Deep Convolutional Neural Network for Image Restoration and Synthesis

A Dissertation Presented

by

Yulun Zhang

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in

Computer Engineering

Northeastern University

Boston, Massachusetts

Aug 2021

To my family.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my deepest and sincerest gratitude to my advisor, Prof. Yun Fu, for his continuous guidance, advice, effort, patience, and encouragement during the past four years. The strong supports from Prof. Fu lie in academic and daily aspects. When I was in need, he is always willing to provide the help. I am truly fortunate to have him as my advisor. This dissertation and my current achievements would not have been possible without his tremendous support.

I would also like to thank my committee members, Prof. Octavia Camps and Prof. Hanspeter Pfister for their valuable time, insightful comments, and suggestions ever since my PhD research proposal.

I would like to thank my collaborators in my master study, Prof. Yongbing Zhang, Prof. Haoqian Wang, Prof. Qionghai Dai, Prof. Jian Zhang, Prof. Dong Xu, Prof. Yu Qiao, Prof. Chao Dong, Prof. Chen Change Loy, Dr. Xintao Wang, and Dr. Ke Yu. I learned a lot from them and established firm research foundations for my PhD study.

I would like to thank all my mentors during my internships at Adobe Research. Many thanks to Dr. Zhaowen Wang, Dr. Zhe Lin, Dr. Zhifei Zhang, Dr. Stephen DiVerdi, Dr. Jose Echevarria, Dr. Jimei Yang, Dr. Chen Fang, and Dr. Yilin Wang for their inspiring discussions and suggestions. Besides, I would like to thank advisor and collaborators during my visiting at Harvard University. Many thanks to Prof. Hanspeter Pfister, Prof. Donglai Wei, Prof. Yalong Yang, Dr. Won-Dong Jang, Salma Abdel Magid, and Zudi Lin. Moreover, I would like to thank my collaborators from other institutions. Great thanks to Yapeng Tian, Dr. Xiaoyu Xiang, Prof. Chenliang Xu, and Prof. Jan Allebach.

In addition, I would like to thank all the members from SMILE Lab, especially my coauthors and collaborators, Prof. Bineng Zhong, Prof. Yu Kong, Prof. Sheng Li, Prof. Zhengming Ding, Prof. Hongfu Liu, Prof. Zhiqiang Tao, Prof. Jun Li, Prof. Gan Sun, Dr. Handong Zhao, Dr. Kunpeng Li, Dr. Kai Li, Dr. Licheng Wang, Dr. Joseph Robinson, Can Qin, Huan Wang, Yu Yin. For other lab members, Bin Sun, Songyao Jiang, Chang Liu, Yue Bai, Sara Al Bunian, Zaid Khan, Yi Xu, Yizhou Wang, Xu Ma, I also thank you very much. I have spent my wonderful four years with these excellent colleagues and left the impressive memory.

I would like to express my gratitude to my parents and wife for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this dissertation.

Abstract of the Dissertation

Deep Convolutional Neural Network for Image Restoration and Synthesis

by

Yulun Zhang

Doctor of Philosophy in Computer Engineering

Northeastern University, Aug 2021

Dr. Yun Fu, Advisor

Deep convolutional neural network (CNN) has been showing excellent performance in computer vision applications, including image restoration and synthesis. This is mainly because the deep features from deep CNNs have more powerful representation ability than classic features. However, to achieve better performance, we still have to further enhance deep features with advanced CNN models by considering the specific domain knowledge.

In this dissertation, we mainly focus on designing efficient deep CNN models for image restoration and synthesis. We first present a residual dense network (RDN) for image restoration by learning hierarchical features. From low-quality input, our RDN obtains residual information, which is essentially important to recover high-quality result. We then try to enhance more informative deep CNN features with various attention mechanisms. Specifically, we propose a residual in residual (RIR) structure to get very deep features, which are adaptively rescaled with our channel attention. We further design residual local and non-local attention blocks to extract features, which capture long-range dependencies between pixels. On the other hand, we investigate the deep CNN features in image synthesis, like style and texture transfer. We propose a flexible and general multimodal style transfer. By visualizing the deep style features, we introduce multimodal style representation, which is achieved with clustering. We then propose multimodal style matching, where we match the clustered sub-style components with local content features under a graph cut formulation. Besides, we investigate image synthesis about texture hallucination with large scaling factor. We propose an efficient high-resolution hallucination network for very large scaling factors.

In summary, this dissertation studies efficient deep CNN models for high-quality image restoration and synthesis. Our proposed deep CNN models have shown promising performance in a wide range of computer vision applications, such as image super-resolution, denoising, deblurring, demosaicing, compression artifacts reduction, neural style transfer, and texture transfer.

Chapter 1

Introduction

1.1 Background

Deep convolutional neural network (CNN) has been achieving promising performance in a wide range of computer vision tasks. Compared with classic features, deep CNN features have more powerful representation ability. However, to achieve better performance, we still have to further enhance deep CNN features with advanced CNN models by considering the specific domain knowledge.

In this dissertation, we mainly focus on designing efficient deep CNN models for image restoration and synthesis. We start with the analyses of deep CNN features in various image restoration and synthesis tasks. Let's take image super-resolution (SR) as an example, the low-resolution (LR) image contains much low-frequency information and lacks high-frequency one, like structural details. To this end, we need to recover more high-frequency information, which can be viewed as residual component in deep feature. Similarly, the noise can also be viewed as residual component, which needs to be learned in the denoising networks. So, learning more sophisticated and efficient deep CNN features are important for image restoration. On the other hand, deep feature representation and matching are also very important in image synthesis. Let's take image style transfer as an example. Most existing methods neglect to investigate the style feature representation or treat the semantic patterns of style image uniformly, resulting unpleasing stylized results. We find the multimodal style representation, based on which we further explore more reasonable matching between style and content deep features.

Consequently, we present a set of novel and efficient approaches. We first present a residual dense network (RDN) for image restoration by learning hierarchical features. From low-quality input,

our RDN obtains residual information, which is essentially important to recover high-quality result. We then try to enhance more informative deep CNN features with various attention mechanisms. Specifically, we propose a residual in residual (RIR) structure to get very deep features, which are adaptively rescaled with our channel attention. We further design residual local and non-local attention blocks to extract features, which capture long-range dependencies between pixels. On the other hand, we investigate the deep CNN features in image synthesis, like style and texture transfer. We propose a flexible and general multimodal style transfer. By visualizing the deep style features, we introduce multimodal style representation, which is achieved with clustering. We then propose multimodal style matching, where we match the clustered sub-style components with local content features under a graph cut formulation. Besides, we investigate image synthesis about texture hallucination with large scaling factor. We propose an efficient high-resolution hallucination network for very large scaling factors.

In this dissertation, we investigate efficient deep CNN models for high-quality image restoration and synthesis. Our proposed deep CNN models have shown promising performance in a wide range of computer vision applications, such as image super-resolution, denoising, deblurring, demosaicing, compression artifacts reduction, neural style transfer, and texture transfer.

In addition, we apply deep CNNs for image enhancement in other domains, like biomedical images, by further considering domain specific knowledge. We propose squeeze and excitation reasoning attention networks (SERAN) for accurate magnetic resonance (MR) image SR [13]. We believe that our investigations about deep CNN based image restoration and synthesis would contribute to other related works [14, 15]. In the future, we would like to explore our efficient deep CNN models for other applications, such as biomedical image analysis, bioinformatics, and compressive imaging. Moreover, we'll investigate more efficient models, which have smaller model size and less computation operations by using model compression techniques, like network pruning.

1.2 Related work

1.2.1 Deep Convolutional Neural Network

LeCun et al. integrated constraints from task domain to enhance network generalization ability for handwritten zip code recognition [16], which can be viewed as the pioneering usage of CNNs. Later, various network structures were proposed with better performance, such as AlexNet [17], VGG [10], and GoogleNet [18]. Recently, He et al. [19] investigated the powerful

effectiveness of network depth and proposed deep residual learning for very deep trainable networks. Such a very deep residual network achieves significant improvements on several computer vision tasks, like image classification and object detection. Huang et al. proposed DenseNet, which allows direct connections between any two layers within the same dense block [20]. With the local dense connections, each layer reads information from all the preceding layers within the same dense block. The dense connection was introduced among memory blocks [21] and dense blocks [2].

1.2.2 Attention Mechanism

Definition of attention mechanism. Attention is a general concept covering all factors that influence selection mechanisms. The basis of many attention models dates back to the Feature Integration Theory proposed in [22] where they stated which visual features are important and how they are combined to direct human attention over pop-out and conjunction search tasks. A major distinction among recent attention models is whether they be scene-driven bottom-up or expectation-driven top-down.

Bottom-up and top-down attention. Bottom-up attention [23] refers to attentional guidance purely by externally driven factors to stimuli that are salient because of their inherent properties relative to the background, such as salient stuff or objects [23, 24]. Regions of interest that attract our attention in a bottom-up manner must be sufficiently distinctive with respect to surrounding features. This attentional mechanism is also called exogenous, automatic, reflexive, or peripherally cued.

This observation has motivated many recent works related to semantic understanding of the scene, such as image-text matching [25, 26], image captioning [27] and Visual Question Answering [28]. Instead of operating on traditional CNN features corresponding to a uniform grid of equally-sized image regions, such bottom-up attention models take such bottom-up attention regions as input, which enables reasoning at the level of objects and other salient regions in a scene. [25] presents an attention algorithm to focus on key words and image regions when aggregating pairwise similarities among visual patches and words. [27] introduces a new attention-based encoder-decoder framework to explore the connections between objects for image captioning. [28] proposes to conduct multi-modal relational reasoning model for Visual Question Answering and it represent interactions between question and image regions by a rich vectorial representation.

Such reasoning processes operated on top of bottom-up attention regions actually further lead to top-down attention [23], which refers to internal guidance of attention based on prior knowledge, willful plans, and current goals. It has also been explored in a wide range of recognition

tasks such as machine translations [29], word embedding learning [30] and image/video classification [31, 32].

Attention mechanism in image SR. In the area of image SR, the goal is to reconstruct an accurate high-resolution image given its low-resolution counterpart where operations are mainly conducted at pixel-level. Therefore, existing attention modules are mainly top-down ones rather than starting from bottom-up attention regions in the scene. The top-down attention module is usually a part of a neural architecture that enables to dynamically highlight relevant features of the input data, which, in recent image SR models, is typically a pool of higher level representations in the different convolutional layers. The core idea behind attention is to compute a weight distribution on the deep CNN features by explicitly modelling inter-dependencies between them, assigning higher values to particular parts according to specific goals.

1.2.3 Image Restoration

Generally, there are two categories of methods for image restoration: model-based [33, 34, 35, 36] and learning-based [37, 21] methods. The model-based methods often formulate the image restoration problems as optimization ones. Numerous regularizers have been investigated to search for better solutions, such as sparsity-based regularizers with learned dictionaries [33] and nonlocal self-similarity inspired ones [35]. Also, instead of using explicitly designed regularizers, Dong et al. [36] proposed a denoising prior driven network for image restoration. Then, we give more details about learning-based methods.

Dong et al. [38] proposed ARCNN for image compression artifact reduction (CAR) with several stacked convolutional layers. Mao et al. [39] proposed residual encoder-decoder networks (RED) with symmetric skip connections, which made the network go deeper (up to 30 layers). Zhang et al. [40] proposed DnCNN to learn mappings from noisy images to noise and further improved performance by utilizing batch normalization [41]. Zhang et al. [37] proposed to learn deep CNN denoiser prior for image restoration (IRCNN) by integrating CNN denoisers into model-based optimization method. However, such methods have limited network depth (e.g., , 30 for RED, 20 for DnCNN, and 7 for IRCNN), limiting the network ability. Simply stacking more layers cannot reach better results due to gradient vanishing problem. On the other hand, by using short term and long-term memory, Tai et al. [21] proposed MemNet for image restoration, where the network depth reached 212 but obtained limited improvement over results with 80 layers. For 31×31 input patches from 91 images, training an 80-layer MemNet takes 5 days using 1 Tesla P40 GPU [21].

The aforementioned DL-based image restoration methods have achieved significant improvement over conventional methods, but most of them lose some useful hierarchical features from the original LQ image. Hierarchical features produced by a very deep network are useful for image restoration tasks (e.g., , image SR). To fix this case, we propose residual dense network (RDN) to extract and adaptively fuse features from all the layers in the LQ space efficiently.

1.2.4 Image Synthesis

Style Transfer. Originating from non-realistic rendering [42], image style transfer is closely related to texture synthesis [43, 44, 45]. Gatys *et al.* [46] were the first to formulate style transfer as the matching of multi-level deep features extracted from a pre-trained deep neural network, which has been widely used in various tasks [47, 48, 49]. Lots of improvements have been proposed based on the works of Gatys *et al.* [46]. Johnson *et al.* [50] trained feed-forward style-specific network and produced one stylization with one model. Sanakoyeu *et al.* [51] further proposed a style-aware content loss for high-resolution style transfer. Jing *et al.* [52] proposed a StrokePyramid module to enable controllable stroke with adaptive receptive fields. However, these methods are either time consuming or have to re-train new models for new styles.

The first arbitrary style transfer was proposed by Chen and Schmidt [53], who matched each content patch to the most similar style patch and swapped them. Luan *et al.* [54] proposed deep photo style transfer by adding a regularization term to the optimization function. Based on markov random field (MRF), Li and Wand [6] proposed CNNMRF to enforce local patterns in deep feature space. Ruder *et al.* [55] improved video stylization with temporal coherence. Although their visual stylizations for arbitrary style are appealing, the results are not stable [55].

Recently, Huang *et al.* [3] proposed real-time style transfer by matching the mean-variance statistics between content and style features. Li *et al.* [4] further introduced whitening and coloring (WCT) by matching the covariance matrices. Li *et al.* boosted style transfer with linear style transfer (LST) [5]. Gu *et al.* [7] proposed deep feature reshuffle (DFR), which connects both local and global style losses used in parametric and non-parametric methods. Sheng *et al.* [8] proposed AvatarNet to enable multi-scale transfer for arbitrary style. Shen *et al.* [56] built meta networks by taking style images as inputs and generating corresponding image transformation networks directly. Mechrez *et al.* [57] proposed contextual loss for image transformation. However, these methods fail to treat the style patterns distinctively and neglect to adaptively match style patterns with content semantic information. For more neural style transfer works, readers can refer to the survey [58].

Texture hallucination. Different from SISR, reference-based SR (Ref-SR) methods attempt to utilize self or external information to enhance the texture. Freeman *et al*. [59] proposed the first work on Ref-SR, which replaced LR patches with fitting HR ones from a database/dictionary. [60, 61] considered the input LR image itself as the database, from which references were extracted to enhance textures. These methods benefit the most from repeated patterns with perspective transformation. Light field imaging is an area of interest for Ref-SR, where HR references can be captured along the LR light field, just with a small offset. Thus, making easier to align the reference to the LR input, facilitating the transfer of high-frequency information in [62, 63]. CrossNet [64] took advantage of deep learning to align the input and reference by estimating the flow between them and achieved SOTA performance.

A more generic scenario for Ref-SR is to relax the constraints on references, i.e., the references could present large spacial/color shift from the input. More extremely, references and inputs could contain unrelated content. Sun *et al*. [65] used global scene descriptors and internet-scale image databases to find similar scenes that provide ideal example textures. Yue *et al*. [66] proposed a similar idea, retrieving similar images from the web and performing global registration and local matching. Recent works [67, 12] leveraged deep models and significantly improved Ref-SR performance, e.g., visual quality and generalization capacity.

Our proposed method further extends the feasible scaling factor of previous Ref-SR methods from $4\times$ to $16\times$. More importantly, as oppose to the previous approach [12], which transfers the high-frequency information from reference as a style transfer task, we conduct texture transfer only in high-frequency band, which reduces the transfer effect on the low-frequency content.

1.3 Dissertation Organization

The rest of this dissertation is organized as follows.

In chapter 2, we develop a novel and efficient residual dense network (RDN) for image restoration (IR). Specifically, we propose residual dense block (RDB) to extract abundant local features via densely connected convolutional layers. To adaptively learn more effective features and stabilize the training of wider network, we propose local feature fusion in RDB. We use global feature fusion to jointly and adaptively learn global hierarchical features in a holistic way. We demonstrate the effectiveness of RDN with several representative IR applications, single image super-resolution, Gaussian image denoising, image compression artifact reduction, and image deblurring.

In chapter 3, we propose a very deep residual channel attention network (RCAN). Specifically, we propose a residual in residual (RIR) structure to form very deep network, which consists of several residual groups with long skip connections. Each residual group contains some residual blocks with short skip connections. Meanwhile, RIR allows abundant low-frequency information to be bypassed through multiple skip connections, making the main network focus on learning high-frequency information. Furthermore, we propose a channel attention mechanism to adaptively rescale channel-wise features by considering interdependencies among channels.

In chapter 4, we design local and non-local attention blocks to extract features that capture the long-range dependencies between pixels and pay more attention to the challenging parts. Specifically, we design trunk branch and (non-)local mask branch in each (non-)local attention block. The trunk branch is used to extract hierarchical features. Local and non-local mask branches aim to adaptively rescale these hierarchical features with mixed attentions. Furthermore, we propose residual local and non-local attention learning to train the very deep network. Our proposed method can be generalized for various image restoration applications, such as image denoising, demosaicing, compression artifacts reduction, and super-resolution.

