we study how the progressive training learning approach [68] can benefit from the extreme image super-resolution task. Our primary scheme is a training method for extreme SR cases, where we generate a relatively low-resolution output at first, and then progressively increase the output resolution by adding an extra network to our model. This scheme alleviates the instability of training since it can reduce the sudden size change of the model by gradually upsampling the image at the middle of the network. Moreover, the quality degradation issue is reduced by the same mechanism that the instability issue is solved. The instability issue caused by training deep convolutional networks can also be solved by our proposed model. This is because progressive learning works as a fine-tuning of the pre-trained network. Last but not least, the progressive training approach benefits the model efficiency on extreme image super-resolution task. With this concept, we can

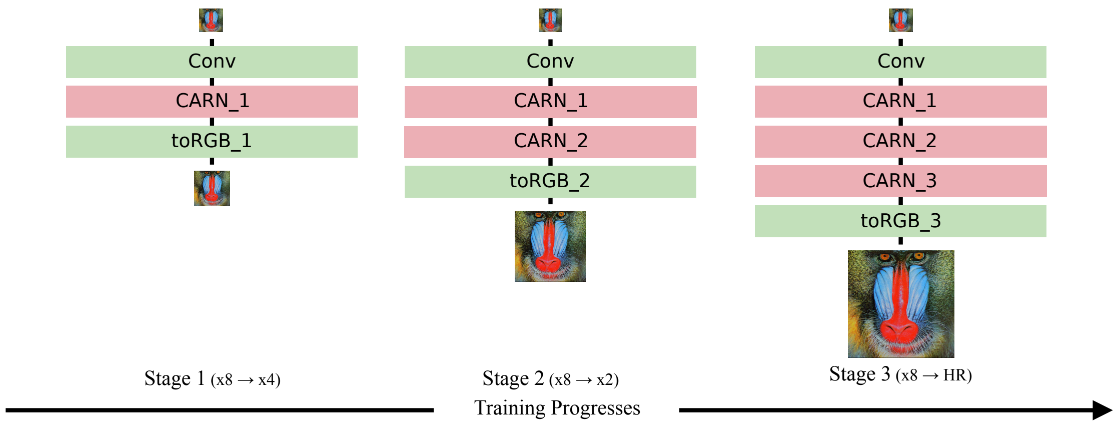


Figure 3.1: **Illustration of progressive training for the extreme SR task.** The number that is denoted in each layer name means the stage number in which the corresponding module is added. For instance, CARN_2 denotes the CARN module that is added in stage two.

achieve high performance even without an *early-upsample* or deeper network so that this help to search a good balance between the high-performance and the lightweightness.

3.2 Background

Progressive training. To further improve the quality of generated high-resolution image, many methods [69, 70, 68] apply a progressive generation scheme to a generative model, which is usually a generative adversarial network [25]. The LapGAN [69] uses a laplacian pyramid [71] so that the model generates the sub-band residual instead of a natural image. During the reconstruction phase, the model reconstructs the natural image by combining the generated residual image and upsampled input LR image. StackGAN [70] generates photo-realistic high-resolution images conditioned on text descriptions via *sketch-refinement*. This method uses a two-stage training procedure which sketches the coarse structure of the objects in the first stage and generates the high-resolution image from the intermediary image in the second stage. To generate very high-resolution images such as 1024 × 1024 resolution, Karras et al., [68] proposed a progressive training methodology for generative adversarial networks. The way progressive training works is that it starts from a low-resolution input and then gradually adds new layers to the model. This makes training the model much easier than the direct learning approach.

Progressive training in SR. Yet, only a few deep learning based SR algorithms apply the progressive approach. Among them, the most successful one is LapSRN [56, 72]. Similarly to the LapGAN, LapSRN uses Laplacian pyramids when restoring images. To this end, from the LR input image, the model progressively increases the image resolution and generates a series of sub-band residual images. The output residual images and the upsampled LR input are combined to produce the final SR image in the reconstruction phase. To further decrease a model size, they share the weights of the component between the multi-level pyramids [72].

3.3 Approach

In this section, we describe the methodology of the proposed progressive method that uses progressive learning based on CARN. Any deep learning-based SISR model can be a backbone network of the progressive approach, but we select CARN because it achieves a good balance between efficiency and performance as we discussed in Chapter 2.

3.3.1 Progressive Cascading Residual Network.

We propose to build our model based on the CARN architecture and progressive learning scheme [68] to effectively reconstruct extremely low-resolution images. The key concept of the methodology is similar to that of Karras et al. [68], but we adapt this scheme for our SR task as shown in Figure 3.1. In detail, we set the number of stages as three in the $\times 8$ scale SR task. That is, in each stage, the model performs $\times 8 \rightarrow \times 4$, $\times 8 \rightarrow \times 2$, and $\times 8 \rightarrow$ HR tasks sequentially.

The training starts from stage one, which produces the $\times 4$ scale image from the first CARN module and the corresponding reconstruction block named *toRGB_1*, as shown in Figure 3.1. The *toRGB* block is a convolution layer that refines the details of the

```
**Algorithm 1:** Overall training process
**Input:** Batch of LR and HR images ($I_{LR}, I_{HR}$),
        # of stages $N$, initial learning rate $\gamma$
**Output:** Trained model $S(I_{LR}; W_S)$
$\Gamma \leftarrow \{\gamma\}$
$S \leftarrow \{f(W_c)\}$ // initial convolution layer
**for** $i \leftarrow 1$ **to** $N$ **do**
    $S \leftarrow S \cup \{M^i\left(W_M^i\right)\}$
    $\Gamma \leftarrow \Gamma \cup \{\gamma\}$
    Attach toRGB_i layer to $S$
    $I_{SR} \leftarrow S(I_{LR}; W_S)$
    Update $W_S$ with $(I_{SR}, I_{HR})$ by corresponding $\Gamma$
    **if** $i < N$ **then**
        Detach toRGB_i layer from $S$
    **end**
    // decay the learning rates of previous modules
    $\Gamma \leftarrow 0.1 \times \Gamma$
**end**
```

upsampled image. After the end of the first stage, we add extra CARN modules to the model and replace the previous reconstruction block with the one that produces the image in double resolution. This training procedure iterates until it reaches the last stage. The output of the final stage is an SR image of the same size as the HR image. The overall training process is shown in Algorithm 1. Note that we use a set-like representation S for the neural network we build and Γ for the set of learning rates.

To further stabilize progressive training, we reduce the learning rate of pre-trained modules ten times. This is a simpler approach than smooth network transition [68], but it also makes the training process stable. The idea behind it is the same as fine-tuning a pre-trained network. However, we found that freezing the pre-trained modules would degrade the overall quality since the information that had to be propagated to earlier blocks could not flow properly.

3.3.2 Why Progressive Training Works?

Applying the progressive training to any SR network alleviates the *instability* of the training problem. This issue comes up when SR tasks are performed using deep architectures, where the final upsampling is usually done abruptly towards the end of the network. This can cause the network's capacity unable to catch up with the sudden increase of the image being processed. In this case, the overall performance might become unstable. On the other hand, the progressive training scheme alleviates this problem by introducing gradual increases in the image resolution.

In terms of the computation cost, using progressive training is also derives a beneficial impact. In case we use a heavier network to avoid the instability of training, this may cause a huge burden of computation. Taking an early-upsample approach may effective to mitigate the unstable training, but this also makes the latency of the entire framework too enlarged, which is not suitable for many applications. The progressive training, however, provides a golden balance between two strategies: alleviate the instability by gradual upsampling without a substantial number of layers. We will discuss the results and model analysis of our approach in Section 3.4.2 and 3.4.3.

3.4 Experiment

3.4.1 Experimental Setting

Dataset. We use the same training and test datasets as described in Chapter 2 except for the image downsample scale. Since the method represented in this chapter tackles the extreme super-resolution task, we downsample the images $\times 8$ scale with Bicubic kernel.

Implementation and training details. We set $B = 4$ and $U = 8$ for all the CARN modules and removed the final convolution layer after the upsampling block since the model produces RGB images at once at the end of the networks (e.g., at the `toRGB` module in Figure 3.1). For the inputs, we use RGB LR images whose patches are of size 48×48

for training. We sample the LR patches randomly and augment them with a random horizontal flip and four 90 degree rotations. We use three-stage progressive training for $\times 8$ scale SR task by setting the batch size 32, 8, and 2 for 1.5×10^5, 2.0×10^5, and 3×10^5 steps, respectively. We train our models with the ADAM optimizer [73] with 10^{-4} as the default learning rate. As described above, we decrease the learning rate of the pre-trained modules by ten times for the training stability.

To boost the performance, we additionally apply geometric self-ensemble [14]. To do this, we flip and rotate LR images to make eight augmented image. Then, we generate SR images from the augmented ones and average these after recovering the original geometry by inverse transformation function. Despite not requiring any extra models, the performance gain is comparable to the inter-model ensemble method. From now on, we add the + symbol to the name of a model to denote the self-ensemble version.

3.4.2 Performance Analysis

To investigate the performance behavior of the proposed method, we analyze the progressive training via an ablation study. Table 3.1 presents the ablation study on the effect of progressive learning. Here, CARN-B3U3 is the default setting of the original model and CARN-B24U4 is the enlarged version of CARN to match the number of network parameters (9.46 million) to the proposed one. This model has 24 cascading blocks and each block has four residual units. Overall, the total number of parameters is the same as that of the progressive CARN, which consists of three B8U4 CARN body modules. Besides, we also show the result of the EDSR [14] to see 1) how progressive training can effectively handle the instability that happens when training a very deep network, and 2) how our approach enhances the model efficiency in terms of both network parameters and the number of operations by preventing to use of excessive capacities.

The CARN-B24U4 outperforms CARN-B3U3 since it has almost eight times more parameters than the latter. The proposed progressive CARN outperforms CARN-B24U4 with a large margin with a similar number of parameters and the progressive model achieves better performance than EDSR as well. The performance gap between the non-

Table 3.1: **Ablation study of the progressive CARN.** We evaluate the number of parameters, running time, and the performance on DIV2K validation dataset $\times 8$ scale. The CARN is the baseline model presented in Chapter 2, and CARN-B24U4 is the enlarged version to match parameters to the progressive CARN. The progressive CARN+ is the model with geometric self-ensemble.

Model	# Params.	MultAdds	DIV2K valid
CARN-B3U3	**1.25M**	**130G**	25.28
CARN-B24U4	9.46M	395G	25.36
EDSR [14]	45.45M	2,862G	25.47
progressive CARN	9.46M	2,144G	25.53
progressive CARN+	9.46M	2,144G	**25.64**

progressive and progressive methods can be attributed to the gradual upsampling manner that alleviates the training instability. For example, CARN-B24U4 and the progressive CARN have nearly identical model sizes, yet the latter outperforms the former. So we can see that having a similar model size does not guarantee similar performance.

We claim that the progressive scheme resulted in an effect that resembles layer-wise training of a deep network. In our case, each intermediary layer receives training signals in the form of LR images. By learning to enlarge the given image to an intermediary-sized image, the progressive model can overcome the performance gap. In addition to the high performance, our model enjoys a better model efficiency compared to the EDSR model. As shown in Table 3.1, progressive CARN uses only 20% of the parameters (45.45M vs. 9.46M) and 75% of the operations used by EDSR (2,862G vs. 2,144G MultAdds), while achieving the superior performance.

The model with self-ensemble also shows a performance gain compared to the one without it. The running time is slower than the without-ensemble version since it cannot be run in parallel to avoid the out-of-memory issue. However, it can be used in many situations since it does not require any extra models when performing ensemble. Also, the running time can be significantly reduced when we run the models in parallel.

Table 3.2: **Quantitative comparison of deep learning-based SR methods.** We evaluate PSNR / SSIM for scaling factor of ×8 on the public benchmark datasets. The two rightmost columns are our methods, and progressive CARN+ denotes the geometric self-ensemble version of progressive CARN. The red text indicates the best performance and blue indicates the second best.

Dataset	Bicubic	LapSRN [56]	CARN	EDSR [14]	Progressive CARN	Progressive CARN+
Set5	24.40/0.658	26.15/0.738	26.72/0.766	27.09/0.781	27.17/0.782	27.28/0.786
Set14	23.10/0.566	24.35/0.620	24.83/0.635	24.96/0.643	25.04/0.644	25.16/0.647
B100	23.67/0.548	24.54/0.586	24.72/0.591	24.81/0.599	24.87/0.600	24.92/0.601
U100	20.74/0.516	21.81/0.581	22.25/0.604	22.55/0.624	22.62/0.626	22.78/0.631

3.4.3 Comparison with State-of-the-art Methods

We compare the proposed model with previous state-of-the-art SR methods [56, 14] on two commonly used image quality metrics: PSNR and the structural similarity index (SSIM) [36]. Table 3.2 shows the quantitative comparisons of the performances for ×8 scale SR over the Set5, Set14, B100, and Urban100 datasets. Here, our proposed progressive CARN outperforms all methods with a large margin over all the datasets. Most of the previous algorithms tend to suffer from the instability problem during training. As mentioned above, this is because the LR images do not have sufficient information to recover, so that upsampling the LR image abruptly can fail to reconstruct the SR image. In addition, our method also shows better performance compared to the progressive upsampling approaches [56]. This advantage can be achieved by progressive training which mitigates the instability of training. Furthermore, the progressive CARN+ achieves even better performance. These observation can also be found in the visual qualitative comparison. As shown in Figure 3.2, our model works better than the others and accurately reconstructs not only stripes and line patterns, but also complex objects such as alphabet type, as depicted in *ppt3* image from the Set14 dataset.

We also visually illustrate the qualitative comparisons among the CARN models and our proposed ones. Somewhat surprisingly, we often observe the degradation issues of the CARN-B24U4, especially for the complex patterns or objects. The images *0821* and *0831* from the DIV2K show the degradation problems that CARN-B24U4 experiences.

However, our proposed method shows a better quality than the the others since it can be trained more stably with a progressive learning scheme.

3.5 Discussion

In Chapter 3, we proposed a progressive cascading residual network that can perform SISR accurately even in an extreme low-resolution scenario. The main idea behind our work is to apply a progressive learning scheme to cascading residual networks. By using the progressive scheme, the training process becomes much easier and more stable, since the model first learns the coarse structure and gradually learns how to restore details in the later stages. In addition, such progressive scheme let the model to achieve a good balance between the high-performance and the efficiency compared to the very deep non-progressive SR networks such as EDSR [14].

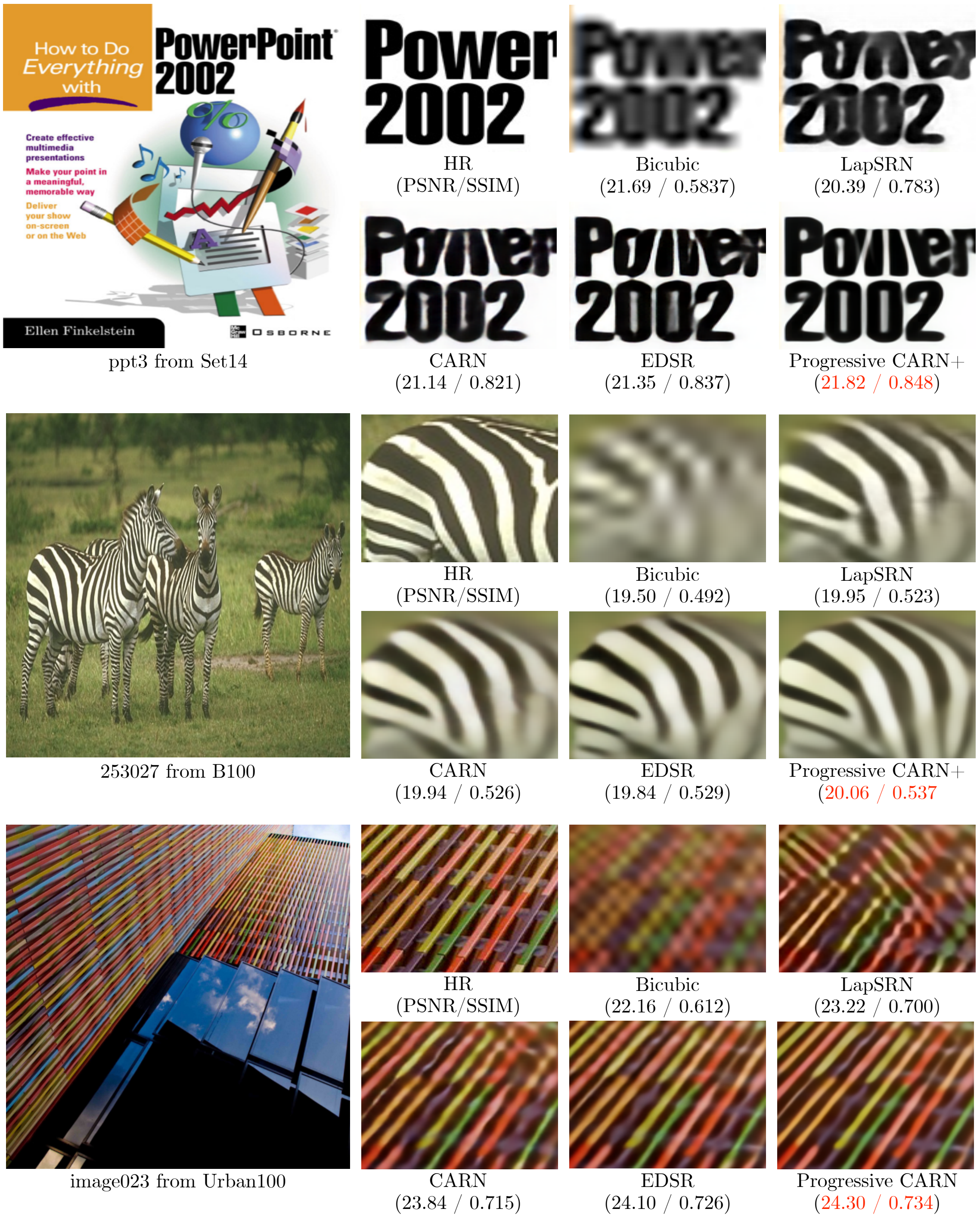


Figure 3.2: **Qualitative comparison on extreme (×8 scale) SR datasets.**

Part II

Data Perspective Efficiency

Chapter 4

Data Augmentation for Low-level Vision

Data augmentation is an effective way to improve the performance of deep networks. Unfortunately, current methods are mostly developed for high-level vision tasks (e.g., image classification) and few are studied for low-level (e.g., image restoration). In this chapter, we provide a comprehensive analysis of the existing data augmentations in the frequency domain. We find that the methods that largely manipulate the spatial information can hinder the image restoration process and hurt the performance. Based on our analyses, we propose CutBlur and mixture-of-augmentation (MoA). CutBlur cuts a low-quality patch and pastes it to the corresponding high-quality image region, or vice versa. The key intuition is to provide enough DA effect while it keeps the pixel distribution intact. This characteristic of CutBlur enables a model to learn not only "how" but also "where" to reconstruct an image. Eventually, the model understands "how much" to restore given pixels, which allows it to generalize better to unseen data distributions. We further improve the restoration performance by MoA that incorporates the curated list of data augmentations. We demonstrate the effectiveness of our methods by conducting extensive experiments on several low-level vision tasks on both single or mixture of distortion tasks. Our results show that CutBlur and MoA consistently and significantly improve the performance especially when the model size is big and the data is collected under real-world environments.

4.1 Overview

Data augmentation (DA) is one of the most practical ways to enhance model performance without additional computation cost in the test phase. While various DA methods [74, 4, 3] have been proposed in several high-level vision tasks, DA in low-level vision has been scarcely investigated. Instead, many image restoration studies, such as super-resolution (SR), have relied on the synthetic datasets [1, 75, 76] that can easily increase the number of training samples by simulating the system degradation functions (e.g., using the bicubic kernel for SR). However, because of the gap between the simulated and real data distribution, models that are trained on synthetic datasets do not exhibit optimal performance in the real environments [77]. Several recent studies have proposed to mitigate this discrepancy by collecting real-world datasets [78, 77, 79]. Still, in many cases, it is often very time-consuming and expensive to obtain a large number of data that are aligned and paired well. Although this is where DA can play an important role, only a handful of studies have been performed in low-level vision tasks [80, 81].

Radu et.al., [81] was the first to study various techniques to improve the performance of SR, one of which was data augmentation. Using rotation and flipping, they reported consistent improvements across models and datasets. However, they only studied simple geometric manipulations with traditional SR models [82, 83] and a very shallow neural network model, SRCNN [1]. To the best of our knowledge, Feng et.al., [80] is the only work that analyzed a recent DA method (Mixup [3]) in the example-based SISR problem. Nonetheless, the authors provided only a limited observation using a single U-Net-like architecture and tested the method with a single dataset, RealSR [77]).

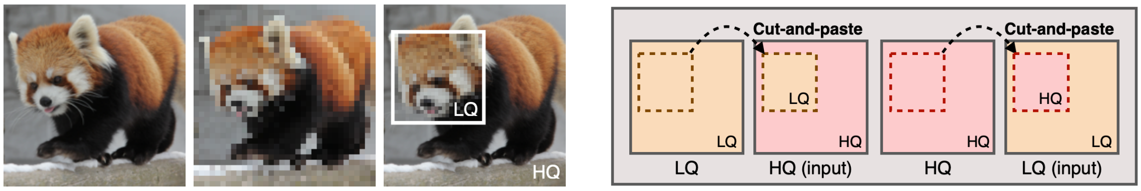
To fill this hole and to better understand the effect of DA methods focusing on low-level vision tasks, a series of analyses are conducted on various models and datasets (Section 4.4). We first categorize the existing augmentation techniques into two groups depending on where the method is applied; pixel-domain [74, 4, 3] and feature-domain [84, 85, 86, 87]. We find that some DA methods harm the image restoration and even hamper the training procedure when directly applied without considering the characteristics of the underlying task. The performance drop is more noticeable especially when a method largely induces the loss or confusion of spatial information between nearby pixels (e.g., Cutout [74] and feature-domain methods). Interestingly, it turns out that basic manipulations like RGB permutation, which do not cause a spatial distortion, provide better improvements than the ones which induce unrealistic patterns or a sharp transition of the structure (e.g., Mixup [3] and CutMix [4]). Our investigation in the frequency domain further reveals that the existing DA methods introduce unwanted changes to the frequency profile of an image, which makes the model hard to learn the intended restoration task.

Based on our analyses, we propose CutBlur, a new augmentation method that is specifically designed for low-level vision tasks. CutBlur cuts and pastes a low quality (LQ) image patch into its corresponding ground-truth high quality (HQ) image patch (Figure 4.1). By having partially LQ and HQ pixel distributions with a random ratio in a single image, it enjoys the regularization effect that encourages a model to learn both *"how"* and *"where"* to resolve the image. One nice side-effect is that the model naturally learns *"how much"* to restore given pixels—it learns to adaptively restore every local part of an image.

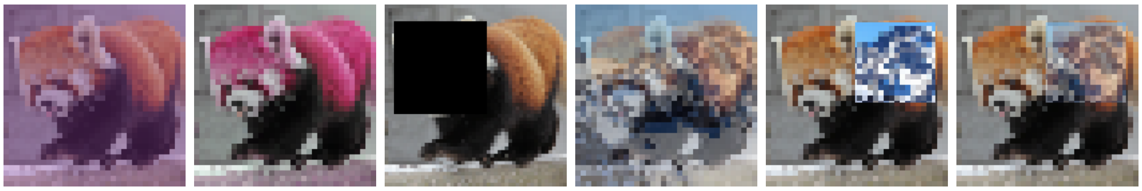
Thanks to this unique property, CutBlur helps the network to generalize better on an unseen pixel distribution and prevents over-correcting it (e.g., over-sharpening in SR and over-smoothing in denoising), which can commonly happen in real-world applications (Section 4.5). This is again clearly seen in the frequency domain analysis, which shows that CutBlur does not alter or add spurious effect on the frequency profile of an image. In addition, we show that the performance can be further boosted by applying several curated DA methods together during the training phase, which we call mixture-of-augmentation (Section 4.3.1). Our experiments demonstrate that the proposed strategy significantly and consistently improves the model performance over various models and datasets.