In chapter **??**, we investigate image synthesis with deep CNNs. We introduce a more flexible and general universal style transfer technique: multimodal style transfer (MST). MST explicitly considers the matching of semantic patterns in content and style images. Specifically, the style image features are clustered into sub-style components, which are matched with local content features under a graph cut formulation. A reconstruction network is trained to transfer each sub-style and render the final stylized result. We also generalize MST to improve some existing methods.

In chapter 5, we further investigate image synthesis about texture hallucination with large scaling factor. We propose an efficient high-resolution hallucination network for very large scaling factors with efficient network structure and feature transferring. We design a wavelet texture loss to enhance more high-frequency components. We further relax the reconstruction constraint with a degradation loss. We also collected a high-resolution painting dataset PaintHD by considering both physical size and image resolution.

In chapter 6, we finally conclude this dissertation.

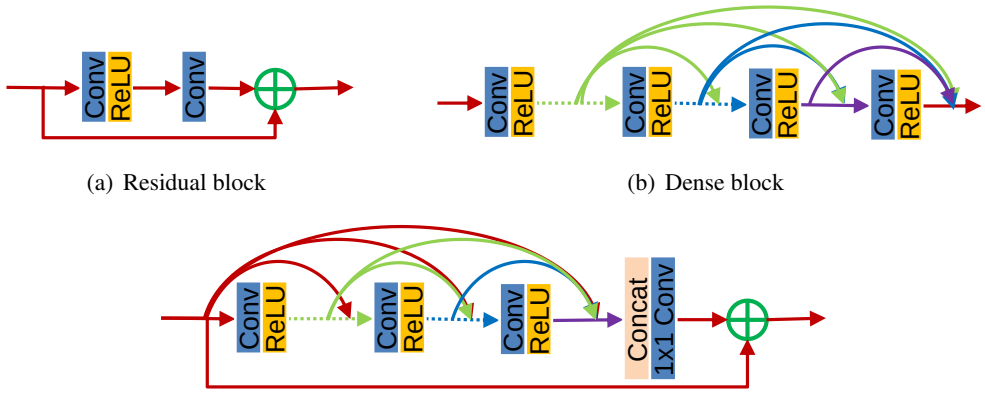
Chapter 2

Residual Dense Network for Image Restoration

2.1 Background and Motivation

Single image restoration (IR) aims to generate a visually pleasing high-quality (HQ) image from its degraded low-quality (LQ) measurement (e.g., downsaled, noisy, compressed, or/and blurred images). Image restoration plays an important and fundamental role and has been widely used in computer vision, ranging from security and surveillance imaging [68], medical imaging [69], to image generation [70]. However, IR is very challenging, because the image degradation process is irreversible, resulting in an ill-posed inverse procedure. To tackle this problem, lots of works have been proposed, such as model-based [33, 34, 35, 36] and learning-based [37, 21] methods. Recently, deep convolutional neural network (CNN) has been widely investigated and achieves promising performance in various image restoration tasks, such as image super-resolution (SR) [71, 72, 73, 74, 75, 76, 77, 61, 78, 2, 79], image denoising (DN) [80, 39, 40, 21, 37, 81], image compression reduction (CAR) [38, 80, 40], and image deblurring [34, 35, 37, 36].

The first challenge is how to introduce deep CNN for image restoration. Dong *et al*. [76], for the first time, proposed SRCNN for image SR with three convolutional (Conv) layers and achieved significant improvement over previous methods. After firstly introducing CNN for image SR, Dong *et al*. [38] further applied CNN for other image restoration tasks, like image CAR. But, it's hard to train by stacking more Conv layers in SRCNN. To ease the difficulty of training deep network, Kim *et al*.proposed VDSR [78] and DRCN [82] by using gradient clipping, residual learning, or



(a) Residual block (b) Dense block

(c) Residual dense block

Figure 2.1: Comparison of prior network structures (a,b) and our residual dense block (c). (a) Residual block in MDSR [1]. (b) Dense block in SRDenseNet [2]. (c) Our proposed residual dense block (RDB), which not only enables previous RDB to connect with each layer of current RDB, but also makes better use of local features.

recursive-supervision. With newly investigated effective building modules and training strategies, better image SR performance could be further achieved. Incorporating residual learning, Zhang *et al*. [40] proposed efficient denoising network. Memory block was proposed by Tai *et al*.to build MemNet for image restoration [21]. Later, Lim *et al*.proposed simplified residual block (Figure 2.1(a) by removing batch normalization layers and came up with a very deep network MDSR [1]. Lim *et al*.further investigated a very wide network EDSR [1] by using residual scaling [83] to stabilizing the training. Convolutional layers in different depth have different size of receptive fields, resulting in hierarchical features. However, these features are not fully used by previous methods. Although MemNet [21] involved a gate unit in memory block to control short-term memory, it failed to build direct connections between local Conv layers and their subsequent ones. As a result, memory block didn't make full use of the features from all the Conv layers within the block either.

What's more, same or similar objects would appear differently in the images due to different scales, angles of view, and aspect ratios. Hierarchical features can capture such characteristics, which would contribute to better reconstruction. However, most learning-based methods (e.g., VDSR [78], LapSRN [84], IRCNN [37], and EDSR [1]) neglect to pursue more clues by making better use of hierarchical features. Although MemNet [21] takes features from several memory block, it failed to extract multi-level features from the original LQ image (e.g., the LR image). Let's take image SR as an example, MemNet would pre-process the original input by interpolating it to desired size. Such a step would not only introduce extra artifacts (e.g., blurring artifacts), but also increase the computation

complexity quadratically. Later, dense block (Figure 2.1(b)) was introduced in SRDenseNet for image SR [2]. The growth rate is a key part in dense block. Based on the investigation in DenseNet [20] and our experiments, higher growth rate contributes to better performance, resulting in wider networks. However, training a wider network with dense blocks would become harder. As investigated in EDSR [1], increasing feature map number above a certain level would make the training more difficult and numerically unstable.

To tackle the issues and limitations above, we propose a simple yet efficient residual dense network (RDN) (Figure 2.2) by extracting the hierarchical features from the original LQ image. Our RDN is built on our proposed residual dense block (Figure 2.1(c)). A straightforward way to obtain hierarchical feature is to directly extract the output of each Conv layer in the LQ space. But, it's impractical, especially for a very deep network. Instead, we propose residual dense block (RDB), which consists of dense connected layers and local feature fusion (LFF) with local residual learning (LRL). Furthermore, the Conv layers of current RDB have direct access to the previous RDB, resulting in a contiguous state pass. We name it as contiguous memory mechanism, which further passes on information that needs to be preserved [20]. So, in each RDB, LFF concatenates the states of preceding and current RDBs and adaptively extracts local dense features. Moreover, LFF helps to stabilize the training of wider networks with high growth rate. After obtaining multi-level local dense features, global feature fusion (GFF) is conducted to adaptively preserve the hierarchical features in a global way. As shown in Figures 2.2 and 2.3, the original LR input directly connects each layer, leading to an implicit deep supervision [85].

In summary, our main contributions are three-fold:

- We propose a unified framework residual dense network (RDN) for high-quality image restoration. The network makes full use of all the hierarchical features from the original LQ image.

- We propose residual dense block (RDB), which can not only read state from the preceding RDB via a contiguous memory (CM) mechanism, but also better utilize all the layers within it via local dense connections. The accumulated features are then adaptively preserved by local feature fusion (LFF).

- We propose global feature fusion to adaptively fuse hierarchical features from all RDBs in the LR space. With global residual learning, we combine the shallow features and deep features together, resulting in global dense features from the original LQ image.

A preliminary version of this work has been presented as a conference version [86]. In the current work, we incorporate additional contents in significant ways:

- We investigate a flexible structure of RDN and apply it for different IR tasks. Such IR applications allow us to further investigate the potential breadth of RDN.
- We investigate more details and add considerable analyses to the initial version, such as block connection, network parameter number, and running time.
- We extend RDN for Gaussian image denoising, compression artifact reduction, and image deblurring. Extensive experiments on benchmark and real-world data demonstrate that our RDN still outperforms existing approaches in these IR tasks.

2.2 Related Work

Existing IR methods mainly include model-based [33, 34, 35, 36] and learning-based [37, 21] methods. Among them, deep learning (DL)-based methods have achieved dramatic advantages against conventional methods in computer vision [87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 11, 97, 98, 99]. Here, we focus on several representative image restoration tasks, such as image super-resolution (SR), denoising (DN), compression artifact reduction (CAR), and image deblurring.

Image Super-Resolution. Deep learning was first investigated for image SR. The most challenging part is how to formulize the mapping between LR and HR images into deep neural network. After constructing the basic networks for image SR, there're still challenging problems to solve, such as how to train bery deep and wide network. We will show how these challenges are solved or alleviated.

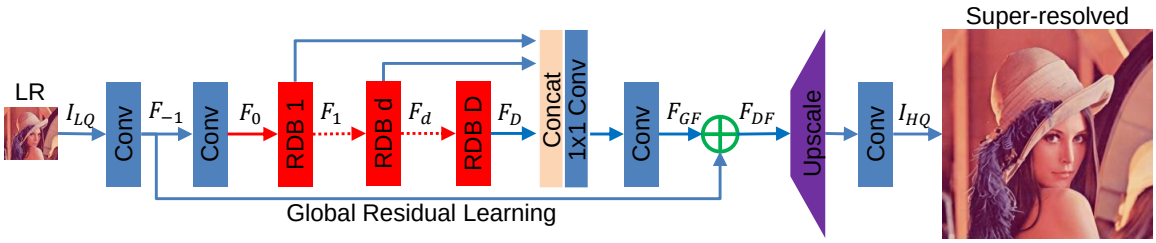
Dong *et al*. [76] proposed SRCNN, applying CNN into image SR for the first time. Based on SRCNN, lots of improvements have been down to pursue better performance. By using residual learning to ease the training of deeper networks, VDSR [78] and IRCNN [37] could train deeper networks with more stacked Conv layers. DRCN [82] also achieved such a deep network by sharing network parameters and recursive learning, which was further employed in DRRN [100]. However, these methods would pre-process the original input by interpolating it to desired size. Such a step would not only introduce extra artifacts (e.g., blurring artifacts), but also increase the computation complexity quadratically. As a result, extracting features from the interpolated LR images would not be able to build direct mapping from the original LR to HR images.

To address the problem above, Dong *et al*. [101] built direct mapping from the original LR image to the HR target by introducing a transposed Conv layer (also known as deconvolution layer). Such a transposed Conv layer was further replaced by an efficient sub-pixel convolution layer in ESPCN [102]. Due to its efficiency, sub-pixel Conv layer was adopted in SRResNet [103], which took advantage of residual learning [19]. By extracting features from the original LR input and upscaling the final LR features with transposed or sub-pixel convolution layer, they could either be capable of real-time SR (e.g., FSRCNN and ESPCN), or be built to be very deep/wide (e.g., SRResNet and EDSR). However, all of these methods neglect to adequately utilize information from each Conv layer, but only upscale the high-level CNN features for the final reconstruction.

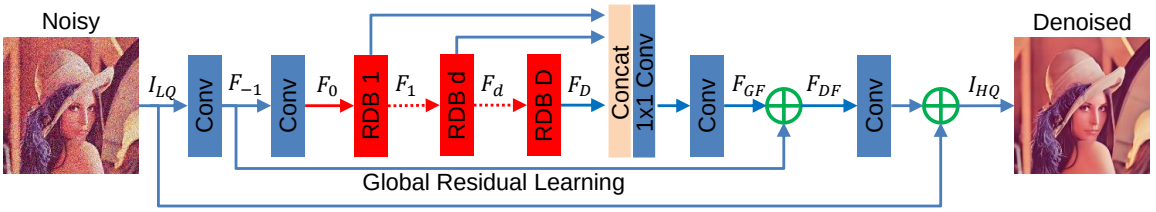
Deep Convolutional Neural Network (CNN). LeCun *et al*.integrated constraints from task domain to enhance network generalization ability for handwritten zip code recognition [16], which can be viewed as the pioneering usage of CNNs. Later, various network structures were proposed with better performance, such as AlexNet [17], VGG [10], and GoogleNet [18]. Recently, He *et al*. [19] investigated the powerful effectiveness of network depth and proposed deep residual learning for very deep trainable networks. Such a very deep residual network achieves significant improvements on several computer vision tasks, like image classification and object detection. Huang *et al*.proposed DenseNet, which allows direct connections between any two layers within the same dense block [20]. With the local dense connections, each layer reads information from all the preceding layers within the same dense block. The dense connection was introduced among memory blocks [21] and dense blocks [2]. More differences between DenseNet/SRDenseNet/MemNet and our RDN would be discussed in Section 2.4.

Deep Learning for Image Restoration. Generally, there are two categories of methods for image restoration: model-based [33, 34, 35, 36] and learning-based [37, 21] methods. The model-based methods often formulate the image restoration problems as optimization ones. Numerous regularizers have been investigated to search for better solutions, such as sparsity-based regularizers with learned dictionaries [33] and nonlocal self-similarity inspired ones [35]. Also, instead of using explicitly designed regularizers, Dong *et al*. [36] proposed a denoising prior driven network for image restoration. Then, we give more details about learning-based methods.

Dong *et al*. [38] proposed ARCNN for image compression artifact reduction (CAR) with several stacked convolutional layers. Mao *et al*. [39] proposed residual encoder-decoder networks (RED) with symmetric skip connections, which made the network go deeper (up to 30 layers). Zhang *et al*. [40] proposed DnCNN to learn mappings from noisy images to noise and further improved performance by utilizing batch normalization [41]. Zhang *et al*. [37] proposed to learn deep CNN



(a) RDN for image super-resolution (SR).



(b) RDN for image denoising (DN), compression artifact reduction (CAR), and deblurring. It should be noted that such a structure can also be used for image SR. But, it would consume more resources (*e.g.*, GPU memory and running time). Here, we use image DN as an example.

Figure 2.2: The architecture of our proposed residual dense network (RDN) for image restoration.

denoiser prior for image restoration (IRCNN) by integrating CNN denoisers into model-based optimization method. However, such methods have limited network depth (e.g., 30 for RED, 20 for DnCNN, and 7 for IRCNN), limiting the network ability. Simply stacking more layers cannot reach better results due to gradient vanishing problem. On the other hand, by using short term and long-term memory, Tai *et al*. [21] proposed MemNet for image restoration, where the network depth reached 212 but obtained limited improvement over results with 80 layers. For 31×31 input patches from 91 images, training an 80-layer MemNet takes 5 days using 1 Tesla P40 GPU [21].

The aforementioned DL-based image restoration methods have achieved significant improvement over conventional methods, but most of them lose some useful hierarchical features from the original LQ image. Hierarchical features produced by a very deep network are useful for image restoration tasks (e.g., image SR). To fix this case, we propose residual dense network (RDN) to extract and adaptively fuse features from all the layers in the LQ space efficiently.

2.3 Residual Dense Network for IR

In Figure 2.2, we show our proposed RDN for image restoration, including image super-resolution (see Figure 2.2(a)), denoising, compression artifact reduction, and deblurring (see Fig-

ure 2.2(b)).

2.3.1 Network Structure

we mainly take image SR as an example and give specific illustrations for image DN and CAR cases.

RDN for image SR. As shown in Figure 2.2(a), our RDN mainly consists of four parts: shallow feature extraction net, residual dense blocks (RDBs), dense feature fusion (DFF), and finally the up-sampling net (UPNet). Let's denote I_{LQ} and I_{HQ} as the input and output of RDN. Specifically, we use two Conv layers to extract shallow features. The first Conv layer extracts features F_{-1} from the LQ input.

$$F_{-1} = H_{SFE1}\left(I_{LQ}\right), \tag{2.1}$$

where $H_{SFE1}\left(\cdot\right)$ denotes convolution operation. F_{-1} is then used for further shallow feature extraction and global residual learning. So, we can further have

$$F_0 = H_{SFE2}\left(F_{-1}\right), \tag{2.2}$$

where $H_{SFE2}\left(\cdot\right)$ denotes convolution operation of the second shallow feature extraction layer and is used as input to residual dense blocks. Supposing we have D residual dense blocks, the output F_d of the d-th RDB can be obtained by

$$\begin{aligned} F_d &= H_{RDB,d}\left(F_{d-1}\right) \\ &= H_{RDB,d}\left(H_{RDB,d-1}\left(\cdots\left(H_{RDB,1}\left(F_0\right)\right)\cdots\right)\right), \end{aligned} \tag{2.3}$$

where $H_{RDB,d}$ denotes the operations of the d-th RDB. $H_{RDB,d}$ can be a composite function of operations, such as convolution and rectified linear units (ReLU) [104]. As F_d is produced by the d-th RDB fully utilizing each convolutional layers within the block, we can view F_d as local feature. More details about RDB will be given in Section 2.3.2.

After extracting hierarchical features with a set of RDBs, we further conduct dense feature fusion (DFF), which includes global feature fusion (GFF) and global residual learning. DFF makes full use of features from all the preceding layers and can be represented as

$$F_{DF} = H_{DFF}\left(F_{-1}, F_0, F_1, \cdots, F_D\right), \tag{2.4}$$

where F_{DF} is the output feature-maps of DFF by utilizing a composite function H_{DFF}. More details about DFF will be shown in Section 2.3.3.

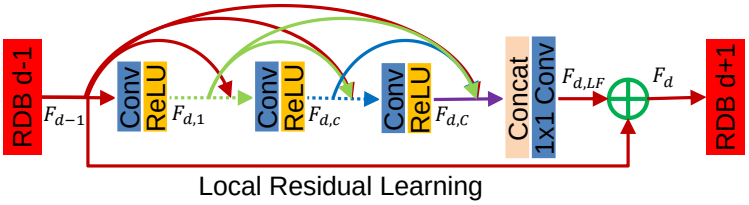


Figure 2.3: Residual dense block (RDB) architecture. We denote the connections between the (d-1)-th RDB and the following convolutional layers as contiguous memory (CM) mechanism.

After extracting local and global features in the LQ space, we stack an up-sampling net (UPNet) in the HQ space. Inspired by [1], we utilize ESPCN [102] in UPNet followed by one Conv layer. The output of RDN can be obtained by

$$I_{HQ} = H_{RDN}\left(I_{LQ}\right), \tag{2.5}$$

where H_{RDN} denotes the function of our RDN.

RDN for image DN, CAR, and Deblurring. When we apply our RDN to image DN, CAR, and Deblurring, the resolution of the input and output keep the same. As shown in Figure 2.2(b), we remove the upscaling module in UPNet and obtain the final HQ output via residual learning

$$I_{HQ} = H_{RDN}\left(I_{LQ}\right) + I_{LQ}. \tag{2.6}$$

Unlike image SR, here, we use residual learning connecting input and output for faster training. It should be noted that the network structure in Figure 2.2(b) is also suitable for image SR. But, it would consume more GPU memory and running time. As a result, we choose the network structure in Figure 2.2(a) for image SR.

2.3.2 Residual Dense Block

We present details about our proposed residual dense block (RDB) shown in Figure 2.3. Our RDB contains dense connected layers, local feature fusion (LFF), and local residual learning, leading to a contiguous memory (CM) mechanism.

Contiguous memory (CM) mechanism. The idea behind CM is that we aim at fusing information from all the Conv layers as much as possible. But, directly fuse feature maps from all the Covn layers is not practical, because it would stack huge amount of features. Instead, we turn to first adaptively fuse information locally and then pass them to the following feature fusions. It is realized by passing the state of preceding RDB to each layer of the current RDB. Let F_{d-1} and F_d

be the input and output of the d-th RDB respectively and both have G_0 feature-maps. The output of c-th Conv layer of d-th RDB can be formulated as

$$F_{d,c} = \sigma\left(W_{d,c}\left[F_{d-1}, F_{d,1}, \cdots, F_{d,c-1}\right]\right), \tag{2.7}$$

where σ denotes the ReLU [104] activation function. $W_{d,c}$ is the weights of the c-th Conv layer, where the bias term is omitted for simplicity. We assume $F_{d,c}$ consists of G (also known as growth rate [20]) feature-maps. $[F_{d-1}, F_{d,1}, \cdots, F_{d,c-1}]$ refers to the concatenation of the feature-maps produced by the $(d-1)$-th RDB, convolutional layers $1, \cdots, (c-1)$ in the d-th RDB, resulting in G_0+$(c-1)$ $\times$G feature-maps. The outputs of the preceding RDB and each layer have direct connections to all subsequent layers, which not only preserves the feed-forward nature, but also extracts local dense feature.

Local feature fusion (LFF). We apply LFF to adaptively fuse the states from preceding RDB and the whole Conv layers in current RDB. As analyzed above, the feature-maps of the $(d-1)$-th RDB are introduced directly to the d-th RDB in a concatenation way, it is essential to reduce the feature number. On the other hand, inspired by MemNet [21], we introduce a 1×1 convolutional layer to adaptively control the output information. We name this operation as local feature fusion (LFF) formulated as

$$F_{d,LF} = H^d_{LFF}\left(\left[F_{d-1}, F_{d,1}, \cdots, F_{d,c}, \cdots, F_{d,C}\right]\right), \tag{2.8}$$

where H^d_{LFF} denotes the function of the 1×1 Conv layer in the d-th RDB. We also find that as the growth rate G becomes larger, very deep dense network without LFF would be hard to train. However, larger growth rate further contributes to the performance, which will be detailed in Section 2.5.1.

Local residual learning (LRL). We introduce LRL in RDB to further improve the information flow and allow larger growth rate, as there are several convolutional layers in one RDB. The final output of the d-th RDB can be obtained by

$$F_d = F_{d-1} + F_{d,LF}. \tag{2.9}$$

It should be noted that LRL can also further improve the network representation ability, resulting in better performance. We introduce more results about LRL in Section 2.5.2. Because of the dense connectivity and local residual learning, we refer to this block architecture as residual dense block (RDB). More differences between RDB and original dense block [20] would be summarized in Section 2.4.

2.3.3 Dense Feature Fusion

After extracting local dense features with a set of RDBs, we further propose dense feature fusion (DFF) to exploit hierarchical features in a global way. DFF consists of global feature fusion (GFF) and global residual learning.

Global feature fusion (GFF). We propose GFF to extract the global feature F_{GF} by fusing features from all the RDBs

$$F_{GF} = H_{GFF}\left([F_1, \cdots, F_D]\right), \tag{2.10}$$

where $[F_1, \cdots, F_D]$ refers to the concatenation of feature maps produced by residual dense blocks $1, \cdots, D$. H_{GFF} is a composite function of 1×1 and 3×3 convolution. The 1×1 convolutional layer is used to adaptively fuse a range of features with different levels. The following 3×3 convolutional layer is introduced to further extract features for global residual learning, which has been demonstrated to be effective in [103].

Global residual learning. We then utilize global residual learning to obtain the feature-maps before conducting up-scaling by

$$F_{DF} = F_{-1} + F_{GF}, \tag{2.11}$$

where F_{-1} denotes the shallow feature-maps. All the other layers before global feature fusion are extensively utilized with our proposed residual dense blocks (RDBs). RDBs produce multi-level local dense features, which are further adaptively fused to form F_{GF}. After global residual learning, we obtain deep dense feature F_{DF}.

It should be noted that Tai *et al*. [21] utilized long-term dense connections in MemNet to recover more high-frequency information. However, in the memory block [21], the preceding layers don't have direct access to all the subsequent layers. The local feature information is not fully used, limiting the ability of long-term connections. In addition, MemNet extracts features in the HQ space, increasing computational complexity. While, inspired by [101, 102, 84, 1], we extract local and global features in the LQ space. More differences between our proposed RDN and MemNet would be shown in Section 2.4. We would also demonstrate the effectiveness of global feature fusion in Section 2.5.2.

2.3.4 Implementation Details

In our proposed RDN, we set 3×3 as the size of all convolutional layers except that in local and global feature fusion, whose kernel size is 1×1. For convolutional layer with kernel size

3×3, we pad zeros to each side of the input to keep size fixed. Shallow feature extraction layers, local and global feature fusion layers have G_0=64 filters. Other layers in each RDB has G=64 filters and are followed by ReLU [104]. For image SR, following [1], we use ESPCNN [102] to upscale the coarse resolution features to fine ones for the UPNet. For image DN and CAR, the up-scaling module is removed from UPNet. The final Conv layer has 3 output channels, as we output color HQ images. However, the network can also process gray images, for example, when we apply RDN for gray-scale image denoising.

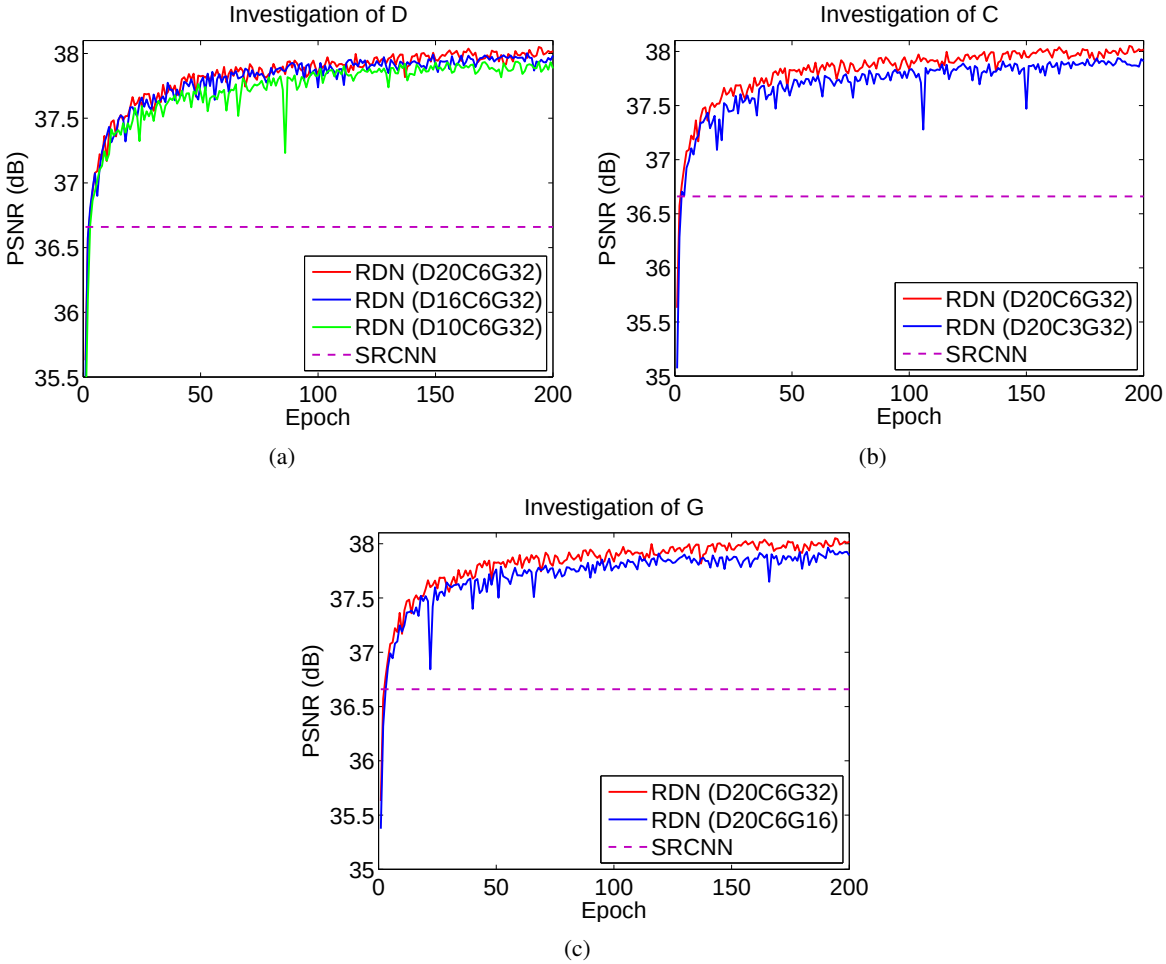


Figure 2.4: Convergence analysis of RDN for image SR ($\times 2$) with different values of D, C, and G.

2.4 Differences with Prior Works

Here, we give more details about the differences between our RDN and several related representative works. We further demonstrate those differences make our RDN more effective in Sections 2.5 and 2.6.

Difference to DenseNet. First, DenseNet [20] is widely used in high-level computer vision tasks (e.g., image classification). While RDN is designed more specific for image restoration without using some operations in DenseNet, such as batch normalization and max pooling. Second, DenseNet places transition layers into two adjacent dense blocks. In RDN, the Conv layers are connected by local feature fusion (LFF) and local residual learning. Consequently, the $(d-1)$-th RDB has direct access to each layer in the d-th RDB and also contributes to the input of $(d+1)$-th RDB. Third, we adopt GFF to make better use of hierarchical features, which are neglected in DenseNet.

Difference to SRDenseNet. First, SRDenseNet [2] introduces the basic dense block with batch normalization from DenseNet [20]. Our residual dense block (RDB) improves it in several ways: (1). We propose contiguous memory (CM) mechanism, building direct connections between the preceding RDB and each layer of the current RDB. (2). Our RDB can be easily extended to be wider and be easy to train with usage of local feature fusion (LFF). (3). RDB utilizes local residual learning (LRL) to further encourage the information flow. Second, SRDenseNet densely connects dense blocks, while there are no dense connections among RDBs. We use global feature fusion (GFF) and global residual learning to extract hierarchical features, based on the fully extracted extracted local features by RDBs. Third, SRDenseNet aims to minimize L_2 loss. However, bing more powerful for performance and convergence [1], L_1 loss function is utilized in RDN for each image restoration task.

Difference to MemNet. In addition to the different choices for loss fuction between MemNet [21] and RDN, we mainly summarize additional three differences. First, MemNet has to interpolate the original LR to the desired size, introducing extra blurry artifacts. While, RDN directly extracts features from the original LR input, which helps to reduce computational complexity and contribute to better performance. Second, in MemNet, most Conv layers within one recursive unit have no direct access to their preceding layers or memory block. While, each Conv layer receives information from preceding RDB and outputs feature for the subsequent layers within same RDB. Plus, LRL further improves the information flow and performance. Third, in MemNet, the memory block neglects to fully utilize the features from preceding block and Conv layers within it. Although dense connections are adopted to connect memory blocks, MemNet fails to extract hierarchical

Table 2.1: PSNR (dB) comparisons under different block connections. The results are obtained with Bicubic (**BI**) degradation model for image SR ($\times 4$).

Block connection	Dense connections		Contiguous memory	
Datasets	SRDenseNet [30]	MemNet [25]	RDN (w/o LRL)	RDN (with LRL)
Set5 [105]	32.02	31.74	32.54	32.61
Set14 [106]	28.50	28.26	28.87	28.93
B100 [107]	27.53	27.40	27.75	27.80
Urban100 [61]	26.05	25.50	26.72	26.85

features from the original LR images. However, with the local dense features extracted by RDBs, our RDN further fuses the hierarchical features from the whole preceding layers in a global way in the LR space.

2.5 Network Investigations

2.5.1 Study of D, C, and G.

In this subsection, we investigate the basic network parameters: the number of RDB (denote as D for short), the number of Conv layers per RDB (denote as C for short), and the growth rate (denote as G for short). We use the performance of SRCNN [108] as a reference. As shown in Figures 2.4(a) and 2.4(b), larger D or C would lead to higher performance. This is mainly because the network becomes deeper with larger D or C. As our proposed LFF allows larger G, we also observe larger G (see Figure 2.4(c)) contributes to better performance. On the other hand, RND with smaller D, C, or G would suffer some performance drop in the training, but RDN would still outperform SRCNN [108]. More important, our RDN allows deeper and wider network, where more hierarchical features are extracted for higher performance.

2.5.2 Ablation Investigation

Table 2.2 shows the ablation investigation on the effects of contiguous memory (CM), local residual learning (LRL), and global feature fusion (GFF). The eight networks have the same RDB number (D = 20), Conv number (C = 6) per RDB, and growth rate (G = 32). We find that local feature fusion (LFF) is needed to train these networks properly, so LFF isn't removed by

Table 2.2: Ablation investigation of contiguous memory (CM), local residual learning (LRL), and global feature fusion (GFF). We observe the best performance (PSNR) on Set5 with scaling factor $\times 2$ in 200 epochs.

	Different combinations of CM, LRL, and GFF							
CM	✗	✓	✗	✗	✓	✓	✗	✓
LRL	✗	✗	✓	✗	✓	✗	✓	✓
GFF	✗	✗	✗	✓	✗	✓	✓	✓
PSNR	34.87	37.89	37.92	37.78	37.99	37.98	37.97	38.06

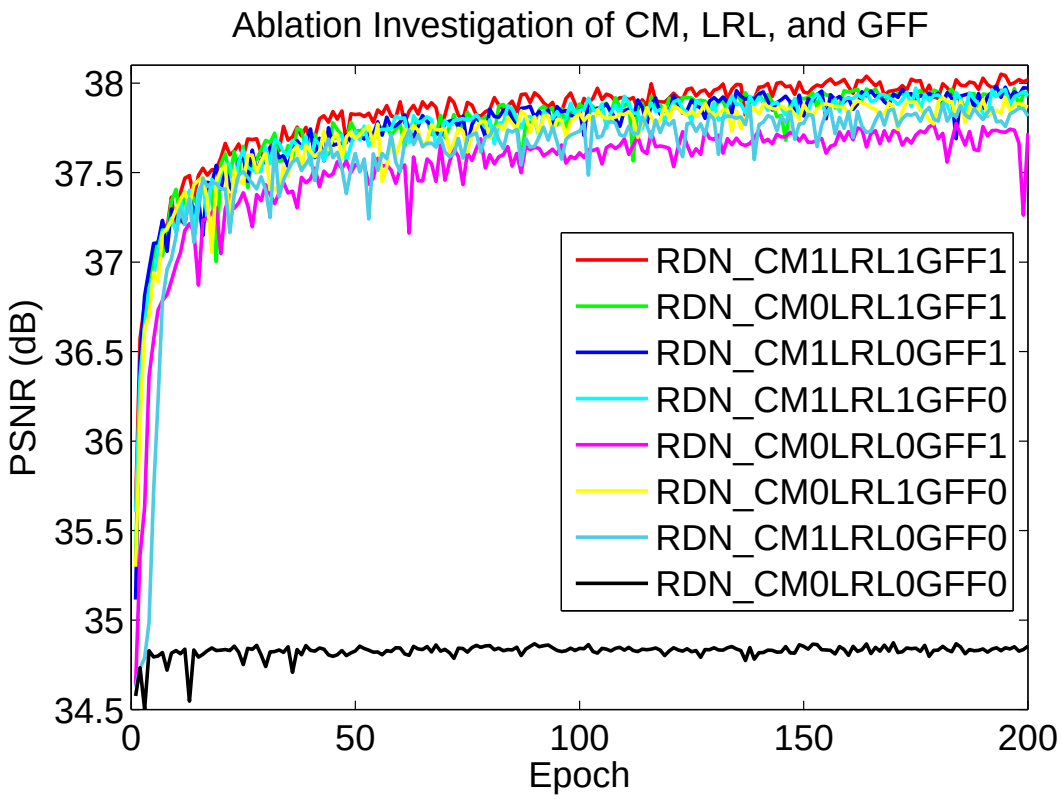


Figure 2.5: Convergence analysis on CM, LRL, and GFF. The curves for each combination is based on the PSNR on Set5 ($\times 2$) in 200 epochs.