4.2 Background

Data augmentation in pixel space. Beyond a simple geometric transformation such as the image flipping and rotation, many advanced DA methods have been proposed to



(a) HQ (b) LQ (c) CutBlur (d) Schematic illustration of CutBlur



(e) Blend (f) RGB perm. (g) Cutout [74] (h) Mixup [3] (i) CutMix [4] (j) CutMixup

Figure 4.1: **Overview of the data augmentation methods.** An illustrative example of our proposed method, CutBlur **(Top)**. CutBlur generates an augmented image by cut-and-pasting the low-quality input image onto the ground-truth high-quality image region and vice versa (Section 4.3.1). Illustrative examples of the existing augmentation techniques and a new variation of CutMix and Mixup, CutMixup **(Bottom)**.

manipulate the pixel space of an input image [74, 4, 3, 88, 89, 90]. Mixup [3] interpolates two images to generate unseen training samples. A regional dropout strategy [74, 88], such as Cutout, erases the random region of the image to enhance the generalization ability of the model. Integrating the advantages of both Cutout and Mixup approaches, CutMix [4] replaces the random region with the other image instead of removing it. This enables the model to fully exploit the entire training data, in contrast to the regional dropout. Recently, some studies have focused on developing an effective way (or framework) of applying DA methods. For example, AutoAugment [89] and its variant [90] have proposed to learn an augmentation policy for a given task and dataset.

Data augmentation in feature space. Instead of directly working on pixels, DA methods in this category manipulate the internal features of the network [91, 85, 84, 92, 86, 87]. They are categorized into three groups: 1) feature mixing, 2) shaking, and 3) dropping. Manifold Mixup [86] blends the latent features as well as the input image to provide a smoother decision boundary. Shake-shake [85] and ShakeDrop [87] perform a stochastic affine transformation to the features so as to give a stronger regularization effect.

Similarly, feature dropping strategies [84, 93] boost the generalization ability by erasing some regions of the feature.

Data augmentation for low-level vision task. Despite its huge success in high-level vision tasks, the effect of the DA has not been actively investigated in the low-level vision tasks. To the best of our knowledge, Feng et.al., [80] is the only work that applied a DA method beyond a geometric transformation to a low-level vision task. They utilized Mixup [3] to train the model on a very small dataset. However, their scope is limited to reducing the overfitting issue of a specific super-resolution model and dataset. In contrast, our work provides a comprehensive analysis of various DA methods, covering various models, datasets, and environments. Recently, Helou et.al., have proposed an interesting work that masks the frequency components of an input image during training [94]. This stochastic frequency masking (SFM) framework simulates the arbitrary blur kernel in the SR task and showed a promising regularization effect on the kernel-overfitting issue that prevails in the modern deep SR models. Although it is not originally proposed as a DA method, we find that, in principle, SFM provides an augmentation effect by perturbing the frequency information. However, SFM needs a pre-processing step in the frequency domain using discrete cosine transforms, which also introduces some hyper-parameters to select a central frequency band. Besides, their results are only restricted to super-resolution and denoising. By contrast, CutBlur works on the image domain with a simple cut-and-paste operation while enjoying similar advantages.

4.3 Approach

We describe CutBlur (Section 4.3.1) and mixture-of-augmentation (Section 4.3.2), a new augmentation method and a general framework for low-level vision tasks. Finally, we describe the experimental settings we used throughout this chapter (Section 4.3.3).

4.3.1 CutBlur

Let $x_{LQ} \in \mathbb{R}^{W \times H \times C}$ and $x_{HQ} \in \mathbb{R}^{sW \times sH \times C}$ be low- and high-quality image patches, respectively. Here, s denotes a scale factor that depends on the task of interest; $s = 1$ in many low-level vision tasks (e.g., image denoising), while $s > 1$ in SR. Because CutBlur requires the same resolution of x_{LQ} and x_{HQ}, we first match the size by using a deterministic kernel if necessary; x^s_{LQ}. The goal of CutBlur is to generate a pair of new training samples $(\hat{x}_{HQ \to LQ}, \hat{x}_{LQ \to HQ})$ by cut-and-pasting the random region of x_{HQ} into the corresponding x^s_{LQ} and vice versa:

$$\begin{aligned} \hat{x}_{HQ \to LQ} &= \mathbf{M} \odot x_{HQ} + (\mathbf{1} - \mathbf{M}) \odot x^s_{LQ}, \\ \hat{x}_{LQ \to HQ} &= \mathbf{M} \odot x^s_{LQ} + (\mathbf{1} - \mathbf{M}) \odot x_{HQ}, \end{aligned} \tag{4.1}$$

where $\mathbf{M} \in \{0, 1\}^{sW \times sH}$ denotes a binary mask indicating where to replace, $\mathbf{1}$ is a binary mask filled with ones, and $\odot$ is an element-wise multiplication. For sampling the mask and its coordinates, we follow the original CutMix [4].

4.3.2 Mixture-of-Augmentation

To maximize the advantages of using DA, we integrate various DA methods (presented in Figure 4.1) into a single framework, which we call a mixture-of-augmentation (MoA) strategy. For each training step, it first determines whether to apply DA or not with decision probability p. When deciding to perform DA, it randomly selects a single method among the pool following the ratio vector $\mathbf{r} = \{r_1, ..., r_k\}$, where k is the number of DAs we use. By default, we set p as 1.0 and sample r from the uniform distribution.

4.3.3 Experimental Setting

We apply our method to the various low-level vision tasks using synthetic and realistic data with single or multiple distortions.

Task and datasets. For image super-resolution task, we generate a synthetic SR dataset by downsampling the HQ image with a bicubic kernel. We use images from the DIV2K [44] as a high-quality. For evaluation, Set5 [45], Set14 [95], B100 [47], Urban100 [48], and Manga109 [65] datasets are used. To test realistic SR scenarios, we train and benchmark the model on RealSR (version 1) [77] and CameraSR [96] datasets.

On single distortion restoration tasks, we perform widely used image restoration tasks: image denoising and JPEG artifacts removal. We distort HQ images from DIV2K [44] to generate the dataset for synthetic denoising and JPEG artifacts removal tasks. Model evaluations are done using Kodak24 and Urban100 [48] for denoising, and Classic5 [97] and LIVE1 [98] datasets for JPEG artifacts removal task, respectively. In addition to the synthetic cases, we perform an experiment on a real denoising task using SIDD [78]. For preparing the dataset, we follow Abdelhamed et.al., [99] and denote this as SIDD+.

In the real world, images can be corrupted with multiple distortions. To simulate this scenario, we follow the recently proposed multi-distortion environments. To generate a mixed distortion dataset [8], a sequence of Gaussian blur, noise, and JPEG compression is applied to DIV2K HQ images with random levels. This provides three groups of datasets by the distortion intensity (mild, moderate, and severe). Among them, we train the model on a moderate level and test it to all levels to verify the generalization ability. In addition, we also experiment on the multi-degradation SR task [100, 101]. Unlike a standard bicubic SR, this adds a Gaussian blur or noise to mimic the image acquired from the real environment. Specifically, we use two settings: `SR+DN` and `SR+BLUR`.

Baselines. To benchmark on super-resolution task, we use four models: SRCNN [1], CARN [102], RCAN [2], and EDSR [76], which have different numbers of parameters from 0.07M to 43.2M (million). When training the models on a synthetic SR dataset, we follow the training protocol of the prior work [76]. That is, the network is pre-trained with $\times 2$ scale, then fine-tuned on $\times 4$ scale dataset. Since we upsample the input image $x_{LQ} \in \mathbb{R}^{W \times H \times C}$ to $x_{LQ}^{s} \in \mathbb{R}^{sW \times sH \times C}$ (as described in Section 4.3.1), we attach desubpixel layer [103] before the first layer for efficient inference. In addition, we change the number of input channels in the first layer from C to s^2C. We observe that such modifications do not harm the SR performance as well as the model efficiency.

In general image restoration task except super-resolution, EDSR [76], RDN [101], and RNAN [104] are used for other low-level vision tasks. The number of parameters of these models ranges from 9.0M to 38.4M. For EDSR, we remove the upsampling blocks of the model to apply it to image restoration tasks that do not involve upscaling.

Evaluation metrics. We evaluate the methods using the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) [49]. PSNR is calculated by the mean-squared error between the two images in the log-space. SSIM [49] measures the similarity of the images based on the luminance, contrast, and structure using the statistics of the given images. Note that we use the Y channel only when calculating metrics on SR and JPEG artifact removal tasks, while using full-RGB channels otherwise.

Augmentation setup. We describe detailed descriptions and settings of every augmentation in Table 4.1. Here, `CutMixup`, `CutBlur`, and `MoA` are the strategies that we have newly proposed in this chapter. In addition, to the best of our knowledge, we are the first to use `RGB permutation` and `Blend` as augmentation methods for low-level vision tasks. The hyper-parameters are described following the original papers' notations. Depending on the task and dataset, we use different MoA decision probability p and ratio vector $\mathbf{r} = \{r_1, ..., r_k\}$. We set $p = 1.0$ for super-resolution and $p = 0.6$ for the rest of restoration tasks. Exceptionally, we use $p = 0.2$ when training SRCNN and CARN on the synthetic SR dataset considering the small capacity of these networks i.e., MoA is applied less. When using MoA, the model takes a single augmentation from the pool. By default, we sample $\mathbf{r}$ from the uniform distribution except for the realistic SR task (e.g., CameraSR and RealSR). For realistic SR task, we adjust the ratio $\mathbf{r}$ to have CutBlur 40% chance more than the other DA's, each of which has 10% chance $(40\% + 10\% + 10\% + 10\% + 10\% + 10\% + 10\% = 100\%)$.

Table 4.1: **A description of data augmentations used in MoA.**

Name	Description	Default α
Cutout [74]	Erase randomly sampled pixels with probability α. Cutout-ed pixels are discarded when calculating loss by masking removed pixels.	0.001
CutMix [4]	Replace randomly selected region to sub-patch from other image. The coordinates are $r_x = \text{Unif}(0, W)$, $r_w = \lambda W$, where $\lambda \sim \text{N}(\alpha, 0.01)$ (same for r_y and r_h).	0.7
Mixup [3]	Blend randomly selected two images. We use default setting of Feng et al., [80] which is: $I' = \lambda I_i + (1-\lambda) I_j$, where $\lambda \sim \text{Beta}(\alpha, \alpha)$.	1.2
CutMixup	CutMix with the Mixup-ed image. CutMix and Mixup procedure use hyper-parameter α_1 and α_2 respectively.	0.7/1.2
RGB perm.	Randomly permute RGB channels.	-
Blend	Blend image with vector $\mathbf{v} = (v_1, v_2, v_3)$, where $v_i \sim \text{Unif}(\alpha, 1)$.	0.6
CutBlur	Perform CutMix with same image but different resolution, producing $\hat{x}_{HQ \to LQ}$ and $\hat{x}_{LQ \to HQ}$. Randomly choose $\hat{x}$ from the [$\hat{x}_{HQ \to LQ}$, $\hat{x}_{LQ \to HQ}$], then provided selected one as input of the network.	0.7
MoA	Use all data augmentations described above. Randomly select single augmentation from the augmentation pool then apply it.	-

4.4 Comprehensive Analysis of Data Augmentation

In this section, we investigate existing DA methods and why they fail in low-level vision tasks (Section 4.4.1). Based on these observations, we analyze the characteristics of our methods (Section 4.4.2, 4.4.3) and show its effectiveness under various conditions (Section 4.4.4). Here, for a concise explanation, we confine our scope of applications to SR. However, we later show that these results are also extended well to other low-level vision tasks such as denoising, JPEG artifact removal, and resolving mixed distortions.

4.4.1 Problems with Existing Data Augmentations

The core idea of many augmentation methods is to partially block or confuse training signals so that a model acquires more generalization power [74, 4]. However, unlike the high-level tasks (e.g., classification) where a model should learn to abstract an image, the local and global relationships among pixels are essential in low-level vision tasks (e.g., denoising and super-resolution). With this characteristic, we hypothesize that an operation

Table 4.2: **Quantitative comparison (PSNR) of data augmentations in SR.** We report the performance gap (δ) of the baseline performance w/ and w/o augmentation on DIV2K ($\times$4) [44] and RealSR ($\times$4) [77] datasets. The dashed line separates the existing DA methods with a set of DA methods that we newly proposed for low-level vision tasks.

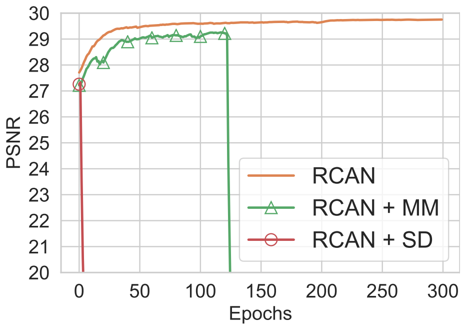
Method	DIV2K (δ)	RealSR (δ)
EDSR [76]	29.21 (+0.00)	28.89 (+0.00)
Cutout [74] (0.1%)	29.22 (+0.01)	28.95 (+0.06)
CutMix [4]	29.22 (+0.01)	28.89 (+0.00)
Mixup [3]	29.26 (+0.05)	28.98 (+0.09)
CutMixup	29.27 (+0.06)	29.03 (+0.14)
RGB perm.	29.30 (+0.09)	29.02 (+0.13)
Blend	29.23 (+0.02)	29.03 (+0.14)
CutBlur	**29.26 (+0.05)**	**29.12 (+0.23)**
Mixture-of-Augmentation (MoA)	**29.30 (+0.09)**	**29.16 (+0.27)**

that severely perturbs spatial information limits model's ability to restore an image.

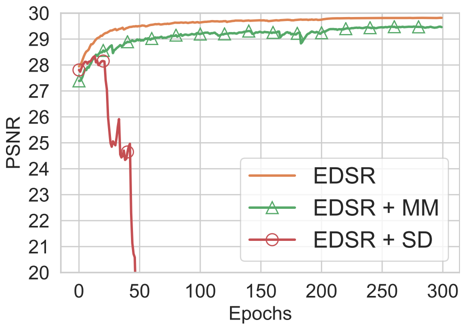
Indeed, DA methods that severely drop or manipulate spatial information give detrimental effects to the SR performance especially when they work in the feature space [91, 84, 93]—every feature augmentation method significantly decreases the performance. To reveal this, we apply feature augmentations [86, 87] to EDSR [76] and RCAN [2] (Figure 4.2). Both Manifold Mixup [86] and ShakeDrop [87] result in inferior performance than the baseline. Specifically, RCAN fails to learn when using any feature-based augmentation. For EDSR, Manifold Mixup is the only one that can be accompanied with, but it also shows a significant performance drop. We hypothesis that the main reason for the catastrophic failure of ShakeDrop is due to its severe manipulation of training signals.

On the other hand, DA methods in pixel space bring some improvements when applied carefully (Table 4.2)[1]. For example, Cutout [74] with default setting drops 25% of pixels in a rectangular shape, and this significantly degrades the performance by 0.1 dB. However, when applied with a 0.1% ratio and erasing random pixels instead of a rectangular region, we find that Cutout gives a positive effect (DIV2K: +0.01 dB and RealSR: +0.06 dB). Note that this drops only 2∼3 pixels when using a 48×48 input patch.

[1]For every experiment, we only used geometric DA methods, flip and rotation. Here, to solely analyze the effect of the DA methods, we did not use the ×2 pre-trained model.



(a) RCAN



(b) EDSR

Figure 4.2: **Training curve comparison of feature space DAs.** `MM` and `SD` denote the model with Manifold Mixup [86] and ShakeDrop [87], respectively.

CutMix [4] shows a marginally better performance than Cutout but less than Mixup [3] (Table 4.2). We hypothesize that this happens because CutMix can use more information than Cutout, while it still generates a drastically sharp transition of image context making boundaries. Mixup avoids this abrupt change by mingling the context of two different images, but at the same time this confuses the SR model by introducing an unnatural pixel distribution. To alleviate these issues, we create a variation of CutMix and Mixup, which we call CutMixup. Interestingly, it gives a better improvement over the others. By getting the best of both methods, CutMixup benefits from minimizing the boundary effect as well as the ratio of the unnatural mixed contexts. Based on these observations, we further test a set of basic operations such as RGB permutation and Blend (adding a constant value to an entire image) that do not incur any structural change of an image. Surprisingly, we observe that these simple methods show promising results in the synthetic DIV2K dataset and a big improvement in RealSR that is considered to be more challenging.

To investigate this more systematically, we analyze the effect of each DA method in the frequency domain (Figure 4.3). To visualize the distribution of frequency content in an image, we use the power spectral density (PSD). The PSD of an image is typically modeled as $1/f^{\alpha}$, which is visible in the profile of the original (unaugmented) image (light green). Here, f is the spatial frequency and $\alpha \in [1, 2]$ varies depending on the scene (natural vs. man-made). Due to the sharp boundary at the cutting edge and the loss

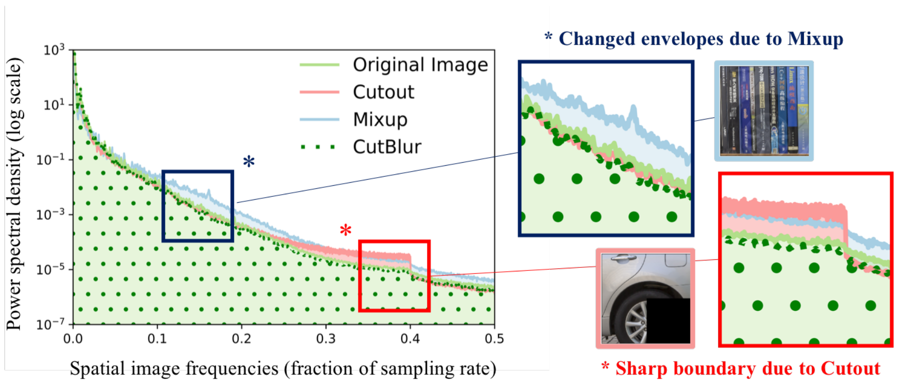


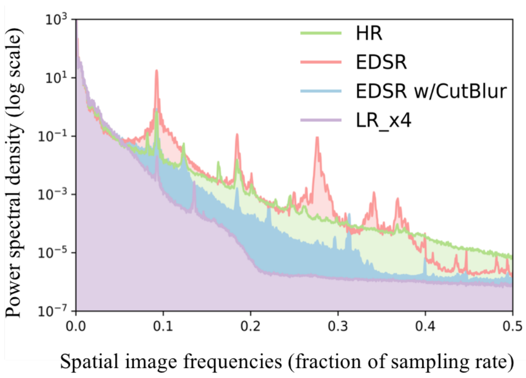
Figure 4.3: **Power spectral density (PSD) of augmented images** The PSD of the original image (light green) is compared to the PSD of its augmented versions; Cutout (red), Mixup (blue), and CutBlur (dotted dark green). While the PSD of CutBlurred image does not largely perturb the image profile, the other methods cause either changed envelopes (Mixup) or a sudden jump (Cutout).

of information, Cutout introduces a sudden jump in the frequency profile with spuriously enhanced high-frequency responses. Mixup significantly changes the PSD envelope in an unnatural way due to the frequency components of the mixing image—note that this is a log scale plot. It is unsurprising that such DA methods incur negative effects for low-level vision tasks. These results naturally leads us to our new augmentation method, CutBlur.

4.4.2 CutBlur

Not only does CutBlur improve the performance (Table 4.2), but it also has several unique and nice properties that cannot be obtained by other DA methods. In this section, we provide a more detailed analysis focused on the properties of CutBlur.

Why CutBlur works for low-level vision tasks? In the previous analysis (Section 4.4.1), we found that sharp transitions or mixed image contents within an image patch, or losing the relationships of pixels can degrade the performance of image restoration models. Therefore, a good DA method for low-level vision tasks should not make unrealistic patterns or information loss while maintaining a good regularization effect. CutBlur satisfies this condition because it cuts and pastes between the pairs of LQ and HQ images of the



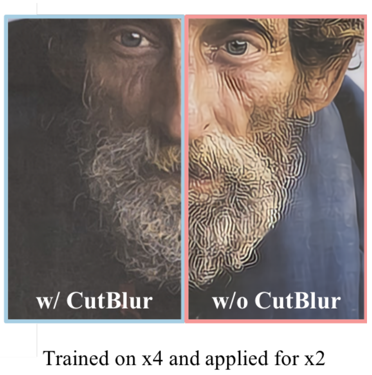


Figure 4.4: **Analysis of the generalization of the baseline w/ and w/o CutBlur.** A slightly higher resolution image is given at test time—the model is trained on x4 task, but x2 task is given (the pixel denseness is doubled). The PSD (**left**) of reconstructed images (**right**) are compared. When the network takes an unexpectedly higher pixel distribution at test time (than it saw during training), the baseline model without CutBlur generalizes poorly and shows various artifacts, while the one trained with CutBlur does not.

same content. By putting the LQ (resp. HQ) image patch onto the corresponding HQ (resp. LQ) location, it can minimize the boundary effect, which majorly comes from a mismatch between the image contents (e.g., Cutout and CutMix). Unlike Cutout, CutBlur can utilize the entire image information while enjoying the regularization effect due to the varied samples of random HQ pixel ratios and locations. This is also clearly seen in the frequency domain analysis (Figure 4.3, dotted dark green). Unlike Mixup, CutBlur minimally hurts the pixel statistics (closely following the original profile) while maintaining the augmentation effect with an immense enhancement of the performance (Table 4.2).

What does a model learn with CutBlur? We hypothesize that the performance enhancement of using CutBlur comes from constraining the SR model to adaptively restore an image, which provides a beneficial regularization effect. More specifically, with CutBlur, the model now has to simultaneously learn both "how" and "where" to super-resolve an image. This naturally leads the model to learn "how much" it should super-resolve a given pixel and thereby generalize better to unseen pixel distributions. Similar to label smoothing [105] that prevents a classifier from making an over-confident decision, CutBlur prevents the model from over-correcting an image.

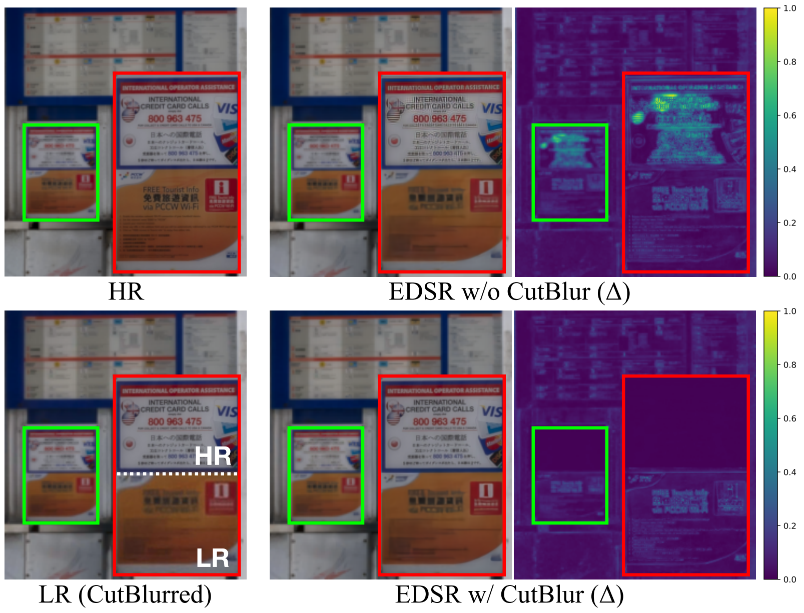


Figure 4.5: **Qualitative comparison of the baseline w/ and w/o CutBlur.** Δ is the absolute residual intensity map between the network output and the ground-truth HR image (**top left**). When the input is augmented by CutBlur (**bottom left**), unlike the baseline (**top right**), CutBlur-model (**bottom right**) not only resolves the HR region (red box) but also reduces Δ of the other LR area (green box).

This can be demonstrated by testing some artificial setups such as giving a model an unexpectedly higher resolution image (Figure 4.4) or a CutBlurred image (Figure 4.5) at test time. In Figure 4.4, the frequency components of LR images lie mostly in low frequencies (the purple line) compared to HR (the light green line). Hence, the goal is to recover high-frequency information using the low-frequency of noisy and coarse-scale signals. Without any specific treatment, when the SR model takes unexpected higher resolution images at test time, it commonly outputs unnaturally over-sharpened predictions especially where the edges are (the right panel of Figure 4.4). The PSD of the baseline model (the red line) clearly shows this with the severely altered PSD envelope, having several peaks in low and high-frequency components. This is a natural behavior since the model learned to super-resolve every given pixel with a fixed scale, no matter how the underlying distribution looks like. CutBlur resolves this issue by showing a mixed pixel distribution to the model during the training phase. As it can be seen in both the reconstructed image and the PSD analysis, when CutBlur is applied (the blue line), the model super-resolves the image (recovers high-frequency components) without introducing serious artifacts (follows the original profile). Figure 4.5 also shows the nice generalization effect of CutBlur. Here, CutBlur-model resolves the given image better than the baseline model in both LR and HR regions. Note that the residual error of CutBlur-model is significantly decreased in both regions. Although this may unfair to compare the models that have been trained with and without such images, we argue that this scenario is realistic; happened in the real-world very often (e.g., out-of-focus images). We will discuss this more in detail in Section 4.5.

CutBlur vs. Giving HR inputs during training. To let a model learn the HR pixel distribution, instead of using CutBlur, one can also think of showing HR images to the model during training. With the EDSR model, CutBlur training showed better performance in PSNR (29.04 dB) than naïvely providing the HR images (28.87 dB) to the network. This is because our CutBlur framework includes the HR input scenario as a special case, where $\mathbf{M} = \mathbf{0}$ or $\mathbf{1}$. On the other hand, giving HR inputs can never simulate the mixed distribution of LR and HR pixels in a single image so that the network can only learn "how", not "where" to super-resolve an image.

4.4.3 Mixture-of-Augmentation

Our analysis shows that, when applied wisely, many DA methods can bring benefits to image restoration models. To get the most out of this observation, we propose mixture-of-augmentation (MoA) that use the curated list of DA methods represented in Table 4.2. With MoA, we achieve the best performance in both synthetic and realistic datasets (Table 4.2). Note that this set of DA methods is never exhaustive, and we believe that there is more room for improvement with new augmentation methods. From now on, unless it is specified, we report all the experimental results using MoA.

Table 4.3: **Analysis on various models and dataset volumes.** We train the models of various sizes on different volumes of DIV2K (×4) dataset. We report PSNR (dB) for each setup w/ and w/o MoA.

Model	Params.	Training Data Size				
		100%	50%	25%	**15%**	**10%**
SRCNN	0.1M	27.95	27.95	27.95	27.93	27.91
+ MoA		-0.02	-0.01	-0.02	-0.02	-0.01
CARN	1.1M	28.80	28.77	28.72	28.67	28.60
+ MoA		+0.00	**+0.01**	**+0.02**	**+0.03**	**+0.04**
RCAN	15.6M	29.22	29.06	29.01	28.90	28.82
+ MoA		**+0.08**	**+0.16**	**+0.11**	**+0.13**	**+0.14**
EDSR	43.2M	29.21	29.10	28.97	28.87	28.77
+ MoA		**+0.08**	**+0.08**	**+0.10**	**+0.10**	**+0.11**

4.4.4 Study on Different Models and Datasets

Various model sizes. It is generally known that a model with a larger capacity benefits more from data augmentation than a small capacity model does. Here, we investigate the relationship between the network size and the performance gain of using MoA to see whether this belief is also valid in low-level vision tasks. In this experiment, we set the decision probability p differently depending on the model size; $p = 0.2$ for small models (SRCNN and CARN) and $p = 1.0$ for the large capacity networks (RCAN and EDSR). When using a full dataset (100%), small models such as SRCNN and CARN do not benefit much from data augmentation in PSNR (Table 4.3)—still, it is noteworthy that the benefit of having CutBlur remains (e.g., suppressing over-sharpening). On the other hand, MoA consistently improves the performance of RCAN (+0.08 dB) and EDSR (+0.08 dB), which have enough capacity to exploit the augmented information.

Various dataset sizes. We further investigate the model performance while decreasing the data size for training using 100%, 50%, 25%, 15% and 10% of the DIV2K dataset(Table 4.3). SRCNN and CARN show none or marginal improvements with our method. This is due to a severe underfitting of small models as it can be also deduced from little differences between the performance of SRCNN with 100% and 10% of the dataset. Here, the effect of DA is minimal due to the lack of model capacity. On the other hand, as soon as the size of a model becomes moderate or big, our method brings a huge benefit in all the settings. The performance gap becomes profound as the dataset size diminishes. With MoA, RCAN trained on only half of the dataset recovers the same performance of using 100% of it ($29.06+0.16 = 29.22$ dB). Our method brings up to 0.16 dB improvement when the dataset size is less than 50%, and this tendency is observed in EDSR as well. This is a promising result because it is usually very hard to acquire a large dataset in many potential real applications.

4.5 Result

In this section, we show the experimental results of our proposed method on diverse low-level vision tasks including image super-resolution (Section 4.5.1), single distortion restoration (Section 4.5.2: denoising and JPEG artifacts removal), and multiple distortions restoration (Section 4.5.3: mixed artifacts, `SR+DN`, and `SR+BLUR`).

4.5.1 Image Super-resolution

Synthetic image super-resolution. In Table 4.4, we compare the performance of the baseline methods on various synthetic benchmark datasets. MoA consistently brings a huge performance gain especially when the models have large capacities. For example, both networks get at least 0.15 dB PSNR gain on Urban100 and 0.20 dB on the Manga109 dataset. As we discussed in Section 4.4.3, even when it does not improve the final performance, all the models enjoy the benefits of better generalization. Qualitative comparison (Figure 4.6) also exhibits the superiority of our method. RCAN and EDSR benefit from the increased performance and successfully resolve the aliasing artifacts (e.g., lines in the first row) and the structural distortions (e.g., holes in the second row).

Realistic image super-resolution. We also demonstrate the benefits of using MoA on real-world SR tasks (Table 4.5). For both RealSR and CameraSR datasets, we observe that MoA successfully boosts the performance in all cases. With MoA, the models gain

Table 4.4: **Quantitative comparison on synthetic SR datasets (scale ×4).** Green/red colors indicate better/worse performance of using MoA than the baseline.

Model	Params.	Bicubic SR				
		Set5 (δ)	Set14 (δ)	B100 (δ)	Urban100 (δ)	Manga109 (δ)
CARN	1.1M	32.00 / 0.8915	28.48 / 0.7787	27.51 / 0.7332	25.85 / 0.7779	30.17 / 0.9034
+ MoA		31.97 / 0.8912	28.48 / 0.7788	27.51 / 0.7332	25.85 / 0.7780	30.16 / 0.9032
RCAN	15.6M	32.50 / 0.8985	28.86 / 0.7879	27.76 / 0.7422	26.76 / 0.8062	31.24 / 0.9169
+ MoA		32.66 / 0.8998	28.92 / 0.7895	27.80 / 0.7436	26.93 / 0.8106	31.46 / 0.9190
EDSR	43.2M	32.50 / 0.8983	28.81 / 0.7871	27.72 / 0.7412	26.66/ 0.8038	31.06 / 0.9151
+ MoA		32.56 / 0.8990	28.88 / 0.7886	27.78 / 0.7430	26.80 / 0.8072	31.25 / 0.9163

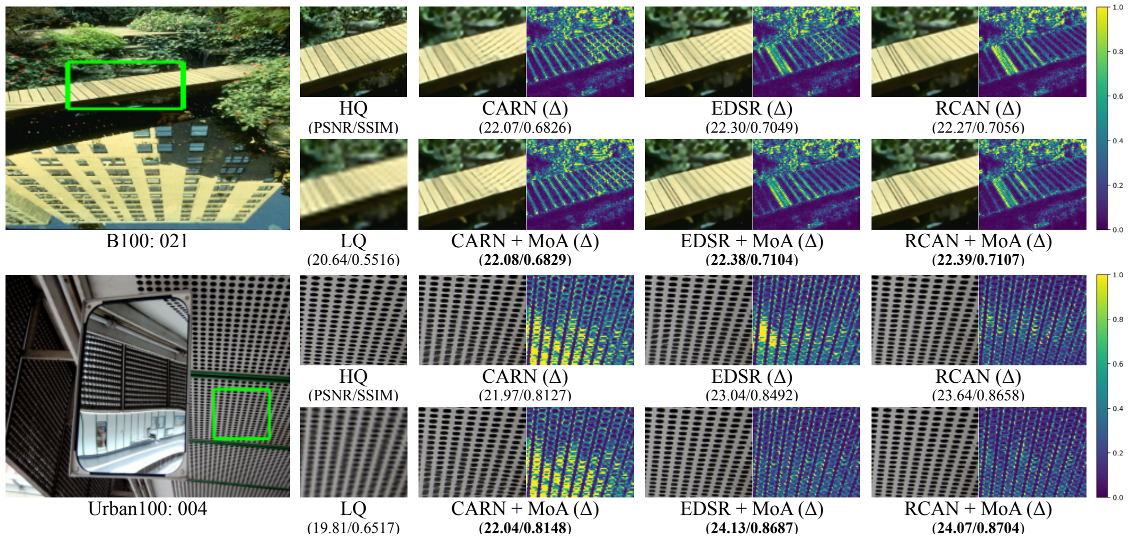


Figure 4.6: **Qualitative comparison on synthetic SR datasets.** Δ is the absolute residual intensity map between the restored and the HQ image.

at most 0.27 dB in PSNR. Figure 4.7 displays the qualitative comparison. Compared to the baselines, the networks utilizing MoA reconstruct the fine details better (e.g., lines in the first row and windows in the second row).

Generalization. In addition to the performance improvement, MoA helps the model to generalize better on various image conditions such as images taken from different devices and unseen pixel resolutions. To test the generalization performance against different devices, we perform a cross-examination using the CameraSR dataset (Table 4.6). In this evaluation, training and test images are captured from different cameras (e.g., train on Nikon and test on iPhone, and vice versa). Because of this train/test inconsistency, the

Table 4.5: **Quantitative comparison on realistic SR datasets**. δ denotes the performance gap between the model w/ and w/o MoA.

Model	Params.	Realistic SR	
		RealSR (δ)	CameraSR (δ)
CARN	1.1M	28.78(+0.00) / 0.8134	30.46(+0.00) / 0.8717
+ MoA		29.00(+0.22) / 0.8204	30.50(+0.04) / 0.8728
RCAN	15.6M	29.22(+0.00) / 0.8254	30.54(+0.00) / 0.8745
+ MoA		29.49(+0.27) / 0.8307	30.79(+0.25) / 0.8780
EDSR	43.2M	28.89(+0.00) / 0.8204	30.53(+0.00) / 0.8742
+ MoA		29.16(+0.27) / 0.8258	30.64(+0.09) / 0.8762

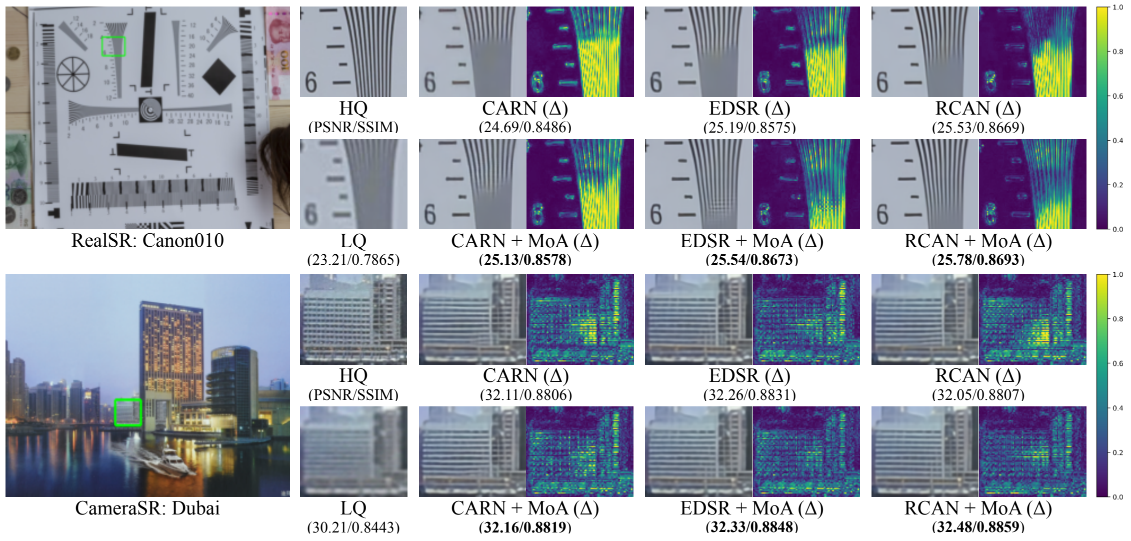


Figure 4.7: **Qualitative comparison on realistic SR datasets.** Δ is the absolute residual intensity map between the restored and the HQ image.

models are required to generalize on unseen distortions, which is more usual in the real-world, where images are taken from various cameras containing device-dependent artifacts. As shown in Table 4.6, the networks with MoA perform significantly better, gaining at most 0.87 dB PSNR in `Nikon`$\rightarrow$`iPhone`, and vice versa. Surprisingly, even though it is a lightweight network, CARN with MoA is shown to be the best model in `Nikon`$\rightarrow$`iPhone`, and it outperforms other models without MoA.

Along with the rapid development of camera sensing module of the edge devices, various types of images have become more popular, which were unavailable in the past without

Table 4.6: **Device generalization ability comparison.** `X`→`Y` indicates that we use images from a device `X` for training and device `Y` for testing.

Model	Params.	CameraSR	
		Nikon→iPhone (δ)	iPhone→Nikon (δ)
CARN	1.1M	22.61(+0.00) / 0.7525	27.86(+0.00) / 0.8138
+ MoA		23.33(+0.72) / 0.7702	28.26(+0.40) / 0.8189
RCAN	15.6M	22.04(+0.00) / 0.7379	27.86(+0.00) / 0.8131
+ MoA		22.91(+0.87) / 0.7621	28.56(+0.70) / 0.8255
EDSR	43.2M	22.28(+0.00) / 0.7596	28.11(+0.00) / 0.8189
+ MoA		23.02(+0.74) / 0.7675	28.63(+0.52) / 0.8265

using expensive auxiliary gadgets. One of the examples is an out-of-focus photography that has recently become available with small devices (e.g., Portrait function in iPhone 11 Pro). To show the effectiveness of CutBlur in the real world, we test a model to super-resolve an out-focused image that we captured by ourselves using iPhone 11 Pro (Figure 4.8). For comparison, we generate LQ input by downsampling the original image through Bicubic kernel (scale $\times 2$). As shown in the figure, the baseline model (EDSR w/o CutBlur) shows degraded performance especially in the focused foreground region—it adds unrealistic textures in the grass. In contrast, CutBlur-model adequately super-resolves both the foreground and background of the image.

4.5.2 Single Distortion Restoration

Synthetic distortion restoration. We compare the models on single distortion restoration tasks: image denoising (color) and JPEG artifact removal. Here, σ and q denote the noise level and the JPEG quality, respectively. Unlike the SR task, even with a big model, MoA seems to influence the performance only marginally (aside from the other side benefits). We conjecture that this result is related to the number of pixel information given to networks. In other words, the effectiveness of DA is diminished due to the richer information. Note that, although we use the DIV2K dataset for both tasks, the total pixel volume used for training is much larger in the restoration tasks than the SR task that involves a downsampling operation. More specifically, the total number for the pixels of

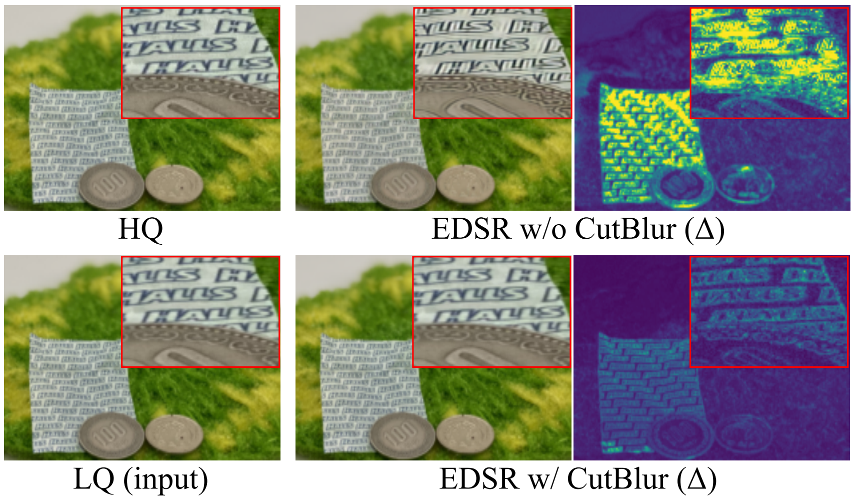


Figure 4.8: **Qualitative comparison on the real-world out-of-focus image.** We capture HQ image by iPhone 11 Pro and downsample (scale $\times 2$) it using bicubic kernel to generate LQ image. EDSR without CutBlur over-sharpens the focused region resulting in unpleasant distortions while the CutBlur-model effectively super-resolves the image.

the denoising and JPEG artifact removal are 2.2B, while 0.1B for the SR ($\times 4$ scale).

Generalization. Still, the advantages of DA methods are not limited to improving performance. To test the enhanced generalization ability of the models on unseen degradation levels, we generate the train and test datasets with different σs and qs. Specifically, we use severely corrupted images as the training dataset and mild distortion for the test dataset: $\sigma = 70 \rightarrow 50$ for denoising and $q = 10 \rightarrow 30$ for JPEG artifact removal, respectively. We would like to note that such train/test inconsistency is a practical scenario in real world because images given in most real use-cases are distorted with an arbitrary and unknown degradation level. For the JPEG artifact removal task, MoA effectively enhances all the restoration models (Table 4.7). The performance is significantly increased by at most 1.02 dB and 0.0171 in PSNR and SSIM, respectively. On the other hand, for the denoising task, MoA seems to negatively impact the model performance of RDN and RNAN in PSNR. Interestingly, one can see that all the models show enhancements in SSIM. We found that the higher PSNR of the baseline models is in fact due to the over-smoothing phenomenon (Figure 4.9). Because the baselines have learned to remove a strong noise ($\sigma = 70$), when the input has a mild noise ($\sigma = 50$), the models deliver over-smoothed

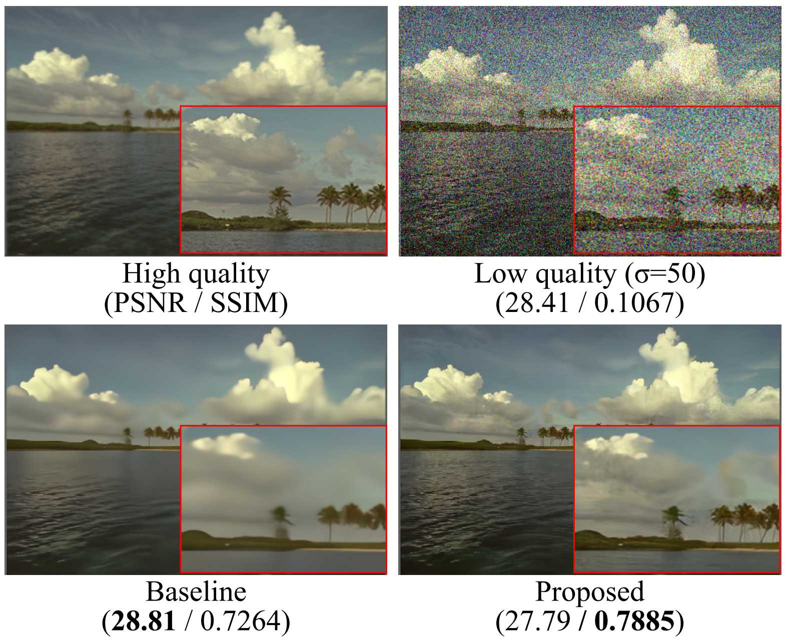


Figure 4.9: **Qualitative comparison of the generalization ability in the denoising task.** Both the baseline and the proposed method are trained using $\sigma = 70$ (severe) and tested with $\sigma = 50$ (mild). Proposed method adequately restores the noise, while the baseline over-smooths the input (resulting in high PSNR) with undesirable artifacts.

Table 4.7: **Quantitative comparison on synthetic distortion datasets.** The networks are trained on the severe case and evaluate on the mild settings (**left:** $\sigma = 70 \rightarrow 30$ for color denoising; **right:** $q = 10 \rightarrow 30$ for JPEG artifact removal). This discrepant environment is to measure the generalization ability to the unseen degradation level. Green/red colors indicate better/worse performance of using MoA than the baseline.

Model	Params.	Image denoising (Kodak24)		JPEG artifact removal (Classic5)	
		$\sigma = 70 \rightarrow 70$ (δ)	$\sigma = 70 \rightarrow 50$ (δ)	$q = 10 \rightarrow 10$ (δ)	$q = 10 \rightarrow 30$ (δ)
EDSR	38.4M	28.23 / 0.7689	27.95 / 0.7385	31.23 / 0.8375	33.28 / 0.8859
+ MoA		28.23 / 0.7686	28.41 / 0.8059	31.24 / 0.8373	34.30 / 0.9016
RDN	22.0M	28.19 / 0.7684	28.13 / 0.7517	31.19 / 0.8362	33.13 / 0.8826
+ MoA		28.20 / 0.7681	27.20 / 0.7854	31.16 / 0.8356	34.17 / 0.8997
RNAN	9.0M	28.10 / 0.7661	27.93 / 0.7450	31.22 / 0.8369	33.34 / 0.8861
+ MoA		28.06 / 0.7635	27.74 / 0.7863	31.09 / 0.8337	34.10 / 0.8984

Table 4.8: **Quantitative comparison on realistic denoising dataset.** δ denotes the performance gap between the model w/ and w/o MoA.

Model	Params.	SIDD+	
		w/o MoA (δ)	w/ MoA (δ)
EDSR	38.4M	36.98(+0.00) / 0.9160	37.16(+0.18) / 0.9231
RDN	22.0M	37.07(+0.00) / 0.9192	37.21(+0.14) / 0.9251
RNAN	9.0M	36.81(+0.00) / 0.9172	36.95(+0.14) / 0.9156

outputs. For example, the tree trunks are erased, the texture of clouds are gone, and the ripples in the water are wiped out. In contrast, with MoA, the networks successfully learn to restore the distorted image while preserving the fine structures, which demonstrates the good regularization effect of our method.

Realistic distortion restoration. To show that MoA brings more benefits in the real-world scenario, we conduct a benchmark on the recently proposed realistic denoising dataset, SIDD+ [78]. Compared to the synthetic distortion dataset, SIDD+ has much larger variation under low-contrast and low-light environments. The key difference is that distortions are applied at arbitrary levels due to the characteristics of the real-world image capturing environment. As depicted in Table 4.8, our method enhances the restoration performance by more than 0.10 dB PSNR for all the networks. This again emphasizes the importance of the generalization ability to handle unseen and arbitrary corruption factors.

Table 4.9: **Quantitative comparison on mixed distortion dataset.** δ denotes the performance gap between the model w/ and w/o MoA. Note that all the models are trained on a moderate level only.

Model	Mixed distortions		
	Mild (δ)	Moderate (δ)	Severe (δ)
EDSR	28.18(+0.00) / 0.7399	27.13(+0.00) / 0.6773	25.84(+0.00) / 0.6154
+ MoA	28.58(+0.40) / 0.7462	27.22(+0.09) / 0.6775	25.89(+0.05) / 0.6149
RDN	28.29(+0.00) / 0.7397	27.17(+0.00) / 0.6773	25.88(+0.00) / 0.6156
+ MoA	28.69(+0.40) / 0.7485	27.27(+0.10) / 0.6793	25.95(+0.07) / 0.6176
RNAN	28.43(+0.00) / 0.7424	27.20(+0.00) / 0.6754	25.91(+0.00) / 0.6145
+ MoA	28.69(+0.27) / 0.7463	27.24(+0.04) / 0.6776	25.90(-0.01) / 0.6155

Table 4.10: **Quantitative comparison on two multi-degradation SR scenarios.** Green/red colors indicate better/worse performance of using MoA than the baseline.

Model	SR+DN			SR+BLUR		
	Set14 (δ)	Urban100 (δ)	Manga109 (δ)	Set14 (δ)	Urban100 (δ)	Manga109 (δ)
CARN	25.25/0.6564	23.24/0.6596	25.62/0.8018	25.26/0.6476	22.62/0.6195	25.11/0.7847
+MoA	25.24/0.6562	23.23/0.6590	25.60/0.8012	25.24/0.6472	22.62/0.6189	25.07/0.7835
RCAN	25.44/0.6653	23.71/0.6841	26.01/0.8161	25.56/0.6584	23.07/0.6443	25.83/0.8084
+MoA	25.46/0.6663	23.78/0.6872	26.11/0.8190	25.58/0.6591	23.16/0.6462	25.95/0.8109
EDSR	25.40/0.6636	23.58/0.6781	25.94/0.8127	25.54/0.6564	22.93/0.6352	25.65/0.8007
+MoA	25.47/0.6666	23.67/0.6836	26.04/0.8171	25.60/0.6596	23.02/0.6409	25.85/0.8078

4.5.3 Multiple Distortion Restoration

Until this point, we have only considered either a single distortion or a degradation task, separately. From now on, we demonstrate the superiority of our method on more challenging tasks, tackling multiple tasks together or mixed artifacts of random intensity levels within a single image.