Table 2.3: Parameter number, PSNR, and test time comparisons. The PSNR values are based on Set14 with Bicubic (**BI**) degradation model ($\times 2$).

Methods	LapSRN [84]	DRRN [100]	MemNet [21]	MDSR [1]	EDSR [1]	RDN (ours)
# param.	812K	297K	677K	8M	43M	22M
PSNR (dB)	33.08	33.23	33.28	33.85	33.92	34.14
Time (s)	0.10	17.81	13.84	0.53	1.64	1.56

default. The baseline (denote as RDN_CM0LRL0GFF0) is obtained without CM, LRL, or GFF and performs very poorly (PSNR = 34.87 dB). This is caused by the difficulty of training [108] and also demonstrates that stacking many basic dense blocks [20] in a very deep network would not result in better performance.

We then add one of CM, LRL, or GFF to the baseline, resulting in RDN_CM1LRL0GFF0, RDN_CM0LRL1GFF0, and RDN_CM0LRL0GFF1 respectively (from 2^{nd} to 4^{th} combination in Table 2.2). We can validate that each component can efficiently improve the performance of the baseline. This is mainly because each component contributes to the flow of information and gradient.

We further add two components to the baseline, resulting in RDN_CM1LRL1GFF0, RDN_CM1LRL0GFF1, and RDN_CM0LRL1GFF1 respectively (from 5^{th} to 7^{th} combination in Table 2.2). It can be seen that two components would perform better than only one component. Similar phenomenon can be seen when we use these three components simultaneously (denote as RDN_CM1LRL1GFF1). RDN using three components performs the best.

We also visualize the convergence process of these eight combinations in Figure 2.5. The convergence curves are consistent with the analyses above and show that CM, LRL, and GFF can further stabilize the training process without obvious performance drop. These quantitative and visual analyses demonstrate the effectiveness and benefits of our proposed CM, LRL, and GFF.

2.5.3 Model Size, Performance, and Test Time

We also compare the model size, performance, and test time with other methods on Set14 ($\times 2$) in Table 2.3. Compared with EDSR, our RDN has half less amount of parameter and obtains better results. Although our RDN has more parameters than that of other methods, RDN achieves comparable (e.g., MDSR) or even less test time (e.g., MemNet). We further visualize the performance and test time comparison in Figure 2.6. We can see that our RDN achieves good trade-off between

Table 2.4: Quantitative results (BI model). Best and second best are **highlighted** and underlined.

Method	Scale	Set5		Set14		B100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	×2	33.66	0.9299	30.24	0.8688	29.56	0.8431	26.88	0.8403	30.80	0.9339
SRCNN [108]	×2	36.66	0.9542	32.45	0.9067	31.36	0.8879	29.50	0.8946	35.60	0.9663
FSRCNN [101]	×2	37.05	0.9560	32.66	0.9090	31.53	0.8920	29.88	0.9020	36.67	0.9710
VDSR [78]	×2	37.53	0.9590	33.05	0.9130	31.90	0.8960	30.77	0.9140	37.22	0.9750
LapSRN [84]	×2	37.52	0.9591	33.08	0.9130	31.08	0.8950	30.41	0.9101	37.27	0.9740
MemNet [21]	×2	37.78	0.9597	33.28	0.9142	32.08	0.8978	31.31	0.9195	37.72	0.9740
EDSR [1]	×2	38.11	0.9602	33.92	0.9195	32.32	0.9013	32.93	0.9351	39.10	0.9773
SRMDNF [79]	×2	37.79	0.9601	33.32	0.9159	32.05	0.8985	31.33	0.9204	38.07	0.9761
D-DBPN [92]	×2	38.09	0.9600	33.85	0.9190	32.27	0.9000	32.55	0.9324	38.89	0.9775
RDN (ours)	×2	38.30	0.9617	34.14	0.9235	32.41	0.9025	33.17	0.9377	39.60	0.9791
RDN+ (ours)	×2	**38.34**	**0.9618**	**34.28**	**0.9241**	**32.46**	**0.9030**	**33.36**	**0.9388**	**39.74**	**0.9794**
Bicubic	×3	30.39	0.8682	27.55	0.7742	27.21	0.7385	24.46	0.7349	26.95	0.8556
SRCNN [108]	×3	32.75	0.9090	29.30	0.8215	28.41	0.7863	26.24	0.7989	30.48	0.9117
FSRCNN [101]	×3	33.18	0.9140	29.37	0.8240	28.53	0.7910	26.43	0.8080	31.10	0.9210
VDSR [78]	×3	33.67	0.9210	29.78	0.8320	28.83	0.7990	27.14	0.8290	32.01	0.9340
LapSRN [84]	×3	33.82	0.9227	29.87	0.8320	28.82	0.7980	27.07	0.8280	32.21	0.9350
MemNet [21]	×3	34.09	0.9248	30.00	0.8350	28.96	0.8001	27.56	0.8376	32.51	0.9369
EDSR [1]	×3	34.65	0.9280	30.52	0.8462	29.25	0.8093	28.80	0.8653	34.17	0.9476
SRMDNF [79]	×3	34.12	0.9254	30.04	0.8382	28.97	0.8025	27.57	0.8398	33.00	0.9403
RDN (ours)	×3	34.78	0.9299	30.63	0.8477	29.33	0.8107	29.02	0.8695	34.58	0.9502
RDN+ (ours)	×3	**34.84**	**0.9303**	**30.74**	**0.8495**	**29.38**	**0.8115**	**29.18**	**0.8718**	**34.81**	**0.9512**
Bicubic	×4	28.42	0.8104	26.00	0.7027	25.96	0.6675	23.14	0.6577	24.89	0.7866
SRCNN [108]	×4	30.48	0.8628	27.50	0.7513	26.90	0.7101	24.52	0.7221	27.58	0.8555
FSRCNN [101]	×4	30.72	0.8660	27.61	0.7550	26.98	0.7150	24.62	0.7280	27.90	0.8610
VDSR [78]	×4	31.35	0.8830	28.02	0.7680	27.29	0.0726	25.18	0.7540	28.83	0.8870
LapSRN [84]	×4	31.54	0.8850	28.19	0.7720	27.32	0.7270	25.21	0.7560	29.09	0.8900
MemNet [21]	×4	31.74	0.8893	28.26	0.7723	27.40	0.7281	25.50	0.7630	29.42	0.8942
SRDenseNet [2]	×4	32.02	0.8930	28.50	0.7780	27.53	0.7337	26.05	0.7819	N/A	N/A
EDSR [1]	×4	32.46	0.8968	28.80	0.7876	27.71	0.7420	26.64	0.8033	31.02	0.9148
SRMDNF [79]	×4	31.96	0.8925	28.35	0.7787	27.49	0.7337	25.68	0.7731	30.09	0.9024
D-DBPN [92]	×4	32.47	0.8980	28.82	0.7860	27.72	0.7400	26.38	0.7946	30.91	0.9137
RDN (ours)	×4	32.61	0.8999	28.93	0.7894	27.80	0.7436	26.85	0.8089	31.45	0.9187
RDN+ (ours)	×4	**32.69**	**0.9007**	**29.01**	**0.7909**	**27.85**	**0.7447**	**27.01**	**0.8120**	**31.74**	**0.9208**
Bicubic	×8	24.40	0.6580	23.10	0.5660	23.67	0.5480	20.74	0.5160	21.47	0.6500
SRCNN [108]	×8	25.33	0.6900	23.76	0.5910	24.13	0.5660	21.29	0.5440	22.46	0.6950
SCN [109]	×8	25.59	0.7071	24.02	0.6028	24.30	0.5698	21.52	0.5571	22.68	0.6963
VDSR [78]	×8	25.93	0.7240	24.26	0.6140	24.49	0.5830	21.70	0.5710	23.16	0.7250
LapSRN [84]	×8	26.15	0.7380	24.35	0.6200	24.54	0.5860	21.81	0.5810	23.39	0.7350
MemNet [21]	×8	26.16	0.7414	24.38	0.6199	24.58	0.5842	21.89	0.5825	23.56	0.7387
MSLapSRN [110]	×8	26.34	0.7558	24.57	0.6273	24.65	0.5895	22.06	0.5963	23.90	0.7564
EDSR [1]	×8	26.96	0.7762	24.91	0.6420	24.81	0.5985	22.51	0.6221	24.69	0.7841
D-DBPN [92]	×8	27.21	0.7840	25.13	0.6480	24.88	0.6010	22.73	0.6312	25.14	0.7987
RDN (ours)	×8	27.23	0.7854	25.25	0.6505	24.91	0.6032	22.83	0.6374	25.14	0.7994
RDN+ (ours)	×8	**27.40**	**0.7900**	**25.38**	**0.6541**	**25.01**	**0.6057**	**23.04**	**0.6439**	**25.48**	**0.8058**

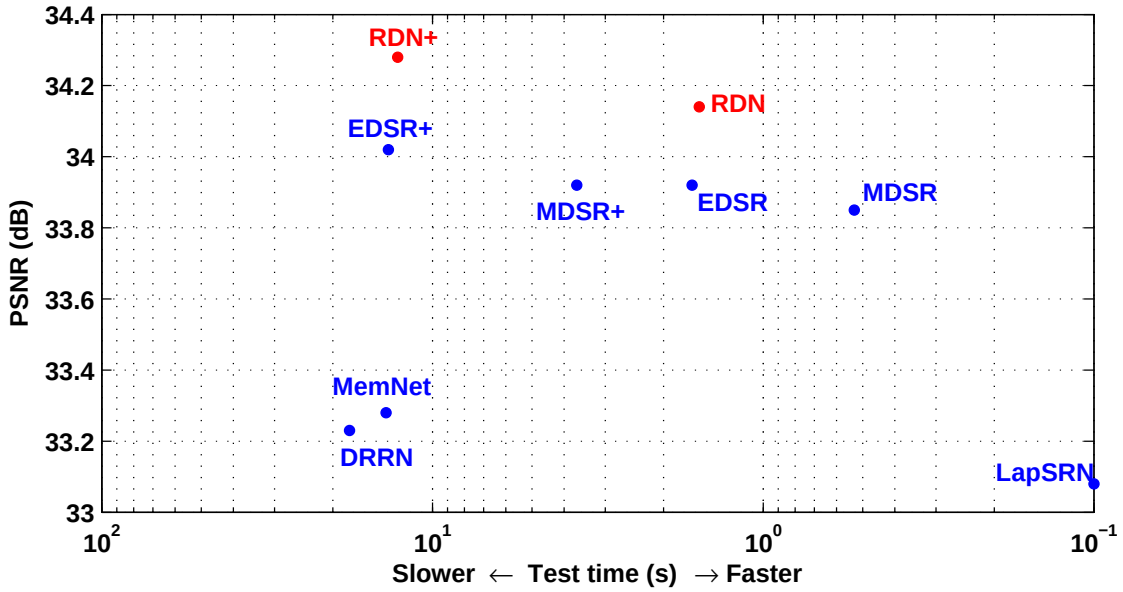


Figure 2.6: PSNR and test time on Set14 with **BI** model ($\times 2$).

the performance and running time.

2.6 Experimental Results

The source code and models of the proposed method can be downloaded at Github [1].

2.6.1 Settings

Here we provide details of experimental settings, such as training/testing data for each tasks.

Training Data. Recently, Timofte *et al.*have released a high-quality (2K resolution) dataset DIV2K for image restoration applications [91]. DIV2K consists of 800 training images, 100 validation images, and 100 test images. We use DIV2K as training data for image SR (except for Bicubic degradation model), color and gray image denoising, CAR, and deblurring. For image SR with Bicubic degradation (**BI**), some compared methods (e.g., D-DBPN [92]) further use Flickr2K [1] as well as DIV2K for training. We also train RDN by using larger training data to investigate whether RDN can further improve performance.

[1] https://github.com/yulunzhang/RDN

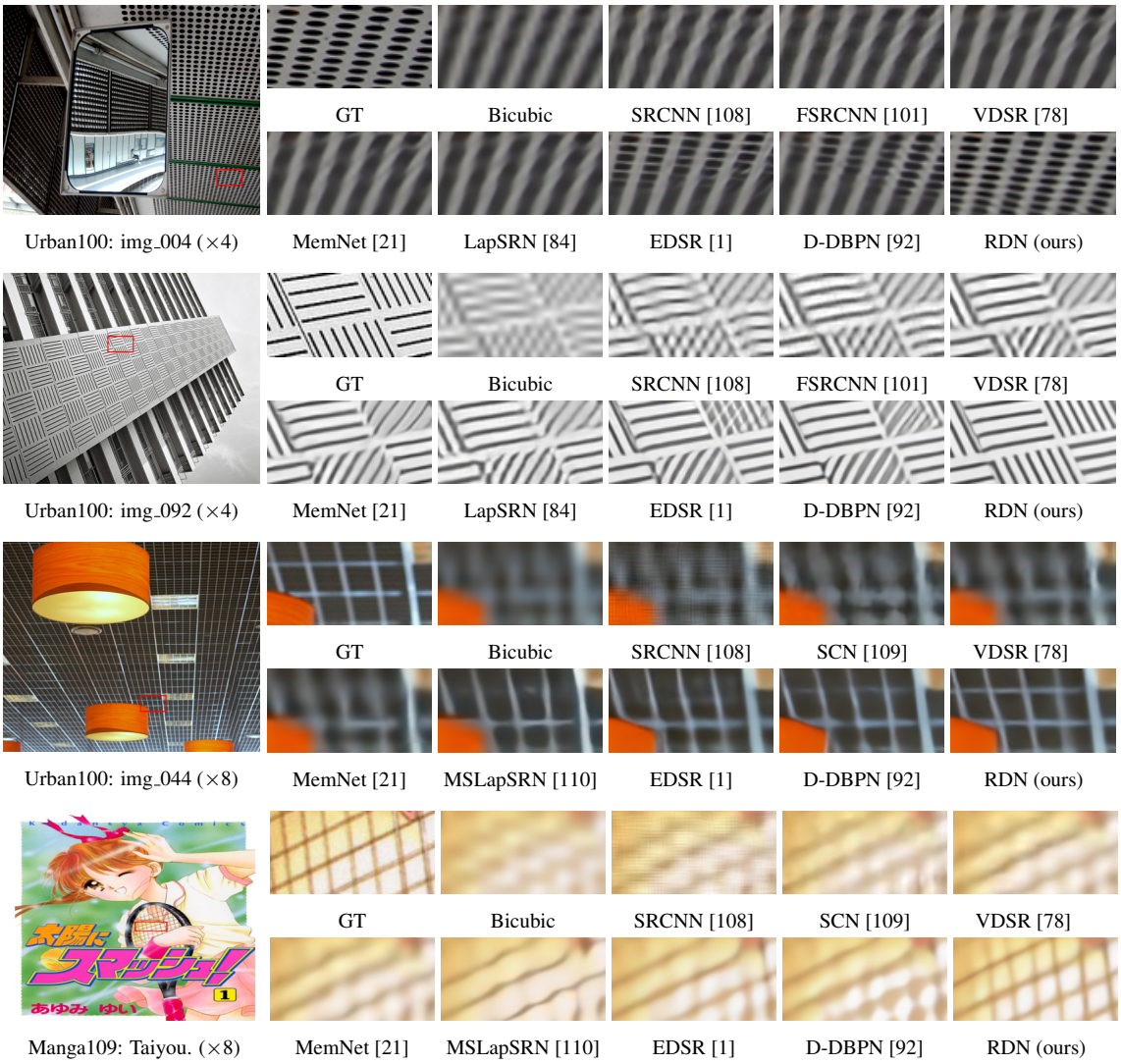


Figure 2.7: Image super-resolution results (**BI** degradation model) with scaling factors $s = 4$ (first two rows) and $s = 8$ (last two rows).

Testing Data and Metrics. For testing, we use five standard benchmark datasets: Set5 [105], Set14 [106], B100 [107], Urban100 [61], and Manga109 [111] for image SR. We use Kodak24 [2], BSD68 [107], and Urban100 [61] for color and gray image DN. LIVE1 [112] and Classic5 [113] are used for image CAR. We use McMaster18 [37], Kodak24, and Urban100 as testing data for image deblurring. The quantitative results are evaluated with PSNR and SSIM [114] on Y channel (i.e., luminance) of transformed YCbCr space.

Degradation Models. For image SR, in order to fully demonstrate the effectiveness of our proposed RDN, we use three degradation models to simulate LR images for image SR. The first one is bicubic downsampling by adopting the Matlab function *imresize* with the option *bicubic* (denote as **BI** for short). We use **BI** model to simulate LR images with scaling factor $\times 2$, $\times 3$, $\times 4$, and $\times 8$. Similar to [37], the second one is to blur HR image by Gaussian kernel of size 7×7 with standard deviation 1.6. The blurred image is then downsampled with scaling factor $\times 3$ (denote as **BD** for short). We further produce LR image in a more challenging way. We first bicubic downsample HR image with scaling factor $\times 3$ and then add Gaussian noise with noise level 30 (denote as **DN** for short). For color and gray image denoising, we add Gaussian noise with noise level σ to the ground truth to obtain the noisy inputs. For image CAR, we use Matlab JPEG encoder [115] to generate compressed inputs. For image deburring, the commonly-used 25×25 Gaussian blur kernel of standard deviation 1.6 is used to blur images first. Then, additive Gaussian noise ($\sigma = 2$) is added to the blurry images to obtain the final inputs.