Mixed artifacts. To generate a mixed distortion dataset, we followed the process of [8]. Here, a sequence of Gaussian blur, Gaussian noise, and JPEG compression is applied to DIV2K HQ images. The range of distortion intensity is preset depending on the group they are in (mild, moderate, and severe), and the level of distortion in the range is randomly sampled for each image. Table 4.9 shows the quantitative comparison on the mixed distortion dataset where the networks are trained on a moderate level alone. Our method successfully enhances the restoration performance across all the models we used. The per-

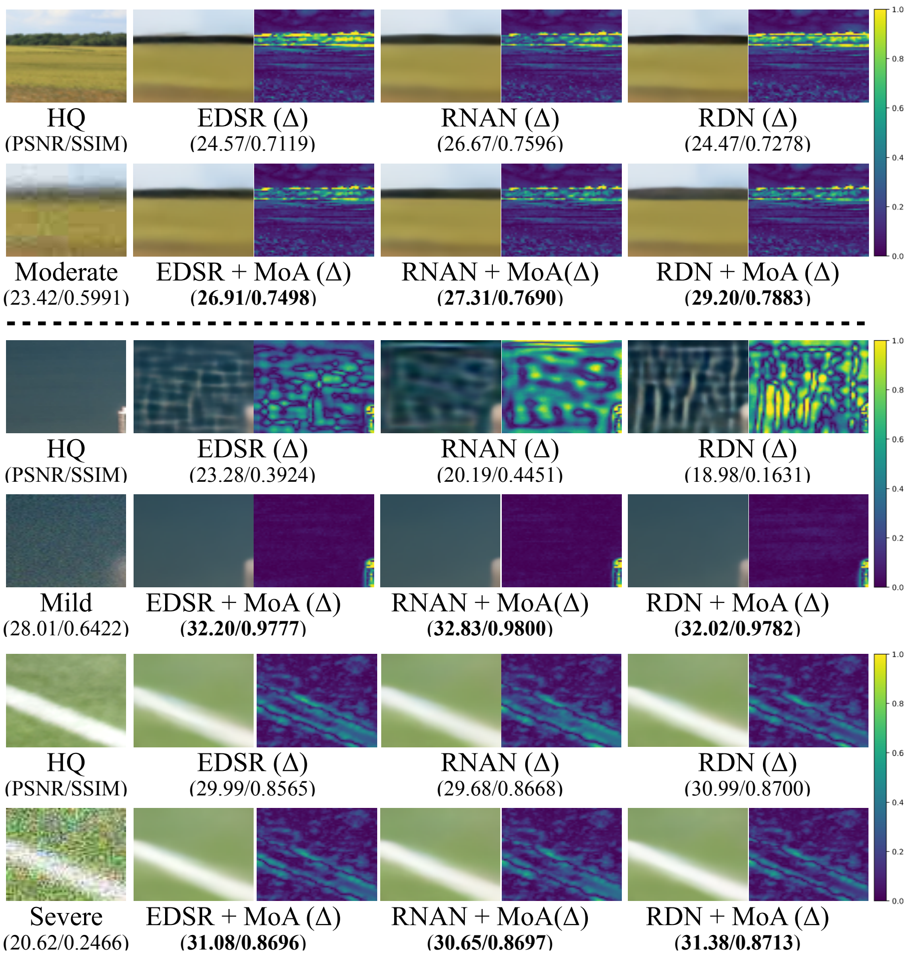


Figure 4.10: **Qualitative comparison on mixed distortion dataset.** Δ is the absolute residual intensity map between the restored and the HQ image. Note that the networks are trained using a *moderate* level (upper dashed line) only; LQ images below the dashed line are distorted from the unseen degradation factors.

formance gains become more notable when transferring to the mild level. This is because CutBlur provides the additional HQ information to the input so that a model acquires the extra capability to avoid over-restoration issue, as we discussed in Figure 4.5. A similar trend can be also seen in the other direction (moderate-to-severe), where the distortion is severer than the usual level during training. Except when the model capacity is small, the networks trained with MoA show better performance.

This trend is more clearly seen in the qualitative results (Figure 4.10). When tested

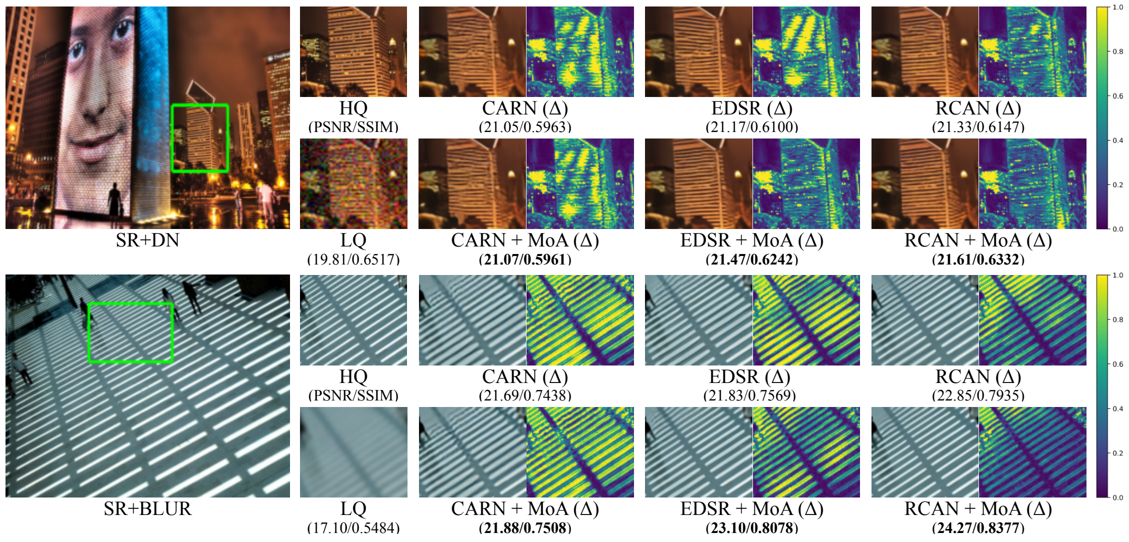


Figure 4.11: **Qualitative comparison on multi-degradation SR datasets.** (**Top**) Bicubic downsample → Gaussian noise (`SR+DN`). (**Bottom**) Gaussian blur → bicubic downsample (`SR+BLUR`). Δ is the residual intensity map between the restored and the HQ.

on the moderate level (the first row, no train/test inconsistency), our method successfully reduces the deviation between the HQ and restored image as shown in the residual intensity map. The visual comparison between the baseline with and without MoA of the first scenario (moderate-to-mild) shows a compelling result; all the baseline networks barely output clean images. In fact, they generate massive aliasing artifacts, and this reveals the vulnerability of image restoration models to the unseen levels of distortions. On the other hand, the models with MoA remarkably remove the present noises in the LQ, which is also summarized by 13.04 dB improvements (RDN) in PSNR. The moderate-to-severe scenario also shows the benefits of using our method (the last row).

Multiple degradation tasks. Here, we consider an image degradation model that involves downsampling with some distortions together. Since one has to solve two objectives simultaneously with a single network, this becomes more challenging than learning to upsample an image or to remove the artifacts, separately. Similar to the previous SR results, the performance of the models with large capacity is improved further. For example, RCAN and EDSR gain an additional 0.10 dB to the baseline on the Manga109 dataset. Figure 4.11 shows a visual comparison on `SR+DN` and `SR+BLUR`, and we see that, with MoA, the networks become better in restoring patterns especially where the aliasing artifact.

4.6 Discussion

The introduced CutBlur and MoA in this chapter are designed to train to train a stronger and more robust model for various low-level vision tasks. We conducted a comprehensive analysis on the effect of the existing DA methods in various aspects, including their frequency domain responses. Based on this analysis, we provided a set of principles to design DA methods for low-level vision tasks, which led to CutBlur and MoA. By learning how and where to restore an image, CutBlur encourages the model to understand how much to correct the underlying pixels. MoA aggregates the best of the curated DA methods enabling further improvements over a single DA. Throughout this chapter, we showed that our proposed strategy consistently and significantly improves the performance across various scenarios especially when the model size is big and the problem becomes closer to real-world environments. The extensive experimental results showed that our method can be applied to various tasks of single distortions, such as denoising and JPEG artifacts removal, and multiple distortions having more than two distortions or degradation in a single image. Last but not least, our method helps the models to generalize better in various conditions such as unseen pixel resolution, noise levels, and device-dependent artifacts, which is beneficial for many real-world applications.

Chapter 5

Unsupervised Image Restoration

In this chapter, we tackle a fully unsupervised super-resolution problem, i.e., neither paired images nor ground truth high-resolution (HR) images. We assume that low resolution (LR) images are relatively easy to collect compared to high resolution (HR) images. By allowing multiple LR images, we build a set of pseudo pairs by denoising and down-sampling LR images and cast the original unsupervised problem into a supervised learning problem but in one level lower. Though this line of study is easy to think of and thus should have been investigated prior to any complicated unsupervised methods, surprisingly, there are currently none. Even more, we show that this simple method outperforms the state-of-the-art unsupervised method with a dramatically shorter latency at runtime, and significantly reduces the gap to the HR supervised models. This simple method should be used as the baseline to beat in the future, especially when multiple LR images are allowed during the training phase. However, even in the zero-shot condition, we argue that this method can serve as a useful baseline to see the gap between supervised and unsupervised frameworks.

5.1 Overview

While most of the deep learning-based SR methods heavily rely on a large number of image pairs (supervised SR), unfortunately, such a large-scale and high quality dataset is not always accessible, especially when we deal with a real environment. A few recent works have proposed a workaround solving an unpaired SR task [106, 107]. Since this setup does not require full supervision of LR and HR pairs but images from each domain, it is a more realistic scenario for many real-world applications. However, in many applications, gathering HR images itself requires a lot of efforts or sometimes even impossible.

To address this, Shocher et al., [9] have proposed a fully unsupervised method, zero-shot super-resolution (ZSSR), which performs both training and testing at runtime using only a single LR test image. By learning a mapping from a scale-down version of the LR image to itself, it learns to exploit internal image statistics to super-resolve the given image. It outperformed the previous internal SR methods [108] in a huge margin with a high flexibility since this can easily be adapted to any unknown downsample kernels.

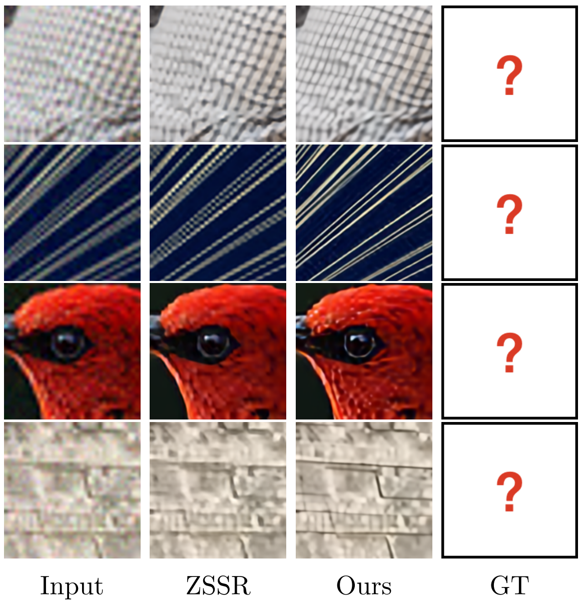


Figure 5.1: **Overview of the unsupervised restoration scenario.** HQ ($\times 4$) [109] images (1st column). ZSSR [9] with BM3D [110] results (2nd column). Our SimUSR with BM3D results (3rd column). Ground truth HQ image (4th column) is not available in this setup. Our method achieves superior SR performance for all the cases.

However, ZSSR has several drawbacks. **1)** It requires an online optimization procedure at runtime. Since it needs at least 1K steps (both forward and backward propagation), the latency of the ZSSR is extremely high. **2)** It is difficult to benefit from a large capacity

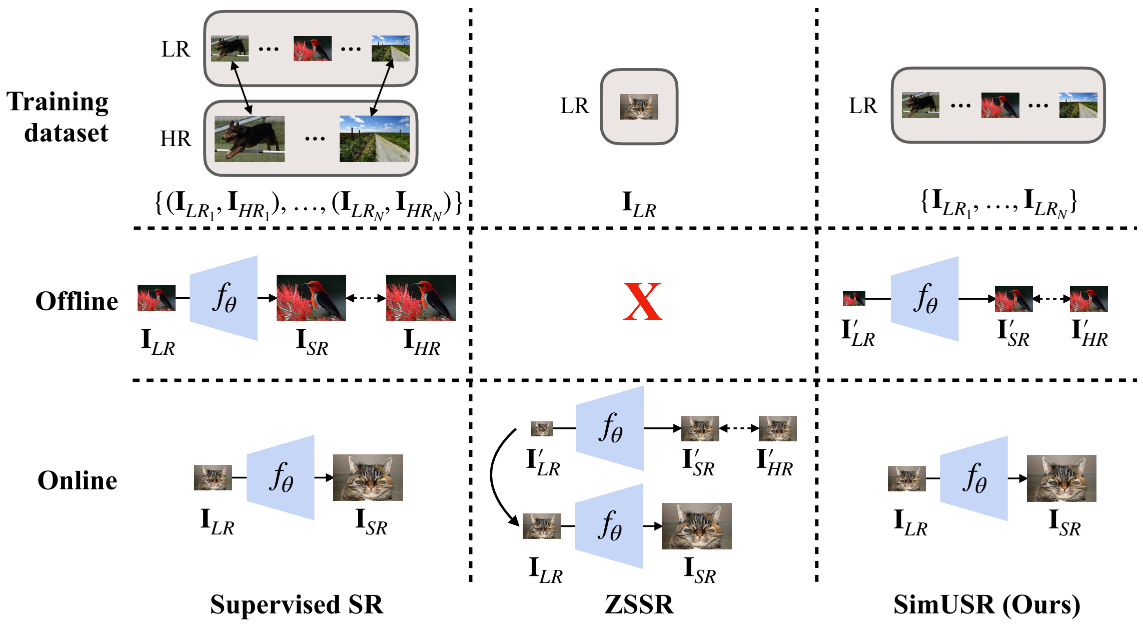


Figure 5.2: **Schematic comparison of the supervised SR, ZSSR [9], and our SimUSR.** We analyze current SR approaches in terms of the training dataset and offline/online phases. The offline phase is operated beforehand the user's inference request (i.e., training of the supervised SR). Online phase denotes runtime. (**1st row**) While supervised SR requires the LR-HR pairs, ZSSR and SimUSR use LR images only, making them more applicable to the real-world scenarios. ZSSR utilizes only a single test LR image and performs both optimization and inference at runtime. (**2nd, 3rd rows**) On the other hand, SimUSR exploits additional LR images and follows a similar procedure to the supervised setup, where the model is first trained offline and inference is done online.

network. Because ZSSR has to perform online training on a single image, the model should be able to quickly adapt to the given image while avoiding the overfitting issue, which limits ZSSR to use a shallow network architecture. **3)** When noises are present in LR images, ZSSR shows deteriorated performance because the model can never learn to denoise, and even after adding a denoising module, it suffers from its restrictive framework. **4)** It does not utilize any prior information at all, which is an excessively restrictive constraint. While collecting LR-HR image pairs are difficult, acquiring LR images only is relatively easy and feasible in many real-world scenarios. Since the internal-based SR methods generally show worse SR performance than the external-based models, it is desirable to exploit every available prior information as long as it stays in the unsupervised regime.

To mitigate these limitations, in this chapter, we propose a simple baseline framework for unsupervised SR (SimUSR) that relaxes the ZSSR into a supervised setting. Instead of using a single image, our SimUSR make pseudo-pairs using multiple LR images. To correctly guide the model, we employ BM3D [110] to remove noises from the LR images when preparing the pairs. Though these are very simple corrections, they bring several benefits: our framework can now exploit every benefit of supervised learning. Thanks to this pseudo-supervision, ample prior information enables a model to reduce the performance gap between the unsupervised (only LR is available) and the supervised setting (HR is available). SimUSR can utilize recently developed network architectures and techniques that provide huge performance gains (Figure 5.1). In addition, since the online training is not necessary, SimUSR can significantly reduce its runtime latency as well. The differences of the supervised SR, ZSSR, and SimUSR are summarized in Figure 5.2. We argue that our assumption is fairly practical while still remaining under the unsupervised learning setup; we only use LR images. Our approach is meaningful in that it investigates the blind spot of the field that should have been addressed but overlooked.

5.2 Background

Supervised Super-resolution. Recently, deep learning-based super-resolution models [111, 112, 17, 113] have shown a dramatic leap over the traditional algorithms [114]. Most of the successful deep SR approaches fall into the supervised setting, where a network is trained on an external dataset having low- and high-resolution pairs. As long as the size of the dataset and the network capacity are large enough, it is well known that the supervised approach provides a better chance to enhance the SR performance [17, 113]. However, it is also true that their performance and generalizability deteriorate dramatically when the dataset size is small and when there exists mismatch between training and testing environments [115]. To mitigate this issue, recent approaches focus on *blind SR*, which assumes that there exist LR and HR pairs but with unknown degradation and downsample kernel [116]. Unlike the above, our proposed method can train a network even when there are no LR and HR pairs.

Unpaired Super-resolution. A few recent works have addressed an unpaired SR task [106, 107, 117] that does not assume a paired setting. Since this setup does not require a full supervision, it is a more realistic scenario for many real-world applications. Most of the methods employ generative adversarial framework [25] so that a generator learns to map HR images into their distorted LR version. Using this generated pairs, they train an SR network in a supervised setting. However, in practice, there are cases where HR images are not even available, which requires a fully unsupervised SR.

Unsupervised Super-resolution. Though the unpaired SR is sometimes considered as an unsupervised SR, we first clarify that unsupervised SR should strictly denote the task without any supervision neither paired images nor HR images. Under this definition, there are only a handful of studies [118, 119, 9] and zero-shot super-resolution (ZSSR) [9] falls into this. ZSSR uses *LR sons* that are downsampled images of the given LR test image (a.k.a *LR father*). Using this pseudo pairs, they train the model in a supervised manner but only exploiting the internal statistics of the given test image. Because every procedure is performed at runtime, ZSSR suffered from high latency. To overcome this, Soh et al., [120] have proposed meta-transfer ZSSR (MZSR). They added a meta-transfer learning phase to exploit the information of the external dataset, which decreased the number of the steps required at runtime. Still, to quickly optimize the network, MZSR was limited to use a simple 8-layer network. Unlike the aforementioned methods, our SimUSR can benefit from the larger capacities of recently developed SR models and short latency at runtime by removing the online update phase, while remaining in the fully unsupervised regime in that it only utilizes the LR images.

5.3 Approach

5.3.1 Zero-shot Super-resolution

Zero-shot super-resolution (ZSSR) [9] tackles the fully unsupervised SR, where only LR images ($\mathbf{I}_{LR}$) are available. To do that, it performs both optimization and inference at runtime using a single test image (Figure 5.2). During the online optimizing phase, they use an test input image ($\mathbf{I}_{LR}$) as *LR father* ($\mathbf{I}_{LR}^{father}$) and generates *LR son* ($\mathbf{I}_{LR}^{son}$) by downsampling LR father with an arbitrary kernel k. Then, they create *pseudo-pair*

$$(\mathbf{I}'_{LR}, \mathbf{I}'_{HR}) = (\mathbf{I}_{LR}^{son}, \mathbf{I}_{LR}^{father}),$$

where $\mathbf{I}_{LR}^{son} = \mathbf{I}_{LR} \downarrow_{s,k}$ and $\mathbf{I}_{LR}^{father} = \mathbf{I}_{LR}$. Here, $\downarrow_{s,k}$ denotes a downsampling operation with an arbitrary kernel k and scale factor s.

With this pseudo-pair, optimizing a SR model now becomes a standard supervised setting. The core idea of ZSSR is to make the model learn internal image-specific statistics of a given test image during the online training. For inference, it generates final SR output ($\mathbf{I}_{SR}$) by feeding $\mathbf{I}_{LR}$ to the trained image-specific network.

5.3.2 SimUSR: Simple Baseline for Unsupervised Super-resolution

We introduce a simple baseline for a fully unsupervised super-resolution task (SimUSR). Similar to the ZSSR [9], our method does not use any HR images for training the network. However, we slightly relax the constraint of ZSSR and assumes that it is relatively easy to collect the LR images, $\{\mathbf{I}_{LR_1}, \dots, \mathbf{I}_{LR_N}\}$, where N is the number of LR images. This allows our method to exploit multiple pseudo-pairs:

$$(\mathbf{I}'_{LR_k}, \mathbf{I}'_{HR_k}) = (\mathbf{I}_{LR_k}^{son}, \mathbf{I}_{LR_k}^{father}), \text{ for } k = 1 \dots N.$$

Here, we generate $\mathbf{I}_{LR}^{son}$ and $\mathbf{I}_{LR}^{father}$ with the same protocol that used in ZSSR.

Though we now lose the generalizability over a single test image, compared to the cost of the relaxation, the benefits are very huge: we can fully enjoy the advantages of the supervised learning framework. More specifically, using these multiple pairs, we can now train a network offline and perform inference online as any supervised model usually does. Our method can be implemented by a simple modification of the supervised SR approach, it gives high flexibility and extensibility. For example, unlike the ZSSR

Table 5.1: **Quantitative comparison on the bicubic SR (scale $\times4$).** We boldface the best performance of both supervised SR and ours.

Dataset	Supervised SR			ZSSR	SimUSR (Ours)		
	CARN	RCAN	EDSR		CARN	RCAN	EDSR
Set5	32.13/0.8937	**32.63/0.9002**	32.46/0.8968	31.13/0.8796	31.94/0.8908	**32.40/0.8962**	32.37/0.8955
Set14	28.60/0.7806	**28.87/0.7889**	28.80/0.7876	28.01/0.7651	28.44/0.7786	**28.71/0.7860**	28.70/0.7855
B100	27.58/0.7349	**27.77/0.7436**	27.71/0.7420	27.12/0.7211	27.49/0.7324	**27.68/0.7394**	27.66/0.7389
U100	26.07/0.7837	**26.82/0.8087**	26.64/0.8033	24.61/0.7282	25.70/0.7740	**26.45/0.7986**	26.31/0.7940
M109	-	**31.22/0.9173**	31.02/0.9148	27.84/0.8657	30.03/0.9014	**30.73/0.9124**	30.59/0.9107

Table 5.2: **Quantitative comparison (PSNR) on SR (scale $\times4$) task with MoA**. We show the effect of MoA on our SimUSR and supervised SR (SSR) model. Note that SSR results are provided to show the improved upper limit again.

Type	Model	Set14	Urban100	Manga109
SimUSR	RCAN (+MoA)	28.80	26.60	30.85
SSR	RCAN (+MoA)	28.92	26.93	31.46

and MZSR [120], which inevitably use shallow networks, we can use any off-the-shelf SR network and technique available, such as data augmentation (Section 5.4.2). In addition, since the runtime of our SimUSR only depends on the network's inference speed, this also gives a huge acceleration in terms of the runtime latency (Section 5.4.4).

5.4 Experiment

In this section, we describe our experimental settings and compare the performance of our method with the supervised SR models and the ZSSR [9]. In Section 5.4.2, we analyze how much our SimUSR improves the performance over the ZSSR and how far we are left to reach the supervised performance. Then, in Section 5.4.3, we apply our method on the NTIRE 2020 SR dataset [109].

5.4.1 Experimental Setting

Baselines. We use ZSSR [9] as our major baseline method. However, since ZSSR and SimUSR are not designed to handle noisy cases, we attach BM3D [110] as a pre-processing step. For our SimUSR, we use various models as our backbone network. We use three SR models: CARN, RCAN [113] and EDSR [17]. Each of the model have different numbers of parameters from 1.1M to 43.2M (million).