Training Setting. Following settings of [1], in each training batch, we randomly extract 16 LQ RGB patches with the size of 48×48 as inputs for image SR, DN, CAR, and deblurring. We randomly augment the patches by flipping horizontally or vertically and rotating 90°. 1,000 iterations of back-propagation constitute an epoch. We implement our RDN with the Torch7 framework and update it with Adam optimizer [116]. The learning rate is initialized to 10^{-4} for all layers and decreases half for every 200 epochs. Training a RDN roughly takes 1 day with a Titan Xp GPU for 200 epochs.

2.6.2 Image Super-Resolution

2.6.2.1 Results with BI Degradation Model

Simulating LR image with BI degradation model is widely used in image SR settings. For BI degradation model, we compare our RDN with state-of-the-art image SR methods: SRCNN [108],

[2] http://r0k.us/graphics/kodak/

Table 2.5: Benchmark results with **BD** and **DN** degradation models. Average PSNR/SSIM values for scaling factor $\times 3$.

Dataset	Model	Bicubic	SRCNN [108]	FSRCNN [101]	VDSR [78]	IRCNN_G [37]	IRCNN_C [37]	RDN (ours)	RDN+ (ours)
Set5	**BD**	28.78/0.8308	32.05/0.8944	26.23/0.8124	33.25/0.9150	33.38/0.9182	33.17/0.9157	34.58/0.9280	**34.70/0.9289**
	DN	24.01/0.5369	25.01/0.6950	24.18/0.6932	25.20/0.7183	25.70/0.7379	27.48/0.7925	28.47/0.8151	**28.55/0.8173**
Set14	**BD**	26.38/0.7271	28.80/0.8074	24.44/0.7106	29.46/0.8244	29.63/0.8281	29.55/0.8271	30.53/0.8447	**30.64/0.8463**
	DN	22.87/0.4724	23.78/0.5898	23.02/0.5856	24.00/0.6112	24.45/0.6305	25.92/0.6932	26.60/0.7101	**26.67/0.7117**
B100	**BD**	26.33/0.6918	28.13/0.7736	24.86/0.6832	28.57/0.7893	28.65/0.7922	28.49/0.7886	29.23/0.8079	**29.30/0.8093**
	DN	22.92/0.4449	23.76/0.5538	23.41/0.5556	24.00/0.5749	24.28/0.5900	25.55/0.6481	25.93/0.6573	**25.97/0.6587**
Urban100	**BD**	23.52/0.6862	25.70/0.7770	22.04/0.6745	26.61/0.8136	26.77/0.8154	26.47/0.8081	28.46/0.8582	**28.67/0.8612**
	DN	21.63/0.4687	21.90/0.5737	21.15/0.5682	22.22/0.6096	22.90/0.6429	23.93/0.6950	24.92/0.7364	**25.05/0.7399**
Manga109	**BD**	25.46/0.8149	29.47/0.8924	23.04/0.7927	31.06/0.9234	31.15/0.9245	31.13/0.9236	33.97/0.9465	**34.34/0.9483**
	DN	23.01/0.5381	23.75/0.7148	22.39/0.7111	24.20/0.7525	24.88/0.7765	26.07/0.8253	28.00/0.8591	**28.18/0.8621**

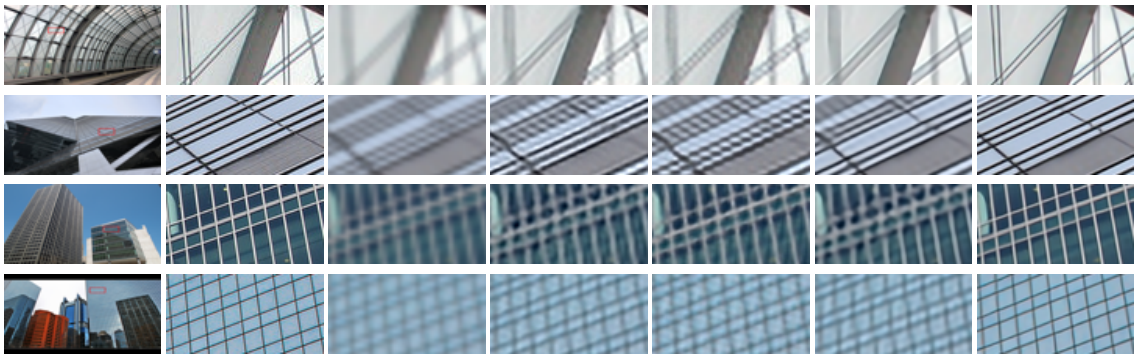


Figure 2.8: Visual results using **BD** degradation model with scaling factor $\times 3$.

FSRCNN [101], SCN [109], VDSR [78], LapSRN [84], MemNet [21], SRDenseNet [2], MSLapSRN [110], EDSR [1], SRMDNF [79], D-DBPN [92], and DPDNN [36]. Similar to [117, 1], we also adopt self-ensemble strategy [1] to further improve our RDN and denote the self-ensembled RDN as RDN+. Here, we also additionally use Flickr2K [91] as training data, which is also used in SRMDNF [79], and D-DBPN [92]. As analyzed above, a deeper and wider RDN would lead to a better performance. On the other hand, as most methods for comparison only use about 64 filters per Conv layer, we report results of RDN by using D = 16, C = 8, and G = 64 for a fair comparison.

Table 2.4 shows quantitative comparisons for $\times 2$, $\times 3$, and $\times 4$ SR. Results of SRDenseNet [2] are cited from their paper. When compared with persistent CNN models (SRDenseNet [2] and MemNet [21]), our RDN performs the best on all datasets with all scaling factors. This indicates the

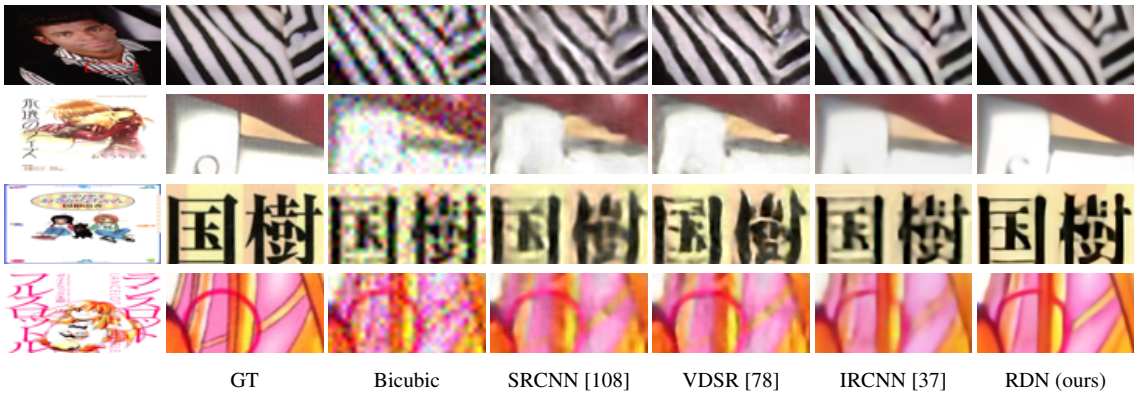


Figure 2.9: Visual results using **DN** degradation model with scaling factor $\times 3$.

better effectiveness of our residual dense block (RDB) over dense block in SRDensenet [2] and the memory block in MemNet [21]. When compared with the remaining models, our RDN also achieves the best average results on most datasets. Specifically, for the scaling factor $\times 2$, our RDN performs the best on all datasets. EDSR [1] uses far more filters (i.e., 256) per Conv layer, leading to a very wide network with a large number of parameters (i.e., 43 M). Our RDN has about half less network parameter number and achieves better performance.

In Figure 2.7, we show visual comparisons on scales $\times 4$ and $\times 8$. We observe that most of compared methods cannot recover the lost details in the LR image (e.g., "img_004"), even though EDSR and D-DBPN can reconstruct partial details. In contrast, our RDN can recover sharper and clearer edges, more faithful to the ground truth. In image "img_092", some unwanted artifacts are generated in the degradation process. All the compared methods would fail to handle such a case, but enlarge the mistake. However, our RDN can alleviate the degradation artifacts and recover correct structures. When scaling factor goes larger (e.g., $\times 8$), more structural and textural details are lost. Even we human beings can hardly distinguish the semantic content in the LR images. Most compared methods cannot recover the lost details either. However, with the usage of hierarchical features through dense feature fusion, our RDN reconstruct better visual results with clearer structures.

2.6.2.2 Results with BD and DN Degradation Models

Following [37], we also show the SR results with BD degradation model and further introduce DN degradation model. Our RDN is compared with SPMSR [75], SRCNN [108], FSRCNN [101], VDSR [78], IRCNN_G [37], and IRCNN_C [37]. We re-train SRCNN, FSRCNN, and VDSR for each degradation model. Table 2.5 shows the average PSNR and SSIM results on Set5,

Historical: img004 SRMDNF [79] D-DBPN [92] RDN (ours)

Figure 2.10: Visual results on real-world images with scaling factor $\times 4$.

Set14, B100, Urban100, and Manga109 with scaling factor $\times 3$. Our RDN and RDN+ perform the best on all the datasets with BD and DN degradation models. The performance gains over other state-of-the-art methods are consistent with the visual results in Figures 2.8 and 2.9.

For **BD** degradation model (Figure 2.8), the methods using interpolated LR image as input would produce noticeable artifacts and be unable to remove the blurring artifacts. In contrast, our RDN suppresses the blurring artifacts and recovers sharper edges. This comparison indicates that extracting hierarchical features from the original LR image would alleviate the blurring artifacts. It also demonstrates the strong ability of RDN for **BD** degradation model.

For **DN** degradation model (Figure 2.9), where the LR image is corrupted by noise and loses some details. We observe that the noised details are hard to recovered by other methods [108, 78, 37]. However, our RDN can not only handle the noise efficiently, but also recover more details. This comparison indicates that RDN is applicable for jointly image denoising and SR. These results with **BD** and **DN** degradation models demonstrate the effectiveness and robustness of our RDN model.

2.6.2.3 Super-Resolving Real-World Images

To further demonstrate the effectiveness of our proposed RDN, we super-resolve historical images with JPEG compression artifacts by $\times 4$. We compare with two state-of-the-art methods: SRMDNF [79] and D-DBPN [92]. As shown in Figure 2.10, the historical image contains letters "HOME" and "ANTI". Both SRMDNF and D-DBPN suffer from blurring and distorted artifacts. On the other hand, our RDN reconstruct sharper and more accurate results. These comparisons indicate the benefits of learning hierarchical features from the original input, allowing our RDN to perform robustly for different or unknown degradation models.

Table 2.6: Quantitative results about gray-scale image denoising. Best and second best results are **highlighted** and underlined

Method	Set12				Kodak24				BSD68				Urban100			
	10	30	50	70	10	30	50	70	10	30	50	70	10	30	50	70
BM3D [118]	34.38	29.13	26.72	25.22	34.39	29.13	26.99	25.73	33.31	27.76	25.62	24.44	34.47	28.75	25.94	24.27
TNRD [80]	34.27	28.63	26.81	24.12	34.41	28.87	27.20	24.95	33.41	27.66	25.97	23.83	33.78	27.49	25.59	22.67
RED [39]	34.89	29.70	27.33	25.80	35.02	29.77	27.66	26.39	33.99	28.50	26.37	25.10	34.91	29.18	26.51	24.82
DnCNN [40]	34.78	29.53	27.18	25.52	34.90	29.62	27.51	26.08	33.88	28.36	26.23	24.90	34.73	28.88	26.28	24.36
MemNet [21]	N/A	29.63	27.38	25.90	N/A	29.72	27.68	26.42	N/A	28.43	26.35	25.09	N/A	29.10	26.65	25.01
IRCNN [37]	34.72	29.45	27.14	N/A	34.76	29.53	27.45	N/A	33.74	28.26	26.15	N/A	34.60	28.85	26.24	N/A
MWCNN [97]	N/A	N/A	**27.74**	N/A	N/A	N/A	N/A	N/A	N/A	N/A	**26.53**	N/A	N/A	N/A	27.42	N/A
N^3Net [98]	N/A	N/A	27.43	25.90	N/A	N/A	N/A	N/A	N/A	N/A	26.39	**25.14**	N/A	N/A	26.82	25.15
NLRN [99]	N/A	N/A	27.64	N/A	N/A	N/A	N/A	N/A	N/A	N/A	26.47	N/A	N/A	29.94	27.38	25.66
FFDNet [81]	34.65	29.61	27.32	25.81	34.81	29.70	27.63	26.34	33.76	28.39	26.30	25.04	34.45	29.03	26.52	24.86
RDN (ours)	35.06	29.94	27.60	26.05	35.17	30.00	27.85	26.54	34.00	28.56	26.41	25.10	35.41	30.01	27.40	25.64
RDN+ (ours)	**35.08**	**29.97**	27.64	**26.09**	**35.19**	**30.02**	**27.88**	**26.57**	**34.01**	**28.58**	26.43	25.12	**35.45**	**30.08**	**27.47**	**25.71**

Table 2.7: Quantitative results about color image denoising. Best and second best results are **highlighted** and underlined

Method	Kodak24				BSD68				Urban100			
	10	30	50	70	10	30	50	70	10	30	50	70
CBM3D [119]	36.57	30.89	28.63	27.27	35.91	29.73	27.38	26.00	36.00	30.36	27.94	26.31
TNRD [80]	34.33	28.83	27.17	24.94	33.36	27.64	25.96	23.83	33.60	27.40	25.52	22.63
RED [39]	34.91	29.71	27.62	26.36	33.89	28.46	26.35	25.09	34.59	29.02	26.40	24.74
DnCNN [40]	36.98	31.39	29.16	27.64	36.31	30.40	28.01	26.56	36.21	30.28	28.16	26.17
MemNet [21]	N/A	29.67	27.65	26.40	N/A	28.39	26.33	25.08	N/A	28.93	26.53	24.93
IRCNN [37]	36.70	31.24	28.93	N/A	36.06	30.22	27.86	N/A	35.81	30.28	27.69	N/A
FFDNet [81]	36.81	31.39	29.10	27.68	36.14	30.31	27.96	26.53	35.77	30.53	28.05	26.39
RDN (ours)	37.31	31.94	29.66	28.20	36.47	30.67	28.31	26.85	36.69	31.69	29.29	27.63
RDN+ (ours)	**37.33**	**31.98**	**29.70**	**28.24**	**36.49**	**30.70**	**28.34**	**26.88**	**36.75**	**31.78**	**29.38**	**27.74**

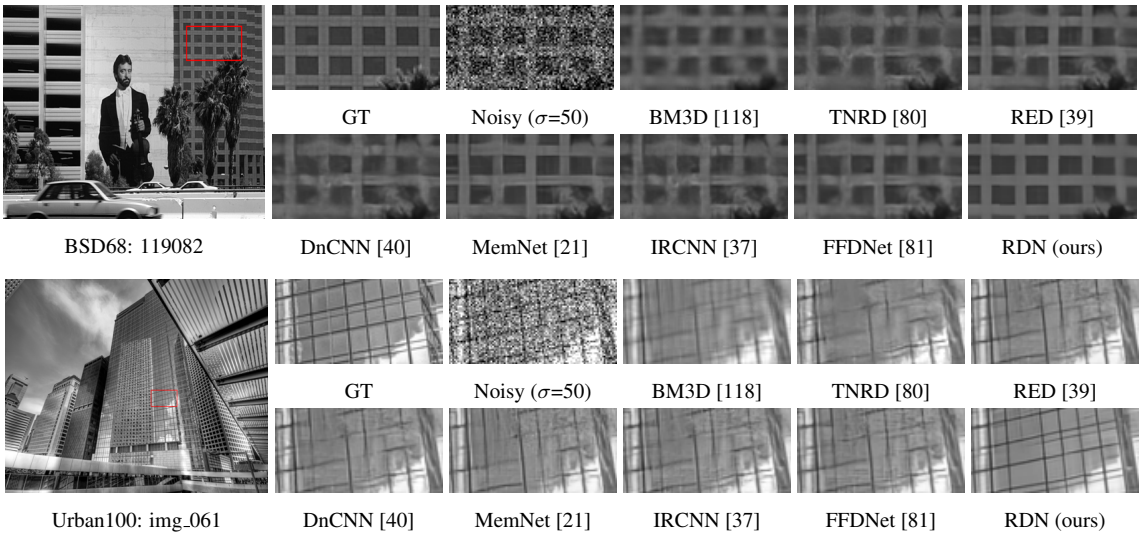


Figure 2.11: Gray-scale image denoising results with noise level σ = 50.

2.6.3 Image Denoising

We compare our RDN with recently leading Gaussian denoising methods: BM3D [118], CBM3D [119], TNRD [80], RED [39], DnCNN [40], MemNet [21], IRCNN [37], MWCNN [97], N^{3}Net [98], NLRN [99], and FFDNet [81]. Kodak24 [3], BSD68 [107], and Urban100 [61] are used for gray-scale and color image denoising. Set12 [40] is also used to test gray-scale image denoising. Noisy images are obtained by adding AWGN noises of different levels to clean images.