Dataset and evaluation. We use the DF2K [121, 17] dataset for the bicubic degradation SR task. However, unlike the Lim et al., [17], we only use the LR images when we train the models. For evaluation, we use Set5 [45], Set14 [122], B100 [47], Urban100 [108], and Manga109 [60] for bicubic SR task. To evaluate our method on the real-world SR task, we use NTIRE 2020 dataset [109]. This dataset is generated with unknown degradation operation to simulate the realistic image processing artifacts. In addition, only non-paired LR and HR images are given so that the model should be trained via unsupervised setup. Same as DF2K, we do not use any of HR images at the training phase. We use PSNR and SSIM to measure the performance. We calculate both metrics on RGB channels for the NTIRE dataset while only using the Y channel for the bicubic SR task.

5.4.2 Bicubic Super-resolution

Here, we compare SimUSR with the ZSSR and the supervised SR models. Though the classical bicubic SR task is not our main task, it provides a testbed to analyze every model simultaneously. This also shows how much gap there exists between the supervised and unsupervised frameworks. For fair comparison, we report our performance using different SR networks as our backbone (CARN, RCAN [113], and EDSR [17]). The quantitative comparison on various benchmark dataset is shown in Table 5.1. Exploiting the additional LR images, our SimUSR shows large improvements over the ZSSR in every case.

More interestingly, by exploiting the recent development of supervised SR techniques, such as data augmentation, SimUSR further reduces the gap toward the supervised learning models (Table 5.2). Note that, while the supervised models can use HR images as ground truth, SimUSR only uses LR images. Therefore, the model should generalize over the learned scale and pixel distributions. Toward this, we used mixture of augmentation (MoA), which is a recent data augmentation method for low-level vision task. MoA is known to not only improve the performance but also enhance the generalization power of the model. By employing the MoA, which ZSSR does not benefit from (results not shown), our performance again increases by 0.09 dB (Set14), 0.15 dB (Urban100), and 0.12 dB (Manga109), which are upto 3.63 dB (Manga109) improvements over the ZSSR. Therefore, from now on, we use MoA with SimUSR by default unless it is specified.

The qualitative results also shows the superior results of SimUSR over the ZSSR (Figure 5.3). In all the cases, SimUSR benefits from the increased performance by using external LR images. This tendency is clearly shown in the residual intensity map between the SR and HR image. For example, our method successfully restores the replicating patterns (1st, 3rd, and 4th rows) while ZSSR has difficulty of recovering distortions. Note that ZSSR is supposed to better learn the internal statistics by repeatedly seeing the same LR image patches, which is in principle good at recovering replicating patterns.

5.4.3 Real-world Super-resolution

In this section, we compare ZSSR [9] and our method on the NTIRE 2020 dataset [109]. We found two observations that 1) ZSSR suffers from noise, and 2) the data augmentation methods, which are used in the original ZSSR, actually harm its SR performance (Table 5.3). Based on this observation, we decided to attach BM3D [110] before the ZSSR network optimization. For a fair comparison, we also use BM3D with our SimUSR. Regarding the data augmentation, we suspect that this is due to ZSSR network's small capacity and the severe spatial distortion by applying strong affine transformations.

By adding an ad-hoc denoiser (BM3D), ZSSR performance is dramatically improved by 0.63dB and 0.0422 in PSNR and SSIM, respectively. And by discarding affine augmentation, we can further enhance the ZSSR to achieve 26.55dB in PSNR (3rd row). With the same setting, our proposed SimUSR outperforms the ZSSR in a huge margin. For example, SimUSR with the lightweight SR network, CARN, already boosts the SR performance

Table 5.3: **Quantitative comparison on the NTIRE 2020 dataset [109]**. We analyze the effect of denoising (`w/ BM3D`) and affine transformations (`w/o Affine`). We also analyze the advantage of applying SimUSR.

Method	w/ BM3D	w/o Affine	PSNR / SSIM
ZSSR			25.82 / 0.6898
	✓		26.45 / 0.7320
	✓	✓	26.55 / 0.7344
SimUSR+CARN	✓	✓	27.19 / 0.7520
SimUSR+RCAN	✓	✓	27.24 / 0.7550
SimUSR+EDSR	✓	✓	**27.28 / 0.7554**

of the ZSSR by 0.64dB and 0.0176 in PSNR and SSIM, respectively. Moreover, thanks to the high flexibility of our method, we can easily improve the performance by simply changing the backbone to any other SR network. For instance, we get another 0.09dB improvement in PSNR by just replacing a backbone network from CARN to RCAN [113]. Figure 5.4 shows the qualitative comparison between the ZSSR and our method with different backbone networks. Similar to the bicubic SR task, SimUSR (with any backbone) provides better restoration results across various cases.

5.4.4 Execution Time

In this section, we evaluate and compare the latency of ZSSR and our SimUSR (Table 5.4). Note that we benchmark the runtime speed on the environment of NVIDIA TITAN X GPU by generating a 1080p SR image on scale factor $\times 4$. Although ZSSR has only 0.23M parameters, it requires a huge amount of runtime (300.83s) since it has to perform optimization and inference at runtime. In contrast, our proposed SimUSR only takes less than two seconds (1.93s) even if we use a heavy SR network (EDSR) as a backbone model. Comparing to the ZSSR, our method is at least 155 times faster than the ZSSR and if we use a lightweight SR network (CARN), 2,500 times faster (0.12s vs. 300.83s).

To embed the SR method to the real application, it is obvious that both SR performance and the latency are important aspects (*e.g.* SR system for the streaming service). However,

Table 5.4: **The number of the parameters and runtime comparison.** We measure the runtime with 480×320 LR images with scale factor ×4.

Method	ZSSR	SimUSR (Ours)		
		CARN	EDSR	RCAN
# Params.	0.23M	1.14M	15.6M	43.2M
Runtime	300.83s	0.12s	1.93s	1.07s

the above analysis reflects that although ZSSR has nice properties, which does not need an HR image, applying it to the real application is challenging because of its high latency. On the other hand, our approach can meet the criteria that real applications demand (on both the performance and speed) by taking advantage of supervised SR. In addition, if necessary, we can further reduce the latency by replacing the backbone to more lightweight network thanks to the flexibility of our method.

5.5 Discussion

In this chapter, we introduce SimUSR, simple but strong baseline framework for unsupervised SR task. We first clarify that unsupervised SR should strictly denote the task without any access to HR images. While complying with this definition, we assume that LR images are relatively easy to obtain in the real-world. Exploiting multiple LR images, we generated a pseudo-pair dataset of LR images and their down-scaled version and use this to train a SR model. This simple conversion allows us to enjoy the advantages of supervised learning. Although this method works very firmly, there are some discussion on the limitations and the future direction of this work.

Limitation. Though the accessibility to multiple LR images is a mild and reasonable relaxation in many cases, there are still many applications and domains that cannot resort on such assumption where collecting the data is very expensive, e.g., medical imaging. In addition, SimUSR heavily relies on the generalizability of a model over different scales and pixel distributions, which can cause unexpected artifacts. Because SimUSR uses bicubic downsampling to prepare the pseudo pairs, this may also cause an implicit bias in the SR model during the training. Finally, it is true that SimUSR is a basic approach that one would easily come up with but overlooked until now. We argue that it should be by no means a new state-of-the-art but serve as a reasonable baseline to beat in the future.

Future work. We showed that our SimUSR framework is a strong baseline but it still has a plenty of room to improve its performance. For example, we used the BM3D as the pre-processing module for removing the noise. This pre-module can be replaced to more effective models [123].

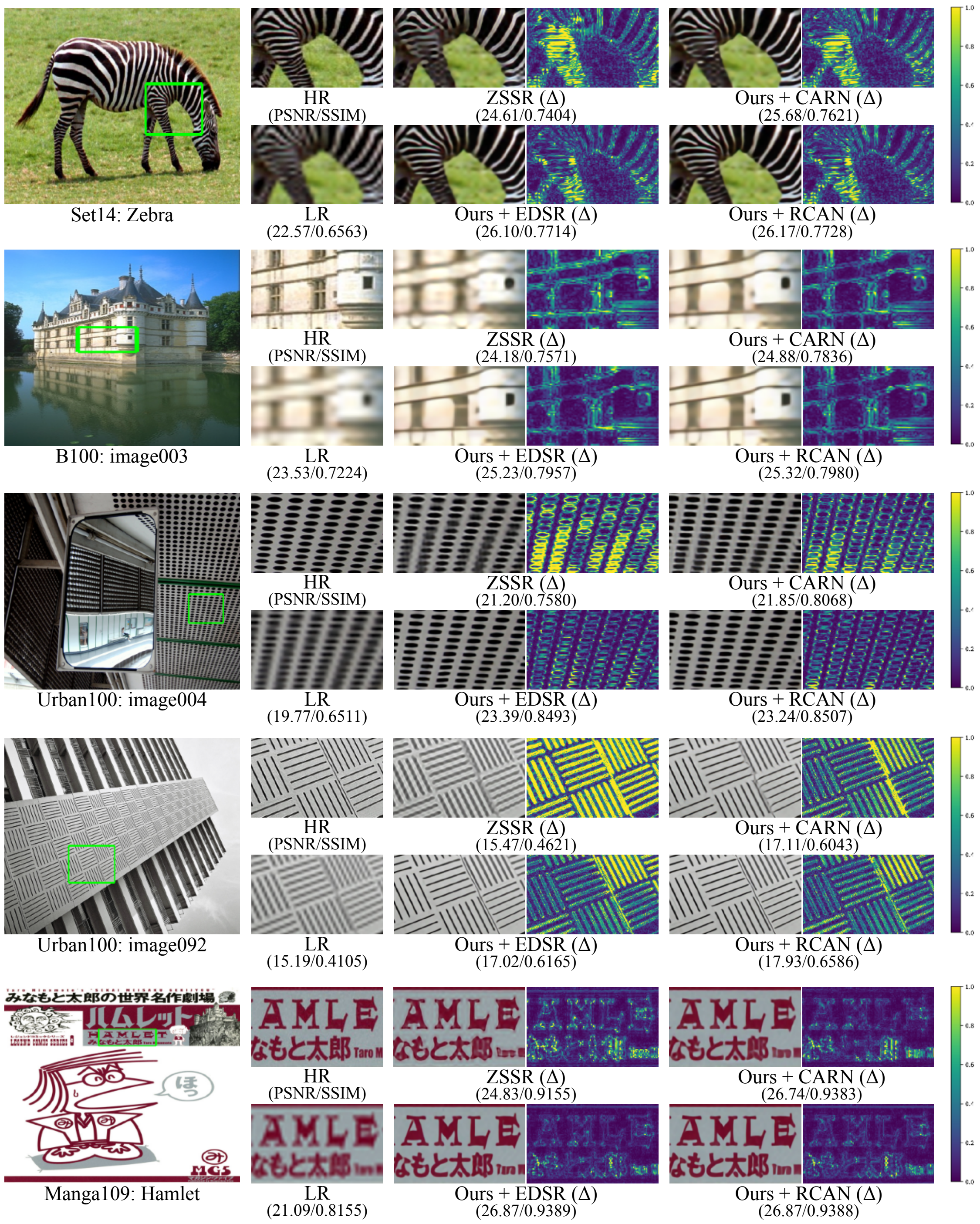

Set14: Zebra
HR (PSNR/SSIM)
ZSSR (Δ) (24.61/0.7404)
Ours + CARN (Δ) (25.68/0.7621)
LR (22.57/0.6563)
Ours + EDSR (Δ) (26.10/0.7714)
Ours + RCAN (Δ) (26.17/0.7728)
B100: image003
HR (PSNR/SSIM)
ZSSR (Δ) (24.18/0.7571)
Ours + CARN (Δ) (24.88/0.7836)
LR (23.53/0.7224)
Ours + EDSR (Δ) (25.23/0.7957)
Ours + RCAN (Δ) (25.32/0.7980)
Urban100: image004
HR (PSNR/SSIM)
ZSSR (Δ) (21.20/0.7580)
Ours + CARN (Δ) (21.85/0.8068)
LR (19.77/0.6511)
Ours + EDSR (Δ) (23.39/0.8493)
Ours + RCAN (Δ) (23.24/0.8507)
Urban100: image092
HR (PSNR/SSIM)
ZSSR (Δ) (15.47/0.4621)
Ours + CARN (Δ) (17.11/0.6043)
LR (15.19/0.4105)
Ours + EDSR (Δ) (17.02/0.6165)
Ours + RCAN (Δ) (17.93/0.6586)
Manga109: Hamlet
HR (PSNR/SSIM)
ZSSR (Δ) (24.83/0.9155)
Ours + CARN (Δ) (26.74/0.9383)
LR (21.09/0.8155)
Ours + EDSR (Δ) (26.87/0.9389)
Ours + RCAN (Δ) (26.87/0.9388)
1.0
0.8
0.6
0.4
0.2
0.0

Figure 5.3: **Qualitative comparison of SimUSR on synthetic SR datasets.** Δ is the absolute residual intensity map between the network output and the ground-truth.

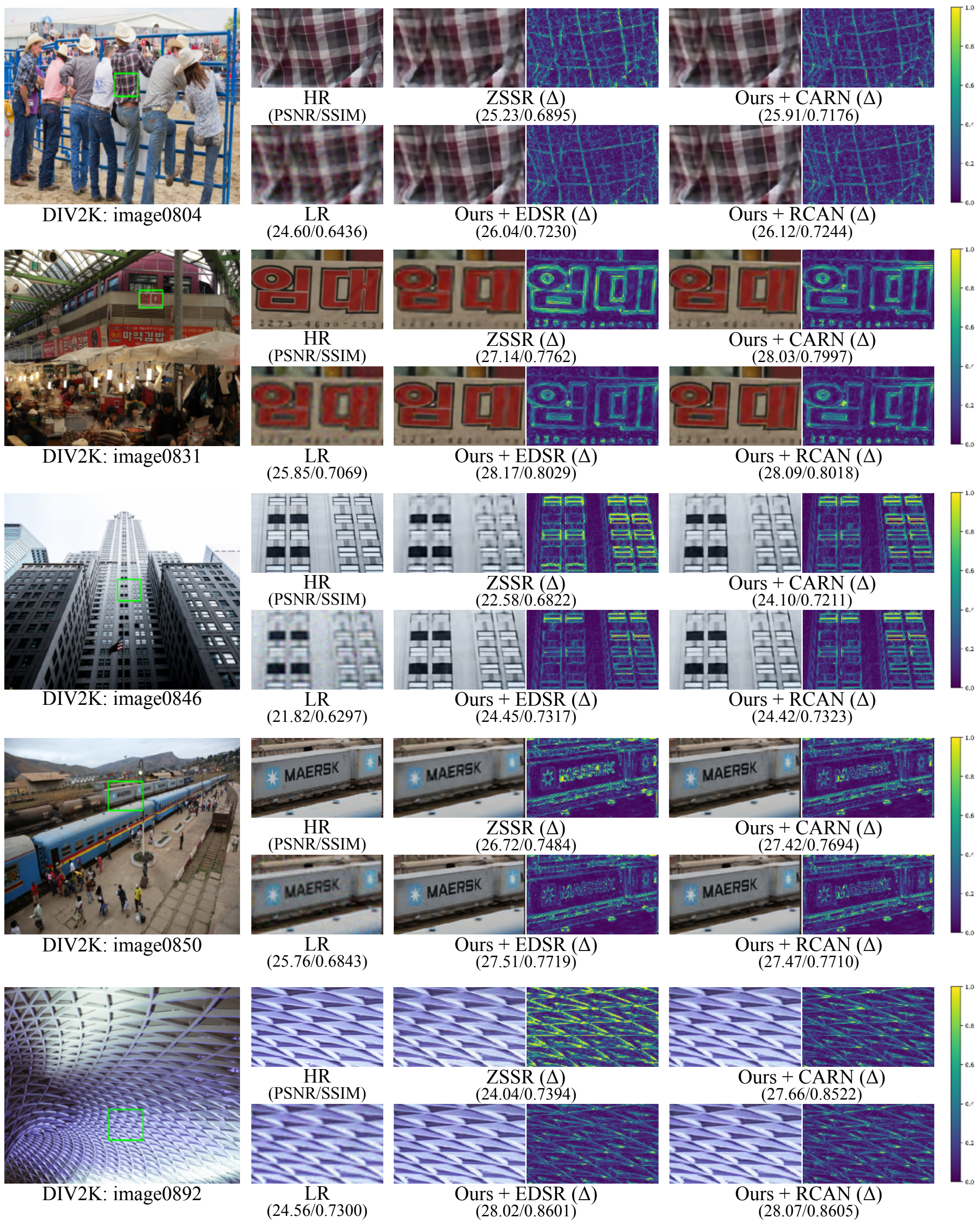


Figure 5.4: **Qualitative comparison of SimUSR on NTIRE 2020 dataset [109]. Δ** is the absolute residual intensity map between the network output and the ground-truth.

Part III

Toward Multi-modal Distortion

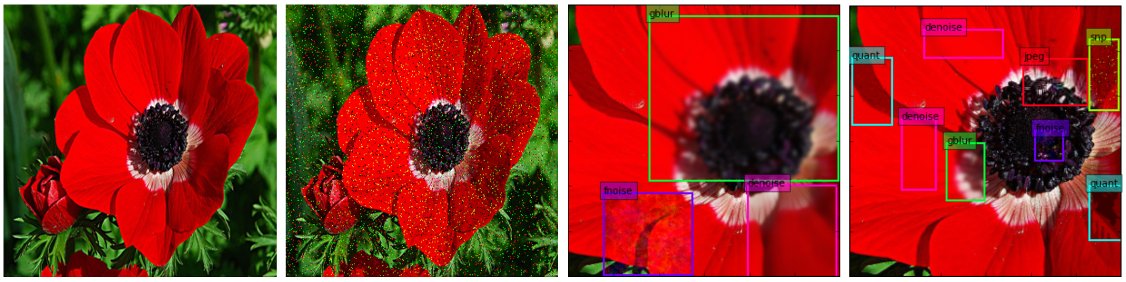
Chapter 6

Investigating the Multi-modality in Distortion

In this chapter, we look through the multi-modal nature represented in distortions. We first introduce a new multi-modality concept in the image distortion scenario, then investigate through distortion recognition task; classification and detection since these are important tasks in many applications. For example when compressing images, if we know the exact location of the distortion, then it is possible to re-compress images by adjusting the local compression level dynamically. We show that the current state-of-the-art object classification and detection networks accurately recognize image distortions as well with little effort the fine-tuning. In Chapter 7 we will discuss on image restoration method in a multi-modal distortion case based on the analyzed insight in this chapter.

6.1 Overview

With the development of the mobile devices, demand for streaming media and cloud service has skyrocketed. These services need a lot of storage to store multimedia, and it is crucial to compress data using lossy compression techniques before storing. A higher compression level is better in terms of storage, but it could cause serious local distortion to images. What if we can detect the region in which the distortions occurred? Then, we



(a) Reference (b) Classification (c) Detection-basic (d) Detection-difficult

Figure 6.1: **Example of the Multi-modal Distortion scenario**. (a) is the reference image, (b) is the distorted image with salt and pepper noise for classification task. (c) and (d) are both for detection task, with different levels of detection difficulty.

may correct the problem by dynamic compression techniques. Such techniques reduce the compression level of detected distortion regions and re-compress using the reduced level.

The primary motivation in this chapter is to build a system that detects the distortion region and performs compression dynamically. Hence, automatic distortion detection is an essential part of this system. However, despite the importance of this task, recent image quality assessment (IQA) methods only focus on predicting perceptual quality scores, such as the mean opinion score (MOS) [124, 125, 126, 127].

One might question the validity of the assumption that multiple or multi-modal distortions exist in a single image. While it is true that distortions in an image are likely to occur globally rather than locally, we consider a situation where individual images are assembled to form a larger one. For example, when creating a panorama picture, the compression of each shot may be subjected to independent distortion. When the individual shots are combined to form the full photo, it might end up with localized distortions.

In recent years, many deep learning-based IQA approaches have been proposed, especially for non-reference IQA (NR-IQA). NR-IQA methods perform image quality assessment without any direct comparison between the reference and the distorted image. The IQA-CNN [124] model is the first such model that applies deep learning in IQA tasks. This model uses a convolutional neural network (CNN) composed of five layers, that achieves the result comparable to the full-reference IQA (FR-IQA) methods, such as FSIM [128].

Deeper networks have been used in [125], whose structure is inspired by the VGGNet [129], and yields results that surpass FR-IQA approaches. DeepBIQ [126] is the first to use pre-trained CNN and show the state-of-the-art result in the IQA task. However, although there exist many outstanding results, the distortion classification task remains mostly unchallenged. Two notable exceptions to this are the IQA-CNN+ and the IQA-CNN++ [127], which predict both the MOS and the distortion type with the similar network used in IQA-CNN. One shortcoming of these models is that they are shallow architectures, and thus might have limited capacity to successfully solve our task.

In addition, to best to our knowledge, distortion detection task with deep learning method has not been applied yet. The reason is twofold: First, there is no sufficiently large distortion detection dataset suitable for deep learning, and second, detection task is a much more challenging problem than classification or predicting MOS is. The difficulty of distortion detection is based on the fact that images can have heterogeneous and multiple distortion types. In general, most IQA datasets contain only homogeneous distortion types that make prediction relatively easy.

To tackle these issues, we introduce a multi-modal distortion scenario and create a new dataset for both distortion classification and detection. Then we apply pre-trained CNNs such as VGGNet [129] and ResNet [130] for distortion classification. Finally, we use deep learning-based detection methods such as single-shot multi-box detector (SSD) [131] to locate the distortion regions.

6.2 Multi-modal Distortion Scenario

In this section, we present a multi-modal distortion scenario and its related benchmark dataset. To do that, we create a new dataset named `Flickr-Distortion` to evaluate image distortion classification and detection task. To make this, we first collect 804 reference images from Flickr with similar ways of NUS-WIDE [132], and make distorted images by using the following eight distortion types: 1) Gaussian white noise (GWN), 2) Gaussian blur (GB), 3) salt and pepper noise (S&P), 4) quantization noise, 5) JPEG compression noise, 6) low-pass noise, 7) denoising and 8) fnoise.

The reason we do not use the LIVE [133] dataset directly is that it contains *global* distortions, whereas we deal with *local* distortions. Furthermore, prevalent distortion dataset such as LIVE [133] or TID2013 [134] have insufficient amount of reference images which are not suitable for training deep learning models.

6.2.1 Distortion Classification

In the dataset for the distortion classification task, each reference image is distorted using eight distortion types with three levels. Thus, a single reference image results in 24 distorted images. The distortion procedure follows the LIVE dataset [133], and includes such distortion types as homogeneous distortion. The distortion is applied to the entire image (Figure 6.1(b)). We create 19,296 distorted images in total, and randomly split data into 60% training, 20% validation, and 20% test set.

We implemented the generation of most noises except fnoise, for which we used the scikit-image [135]. Detailed values we use are as follows: Gaussian noise is generated with three values of variances: {0.0125, 0.025, 0.05}, and the amount of salt and pepper noise are the same as Gaussian noise. For the Gaussian blur, we use the three sigma {1.5, 3, 6} and in JPEG compression we use {20, 10, 5} quality levels. To implement the low-pass noise, we simply scaled images with ratios {0.3, 0.1, 0.03}, and resized them to the original image size. We implemented denoising with non-local means algorithm [136] using factors {0.04, 0.06, 0.08}, but with fixed batch size and patch distance of 7 and 11, respectively. Finally, fnoise is implemented using noise level $1/f$ with factor $f \in \{2.5, 5, 10\}$.

6.2.2 Distortion Detection

Unlike the `Classification` dataset, each image in the detection dataset can have heterogeneous and multiple distortions, as shown in Figure 6.1(c). Since the SSD network used in the detection task accepts images of dimension 300x300 as input, we crop the center of the correct size before applying distortion.

Distortion levels are chosen uniformly at random with a range of minimum and maximum values used in the `Classification` dataset. When choosing the distortion regions, we sample the number of regions in a single image from a uniform distribution over the interval [1, 4]. The ratio of region size to image size is also picked uniformly from [0.3, 0.7] (average is 0.43). We generate 20 images per reference image, with a total of 16,080 images created. For evaluation, we split data into 80% training and 20% for the test. The assumption on the number of regions and sizes in the `Detection` dataset is quite reasonable. However in practice, there may be many small regions of distortions. Therefore, as in Figure 6.1(d), we created another dataset named `Detection-difficult` sets (Above dataset is named with `Detection-basic`). In this dataset, the minimum and the maximum number of regions per image are 5 and 9, respectively. The ratio of region and image size is changed to 0.1 and 0.3, with an average of 0.18.