2.6.3.1 Gray-scale Image Denoising

The PSNR results are shown in Table 2.6. One can see that on all the 4 test sets with 4 noise levels, our RDN+ achieves higher average PSNR values than most of compared methods. On average, for noise level $\sigma = 50$, our RDN achieves 0.28 dB, 0.22 dB, 0.11 dB, and 0.88 dB gains over FFDNet [81] on four test sets respectively. Gains on Urban100 become larger, which is mainly because our method takes advantage of a larger scope of context information with hierarchical features. Moreover, for noise levels σ = 30, 50, and 70, the gains over BM3D of RDN are larger than 0.7 dB, breaking the estimated PSNR bound (0.7 dB) over BM3D in [120]. It should also be noted that our RDN achieves moderate gains over MemNet on BSD68. This is mainly because BSD68 is formed from 100 validation images in Berkeley Segmentation Dataset (BSD) [107].

[3] http://r0k.us/graphics/kodak/

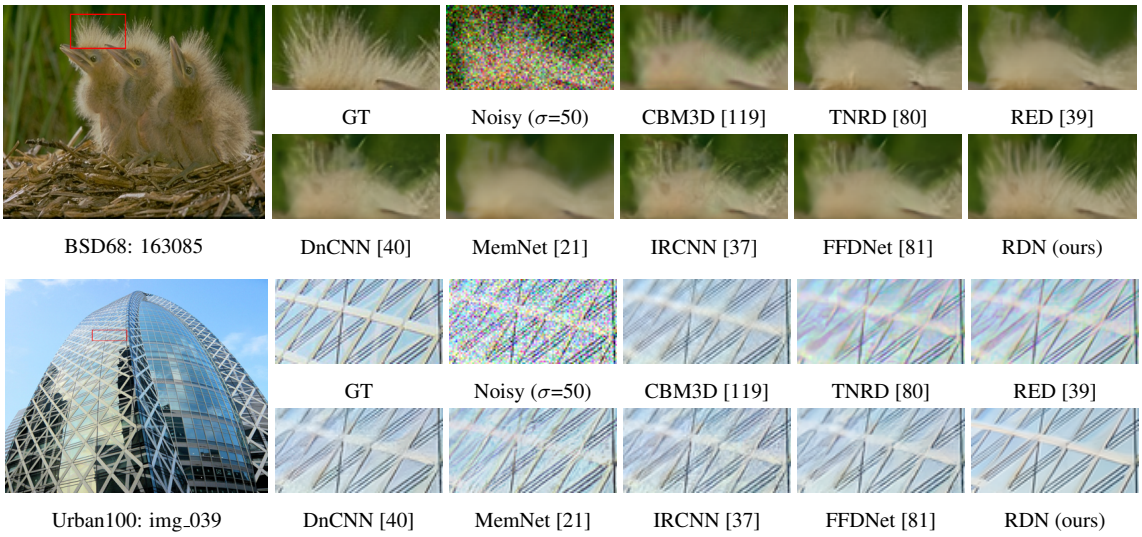


Figure 2.12: Color image denoising results with noise level $\sigma = 50$.

But MemNet [21] used these 100 validation images and 200 training images in BSD as training data. So it's reasonable for MemNet to perform pretty well on BSD68. Moreover, MWCNN [97], N^3Net [98], and NLRN [99] also have achieved good performance for some noise levels, which indicates that Wavelet transformation and non-local processing are promissing candidates for further image denoising improvements.

We show visual gray-scale denoised results of different methods in Figure 2.11. We can see that BM3D preserves image structure to some degree, but fails to remove noise deeply. TNRD [80] tends to generate some artifacts in the smooth region. RED [39], DnCNN [40], MemNet [21], and IRCNN [37] would over-smooth edges. The main reason should be the limited network ability for high noise level (e.g., $\sigma = 50$). In contrast, our RDN can remove noise greatly and recover more details (e.g., the tiny lines in "img_061"). Also, the gray-scale visual results by our RDN in the smooth region are more faithful to the clean images (e.g., smooth regions in "119082" and "img_061").

2.6.3.2 Color Image Denoising

We generate noisy color images by adding AWGN noise to clean RGB images with different noise levels $\sigma = 10$, 30, 50, and 70. The PSNR results are listed in Table 2.7. We apply gray image denoising methods (e.g., MemNet [21]) for color image denoising channel by channel. Lager gains over MemNet [21] of our RDN indicate that denoising color images jointly perform better than

Figure 2.13: Visual denoised results of the real noisy image "Dog". The noise level of R, G, B channels are estimated as 16.8, 17.0, and 16.6 respectively.

denoising each channel separately. Take σ=50 as an example, our RDN obtains 0.56 dB, 0.35, and 1.24 dB improvements over FFDNet [81] on three test sets respectively. Residual learning and dense feature fusion allows RDN to go wider and deeper, obtain hierarchical features, and achieve better performance.

We also show color image denoising visual results in Figure 2.12. CBM3D [119] tends to produce artifacts along the edges. TNRD [80] produces artifacts in the smooth area and is unable to recover clear edges. RED [39], DnCNN [40], MemNet [21], IRCNN [37], and FFDNet [81] could produce blurring artifacts along edges (e.g., the structural lines in "img_039"). Because RED [39] and MemNet [21] were designed for gray image denoising. In our experiments on color image denoising, we conduct RED [39] and MemNet [21] in each channel. Although DnCNN [40], IRCNN [37], and FFDNet [81] directly denoise noisy color images in three channels, they either fail to recover sharp edges and clean smooth area. In contrast, our RDN can recover shaper edges and cleaner smooth area.

2.6.3.3 Real-World Color Image Denoising

To further demonstrate the effectiveness of our proposed RDN, we compare it with recent leading method MCWNNM [121] on real noisy image. Following the same settings as [121], we estimate the noise levels $(\sigma_r, \sigma_g, \sigma_b)$ via some noise estimation methods [122]. Inspired by [121], we finetune RDN with the noise level σ=$\sqrt{\left(\sigma_r^2 + \sigma_g^2 + \sigma_b^2\right)/3}$. Due to limited space, we only show the denoised results on the real noisy image "Dog" (with 192$\times$192 pixels) [123], which doesn't have ground truth.

We show real-world color image denoising in Figure 2.13. Although MCWNNM considers the specific noise levels of each channel, its result still suffers from over-smoothing artifacts and loses some details (e.g., the moustache). In contrast, our RDN handles the noise better and preserves more details and shaper edges, which indicates that RDN is also suitable for real applications.

Table 2.8: Quantitative results about image compression artifact reduction. Best and second best results are highlighted and underlined

Dataset	Quality	JPEG		SA-DCT [113]		ARCNN [38]		TNRD [80]		DnCNN [40]		RDN (ours)		RDN+ (ours)	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
LIVE1	10	27.77	0.7905	28.65	0.8093	28.98	0.8217	29.15	0.8111	29.19	0.8123	29.67	0.8247	**29.70**	**0.8252**
	20	30.07	0.8683	30.81	0.8781	31.29	0.8871	31.46	0.8769	31.59	0.8802	32.07	0.8882	**32.10**	**0.8886**
	30	31.41	0.9000	32.08	0.9078	32.69	0.9166	32.84	0.9059	32.98	0.9090	33.51	0.9153	**33.54**	**0.9156**
	40	32.35	0.9173	32.99	0.9240	33.63	0.9306	N/A	N/A	33.96	0.9247	34.51	0.9302	**34.54**	**0.9304**
Classic5	10	27.82	0.7800	28.88	0.8071	29.04	0.8111	29.28	0.7992	29.40	0.8026	30.00	0.8188	**30.03**	**0.8194**
	20	30.12	0.8541	30.92	0.8663	31.16	0.8694	31.47	0.8576	31.63	0.8610	32.15	0.8699	**32.19**	**0.8704**
	30	31.48	0.8844	32.14	0.8914	32.52	0.8967	32.78	0.8837	32.91	0.8861	33.43	0.8930	**33.46**	**0.8932**
	40	32.43	0.9011	33.00	0.9055	33.34	0.9101	N/A	N/A	33.77	0.9003	34.27	0.9061	**34.29**	**0.9063**

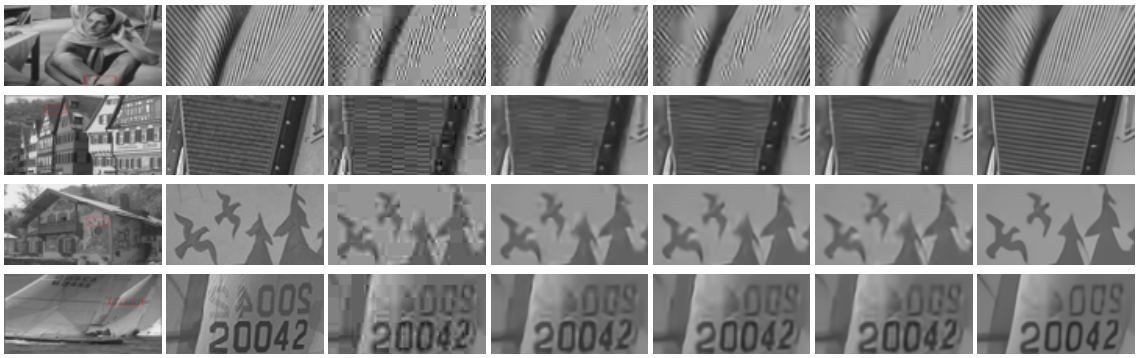


Figure 2.14: Image compression artifacts reduction results with JPEG quality q = 10.

2.6.4 Image Compression Artifact Reduction

We further apply our RDN to reduce image compression artifacts. We compare our RDN with SA-DCT [113], ARCNN [38], TNRD [80], and DnCNN [40]. We use Matlab JPEG encoder [115] to generate compressed test images from LIVE1 [112] and Classic5 [113]. Four JPEG quality settings q = 10, 20, 30, 40 are used in Matlab JPEG encoder. Here, we only focus on the

Table 2.9: PSNR (dB)/SSIM results about image deblurring.

Set	McMaster18		Kodak24		Urban100	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Blurry	27.00	0.7817	26.09	0.7142	22.38	0.6732
IRCNN [37]	32.50	0.8961	30.40	0.8513	27.70	0.8577
RDN (ours)	33.60	0.9157	30.88	0.8718	29.34	0.8886

Urban100: img_012 | HQ | Blurry | IRCNN [37] | RDN (ours)

Figure 2.15: Visual results on image deblurring.

compression artifact reduction (CAR) of Y channel (in YCbCr space) to keep fair comparison with other methods.

We report PSNR/SSIM values in Table 2.8. As we can see, our RDN and RDN+ achieve higher PSNR and SSIM values on LIVE1 and Classic5 with all JPEG qualities than other compared methods. Taking $q = 10$ as an example, our RDN achieves 0.48 dB and 0.60 dB improvements over DnCNN [40] in terms of PSNR. Even in such a challenging case (very low compression quality), our RDN can still obtain great performance gains over others. Similar improvements are also significant for other compression qualities. These comparisons further demonstrate the effectiveness of our proposed RDN.

Visual comparisons are further shown in Figure 2.14, where we provide comparisons under very low image quality (q=10). Although ARCNN [38], TNRD [80], and DnCNN [40] can remove blocking artifacts to some degree, they also over-smooth some details (e.g., 1st and 2nd rows in Figure 2.14) and cannot deeply remove the compression artifacts around content structures (e.g., 3rd and 4th rows in Figure 2.14). While, RDN has stronger network representation ability to distinguish compression artifacts and content information better. As a result, RDN recovers more details with consistent content structures.

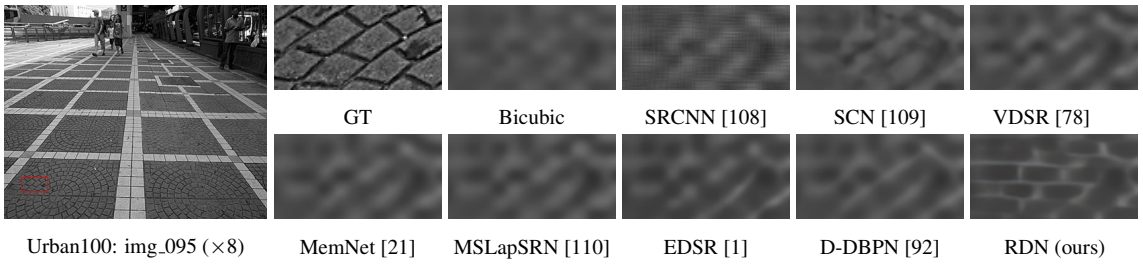


Figure 2.16: Failure cases for image super-resolution ($\times 8$).

2.6.5 Image Deblurring

A common practice to synthesize blurry images is first apply a blur kernel and then add Gausian noise with noise level σ [37, 36]. There are several blur kernels, such as Gaussian and motion blur kernels. Here, we focus on the commonly-used 25×25 Gaussian blur kernel of standard deviation 1.6. The additive Gaussian noise ($\sigma = 2$) is added to the blurry images.

We compare with IRCNN [37] and use McMaster18 [37], Kodak24, and Urban100 as test sets. We provide quantitative results in Table 2.9, where our RDN achieves large improvements over IRCNN for each test set. We further show visual results in Figure 2.15, where IRCNN still outputs some blurry structures. While, our RDN reconstructs much sharper edges and tiny details. These quantitative and qualitative comparisons demonstrate that our RDN also performs well for image deblurring.

2.7 Discussions

Here, we give a brief view of the benefits and limitations of our RDN and challenges in image restoration.

Benefits of RDN. RDB serves as the basic build module in RDN, which takes advantage of local and global feature fusion, obtaining very powerful representational ability. RDN uses less network parameters than residual network while achieves better performance than dense network, leading to a good tradeoff between model size and performance. RDN can be directly applied or generilized to several image restoration tasks with promissing performance.

Limitations of RDN. In some challenging cases (e.g., large scaling factor), RDN may fail to obtain proper details. As shown in Figure 2.16, although other methods fail to recover proper structures, our RDN cannot generates right structures either. The main reasons of this failure case may be that our RDN cannot recover similar textures based on the limited input information. Instead,

RDN would generate most likely texture patterns learned from the training data.

Challenges in Image Restoration. Extreme cases make the image restoration tasks much harder, such as very large scaling factors for image SR, heavy noise for image DN, low JPEG quality in image CAR, and heavy blurring artifacts in image deblurring. Complex desegregation processes in the real world make it difficult for us to formulate the degradation process. Then it may make the data preparation and network training harder.

Future Works. We believe that RDN can be further applied to other image restoration tasks, such as image demosaicing, derain, and dehazing. On the other hand, it's worth to further improve the performance of RDN. As indicated in Figure 2.16, RDN generates better local structures than others, while the global recovered structures are wrong. How to learn stronger local and global feature representations is worth to investigate. Plus, RDN may further benefit from adversarial training, which may help to alleviate some blurring and over-smoothing artifacts.

2.8 Summary

In this chapter, based on our proposed residual dense block (RDB), we propose deep residual dense network (RDN) for image restoration. RDB allows direct connections from preceding RDB to each Conv layer of current RDB, which results in a contiguous memory (CM) mechanism. We proposed LFF to adaptively preserve the information from the current and previous RDBs. With the usage of LRL, the flow of gradient and information can be further improved and training wider network becomes more stable. Extensive benchmark and real-world evaluations well demonstrate that our RDN achieves superiority over state-of-the-art methods for several image restoration tasks.

Chapter 3

Residual Channel Attention Network

3.1 Background and Motivation

We address the problem of reconstructing an accurate high-resolution (HR) image given its low-resolution (LR) counterpart, usually referred as single image super-resolution (SR) [124]. Image SR is used in various computer vision applications, ranging from security and surveillance imaging [68], medical imaging [69] to object recognition [125]. However, image SR is an ill-posed problem, since there exists multiple solutions for any LR input. To tackle such an inverse problem, numerous learning based methods have been proposed to learn mappings between LR and HR image pairs.

Recently, deep convolutional neural network (CNN) based methods [108, 101, 109, 78, 100, 84, 110, 125, 21, 1, 79, 92, 86] have achieved significant improvements over conventional SR methods. Among them, Dong *et al*. [76] proposed SRCNN by firstly introducing a three-layer CNN for image SR. Kim *et al*. increased the network depth to 20 in VDSR [78] and DRCN [82], achieving notable improvements over SRCNN. Network depth was demonstrated to be of central importance for many visual recognition tasks, especially when He at al. [19] proposed residual net (ResNet), which reaches 1,000 layers with residual blocks. Such effective residual learning strategy was then introduced in many other CNN-based image SR methods [100, 21, 103, 1, 125]. Lim *et al*. [1] built a very wide network EDSR and a very deep one MDSR (about 165 layers) by using simplified residual blocks. The great improvements on performance of EDSR and MDSR indicate that the depth of representation is of crucial importance for image SR. However, to the best of our knowledge, simply stacking residual blocks to construct deeper networks can hardly obtain better improvements. Whether deeper networks can further contribute to image SR and how to construct very deep trainable

networks remains to be explored.

On the other hand, most recent CNN-based methods [108, 101, 109, 78, 100, 84, 110, 125, 21, 1, 79] treat channel-wise features equally, which lacks flexibility in dealing with different types of information (e.g., low- and high-frequency information). Image SR can be viewed as a process, where we try to recover as more high-frequency information as possible. The LR images contain most low-frequency information, which can directly forwarded to the final HR outputs and don't need too much computation. While, the leading CNN-based methods (e.g., EDSR [1]) would extract features from the original LR inputs and treat each channel-wise feature equally. Such process would wastes unnecessary computations for abundant low-frequency features, lacks discriminative learning ability across feature channels, and finally hinders the representational power of deep networks.

To practically resolve these problems, we propose a residual channel attention network (RCAN) to obtain very deep trainable network and adaptively learn more useful channel-wise features simultaneously. To ease the training of very deep networks (e.g., over 400 layers), we propose residual in residual (RIR) structure, where the residual group (RG) serves as the basic module and long skip connection (LSC) allows residual learning in a coarse level. In each RG module, we stack several simplified residual block [1] with short skip connection (SSC). The long and short skip connection as well as the short-cut in residual block allow abundant low-frequency information to be bypassed through these identity-based skip connections, which can ease the flow of information. To make a further step, we propose channel attention (CA) mechanism to adaptively rescale each channel-wise feature by modeling the interdependencies across feature channels. Such CA mechanism allows our proposed network to concentrate on more useful channels and enhance discriminative learning ability. As shown in Figure 3.1, our RCAN achieves better visual SR result compared with state-of-the-art methods.

Overall, our contributions are three-fold: (1) We propose the very deep residual channel attention networks (RCAN) for highly accurate image SR. Our RCAN can reach much deeper than previous CNN-based methods and obtains much better SR performance. (2) We propose residual in residual (RIR) structure to construct very deep trainable networks. The long and short skip connections in RIR help to bypass abundant low-frequency information and make the main network learn more effective information. (3) We propose channel attention (CA) mechanism to adaptively rescale features by considering interdependencies among feature channels. Such CA mechanism further improves the representational ability of the network.

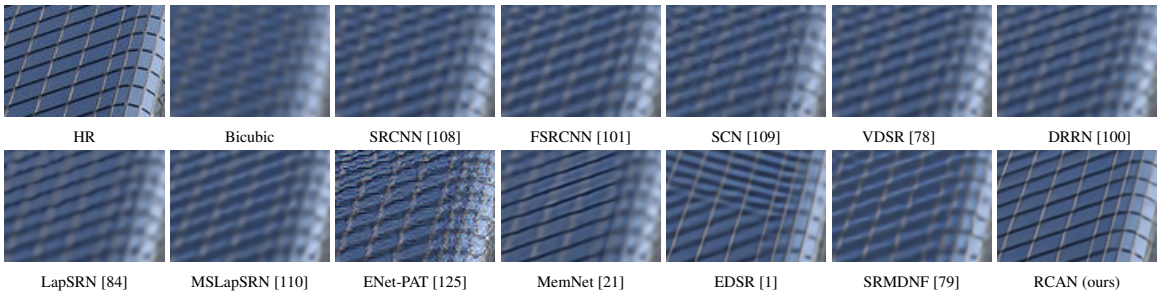


Figure 3.1: Visual results with Bicubic (BI) degradation (4×) on "img_074" from Urban100.

3.2 Related Work

Numerous image SR methods have been studied in the computer vision community [108, 101, 109, 78, 100, 84, 110, 125, 21, 1, 79, 61]. Attention mechanism is popular in high-level vision tasks, but is seldom investigated in low-level vision applications [31]. Due to space limitation, here we focus on works related to CNN-based methods and attention mechanism.

Deep CNN for SR. The pioneer work was done by Dong *et al*. [76], who proposed SRCNN for image SR and achieved superior performance against previous works. By introducing residual learning to ease the training difficulty, Kim *et al*. proposed VDSR [78] and DRCN [82] with 20 layers and achieved significant improvement in accuracy. Tai *et al*. later introduced recursive blocks in DRRN [100] and memory block in MemNet [21]. These methods would have to first interpolate the LR inputs to the desired size, which inevitably loses some details and increases computation greatly.

Extracting features from the original LR inputs and upscaling spatial resolution at the network tail then became the main choice for deep architecture. A faster network structure FSRCNN [101] was proposed to accelerate the training and testing of SRCNN. Ledig *et al*. [103] introduced ResNet [19] to construct a deeper network, SRResNet, for image SR. They also proposed SRGAN with perceptual losses [50] and generative adversarial network (GAN) [126] for photo-realistic SR. Such GAN based model was then introduced in EnhanceNet [125], which combines automated texture synthesis and perceptual loss. Although SRGAN and Enhancenet can alleviate the blurring and oversmoothing artifacts to some degree, their predicted results may not be faithfully reconstructed and produce unpleasing artifacts. By removing unnecessary modules in conventional residual networks, Lim *et al*. [1] proposed EDSR and MDSR, which achieve significant improvement. However, most of these methods have limited network depth, which has demonstrated to be very

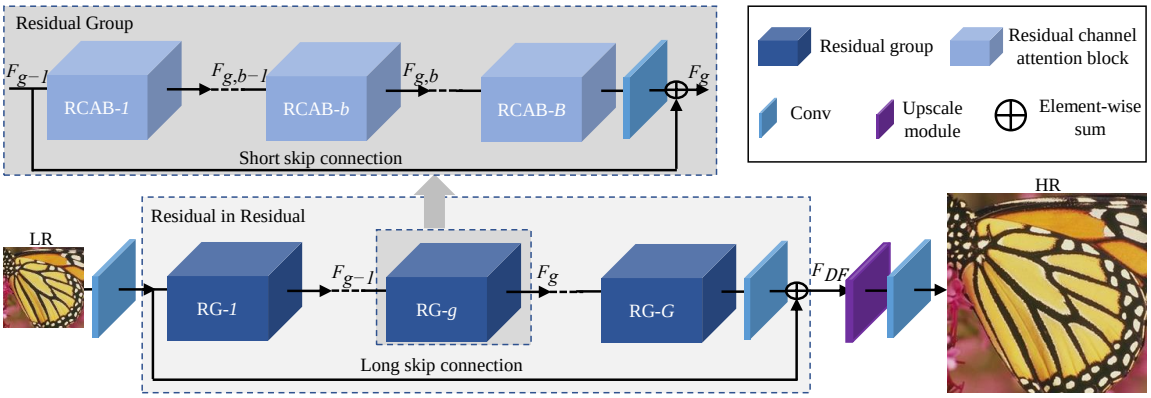


Figure 3.2: Network architecture of our residual channel attention network (RCAN).

important in visual recognition tasks [19] and can reach to about 1,000 layers. Simply stacking residual blocks in MDSR [1], very deep networks can hardly achieved improvements. Furthermore, most of these methods treat the channel-wise features equally, hindering better discriminative ability for different types of features.

Attention mechanism. Generally, attention can be viewed as a guidance to bias the allocation of available processing resources towards the most informative components of an input [31]. Recently, tentative works have been proposed to apply attention into deep neural networks [127, 128, 31], ranging from localization and understanding in images [129, 130] to sequence-based networks [131, 132]. It's usually combined with a gating function (e.g., sigmoid) to rescale the feature maps. Wang *et al*. [128] proposed residual attention network for image classification with a trunk-and-mask attention mechanism. Hu *et al*. [31] proposed squeeze-and-excitation (SE) block to model channel-wise relationships to obtain significant performance improvement for image classification. However, few works have been proposed to investigate the effect of attention for low-level vision tasks (e.g., image SR).

In image SR, high-frequency channel-wise features are more informative for HR reconstruction. If our network pays more attention to such channel-wise features, it should be promising to obtain improvements. To investigate such mechanism in very deep CNN, we propose very deep residual channel attention networks (RCAN), which we will detail in next section.

3.3 Residual Channel Attention Network (RCAN)

3.3.1 Network Architecture

As shown in Figure 3.2, our RCAN mainly consists four parts: shallow feature extraction, residual in residual (RIR) deep feature extraction, upscale module, and reconstruction part. Let's denote I_{LR} and I_{SR} as the input and output of RCAN. As investigated in [103, 1], we use only one convolutional layer (Conv) to extract the shallow feature F_0 from the LR input

$$F_0 = H_{SF}\left(I_{LR}\right), \tag{3.1}$$

where $H_{SF}\left(\cdot\right)$ denotes convolution operation. F_0 is then used for deep feature extraction with RIR module. So we can further have

$$F_{DF} = H_{RIR}\left(F_0\right), \tag{3.2}$$

where $H_{RIR}\left(\cdot\right)$ denotes our proposed very deep residual in residual structure, which contains G residual groups (RG). To the best of our knowledge, our proposed RIR achieves the largest depth so far and provides very large receptive field size. So we treat its output as deep feature, which is then upscaled via a upscale module

$$F_{UP} = H_{UP}\left(F_{DF}\right), \tag{3.3}$$

where $H_{UP}\left(\cdot\right)$ and F_{UP} denote a upscale module and upscaled feature respectively.

There're several choices to serve as upscale modules, such as deconvolution layer (also known as transposed convolution) [101], nearest-neighbor upsampling + convolution [133], and ESPCN [102]. Such post-upscaling strategy has been demonstrated to be more efficient for both computation complexity and achieve higher performance than pre-upscaling SR methods (e.g., DRRN [100] and MemNet [21]). The upscaled feature is then reconstructed via one Conv layer

$$I_{SR} = H_{REC}\left(F_{UP}\right) = H_{RCAN}\left(I_{LR}\right), \tag{3.4}$$

where $H_{REC}\left(\cdot\right)$ and $H_{RCAN}\left(\cdot\right)$ denote the reconstruction layer and the function of our RCAN respectively.

Then RCAN is optimized with loss function. Several loss functions have been investigated, such as L_2 [108, 101, 109, 78, 100, 125, 21, 79, 92], L_1 [1, 84, 110, 86], perceptual and adversarial losses [125, 103]. To show the effectiveness of our RCAN, we choose to optimize same loss function

as previous works (e.g., L_1 loss function). Given a training set $\left\{I_{LR}^i, I_{HR}^i\right\}_{i=1}^N$, which contains N LR inputs and their HR counterparts. The goal of training RCAN is to minimize the L_1 loss function

$$L\left(\Theta\right) = \frac{1}{N}\sum_{i=1}^{N}\left\|H_{RCAN}\left(I_{LR}^i\right) - I_{HR}^i\right\|_1, \tag{3.5}$$

where Θ denotes the parameter set of our network. The loss function is optimized by using stochastic gradient descent. More details of training would be shown in Section 3.4.1. As we choose the shallow feature extraction $H_{SF}\left(\cdot\right)$, upscaling module $H_{UP}\left(\cdot\right)$, and reconstruction part $H_{UP}\left(\cdot\right)$ as similar as previous works (e.g., EDSR [1] and RDN [86]), we pay more attention to our proposed RIR, CA, and the basic module RCAB.

3.3.2 Residual in Residual (RIR)

We now give more details about our proposed RIR structure (see Figure 3.2), which contains G residual groups (RG) and long skip connection (LSC). Each RG further contains B residual channel attention blocks (RCAB) with short skip connection (SSC). Such residual in residual structure allows to train very deep CNN (over 400 layers) for image SR with high performance.

It has been demonstrated that stacked residual blocks and LSC can be used to construct deep CNN in [1]. In visual recognition, residual blocks [19] can be stacked to achieve more than 1,000-layer trainable networks. However, in image SR, very deep network built in such way would suffer from training difficulty and can hardly achieve more performance gain. Inspired by previous works in SRRestNet [103] and EDSR [1], we proposed residual group (RG) as the basic module for deeper networks. A RG in the g-th group is formulated as

$$F_g = H_g\left(F_{g-1}\right) = H_g\left(H_{g-1}\left(\cdots H_1\left(F_0\right)\cdots\right)\right), \tag{3.6}$$

where H_g denotes the function of g-th RG. F_{g-1} and F_g are the input and output for g-th RG. We observe that simply stacking many RGs would fail to achieve better performance. To solve the problem, the long skip connection (LSC) is further introduced in RIR to stabilize the training of very deep network. LSC also makes better performance possible with residual learning via

$$F_{DF} = F_0 + W_{LSC}F_G = F_0 + W_{LSC}H_g\left(H_{g-1}\left(\cdots H_1\left(F_0\right)\cdots\right)\right), \tag{3.7}$$

where W_{LSC} is the weight set to the Conv layer at the tail of RIR. The bias term is omitted for simplicity. LSC can not only ease the flow of information across RGs, but only make it possible for RIR to learning residual information in a coarse level.

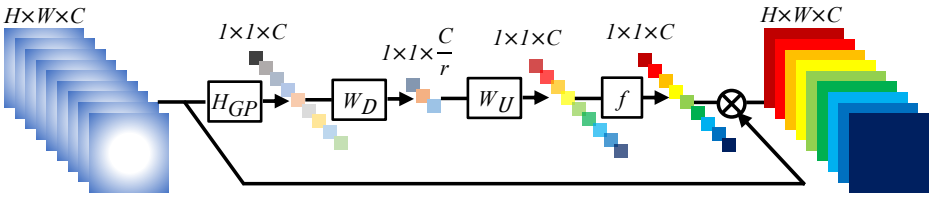


Figure 3.3: Channel attention (CA). $\otimes$ denotes element-wise product.

As discussed in Section 3.1, there are lots of abundant information in the LR inputs and features and the goal of SR network is to recover more useful information. The abundant low-frequency information can be bypassed through identity-based skip connection. To make a further step towards residual learning, we stack B residual channel attention blocks in each RG. The b-th residual channel attention block (RCAB) in g-th RG can be formulated as

$$F_{g,b} = H_{g,b}\left(F_{g,b-1}\right) = H_{g,b}\left(H_{g,b-1}\left(\cdots H_{g,1}\left(F_{g-1}\right)\cdots\right)\right), \tag{3.8}$$

where $F_{g,b-1}$ and $F_{g,b}$ are the input and output of the b-th RCAB in g-th RG. The corresponding function is denoted with $H_{g,b}$. To make the main network pay more attention to more informative features, a short skip connection (SSC) is introduced to obtain the block output via

$$F_g = F_{g-1} + W_g F_{g,B} = F_{g-1} + W_g H_{g,B}\left(H_{g,B-1}\left(\cdots H_{g,1}\left(F_{g-1}\right)\cdots\right)\right), \tag{3.9}$$

where W_g is the weight set to the Conv layer at the tail of g-th RG. The SSC further allows the main parts of network to learn residual information. With LSC and SSC, more abundant low-frequency information is easier bypassed in the training process. To make a further step towards more discriminative learning, we pay more attention to channel-wise feature rescaling with channel attention.

3.3.3 Channel Attention (CA)

Previous CNN-based SR methods treat LR channel-wise features equally, which is not flexible for the real cases. In order to make the network focus on more informative features, we exploit the interdependencies among feature channels, resulting in a channel attention (CA) mechanism (see Figure 3.3).

How to generate different attention for each channel-wise feature is a key step. Here we mainly have two concerns: First, information in the LR space has abundant low-frequency and valuable high-frequency components. The low-frequency parts seem to be more complanate. The

high-frequency components would usually be regions, being full of edges, texture, and other details. On the other hand, each filter in Conv layer operates with a local receptive field. Consequently, the output after convolution is unable to exploit contextual information outside of the local region.

Based on these analyses, we take the channel-wise global spatial information into a channel descriptor by using global average pooling. As shown in Figure 3.3, let $X = [x_1, \cdots, x_c, \cdots, x_C]$ be an input, which has C feature maps with size of $H \times W$. The channel-wise statistic $z \in \mathbb{R}^C$ can be obtained by shrinking X through spatial dimensions $H \times W$. Then the c-th element of z is determined by

$$z_c = H_{GP}(x_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x_c(i, j), \tag{3.10}$$

where $x_c(i, j)$ is the value at position (i, j) of c-th feature x_c. $H_{GP}(\cdot)$ denotes the global pooling function. Such channel statistic can be viewed as a collection of the local descriptors, whose statistics contribute to express the whole image [31]. Except for global average pooling, more sophisticated aggregation techniques could also be introduced here.

To fully capture channel-wise dependencies from the aggregated information by global average pooling, we introduce a gating mechanism. As discussed in [31], the gating mechanism should meet two criteria: First, it must be able to learn nonlinear interactions between channels. Second, as multiple channel-wise features can be emphasized opposed to one-hot activation, it must learn a non-mututually-exclusive relationship. Here, we opt to exploit simple gating mechanism with sigmoid function

$$s = f(W_U \delta(W_D z)), \tag{3.11}$$

where $f(\cdot)$ and $\delta(\cdot)$ denote the sigmoid gating and ReLU [134] function, respectively. W_D is the weight set of a Conv layer, which acts as channel-downscaling with reduction ratio r. After being activated by ReLU, the low-dimension signal is then increased with ratio r by a channel-upscaling layer, whose weight set is W_U. Then we obtain the final channel statistics s, which is used to rescale the input x_c

$$\widehat{x}_c = s_c \cdot x_c, \tag{3.12}$$

where s_c and x_c are the scaling factor and feature map in the c-th channel. With channel attention, the residual component in the RCAB is adaptively rescaled.

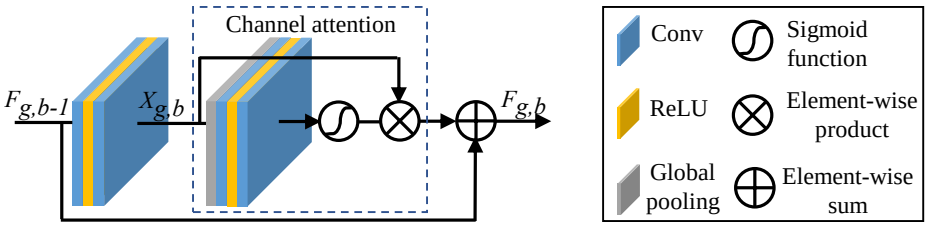


Figure 3.4: Residual channel attention block (RCAB).

3.3.4 Residual Channel Attention Block (RCAB)

As discussed above, residual groups and long skip connection allow the main parts of network to focus on more informative components of the LR features. Channel attention extracts the channel statistic among channels to further enhance the discriminative ability of the network.