6.3 Experimental Analysis

In this section, we first describe the analysis protocols and detailed model settings in Section 6.3.1. Then, we interpret the experimental results in Section 6.3.2. For ease of exposition, we separate the analysis report by classification and detection. Note that except for the pre-training of the network, we train and test using our proposed multi-modal distortion dataset (Section 6.2).

6.3.1 Experimental Setting

For the multi-modal distortion analysis, we use a convolutional neural network (CNN) to classify or detect the distortions within a single image. Here, we briefly explain the detailed setups and protocols used in our experimental analysis.

Distortion Classification. Throughout this chapter, we experiment with VGG-16 [129] and ResNet-101 [130]. Both networks are variants of CNN which consist of several convolutions, pooling, and fully-connected (FC) layers to recognize images. VGG-16 has

13 convolution layers and 3 FC layers. Because of the simplicity of this network, many researchers use the VGG-16 as a base network. The Atrous VGGNet is introduced by DeepLab [137] with an architecture similar to the VGGNet, but with a difference in the number of parameters in the final fully-connected layers, and its use of Atrous convolution that allows for the processing of arbitrary-sized field-of-views. ResNet uses *residual connections* to avoid the degradation problem. Without residual connections, deep networks are known to not only overfit but also show increasing training error. Unlike the VGGNet, ResNet-101 uses 101 layers with only the last layer being fully connected. Additionally, a global average pooling technique is used to reduce the number of parameters.

In practice, training the entire CNN from scratch is a difficult and time-consuming job. Also, if the dataset does not have sufficient training data, training does not converge well. Therefore, it is common to use pre-trained networks which have been trained on large external datasets such as ImageNet [138]. This transfer learning strategy works well if the distribution of the source dataset (used for pre-trained) and target dataset are similar. As ImageNet and our dataset have a similar distribution, we use CNNs pre-trained on ImageNet for all our experiments.

Distortion Detection. For the distortion detection, we use the single-shot multibox detector (SSD) [131]. With the development of CNN, many detection methods have been proposed such as R-CNN [139], Faster R-CNN [140], YOLO [141], OverFeat [142], and SSD [131]. R-CNN and its variants perform state-of-the-art detection, while inference time is very slow due to the limitation of their architecture. On the other hand, YOLO and SSD are real-time detection algorithms, with SSD outperforming YOLO. SSD computes multi-scale feature maps for detection by adding extra convolution layers at the end of the base network. Then six output feature maps from different convolution layers are concatenated to form the final layer. With this idea, SSD effectively detects objects of various sizes using a single, simple architecture, since the output maps from the lower layer tend to capture fine-grained details of the object. Predictors of SSD rely on convolution layers instead of the conventional fully-connected layers, to reduce inference time.

In our experiments, we use the best setting for SSD: 1) Use Atrous VGG-16 as a baseline

Table 6.1: **Quantitative results of distortion classification scenario.** We measure both classification accuracy and inference speed in this task. VGG- and ResNet-based models outperform IQA-CNN variants.

Method	w/o finetuning	w/ finetuning	Frame per sec.
IQA-CNN+	-	0.820	166
IQA-CNN++	-	0.782	**250**
VGG-16	0.858	0.984	83
Atrous VGG-16	0.855	0.984	90
ResNet-101	0.926	**0.988**	20

network since it shows a similar result with faster running time. 2) We use 300x300 as the input dimension. If we increase the input dimension to 500x500, the inference becomes much slower while the performance gain is relatively small. This shows that using 300x300 input can capture small-sized objects reasonably well in a short time via multi-scale feature maps. 3) On the contrary to the SSD used in the object detection task, the only data augmentation we use is the horizontal flip. This is because affine transformations might corrupt the details of the distortion, such as in the case of scaling and shearing.

6.3.2 Experimental Result

Distortion Classification. We first evaluate the distortion classification task using pre-trained networks. To do this, we remove the last fully-connected layer in the pre-trained network and freeze all layers in the network. Then, we add a new fully-connected layer suited for the number of classes of our dataset. Only this new layer is trained from scratch. Since our classification dataset is homogeneously distorted as in LIVE [133], we center-crop the images so the size fits the input layer of the network without concern for equal distribution of the distortion. We also evaluate a fine-tuned network. The training procedure of the fine-tuned network is the same as that of the non-fine-tune version, but in this case, we let the gradient propagate through all layers.

The result of the classification task is given in Table 6.1. To verify what effect pre-training has, we use IQA-CNN+ and IQA-CNN++[1] [127] as the baseline. As can be

[1]We re-implemented these networks. Note that we remove linear regression layer for predicting MOS.

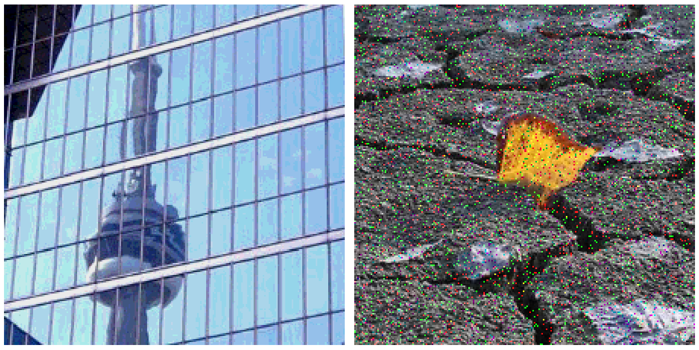

(a) quantization→fnoise (b) S&P→GWN

Figure 6.2: **Example of mis-classified distortions.** (a) has quantization distortion but is predicted as fnoise. (b) Salt and pepper noise classified as Gaussian white noise.

seen in the results, the pre-trained networks outperform baseline networks. Since all pre-trained networks have deeper architectures compared to the baseline, they are suitable for complex data due to the high network capacity. Moreover, the fine-tuning procedure makes the network better adapt to the new data. Among non fine-tuned pre-trained networks, ResNet outperforms the VGGNet family. This is due to the output of VGGNet being 4096-dimensional, which is twice as large as that of ResNet. However, all three networks show similar performance after being fine-tuned. We conjecture that this is probably because all networks have relatively large enough capacity to handle this task. Unlike the accuracy, inference time shows a large gap among different architectures. ResNet is much slower than the VGGNet family, and Atrous VGGNet is faster than the vanilla VGGNet since Atrous VGGNet subsamples parameters in its final two fully-connected layers.

In most cases, our model can classify distortion types very well. However, as in Figure 6.2, some images are commonly mis-classified. For example, salt and pepper noise is often mistaken as Gaussian white noise as seen in Figure 6.2(b) and vice versa. Figure 6.2(a) shows a case where the image does not show an abrupt color change, in which case the model also confuses quantization noise with fnoise. Such problems can be alleviated if we directly compare the given image with a reference image, but in practice, there are restrictions on using reference images.

Distortion Detection In this section, we present the results of the distortion detection

Table 6.2: **Quantitative results of distortion detection scenario.** We measure the detection performance via mAP and inference speed over a variety of IoU threshold values. Atrous VGG performs slightly better than ResNet with higher FPS.

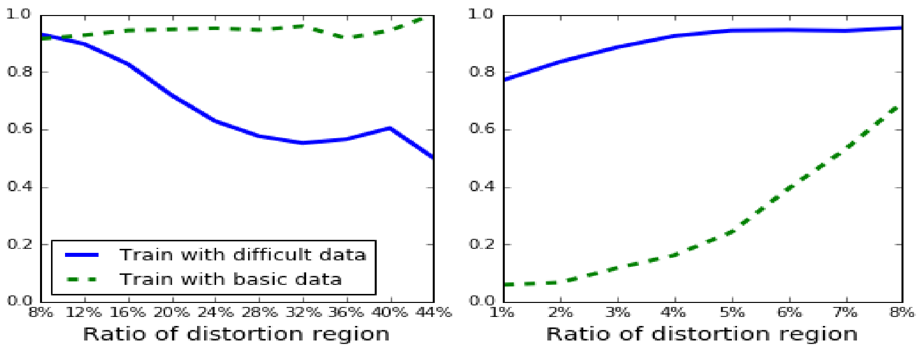
Method	Frame per sec.	mAP. IoU:		
		@ 0.5	@ 0.75	@ 0.9
ResNet-101	16	0.910	0.900	0.842
Atrous VGG-16	**63**	**0.919**	**0.906**	**0.873**

Table 6.3: **Transfer experiment results on multi-modal distortion detection.**

Train data → Test data	mAP. IoU:		
	@0.5	@0.75	@0.9
`basic` → `basic`	0.919	0.906	0.873
`difficult` → `basic`	0.915	0.864	0.728
`basic` → `difficult`	0.717	0.467	0.109
`difficult` → `difficult`	0.908	0.895	0.785

experiment with SSD. Here, we only use the Atrous VGG-16 and ResNet-101 since VGG-16 and Atrous VGG-16 have similar performance but Atrous VGG-16 is faster. The IQA-CNN+ achieves reasonable results with very fast inference time, however, since this model only has a single convolution layer, it is not appropriate to use the SSD that needs multiple convolution layers.

When training the network, we use the pre-trained CNN, which is fine-tuned over the `Classification` dataset, as a base network and stack SSD layer on top of the base network. In the evaluation step, we use the mean average precision (mAP) metric which measures the average precision of each class when the intersection over union (IoU) of the bounding box is one of $\{0.5, 0.75, 0.9\}$. In typical object detection tasks, performance evaluation is usually done with IoU @0.5. However, in our assumed scenario of finding the distortion regions to apply local dynamic compression, finding accurate boxes is vital. This is why we use a variety of IoU thresholds to assess the degree of our algorithm's region detection accuracy. Result of experiments are in Table 6.2. Surprisingly, there does not seem to be any advantages of using ResNet over VGGNet in this experiment. We believe that this is because VGGNet's capacity is large enough to fit the given data. In addition, Atrous VGG-16 excels ResNet-101 in terms of inference time, and it can be done



(a) Test Acc. on `basic` (b) Test Acc. on `difficult`

Figure 6.3: **Relationship between distortion size and detection accuracy with IOU@0.9.** (a) shows test accuracy with *basic* data and (b) is for *difficult* scenario.

in real-time on state-of-the-art GPUs such as Maxwell TITAN X.

As described in Section 6.2.2, distortion may occur in small local regions in the real world. Therefore, evaluating using only the *basic* dataset might not be a desirable strategy. To further investigate, we conducted a set of transfer learning experiments that evaluates the four combinations of training-testing scenarios, where the training and testing datasets can be either `basic` or `difficult`. Table 6.3 shows that the models trained and tested on the same type of dataset yield the best performance. This is natural since the model trained using only the *basic* set cannot catch distortions with the small region, while training only on the *difficult* set tends to drive the detector towards finding small regions. Note that in `basic` $\rightarrow$ `difficult` case, the trained model performs poorly when the IoU threshold is large, since it misses most small-size regions. The graph in Figure 6.3 shows how the accuracy changes as the size of the ground-truth regions changes. This also illustrates that training with `basic` data is good when the area of the region is large but performs worse for small sizes when trained on `difficult` data. To sum up, it is crucial to match the settings between train and test data. But since we do not know much about the test set, it is better to train on the `difficult` set to better-cope with possible worst-case scenario based on the assumption that distortion may occur in the small local region.

6.4 Discussion

In this chapter, we investigated and analyzed the novel problem of multi-modal distortion scenario by using distortion classification and detection. To do that, we created a new `Flickr-Distortion` dataset to train on. In distortion classification, we used fine-tuned models that have been pre-trained on the ImageNet data, to reach convergence quickly. By doing so, we discovered that fine-tuned CNNs outperform other baseline models such as IQA-CNN+. In the multi-modal distortion detection scenario, which also has not been addressed previously, we found that deep learning-based methods can efficiently classify and detect various distortions.

We expect that our analysis can motivate the usefulness of assuming the multi-modality of the distortion in many applications such as dynamic compression techniques or image reconstruction. In addition, some notable findings based on the analysis also can give some insight into the community. One of our main discoveries is that the difference in the quality of training images and the testing images significantly affects the overall performance. This is not necessarily a surprising fact from a machine learning point-of-view, but it does have important practical implications. Since we do not know in advance the quality of the image we process, it might be difficult to guarantee the performance of our final system. We propose that we deploy our system after training on image sets consisting mostly of `difficult` images, to cope with the worst-case scenario.

Future directions. We are planning to further develop our system to handle multiple distortions in a specialized manner. Our current system tries to classify and detect multiple distortions using a single structure. To account for multiple distortions, one must have a high-capacity system that could potentially lead to overfitting. If we could devise an ensemble-like system that specializes in each distortion type, the system might be able to focus on quality-neutral generalization within each distortion. In Chapter 7, we will present the image restoration task on the multi-modal distortion scenario by utilizing the analysis derived in this chapter.

Chapter 7

Multi-modal Distortion Restoration

In this chapter, we introduce the image restoration method for multi-modal distortion. As discussed in the earlier part, scenarios that assuming a single distortion only may not be suitable for many real-world applications. To deal with such cases, some studies have proposed sequentially combined distortions datasets. Viewing in a different point of combining, we introduce a spatially heterogeneous distortion dataset in which multiple corruptions are applied to the different locations of each image. This dataset is similar to the one in Chapter 6, but we refine it to be more realistic. In addition, we propose a mixture of experts network to effectively restore a multi-distortion image. Motivated by the insight in Chapter 6 and the multi-task learning literature, we design our network to have multiple paths that learn both common and distortion-specific representations. The proposed network performs both distortion recognition (in an unsupervised way) and restoration simultaneously so that it does not require any additional processing pipeline. By doing so, our model is effective for restoring real-world distortions and efficient in terms of model serving perspective.

7.1 Overview

As we observed in the previous chapters, the performance of the image restoration methods has been significantly improved by the use of a deep learning-based approach.

However, most of these models assume that the image is corrupted by a single distortion only (Figure 7.1b), which may not be suitable for the real scenarios. In real-world applications, there can be mixed or multiple distortions in one image such as a JPEG artifact of the blurry image or a photo taken on a hazy and rainy day. To cope with such a multi-modal nature, some studies have proposed new datasets [8, 143] or methods [144, 145] recently. They generated datasets by overlapping multiple distortions sequentially, which makes the assumption more realistic than the single distortion (Figure 7.1c).

Viewing a different point of the multi-modality in distortion, in this chapter, we will discuss a spatially heterogeneous distortion scenario. In this circumstance, distortions are applied in different regions of the image (Figure 7.1d). This concept makes the scenario a proxy environment of real-world applications such as multi-camera systems. For example, in the case where the images are acquired from various devices or post-processors, stitching these images may produce output that has different quality regions, thus degrading the recognition performance. Because of the nature of the spatial-heterogeneity, it is crucial to catch both *what* and *where* the corruptions are, unlike the existing multi-modal distortion datasets [8, 143] which spread corruptions to the entire image. The overall motivation and the core spirit of viewing multi-modality in distortion are identical as we discussed in Chapter 6. Here, we go beyond the previous environment more realistically since the dataset used in Chapter 6 is spatially sparse thus may not be ideal to potential applications in which multiple images acquired from different devices with multi-modal distortion (i.e., image stitching process).

To address the above requirements, we propose a mixture of experts with a parameter sharing network (MEPSNet) that effectively restores an image corrupted by the spatially-varying distortions. Motivated by the multi-task learning [147, 148] and the mixture of experts [149], we build our network to have a multi-expert system (Figure 7.2). With this approach, individual distortion can be treated as a single task, thus the model divides and distributes the task to appropriate experts internally. By doing so, experts are able to concentrate on restoring only the given single distortion. We experimentally observed that each expert learns a particular distortion distribution as well. Note that even though we build every expert to be identical, any type of the network design can be adapted thanks

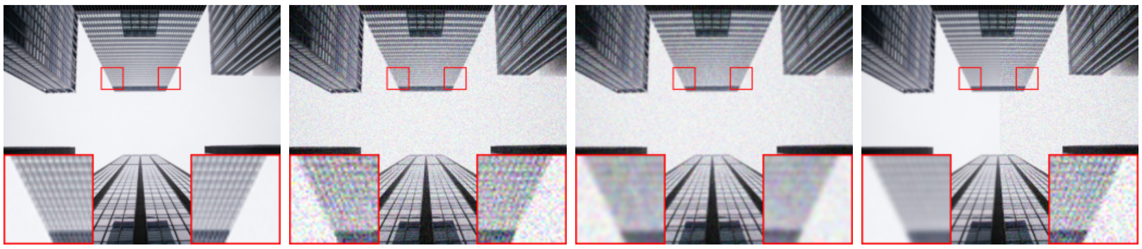


Figure 7.1: **Comparison of the modality assumptions in distortion.** **(a)** Clean image. **(b)** Single distortion (Gaussian noise). Only one corruption is applied to the image. **(c)** Mixed distortions [8]. Multiple distortions corrupt the image in sequentially (Gaussian blur and Gaussian noise), but no variation on the spatial domain. **(d)** Our proposed spatially-variant distortion. Instead of mixing in sequentially, we *spatially* combine heterogeneous distortions (left: Gaussian blur, right: Gaussian noise).

to the flexibility and modularity of our framework.

However, naively constructing the sub-networks (experts) may limit the power of using MTL. Misra et.al., [10] investigated the trade-offs amongst different combinations of the shared and the task-specific architectures and revealed that the performance mostly depends on the tasks, not on the proportion of the shared units. Based on the above investigation, they proposed a cross-stitch unit so that the network can learn an optimal combination of shared and task-specific representations. Following the analysis of Misra et.al., [10], we use the soft parameter sharing [146] to guide experts to learn both shared and distortion-specific information effectively. In this approach, convolutional layers of the experts only contain the coefficient vector, not the entire weights and biases. Instead, these are stored in the global template bank thus layers adaptively generate their parameters by a linear combination between the coefficient vector and the templates. It allows each expert to grasp not only the characteristics of the individual distortions but also common representation from the various corruptions automatically. In addition, the number of the parameters is decoupled to the number of the experts and we can increase the experts only using negligible additional parameters. Our experiments show that MEPSNet outperforms other restoration methods including OWAN [144] which is specifically designed network to manage multiple distortions.

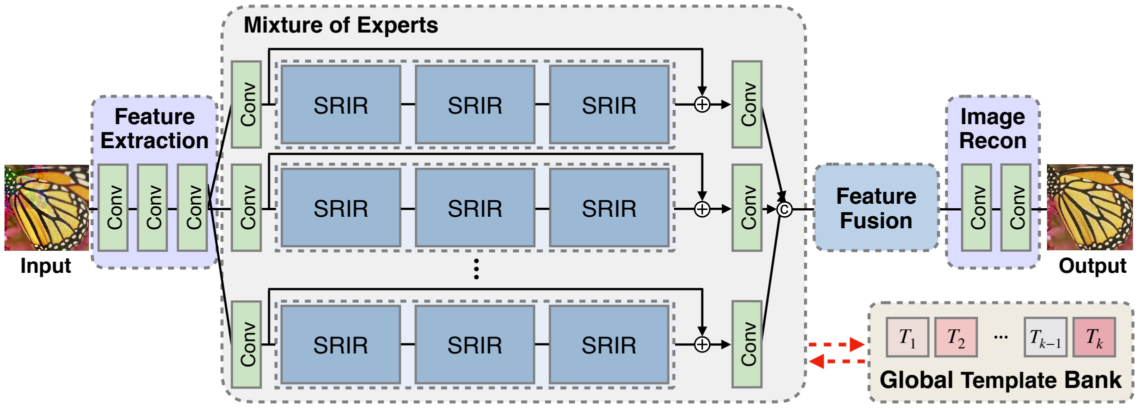


Figure 7.2: **Overview of the MEPSNet.** Our network is composed of 1) feature extraction, 2) mixture of experts, 3) global template bank, 4) feature fusion, and 5) reconstruction modules. Here, SRIR denotes our proposed shared residual-in-residual block. Ⓒ and ⊕ symbols are concatenation and element-wise addition operations, respectively. The mixture of experts unit has several pathways, each with multiple SRIR blocks. The parameters of all SRIRs are soft-shared through the global template bank [146].

7.2 Background

Multiple image distortion restoration. In real-world applications, multiple distortions can damage entire images, or only the partial regions. Restoring such images using the model trained on a single distortion dataset may produce undesirable artifacts due to the mismatched distribution. To close the gap between the real and simulated data, recent studies have proposed new datasets [8, 143] and methods [144, 145] for multi-distortion restoration task. In their datasets, images are damaged with sequentially applied distortions [8, 143] or only small parts of the image are corrupted. To restore multiple distortions, Yu et al., [8] used the toolbox that has several distortion specialized tools. Then, the framework learns to choose the appropriate tool given the current image. Similarly, path-restore [145] and OWAN [144] adopt a multi-path approach so that the models dynamically select an appropriate pathway for each image regions or distortions. Although our method is also motivated by the multi-path scheme, we have two key differences. First, our proposed network is built for restoring spatially-varying distortions. Second, by cooperatively using a mixture of experts and parameter sharing strategies, we can achieve more advanced performance than the other competitors.

Multi-task learning. Multi-task learning (MTL) guides a model to learn both common and distinct representations across different tasks [147, 148]. One of the widely used approaches for MTL is combining a shared and task-specific modules [147]. Based on this work, numerous studies have been investigated the power of MTL in various tasks [10, 150, 151]. Among them Misra et al., [10] is one notable work; they proposed a cross-stitch unit to optimize the best combination settings for given tasks with a end-to-end training. Without this module, the optimal point depends on the tasks, and the searching process may become cumbersome. Hinted by this work, our network also learns to find the balance between the shared and the distortion-specific representations using the soft parameter sharing approach.

7.3 Spatially Heterogeneous Distortion Scenario

In this section, we introduce a novel spatially-heterogeneous distortion dataset (SHDD). Our proposed dataset is designed to simulate the scenario where the image is corrupted by spatially varying distortions. To implement this idea, we synthetically generate corrupted images using *divide-and-distort* procedure. That is, we divide clean images into the multiple blocks (divide), and corrupt each block with selected distortions (distort).

In *divide* phase, we split images according to the virtual horizontal or vertical lines (Figure 7.3). These lines are randomly arranged so as to prevent the model from memorizing the position of the resulting regions. We create three levels of difficulties (easy, moderate, and difficult), by varying the number of split regions. The reason for creating a multi-level dataset is two-fold. First, we consider the relationship between the restoration hardness and the number of regions presented in a single image. Second, we would like to explore the robustness of the model by training on one level and evaluating it on others. In *distort* stage, we corrupt each block with randomly selected distortion. We use **1)** Gaussian noise, **2)** Gaussian blur, **3)** f-noise, **4)** contrast change, and **5)** identity (no distortion). Note that we include *identity* to the distortion pool. By including it, we can

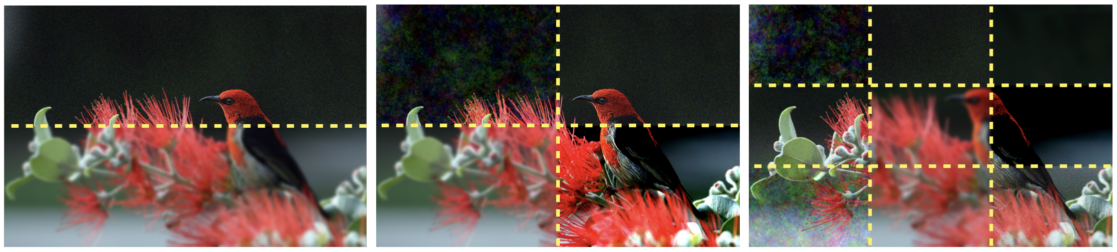

(a) Easy (b) Moderate (c) Difficult

Figure 7.3: **Examples of the spatially-heterogeneous distortion dataset (SHDD).** We separate our dataset into three levels (`easy`, `moderate`, and `difficult`) according to the number of the blocks in a single image. To generate a dataset, we first split image to sub-images using the virtual perforated line (divide phase) and corrupt each region with different distortions (distort phase). Best viewed on display.

measure the generalizability of the model in depth since deep restoration methods tend to over-sharpen or over-smooth when the input is already of high-quality. In addition, it simulates more realistic cases where the real-world scenarios suffer very often (*i.e.* merging clean image to the corrupted ones).

We build SHDD based on DIV2K [121]. This dataset has 800 and 100 images for training and validation respectively. We use half of the DIV2K validation dataset as validation of SHDD and the rest of half for testing. For each of the high-quality images, we generate 12 distorted images (training dataset: 9,600 = 800×12 images) to cover data samples as densely as possible since SHDD is inherently sparse due to the spatially-varying distortions. We set {easy, moderate, difficult}-levels by chopping each image {2, 3, 4}-times (Figure 7.3). The distortions used in SHDD are carefully selected following the recent image distortion datasets [152, 153]. These reflect the real-world scenario, especially for image acquisition and registration stage. When applying the distortions, we randomly sample its strength from following ranges: **1)** [0.005, 0.02]-variances for Gaussian white noise, **2)** [1.0, 2.5]-variances for Gaussian blur, **3)** [6.0, 10.0]-scales for f-noise (pink noise), and **4)** [25.0, 40.0]-levels for contrast change.

7.4 Approach

Our proposed mixture of experts with a parameter sharing network (MEPSNet) is composed of five parts: feature extraction, mixture of experts, template bank, feature fusion, and reconstruction blocks (Figure 7.2). In Section 7.4.1, we show an overview of our proposed method. Then, we describe the multi-expert architecture and the feature fusion module in Section 7.4.2 and 7.4.3.

7.4.1 Model Overview

Let's denote $\mathbf{X}$ and $\mathbf{y}$ as a distorted and a clean image, respectively. Given the image $\mathbf{X}$, a feature extractor f_{ext} computes the intermediate feature $\mathbf{F}_0 = f_{ext}(\mathbf{X})$. To extract informative features, the extraction module has multiple convolutional layers (we use three layers) and their dimensions are gradually expanded up to 256 unlike the recent image restoration methods [17, 113]. We observed that the capacity of the extraction module makes the impact to the performance. We conjecture that it is due to the usage of multiple distortion-specialized experts. With this concept, it is crucial to extract informative shared representation to encourage the individual experts concentrate solely on their own goal. Extracted intermediate feature $\mathbf{F}_0$ is then fed into the mixture of experts module which outputs a concatenated feature $\mathbf{F}_D$ as in below.

$$\mathbf{F}_D = \left[f^k_{exp}(\mathbf{F}_0) \right], \quad \text{for } k = 1 \ldots N \tag{7.1}$$

Here, N is the number of experts, f_{exp} and [.] denote the expert branch and the channel-wise concatenation respectively. With this deep feature $\mathbf{F}_D$, we finally generate restored image $\hat{\mathbf{y}}$ by Equation 7.2. To guide the reconstruction module to gather multiple information adequately, we attach the attentive feature fusion module f_{fuse} before the image reconstruction unit.

$$\hat{\mathbf{y}} = f_{recon}(f_{fuse}(\mathbf{F}_D)). \tag{7.2}$$

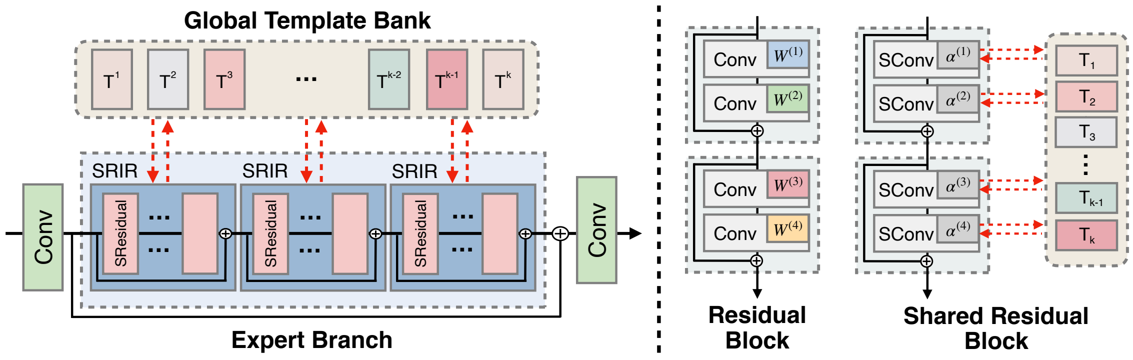


Figure 7.4: **An illustration of the mixture of experts module. (Left)** Expert branch with parameter sharing. Experts are shared using global template bank (other branches are omitted). In each pathw, there exist three shared residual-in-residual (SRIR) units which have several shared residual blocks. **(Right)** Comparison of the standard and our shared residual blocks. While convolutional layers of the standard block have their own parameters $\{W^{(1)}, \ldots, W^{(4)}\}$, ours only have coefficient. Instead, weights are adaptively generated using coefficients $\{\alpha^{(1)}, \ldots, \alpha^{(4)}\}$ and templates $\{T^1, \ldots, T^k\}$.

We optimize our MEPSNet using a pixel-wise L_2 loss function. While several criteria for training restoration network have been investigated [154, 111], we observed that there is no performance gain of using other loss functions in our task.

7.4.2 Mixture of Parameter Shared Experts

Mixture of experts module. In our network, this module is the key component to successfully restore heterogeneous distortions. As shown in Figure 7.2, multiple branches, dubbed as *experts*, are positioned in between the feature extraction and the feature fusion blocks. Each expert has the same structure, which consists of three contiguous shared residual-in-residual (SRIR) units and few convolutional layers (green boxes in Figure 7.2) that envelope the SRIR blocks. Following the prior works [17, 113], we use a long skip connection to bridge across multiple intermediate features for stable training. The SRIR is composed of multiple residual blocks [17, 53] as shown in Figure 7.4 (left). We also employ additional shortcut connection between the residual blocks to further stabilize the training [113]. Note that the structure of the experts is not restricted to be identical; they could be the networks with different receptive fields [155] or disparate operations [144].

However, we choose to set all the experts to have same structure considering both the simplicity and the performance.

In contrast to the conventional mixture of experts [149], our experts module does not have an external gating network. Instead, distilled information adaptively selects their importance themselves by the self-attention scheme (Section 7.4.3). Since we do not attach additional gating mechanism, now the formulation of our mixture of experts module is related to the multi-branch networks [156, 157]. However, we observed that the vanilla multi-branch network requires very careful tuning to stabilize the training and even shows degraded performance when we increase the number of the branches or the depths of each branch. We hypothesize that the degradation issue of using multi-branch system arises due to the *isolated branch* structure. That is, no bridge exists between the branches, thus experts (branches) learn all the representations on their own way without referring others. This is inefficient since some information are sufficient to be extracted once and shared to others. To mitigate such issue, we employ soft parameter sharing scheme.

Soft parameter sharing. We use this scheme [146] to guide the experts in acquiring both shared and distortion-specific information effectively. Contrary to the hard parameter sharing (*i.e.* recursive network), parameters of the layer are generated by a linear combination of the template tensors in the bank. We set the bank as global (Figure 7.4, left) so that all the SRIRs are shared altogether. The SRIR has shared residual blocks (SResidual) which communicate with a template bank. The SResidual is composed of several shared convolutional layers (SConv), and the parameters of the SConv are adaptively generated through the template bank (Figure 7.4, right). In detail, a standard convolutional layer stores weights $W \in \mathbb{R}^{C_{in} \times C_{out} \times S \times S}$ (S is kernel size). In contrast, our SConv only contains a coefficient vector $\alpha \in \mathbb{R}^K$, where K is the number of templates in the bank. Instead, the global template bank holds all the weights as $\{\mathbf{T}_1, \dots \mathbf{T}_K\}$ where $\mathbf{T}_k$ is the $[C_{in} \times C_{out} \times S \times S]$-dimensional tensor. By referring these templates, each layer generates their adaptive weight $\tilde{W}$ as

$$\tilde{W} = \sum_{j=1}^{K} \alpha_j \mathbf{T}_j. \tag{7.3}$$

Jointly using a parameter sharing and the mixture of experts provides two advantages: First, the number of the parameters is determined by the number of templates K, not the experts. Second, it improves the restoration performance compared to the model without a parameter sharing. In detail, we share the parameters not only within the experts but also between the branches. This allows every expert to jointly optimize the common representations from various distortions while each expert produces their specialized features. We can also interpret the benefit of the parameter sharing as in the multi-task learning literature. In a multi-task learning context, finding a good balance between the task-specific and the shared representations is cumbersome job and moreover, the optimal point depends on the tasks themselves [10]. To find the best combination without human-laboring, they share the intermediate representations using a cross-stitch unit [10]. Our approach has an analogous motivation to them but we tackle this with a parameter sharing.

7.4.3 Attentive Feature Fusion

As described in Section 7.4.2, each expert branch generates their specific high-level features. Our attentive feature fusion module takes concatenated features $\mathbf{F}_D$, which is the output of the mixture of experts module, and fuses this information via channel-wise attention mechanism [113, 158]. With given feature $\mathbf{F}_D \in \mathbb{R}^{C\times H\times W}$, we first apply global average pooling to make C-dimensional channel descriptor $\mathbf{F}_{CD} \in \mathbb{R}^C$ as in below.

$$\mathbf{F}_{CD} = \frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W}\mathbf{F}_D(i,j), \tag{7.4}$$

where $\mathbf{F}_D(i,j)$ denotes the (y,x) position of the feature $\mathbf{F}_D$. With $\mathbf{F}_{CD}$, we calculate the scaling vector $\mathbf{S}$ using a two-layer network followed by a simple gating scheme. Then, $\mathbf{F}_F$ is produced by multiplying $\mathbf{S}$ and $\mathbf{F}_D$ in a channel-wise manner. Finally, the reconstruction block receives this feature and generates a restored image $\hat{\mathbf{y}}$.

$$\begin{aligned}\mathbf{S} &= \sigma(W_2 \cdot \delta(W_1 \cdot \mathbf{F}_{CD})),\\ \mathbf{F}_F &= \mathbf{S} \cdot \mathbf{F}_D. \end{aligned} \tag{7.5}$$

Here, $\sigma(.)$ and $\delta(.)$ are sigmoid and ReLU respectively, and $\{W_1, W_2\}$ denotes the weight set of convolutional layers. With this attentive feature fusion module, diverse representations inside of the $\mathbf{F}_D$ are adaptively selected. Unlike ours, previous mixture of experts methods [149] generate attention vector using the external network. However, we observed that such design choice does not work well in our task. Related to the isolation issue of the vanilla multi-branch network, as described in Section 7.4.2, we suspect that isolated external gating network cannot judge how to select features from the multiple experts adequately. In contrast, our fusion module is based on the self-attention [113, 158, 159]. With this concept, attentive feature fusion unit is now closely linked to the main expert module so that is able to decide which feature to take or not more clearly.

7.5 Experiment

Implementation details. We train all the models on moderate level of SHDD. The reason for using single level only for training is to measure the generalizability of the model by evaluating on unseen (easy and difficult) cases. In each training batch, 16 patches with a size of 80×80 are used as input. We train the model for 1.2M iterations using ADAM optimizer [160] with settings of $(\beta_1,\ \beta_2,\ \epsilon) = (0.9,\ 0.99,\ 10^{-8})$, and weight decay as 10^{-4}. We initialize the network parameters following He *et al.* [51]. The learning rate is initially set to 10^{-4} and halved at 120K and 300K iterations. Unless mentioned, our network consists of three experts, each of which has three SRIRs. We choose the number of SResidual blocks in SRIR to 12 and the number of the templates K as 16.

7.5.1 Comparison with State-of-the-art Methods

Baseline. we use following restoration methods: DnCNN [161], VDSR [112], OWAN [144] and RIDNet [162]. OWAN is proposed to restore multiple distortions while others are for a single distortion. We modify VDSR by stacking convolutional layers four times than the original ones to match the number of the parameters to the others. For OWAN and RIDNet, we use author's official code.

Evaluation on SHDD. We compare the MEPSNet to the baselines on SHDD using pixel-driven metrics such as PSNR and SSIM. Table 7.1 shows the quantitative comparison on the different levels of the SHDD test set. In this benchmark, our proposed method consistently outperforms the others. For example, the performance gain of the MEPSNet in moderate level is +0.53 dB PSNR compared to the second best method, RIDNet. In addition, MEPSNet achieves the best performance on the unseen settings as well, and especially shows the superior PSNR on difficult level, +0.41 dB to the second best. It should be noted that OWAN [144] is also devised for the multi-distortion restoration. However, their performance is much lower than both ours and RIDNet. We conjecture that isolating all the operation layers and attention layer results in degraded performance. On the other hand, ours can fully enjoy the effect of using multi-route by sharing the parameters altogether. Figure 7.5 shows the qualitative results of our model. For the contrast change distortion (1st and 4th rows), the other methods create unpleasant spots (OWAN, RIDNet) or regions (DnCNN) while ours successfully reconstructs the original color. Similarly, MEPSNet effectively restores the other corruptions, such as f-noise (2nd row) or Gaussian blur (3rd row).

Evaluation on image recognition tasks. To further compare the performance of our method, we use image recognition tasks: object detection and semantic segmentation. To be specific, we distort images of COCO dataset [163] with same protocols of SHDD. Then, we restore distorted images using the trained models on SHDD. We evaluate mean average precision (mAP) score using faster R-CNN [164] and mask R-CNN [150] for detection and segmentation respectively. As in Table 7.2, mAPs of the distorted images are significantly lower than the clean cases. Restored results with our proposed MEPSNet show the best

Table 7.1: **Quantitative comparison on SHDD.** We benchmark the MEPSNet in three levels along with the other deep learning-based restoration methods.

Method	Levels of SHDD		
	Easy	Moderate	Difficult
DnCNN [161]	25.29 / 0.7110	25.54 / 0.7354	26.70 / 0.7723
VDSR [112]	27.34 / 0.7709	25.73 / 0.7701	25.95 / 0.7760
OWAN [144]	30.95 / 0.9181	29.77 / 0.9112	29.27 / 0.9098
RIDNet [162]	34.19 / 0.9361	32.94 / 0.9317	32.30 / 0.9282
MEPSNet (ours)	**34.23 / 0.9369**	**33.47 / 0.9331**	**32.71 / 0.9284**

Table 7.2: **Quantitative comparison (mAP) on object detection and semantic segmentation tasks.** We use faster R-CNN [164] and mask R-CNN [150] to measure mAP for object detection and instance segmentation, respectively.

Task	Clean	Distorted	DnCNN	VDSR	OWAN	RIDNet	MEPSNet
Detection	40.2	26.4	26.8	25.6	28.6	**29.5**	**29.5**
Segmentation	37.2	24.4	24.8	23.6	26.5	27.4	**27.5**

mAP than the other methods and RIDNet [162] is the only method comparable to ours.

7.5.2 Model Analysis

In this section, we dissect our proposed MEPSNet through the internal analysis. Unless mentioned, we set the MEPSNet to have three SRIRs each of which includes twelve SResidual blocks. We trained our model using 48×48 input patches.

Ablation study. In Table 7.3, we analyze how the mixture of experts (ME) and the parameter sharing (PS) affect the restoration performance. First, using PS (2nd row) outperforms the baseline (1st row) by a huge margin only using half of the parameters. We hypothesize that the PS through the global template bank successfully guides the model to combine low- and high-level features internally. The advantages of combining the multiple features are also verified in recent restoration methods [165], and network with PS (via template bank) enjoy the fruitful results by an alternative implementation of the feature aggregating.

Simultaneously applying PS and ME additionally gives dramatic improvements (2nd vs.

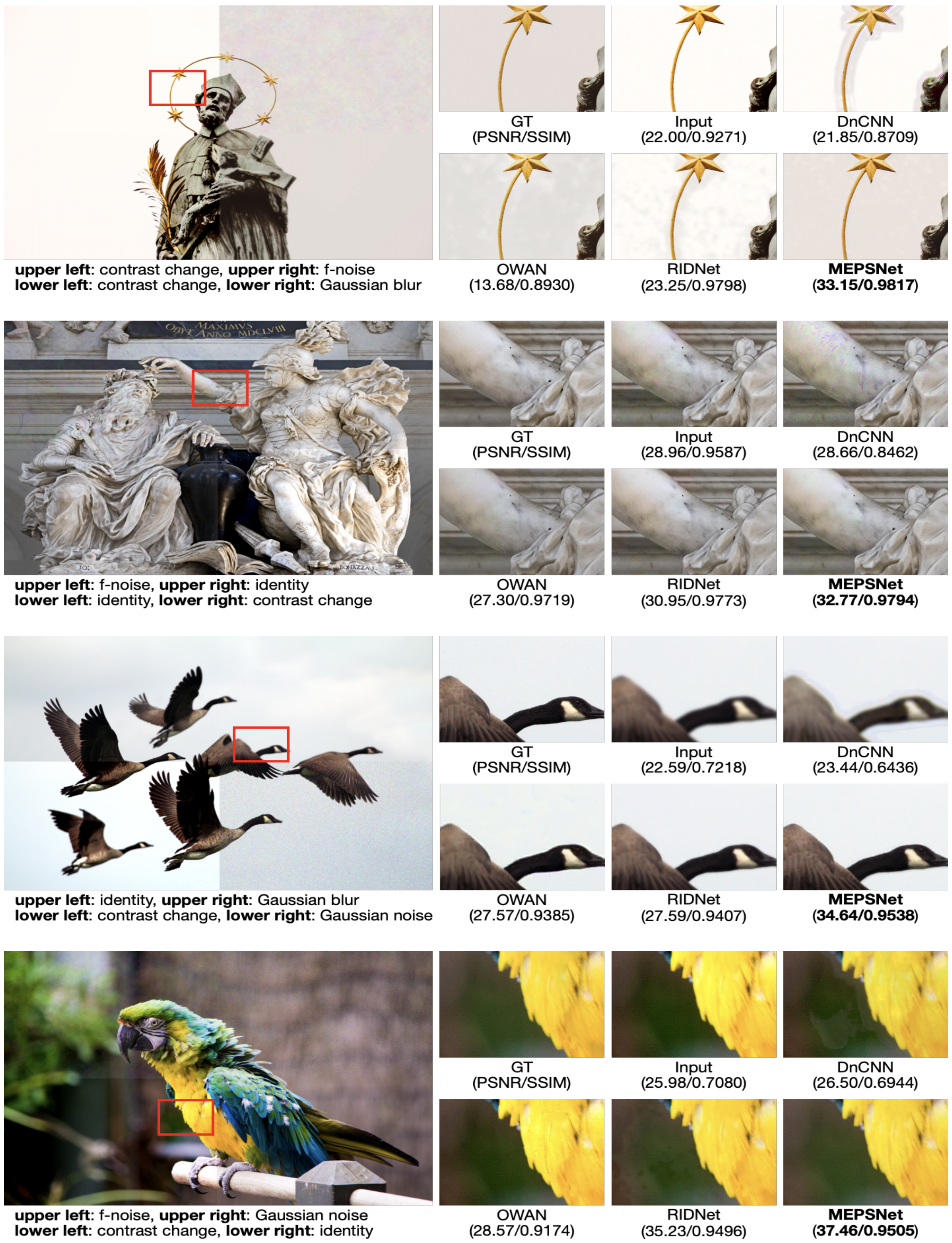


Figure 7.5: **Qualitative comparison on SHDD.**

Table 7.3: **Ablation study of MEPSNet.** ME and PS denote the mixture of experts and parameter sharing, respectively. Using both modules dramatically improves the performance of the baseline and successfully suppressing the number of the parameters.

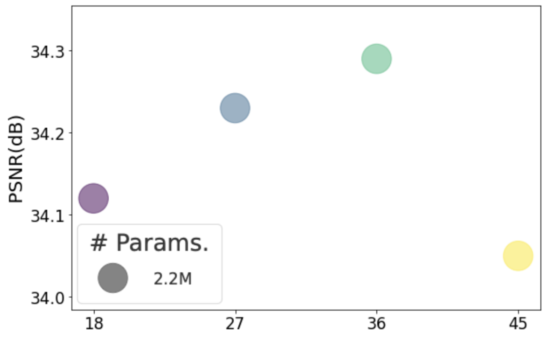
ME	PS	# Experts	# Params.	PSNR / SSIM
		1	3.9M	29.36 / 0.8835
	✓	1	1.9M	33.55 / 0.9322
✓	✓	3	2.2M	34.29 / 0.9353
✓	✓	5	2.6M	**34.39/0.9362**

Table 7.4: **Effect of the multi experts and parameter sharing under the layer constraint scenario.** We force the number of the residual (or SResidual) blocks in entire mixture of experts module as 36. That is, expert has 36 blocks for single expert case (1st, 3rd rows), whereas 12 blocks for each when using three experts (2nd, 4th rows).

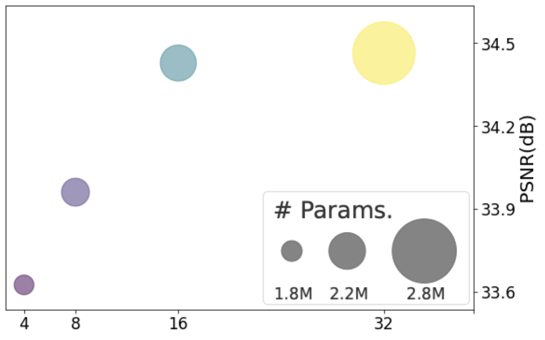
# Blocks	# Experts	PS	PSNR / SSIM
36	1		29.36 / 0.8835
	3		**32.21/0.9226**
	1	✓	33.55 / 0.9322
	3	✓	**33.78/0.9334**

3rd rows). Even though we triple the number of experts, the total number of parameters is marginally increased by only 15%, thanks to the parameter sharing scheme. Increasing the number of the experts to five (4th row) further boosts the performance as well. However, unless we share the parameters, using five experts increases about 40% of the parameters compared to the single expert network due to the additional coefficients and extra burden to the fusion module. Considering the trade-off between the number of the parameters and the performance, we choose to use three experts for the final model.

To analyze the impact of the mixture of experts and parameter sharing more clearly, we conduct an experiment based on the layer constraint setting as in Table 7.4. In this scenario, the number of the residual (or SResidual) blocks in the entire mixture of experts module is fixed to 36. Without a multi-expert (1st, 3rd rows), models are three times deeper than the others (2nd, 4th rows). However, single expert models result in degraded performance than the multi-expert. It may contradict the recent trends of single distortion restoration task [17, 113]: deeper network is better than the shallow one. Such a result may indicate that it is necessary to view multi-distortion restoration task on a different



(a) PSNR vs. # of SResidual blocks



(b) PSNR vs. # of templates

Figure 7.6: **Effect of the number of the SResidual blocks and the templates. (a)** Varying the number of the blocks of each expert. The number of the experts are fixed as three for all the cases. With parameter sharing, they all have similar parameters. **(b)** Increasing the number of the templates in the global bank from 4 to 32.

angle to the single distortion restoration literature.

Effect of the number of layers and templates. In Figure 7.6a, we fix the number of the experts to three and vary the number of the SResidual blocks for each of the expert. Not surprisingly, we can stack more layers without a sudden increase in the number of the parameters. The performances are consistently improved except the 45 blocks case may due to the unstable training of the extremely deep network. Increasing the number of the templates also gives the progressive gains as show in Figure 7.6b. With diverse templates, layers can generate more complex and advanced weight combinations so that it is possible to restore complicated distortion patterns.

Feature visualization. Figure 7.7 shows the output feature map of the mixture of experts module. The model without a mixture of experts (Figure 7.7b) struggles to recover all the distortions simultaneously while ours separates the role to each other (Figure 7.7c-e). For example, expert 1 produces coarse and large activations, implying that it mainly deals with contrast change and partially reduces the color tone of the f-noise and Gaussian noise. On the other hand, expert 2 concentrates on recovering edges for the Gaussian blur. The expert 3 also focuses on the primitives but finer elements than the expert 2.

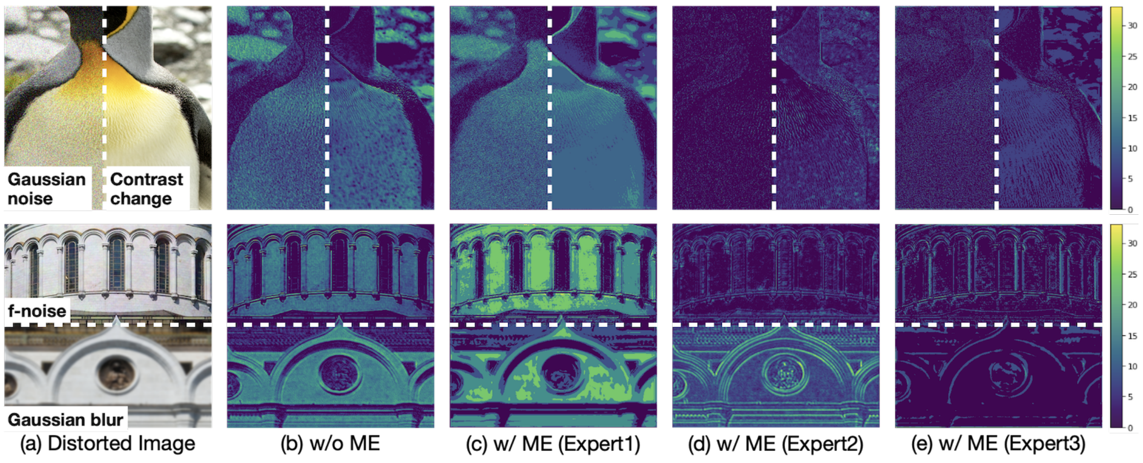


Figure 7.7: **Visualization of the extracted feature maps from the experts. (a)-upper** Distorted image by Gaussian noise (left) and contrast change (right). **(a)-lower** Distorted image by f-noise (top) and Gaussian blur (bottom). **(b)** Generated feature map of the single expert module, without mixture of experts. **(c-e)** Feature maps produces by three different experts when using multi-expert system.

7.6 Discussion

In this chapter, we presented the spatially-heterogeneous distortion dataset (SHDD) and the mixture of experts with a parameter sharing network (MEPSNet) for effective distortion restoration. The proposed SHDD assumes the cases where the multi-modal corruptions are applied to the different locations. To appropriately handle the above scenario, our method is motivated by the analysis from the multi-task learning contexts [10]. By jointly utilizing the mixture of experts scheme [149] and the parameter sharing technique [146], MEPSNet outperforms the other image restoration methods on both the pixel-based metrics and the indirect measures. As future work, we plan to integrate the spatially-heterogeneous and the sequentially-combined distortions [8] concepts to further reduce the disparity between the simulated and the real-world environments.

Chapter 8

Conclusion and Discussion

We have studied how deep learning-based image restoration methods achieve "efficiency" in various perspectives. This chapter summarises key insights derived from the previous chapters. We close the thesis with future directions of my work and guide some notable long-term perspectives for future image restoration research.

8.1 Key Insight

In this section, we revisit each chapter with a brief summary of key insights and methodologies developed to build an efficient image restoration model.

Part I: Model Perspective Efficiency. We first have done rethinking the true efficiency of the model that it is necessary to consider not only the model parameters but also the model operations (Chapter 2). Although this is an obvious aspect when we apply the deep image restoration methods in real applications due to the latency issue, many previous studies have overlooked it. To achieve this, we have developed a lightweight super-resolution network with a novel cascading connection and by adapting some network design techniques. In Chapter 3, we have gone beyond by focusing on the extreme super-resolution. We have found that many previous methods (including the lightweight model proposed in Chapter 2) suffer the training instability, resulting in an inferior performance.

This could be alleviated by providing upsampled input images to the network, however, it leads to a huge computation burden. By bridging two approaches, which are the early- and later-upsampling, we have taken the progressive upsampling so that the model achieves a good balance point between the performance and the efficiency.

Part II: Data Perspective Efficiency. One of the major open questions on the image restoration community is the distribution shift between the training and test dataset. To deal with this issue, some recent studies have introduced a new dataset that reflects the real environment. However, collecting such a dataset is expensive and even impossible for some cases. To detour this, in Chapter 4, we have analyzed the effects of the data augmentation in the image restoration task and have observed that appropriate data augmentations give beneficial impacts, especially for the realistic dataset. With the data augmentation methods, now we can efficiently train the big restoration models with a little amount of the real-world dataset. We have further investigated the more limited scenario where no high-quality images exist (i.e., unsupervised image restoration). By carefully modifying the zero-shot image restoration method [9], we have reformulated the unsupervised environment into the supervised training regime. With this simple concept, the proposed method has shown outperformed results compared to the previous method.

Part III: Toward Multi-modal Distortion. Images acquired from real-world devices contain multiple distortions. Unfortunately, many previous studies only have focused on the uni-modal distortion so that most of the methods significantly degraded in the real environment. To close these gaps, we have studied several questions: what is the multi-modal nature of distortions? How we infuse the multi-modality to the image restoration model? How we make an efficient multi-modal distortion restoration method? In Chapter 6, we have discussed the assumption that multiple distortions appear in different sub-regions of image. With this concept, we have analyzed whether deep networks can recognize such distortions. The results have shown that modern deep methods are able to catch the subtle difference of the multiple distortions very accurately, implying that we can leverage this idea to a multi-modal distortion restoration task. In Chapter 7, we have designed a multi-modal distortion restoration network referring to the insight derived in Chapter 6. One possible approach is the *restoration pipeline* where the framework consists of the distortion recognition stage and the series of distortion-specific restoration networks. Such framework is straightforward to use, however, we have not considered this due to the efficiency issue (for both latency and model size). In contrast, we have built a single network system that is composed of multi-expert modules so to make each expert takes charge of individual distortion. With this architecture, the proposed network has achieved state-of-the-art performance with reasonable efficiency.

8.2 Future Direction

This thesis has explored the efficiencies of deep image restoration methods in multiple directions, however, there remain many unanswered questions and interesting aspects to investigate. In this section, we point some potential future research topics toward an efficient image restoration method as well as more long-term future directions of this field.

8.2.1 Follow-up Topics

More realistic multi-modal distortions. In Chapter 6 and 7, we have studied the multi-modality in distortions. To make the dataset, we have corrupted the images synthetically based on the multi-modal assumption. Because of the synthetic distortion process, there still remains a distance between the real environment. As future work, closing the gap between synthetic and realistic datasets is meaningful to the community.

Stronger data augmentation. In high-level vision tasks, there have been a lot of data augmentation methods introduced beyond Mixup [3] and CutMix [4]. We believe that in low-level vision, better data augmentations over CutBlur are not discovered yet. Then, what are the possible candidates? Beyond the pixel-level inspection, rethinking the image as the frequency signal would be one possible approach as having done in SFM [94].

Model efficiency in the wild. Although the proposed networks in this thesis have accomplished the high efficiency on the number of the operations (MultAdds mostly), yet it is unclear that MultAdds and actual inference time are perfectly aligned. As we have seen in Chapter 2, we can get some hints on this that MultAdds do not reflect the latency very accurately especially when inference on GPU. There are many potential reasons for this result, but we suspect that it because 1) internal implementation of the operations is framework-dependent, thus for some cases, it hinders the efficiency improvement when the features are not implemented in hardware-friendly (e.g., group convolution). 2) For CARN, the cascading module is the key module for efficiency. However, because of the multiple concatenation operations, this could slow down the actual runtime in GPU as also described in [42]. For example, it has studied that the fragmented network design or element-wise operations including concatenation (that CARN mostly depends on to boost the performance) potentially increase the memory access cost, resulting in worse GPU latency. Overall, the future direction of the model efficiency should be preceded with the understanding of the hardware feature to "real" efficiency in the real world.

Adaptive efficiency control. Is efficiency is the ultimate solution for all the cases? When the image is severely corrupted, one might want to restore it back to a clear image even it takes extra time or burden. However, since most of the model efficiency modules are "static", this cannot handle the adaptive control from the users' preference. With this respect, one solution is the recognize-restoration process. That is, if we can recognize the distortion information with a little overhead, the dynamic module gives high flexibility of the efficiency that reflects the users' requirement or severity of the distortions.

8.2.2 Long-term Topics

Tackling the distortion distribution shift. This is an open and challenging question to make an image restoration model robust on the distribution shift. It occurs frequently when we train the model with a synthetic distortion dataset and inference the images corrupted with slightly different (but human may not notice) distortions. The most straightforward approach is collecting a massive number of real dataset but it is time-consuming and expensive to make low- and high-quality pairs and sometimes even impossible (e.g., medical imaging dataset). Some recent studies try to mimic the realistic distortions using GAN but are the GAN able to model the subtle difference of the synthetic and realistic distortions? We should think about whether the generative models also good at mimicking the low-level or even frequency signals. Taking the other direction, combining the internal- and external-based restoration method is also another solution. That is, the internal-based module first extract the image statistics and its related image-specific distortion information and guide the external-based model to give extra robustness

Can image restoration benefit other tasks? Despite the great performance of the deep recognition system, this is vulnerable to the undesired distortions in the input image at the inference stage. In this case, can the restoration models help the high-level vision system? One notable work is Liu et al., [166]; they jointly train the distortion restoration and image segmentation networks so that the prior information of each module improves both the restoration and segmentation performance. Another aspect would be the restoration-recognition pipeline framework that restores the input image first and recognize this as we have studied in Chapter 7. However, both approaches possibly yield significantly degraded performance when the distribution shift of training and test is occurring. Since the recognition module is dependent on the restoration counterpart, the error or unpleasant artifact could damage the overall system. Nevertheless, such a paradigm is worth researching in a long-term view for the robustness of the artificial visual system.

Besides, the image super-resolution has the potential to charity the image generation task. Although recent methods such as BigGAN [167] or StyleGAN [168] have ability to generate a high-resolution image, they still suffer several drawbacks. First, the computation burden of the high-resolution ($> 512 \times 512$) is exponentially increased which is not easily maintainable even for the modern devices. Second, the detailed textures of high-resolution images look odd even when they perfectly draw global concepts such as object shape or coarse texture. We can arise the natural question on these observations: Can the super-resolution network improve the finer texture of GAN-generated image? If yes, generating an overall image with a standard generative model first and refine the fine texture with a super-resolution network (or sub-module) might be the available option. For the computation perspective, not-so-BigGAN [169] is one example study that they adopt super-resolution module to the BigGAN to reduce the training time. However, again, understanding of the distribution shift issue should be preceded in this scenario.

Bibliography

[1] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[2] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, “Image super-resolution using very deep residual channel attention networks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[3] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond empirical risk minimization,” in *International Conference on Learning Representations (ICLR)*, 2018.

[4] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, “Cutmix: Regularization strategy to train strong classifiers with localizable features,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[5] M. Caron, P. Bojanowski, J. Mairal, and A. Joulin, “Unsupervised pre-training of image features on non-curated data,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2959–2968, 2019.

[6] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[7] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in *International Conference on Machine Learning (ICML)*, 2020.

[8] K. Yu, C. Dong, L. Lin, and C. Change Loy, “Crafting a toolchain for image restoration by deep reinforcement learning,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[9] A. Shocher, N. Cohen, and M. Irani, ““zero-shot” super-resolution using deep internal learning,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, “Cross-stitch networks for multitask learning,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] H. Huang, H. He, X. Fan, and J. Zhang, “Super-resolution of human face image using canonical correlation analysis,” *Pattern Recognition*, vol. 43, no. 7, pp. 2532–2543, 2010.

[12] K. Nguyen, C. Fookes, S. Sridharan, M. Tistarelli, and M. Nixon, “Super-resolution for biometrics: A comprehensive survey,” *Pattern Recognition*, vol. 78, pp. 23–42, 2018.

[13] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in *Proceedings of the European conference on computer vision (ECCV)*, 2014.

[14] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, “Enhanced deep residual networks for single image super-resolution,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[15] T. Tong, G. Li, X. Liu, and Q. Gao, “Image super-resolution using dense skip connections,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[16] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy, “Esrgan: Enhanced super-resolution generative adversarial networks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[17] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[18] J. Kim, J. Kwon Lee, and K. Mu Lee, "Deeply-recursive convolutional network for image super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[19] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memnet: A persistent memory network for image restoration," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[20] W. Lai, J. Huang, N. Ahuja, and M. Yang, "Fast and accurate image super-resolution with deep laplacian pyramid networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[21] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *Proceedings of the European conference on computer vision (ECCV)*, 2016.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[23] Y. Tai, J. Yang, and X. Liu, "Image super-resolution via deep recursive residual network," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[24] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[25] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair,

A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[26] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proceedings of the European conference on computer vision (ECCV)*, 2016.

[27] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[28] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[29] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[30] L. Wang, Z. Huang, Y. Gong, and C. Pan, "Ensemble based deep networks for image super-resolution," *Pattern Recognition*, vol. 68, pp. 191–198, 2017.

[31] Y. Wang, F. Perazzi, B. McWilliams, A. Sorkine-Hornung, O. Sorkine-Hornung, and C. Schroers, "A fully progressive approach to single-image super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

[32] M. S. Sajjadi, B. Scholkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[33] M. Waleed Gondal, B. Scholkopf, and M. Hirsch, "The unreasonable effectiveness of texture transfer for single image super-resolution," in *Proceedings of the European conference on computer vision (ECCV) Workshops*, 2018.

[34] L. Gatys, A. S. Ecker, and M. Bethge, “Texture synthesis using convolutional neural networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[35] A. Jolicoeur-Martineau, “The relativistic discriminator: a key element missing from standard gan,” *arXiv preprint arXiv:1807.00734*, 2018.

[36] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[37] H. Talebi and P. Milanfar, “Nima: Neural image assessment,” *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3998–4011, 2018.

[38] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[40] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.

[41] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.

[42] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “Shufflenet v2: Practical guidelines for efficient cnn architecture design,” in *Proceedings of the European conference on computer vision (ECCV)*, 2018.

[43] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[44] E. Agustsson and R. Timofte, “Ntire 2017 challenge on single image super-resolution: Dataset and study,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[45] M. Bevilacqua, A. Roumy, C. Guillemot, and M. Alberi-Morel, “Low-complexity single-image super-resolution based on nonnegative neighbor embedding,” in *Proceedings of the British Machine Vision Conference (BMVC)*, 2012.

[46] J. Yang, J. Wright, T. S. Huang, and Y. Ma, “Image super-resolution via sparse representation,” *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[47] D. Martin, C. Fowlkes, D. Tal, J. Malik, *et al.*, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2001.

[48] J.-B. Huang, A. Singh, and N. Ahuja, “Single image super-resolution from transformed self-exemplars,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[49] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, *et al.*, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[50] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, “The unreasonable effectiveness of deep features as a perceptual metric,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[51] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015.

[52] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedfor-

ward neural networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2010.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[54] J. Kim, J. Kwon Lee, and K. Mu Lee, "Deeply-recursive convolutional network for image super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[55] J.-S. Choi and M. Kim, "A deep convolutional neural network with selection units for super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[56] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[57] H. Ren, M. El-Khamy, and J. Lee, "Image super resolution based on fusing multiple convolution neural networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[58] Y. Fan, H. Shi, J. Yu, D. Liu, W. Han, H. Yu, Z. Wang, X. Wang, and T. S. Huang, "Balanced two-stage residual networks for image super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[59] Y. Blau and T. Michaeli, "The perception-distortion tradeoff," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[60] Y. Matsui, K. Ito, Y. Aramaki, A. Fujimoto, T. Ogawa, T. Yamasaki, and K. Aizawa, "Sketch-based manga retrieval using manga109 dataset," *Multimedia Tools and Applications*, vol. 76, no. 20, pp. 21811–21838, 2017.

[61] P. Navarrete Michelini, H. Liu, and D. Zhu, "Multi–scale recursive and perception–distortion controllable image super–resolution," in *Proceedings of the European conference on computer vision (ECCV) Workshops*, 2018.

[62] S. Vasu, N. Thekke Madam, and A. Rajagopalan, "Analyzing perception-distortion tradeoff using enhanced perceptual super-resolution network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[63] Q. Yang, Y. Zhang, and T. Zhao, "Example-based image super-resolution via blur kernel estimation and variational reconstruction," *Pattern Recognition Letters*, vol. 117, pp. 83–89, 2019.

[64] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, "Non-local recurrent network for image restoration," in *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[65] Y. Matsui, K. Ito, Y. Aramaki, A. Fujimoto, T. Ogawa, T. Yamasaki, and K. Aizawa, "Sketch-based manga retrieval using manga109 dataset," *Multimedia Tools and Applications*, vol. 76, no. 20, pp. 21811–21838, 2017.

[66] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[67] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, *et al.*, "Going deeper with convolutions," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[68] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *International Conference on Learning Representations (ICLR)*, 2018.

[69] E. L. Denton, S. Chintala, R. Fergus, *et al.*, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[70] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[71] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," in *Readings in Computer Vision*, pp. 671–679, Elsevier, 1987.

[72] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Fast and accurate image super-resolution with deep Laplacian pyramid networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[73] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization.," in *International Conference on Learning Representations (ICLR)*, 2015.

[74] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[75] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Fast and accurate image super-resolution with deep laplacian pyramid networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2599–2613, 2018.

[76] B. Lim, S. Son, H. Kim, S. Nah, and K. Mu Lee, "Enhanced deep residual networks for single image super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[77] J. Cai, H. Zeng, H. Yong, Z. Cao, and L. Zhang, "Toward real-world single image super-resolution: A new benchmark and a new model," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[78] A. Abdelhamed, S. Lin, and M. S. Brown, "A high-quality denoising dataset for smartphone cameras," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[79] X. Zhang, R. Ng, and Q. Chen, “Single image reflection separation with perceptual losses,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[80] R. Feng, J. Gu, Y. Qiao, and C. Dong, “Suppressing model overfitting for image super-resolution networks,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[81] R. Timofte, R. Rothe, and L. Van Gool, “Seven ways to improve example-based single image super resolution,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[82] S. Gu, W. Zuo, Q. Xie, D. Meng, X. Feng, and L. Zhang, “Convolutional sparse coding for image super-resolution,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2015.

[83] R. Timofte, V. De Smet, and L. Van Gool, “A+: Adjusted anchored neighborhood regression for fast super-resolution,” in *Asian Conference on Computer Vision (ACCV)*, 2014.

[84] G. Ghiasi, T.-Y. Lin, and Q. V. Le, “Dropblock: A regularization method for convolutional networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[85] X. Gastaldi, “Shake-shake regularization,” *arXiv preprint arXiv:1705.07485*, 2017.

[86] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio, “Manifold mixup: Better representations by interpolating hidden states,” in *International Conference on Machine Learning (ICML)*, 2019.

[87] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise, “Shakedrop regularization for deep residual learning,” *IEEE Access*, vol. 7, pp. 186126–186136, 2019.

[88] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, “Random erasing data augmentation,” *arXiv preprint arXiv:1708.04896*, 2017.

[89] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[90] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim, "Fast autoaugment," in *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[91] J. Choe and H. Shim, "Attention-based dropout layer for weakly supervised object localization," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[92] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[93] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[94] M. El Helou, R. Zhou, and S. Süsstrunk, "Stochastic frequency masking to improve super-resolution and denoising networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[95] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[96] C. Chen, Z. Xiong, X. Tian, Z.-J. Zha, and F. Wu, "Camera lens super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[97] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images," *IEEE Transactions on Image Processing*, vol. 16, no. 5, pp. 1395–1411, 2007.

[98] H. R. Sheikh, Z. Wang, L. Cormack, and A. C. Bovik, "Live image quality assessment database release 2," 2005.

[99] A. Abdelhamed, M. Afifi, R. Timofte, and M. S. Brown, "Ntire 2020 challenge on real image denoising: Dataset, methods and results," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020.

[100] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep cnn denoiser prior for image restoration," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[101] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[102] N. Ahn, B. Kang, and K.-A. Sohn, "Fast, accurate, and lightweight super-resolution with cascading residual network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[103] T. Vu, C. Van Nguyen, T. X. Pham, T. M. Luu, and C. D. Yoo, "Fast and efficient image quality enhancement via desubpixel convolutional neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[104] Y. Zhang, K. Li, K. Li, B. Zhong, and Y. Fu, "Residual non-local attention networks for image restoration," in *International Conference on Learning Representations (ICLR)*, 2019.

[105] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[106] M. Fritsche, S. Gu, and R. Timofte, "Frequency separation for real-world super-resolution," *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2019.

[107] A. Lugmayr, M. Danelljan, and R. Timofte, "Unsupervised learning for real-world super-resolution," *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2019.

[108] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[109] A. Lugmayr, M. Danelljan, R. Timofte, *et al.*, "Ntire 2020 challenge on real-world image super-resolution: Methods and results," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020.

[110] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[111] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[112] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[113] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[114] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Asian Conference on Computer Vision (ACCV)*, 2014.

[115] R. Feng, J. Gu, Y. Qiao, and C. Dong, "Suppressing model overfitting for image super-resolution networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[116] J. Gu, H. Lu, W. Zuo, and C. Dong, "Blind super-resolution with iterative kernel correction," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[117] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, and L. Lin, "Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

[118] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[119] R. Heckel and P. Hand, "Deep decoder: Concise image representations from untrained non-convolutional networks," *International Conference on Learning Representations (ICLR)*, 2019.

[120] J. W. Soh, S. Cho, and N. I. Cho, "Meta-transfer learning for zero-shot super-resolution," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[121] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[122] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[123] A. Krull, T.-O. Buchholz, and F. Jug, "Noise2void-learning denoising from single noisy images," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[124] L. Kang, P. Ye, Y. Li, and D. Doermann, "Convolutional neural networks for no-reference image quality assessment," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[125] S. Bosse, D. Maniry, T. Wiegand, and W. Samek, "A deep neural network for image quality assessment," in *IEEE International Conference on Image Processing (ICIP)*, 2016.

[126] S. Bianco, L. Celona, P. Napoletano, and R. Schettini, "On the use of deep learning for blind image quality assessment," *Signal, Image and Video Processing*, vol. 12, no. 2, pp. 355–362, 2018.

[127] L. Kang, P. Ye, Y. Li, and D. Doermann, "Simultaneous estimation of image quality and distortion via multi-task convolutional neural networks," in *IEEE International Conference on Image Processing (ICIP)*, 2015.

[128] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "Fsim: a feature similarity index for image quality assessment," *IEEE Transactions on Image Processing*, vol. 20, no. 8, pp. 2378–2386, 2011.

[129] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[130] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[131] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed, "Ssd: Single shot multibox detector," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2015.

[132] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "Nus-wide: a real-world web image database from national university of singapore," in *Proceedings of the ACM International Conference on Image and Video Retrieval*, p. 48, 2009.

[133] H. R. Sheikh, Z. Wang, L. Cormack, and A. C. Bovik, "Live image quality assessment database release 2," 2005.

[134] N. Ponomarenko, L. Jin, O. Ieremeiev, V. Lukin, K. Egiazarian, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, *et al.*, "Image database tid2013: Peculiarities,

results and perspectives," *Signal Processing: Image Communication*, vol. 30, pp. 57–77, 2015.

[135] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in python," *PeerJ*, vol. 2, p. e453, 2014.

[136] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[137] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[138] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[139] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[140] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[141] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[142] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

[143] X. Liu, M. Suganuma, X. Luo, and T. Okatani, "Restoring images with unknown degradation factors by recurrent use of a multi-branch network," *arXiv preprint arXiv:1907.04508*, 2019.

[144] M. Suganuma, X. Liu, and T. Okatani, "Attention-based adaptive selection of operations for image restoration in the presence of unknown combined distortions," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[145] K. Yu, X. Wang, C. Dong, X. Tang, and C. C. Loy, "Path-restore: Learning network path selection for image restoration," *arXiv preprint arXiv:1904.10343*, 2019.

[146] P. Savarese and M. Maire, "Learning implicitly recurrent cnns through parameter sharing," *International Conference on Learning Representations (ICLR)*, 2019.

[147] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[148] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[149] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation*, vol. 3, no. 1, pp. 79–87, 1991.

[150] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[151] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi, "Modeling task relationships in multi-task learning with multi-gate mixture-of-experts," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

[152] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[153] N. Ponomarenko, L. Jin, O. Ieremeiev, V. Lukin, K. Egiazarian, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, *et al.*, "Image database tid2013: Peculiarities, results and perspectives," *Signal processing: Image communication*, vol. 30, pp. 57–77, 2015.

[154] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[155] Y. Wang, X. Tao, X. Qi, X. Shen, and J. Jia, "Image inpainting via generative multi-column convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2018.

[156] H. Ren, M. El-Khamy, and J. Lee, "Image super resolution based on fusing multiple convolution neural networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017.

[157] X. Du, M. El-Khamy, J. Lee, and L. Davis, "Fused dnn: A deep neural network fusion approach to fast and robust pedestrian detection," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[158] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[159] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[160] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[161] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[162] S. Anwar and N. Barnes, "Real image denoising with feature attention," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[163] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Proceedings of the European conference on computer vision (ECCV)*, 2014.

[164] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[165] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017.

[166] D. Liu, B. Wen, X. Liu, Z. Wang, and T. S. Huang, "When image denoising meets high-level vision tasks: A deep learning approach," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[167] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *International Conference on Learning Representations (ICLR)*, 2019.

[168] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[169] S. Han, A. Srivastava, C. Hurwitz, P. Sattigeri, and D. D. Cox, "not-so-biggan: Generating high-fidelity images on a small compute budget," *arXiv preprint arXiv:2009.04433*, 2020.