At the same time, inspired by the success of residual blocks (RB) in [1], we integrate CA into RB and propose residual channel attention block (RCAB) (see Figure 3.4). For the b-th RB in g-th RG, we have

$$F_{g,b} = F_{g,b-1} + R_{g,b}\left(X_{g,b}\right) \cdot X_{g,b}, \tag{3.13}$$

where $R_{g,b}$ denotes the function of channel attention. $F_{g,b}$ and $F_{g,b-1}$ are the input and output of RCAB, which learns the residual $X_{g,b}$ from the input. The residual component is mainly obtained by two stacked Conv layers

$$X_{g,b} = W_{g,b}^{2}\delta\left(W_{g,b}^{1}F_{g,b-1}\right), \tag{3.14}$$

where $W_{g,b}^{1}$ and $W_{g,b}^{2}$ are weight sets the two stacked Conv layers in RCAB.

We further show the relationships between our proposed RCAB and residual block (RB) in [1]. We find that the RBs used in MDSR and EDSR [1] can be viewed as special cases of our RCAB. For RB in MDSR, there is no rescaling operation. It is the same as RCAB, where we set $R_{g,b}\left(\cdot\right)$ as constant 1. For RB with constant rescaling (e.g., 0.1) in EDSR, it is the same as RCAB with $R_{g,b}\left(\cdot\right)$ set to be 0.1. Although the channel-wise feature rescaling is introduced to train a very wide network, the interdependencies among channels are not considered in EDSR. In these cases, the CA is not considered.

Based on residual channel attention block (RCAB) and RIR structure, we construct a very deep RCAN for highly accurate image SR and achieve notable performance improvements over previous leading methods. More discussions about the effects of each proposed component are shown in Section 3.4.2.

Table 3.1: Investigations of RIR (including LSC and SSC) and CA. We observe the best PSNR (dB) values on Set5 ($2\times$) in 5×10^4 iterations.

Residual in Residual (RIR)	LSC	✗	✓	✗	✓	✗	✓	✗	✓
	SSC	✗	✗	✓	✓	✗	✗	✓	✓
Channel attention (CA)		✗	✗	✗	✗	✓	✓	✓	✓
PSNR on Set5 ($2\times$)		37.45	37.77	37.81	37.87	37.52	37.85	37.86	37.90

3.3.5 Implementation Details

Now we specify the implementation details of our proposed RCAN. We set RG number as G=10 in the RIR structure. In each RG, we set RCAB number as 20. We set 3×3 as the size of all Conv layers except for that in the channel-downscaling and channel-upscaling, whose kernel size is 1×1. For Conv layers with kernel size 3×3, zero-padding strategy is used to keep size fixed. Conv layers in shallow feature extraction and RIR structure have C=64 filters, except for that in the channel-downscaling. Conv layer in channel-downscaling has $\frac{C}{r}$=4 filters, where the reduction ratio r is set as 4. For upscaling module $H_{UP}(\cdot)$, we follow [102, 1, 86] and use ESPCNN [102] to upscale the coarse resolution features to fine ones. The final Conv layer has 3 filters, as we output color images. While, our network can also process gray images.

3.4 Experiments

3.4.1 Settings

We clarify the experimental settings about datasets, degradation models, evaluation metric, and training settings.

Datasets and degradation models. Following [91, 1, 86, 79], we use 800 training images from DIV2K dataset [91] as training set. For testing, we use five standard benchmark datasets: Set5 [105], Set14 [106], B100 [107], Urban100 [61], and Manga109 [111]. We conduct experiments with Bicubic (BI) and blur-downscale (BD) degradation models [37, 79, 86].

Evaluation metrics. The SR results are evaluated with PSNR and SSIM [114] on Y channel (i.e., luminance) of transformed YCbCr space. We also provide performance (e.g., top-1 and top-5 recognition errors) comparisons on object recognition by several leading SR methods.

Training settings. Data augmentation is performed on the 800 training images, which are randomly rotated by 90°, 180°, 270° and flipped horizontally. In each training batch, 16 LR

color patches with the size of 48×48 are extracted as inputs. Our model is trained by ADAM optimizor [116] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The initial leaning rate is set to 10^{-4} and then decreases to half every 2×10^5 iterations of back-propagation. We use PyTorch [135] to implement our models with a Titan Xp GPU.[1]

3.4.2 Effects of RIR and CA

We study the effects of residual in residual (RIR) and channel attention (CA).

Residual in residual (RIR). To demonstrate the effect of our proposed residual in residual structure, we remove long skip connection (LSC) or/and short skip connection (SSC) from very deep networks. Specifically, we set the number of residual block as 200, namely 10 residual groups, resulting in very deep networks with over 400 Conv layers. In Table 3.1, when both LSC and SSC are removed, the PSNR value on Set5 ($\times 2$) is relatively low, no matter channel attention (CA) is used or not. For example, in the first column, the PSNR is 37.45 dB. After adding RIR, the performance reaches 37.87 dB. When CA is added, the performance can be improved from 37.52 dB to 37.90 by using RIR. This indicates that simply stacking residual blocks is not applicable to achieve very deep and powerful networks for image SR. The performance would increase with LSC or SSC and can obtain better results by using both of them. These comparisons show that LSC and SSC are essential for very deep networks. They also demonstrate the effectiveness of our proposed residual in residual (RIR) structure for very deep networks.

Channel attention (CA). We further show the effect of channel attention (CA) based on the observations and discussions above. When we compare the results of first 4 columns and last 4 columns, we find that networks with CA would perform better than those without CA. Benefitting from very large network depth, the very deep trainable networks can achieve a very high performance. It's hard to obtain further improvements from such deep networks, but we obtain improvements with CA. Even without RIR, CA can improve the performance from 37.45 dB to 37.52 dB. These comparisons firmly demonstrate the effectiveness of CA and indicate adaptive attentions to channel-wise features really improves the performance.

3.4.3 Results with Bicubic (BI) Degradation Model

We compare our method with 11 state-of-the-art methods: SRCNN [108], FSRCNN [101], SCN [109], VDSR [78], LapSRN [84], MemNet [21], EDSR [1], SRMDNF [79], D-DBPN [92],

[1]The RCAN source code is available at https://github.com/yulunzhang/RCAN.

Table 3.2: Quantitative results (BI model). Best and second best are **highlighted** and underlined.

Method	Scale	Set5		Set14		B100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	×2	33.66	0.9299	30.24	0.8688	29.56	0.8431	26.88	0.8403	30.80	0.9339
SRCNN [108]	×2	36.66	0.9542	32.45	0.9067	31.36	0.8879	29.50	0.8946	35.60	0.9663
FSRCNN [101]	×2	37.05	0.9560	32.66	0.9090	31.53	0.8920	29.88	0.9020	36.67	0.9710
VDSR [78]	×2	37.53	0.9590	33.05	0.9130	31.90	0.8960	30.77	0.9140	37.22	0.9750
LapSRN [84]	×2	37.52	0.9591	33.08	0.9130	31.08	0.8950	30.41	0.9101	37.27	0.9740
MemNet [21]	×2	37.78	0.9597	33.28	0.9142	32.08	0.8978	31.31	0.9195	37.72	0.9740
EDSR [1]	×2	38.11	0.9602	33.92	0.9195	32.32	0.9013	32.93	0.9351	39.10	0.9773
SRMDNF [79]	×2	37.79	0.9601	33.32	0.9159	32.05	0.8985	31.33	0.9204	38.07	0.9761
D-DBPN [92]	×2	38.09	0.9600	33.85	0.9190	32.27	0.9000	32.55	0.9324	38.89	0.9775
RDN [86]	×2	38.24	0.9614	34.01	0.9212	32.34	0.9017	32.89	0.9353	39.18	0.9780
RCAN (ours)	×2	38.27	0.9614	34.12	0.9216	32.41	0.9027	33.34	0.9384	39.44	0.9786
RCAN+ (ours)	×2	**38.33**	**0.9617**	**34.23**	**0.9225**	**32.46**	**0.9031**	**33.54**	**0.9399**	**39.61**	**0.9788**
Bicubic	×3	30.39	0.8682	27.55	0.7742	27.21	0.7385	24.46	0.7349	26.95	0.8556
SRCNN [108]	×3	32.75	0.9090	29.30	0.8215	28.41	0.7863	26.24	0.7989	30.48	0.9117
FSRCNN [101]	×3	33.18	0.9140	29.37	0.8240	28.53	0.7910	26.43	0.8080	31.10	0.9210
VDSR [78]	×3	33.67	0.9210	29.78	0.8320	28.83	0.7990	27.14	0.8290	32.01	0.9340
LapSRN [84]	×3	33.82	0.9227	29.87	0.8320	28.82	0.7980	27.07	0.8280	32.21	0.9350
MemNet [21]	×3	34.09	0.9248	30.00	0.8350	28.96	0.8001	27.56	0.8376	32.51	0.9369
EDSR [1]	×3	34.65	0.9280	30.52	0.8462	29.25	0.8093	28.80	0.8653	34.17	0.9476
SRMDNF [79]	×3	34.12	0.9254	30.04	0.8382	28.97	0.8025	27.57	0.8398	33.00	0.9403
RND [86]	×3	34.71	0.9296	30.57	0.8468	29.26	0.8093	28.80	0.8653	34.13	0.9484
RCAN (ours)	×3	34.74	0.9299	30.65	0.8482	29.32	0.8111	29.09	0.8702	34.44	0.9499
RCAN+ (ours)	×3	**34.85**	**0.9305**	**30.76**	**0.8494**	**29.39**	**0.8122**	**29.31**	**0.8736**	**34.76**	**0.9513**
Bicubic	×4	28.42	0.8104	26.00	0.7027	25.96	0.6675	23.14	0.6577	24.89	0.7866
SRCNN [108]	×4	30.48	0.8628	27.50	0.7513	26.90	0.7101	24.52	0.7221	27.58	0.8555
FSRCNN [101]	×4	30.72	0.8660	27.61	0.7550	26.98	0.7150	24.62	0.7280	27.90	0.8610
VDSR [78]	×4	31.35	0.8830	28.02	0.7680	27.29	0.0726	25.18	0.7540	28.83	0.8870
LapSRN [84]	×4	31.54	0.8850	28.19	0.7720	27.32	0.7270	25.21	0.7560	29.09	0.8900
MemNet [21]	×4	31.74	0.8893	28.26	0.7723	27.40	0.7281	25.50	0.7630	29.42	0.8942
EDSR [1]	×4	32.46	0.8968	28.80	0.7876	27.71	0.7420	26.64	0.8033	31.02	0.9148
SRMDNF [79]	×4	31.96	0.8925	28.35	0.7787	27.49	0.7337	25.68	0.7731	30.09	0.9024
D-DBPN [92]	×4	32.47	0.8980	28.82	0.7860	27.72	0.7400	26.38	0.7946	30.91	0.9137
RDN [86]	×4	32.47	0.8990	28.81	0.7871	27.72	0.7419	26.61	0.8028	31.00	0.9151
RCAN (ours)	×4	32.63	0.9002	28.87	0.7889	27.77	0.7436	26.82	0.8087	31.22	0.9173
RCAN+ (ours)	×4	**32.73**	**0.9013**	**28.98**	**0.7910**	**27.85**	**0.7455**	**27.10**	**0.8142**	**31.65**	**0.9208**
Bicubic	×8	24.40	0.6580	23.10	0.5660	23.67	0.5480	20.74	0.5160	21.47	0.6500
SRCNN [108]	×8	25.33	0.6900	23.76	0.5910	24.13	0.5660	21.29	0.5440	22.46	0.6950
SCN [109]	×8	25.59	0.7071	24.02	0.6028	24.30	0.5698	21.52	0.5571	22.68	0.6963
VDSR [78]	×8	25.93	0.7240	24.26	0.6140	24.49	0.5830	21.70	0.5710	23.16	0.7250
LapSRN [84]	×8	26.15	0.7380	24.35	0.6200	24.54	0.5860	21.81	0.5810	23.39	0.7350
MemNet [21]	×8	26.16	0.7414	24.38	0.6199	24.58	0.5842	21.89	0.5825	23.56	0.7387
MSLapSRN [110]	×8	26.34	0.7558	24.57	0.6273	24.65	0.5895	22.06	0.5963	23.90	0.7564
EDSR [1]	×8	26.96	0.7762	24.91	0.6420	24.81	0.5985	22.51	0.6221	24.69	0.7841
RCAN (ours)	×8	27.31	0.7878	25.23	0.6511	24.98	0.6058	23.00	0.6452	25.24	0.8029
RCAN+ (ours)	×8	**27.47**	**0.7913**	**25.40**	**0.6553**	**25.05**	**0.6077**	**23.22**	**0.6524**	**25.58**	**0.8092**

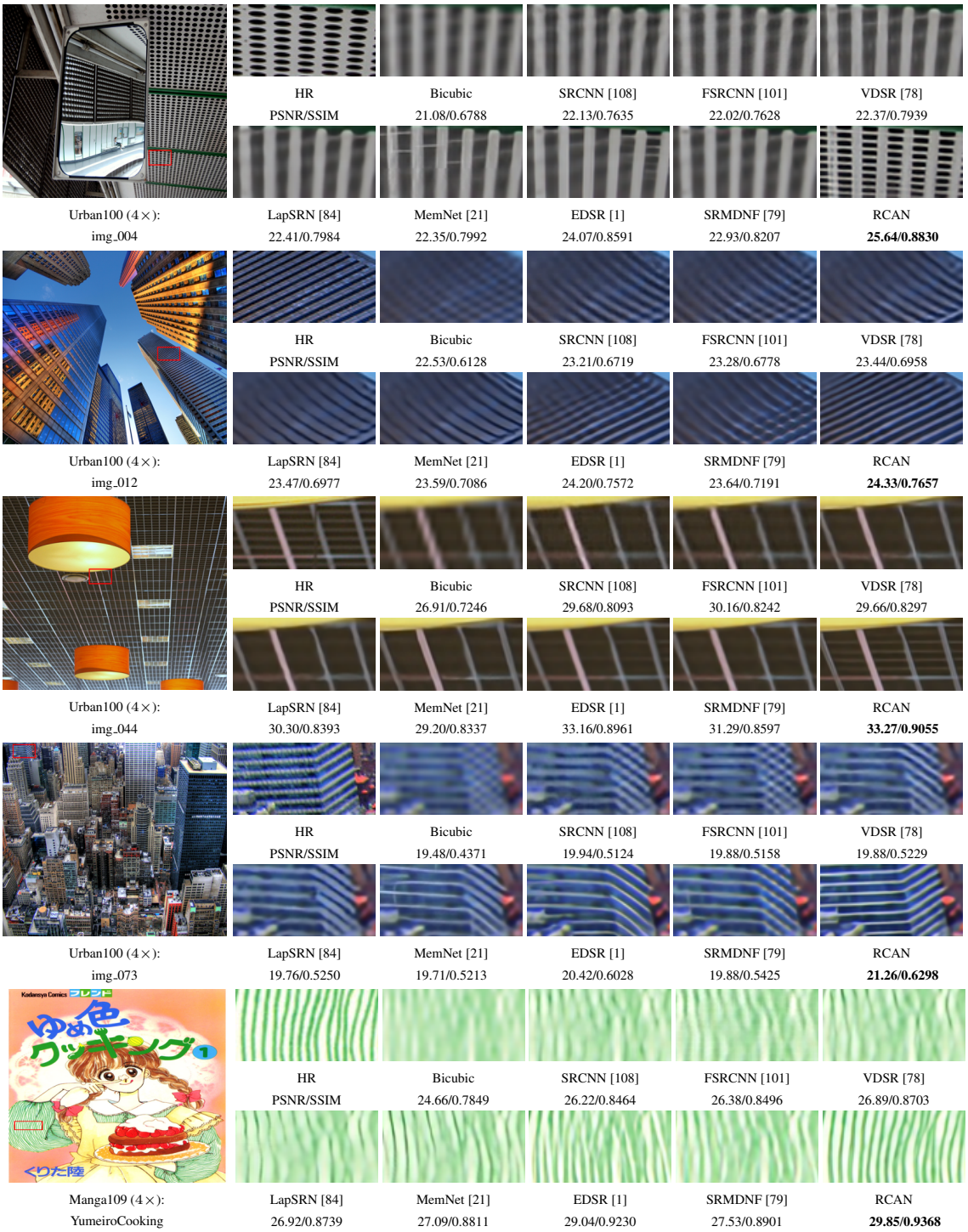


Figure 3.5: Visual comparison for $4\times$ SR with BI model on Urban100 and Manga109 datasets. The best results are **highlighted**.

and RDN [86]. Similar to [117, 1, 86], we also introduce self-ensemble strategy to further improve our RCAN and denote the self-ensembled one as RCAN+. More comparisons are provided in supplementary material.

Quantitative results by PSNR/SSIM. Table 3.2 shows quantitative comparisons for $\times 2$, $\times 3$, $\times 4$, and $\times 8$ SR. The results of D-DBPN [92] are cited from their paper. When compared with all previous methods, our RCAN+ performs the best on all the datasets with all scaling factors. Even without self-ensemble, our RCAN also outperforms other compared methods.

On the other hand, when the scaling factor become larger (e.g., 8), the gains of our RCAN over EDSR also becomes larger. For Urban100 and Manga109, the PSNR gains of RCAN over EDSR are 0.49 dB and 0.55 dB. EDSR has much larger number of parameters (43 M) than ours (16 M), but our RCAN obtains much better performance. Instead of constantly rescaling the features in EDSR, our RCAN adaptively rescales features with channel attention (CA). CA allows our network to further focus on more informative features. This observation indicates that very large network depth and CA improve the performance.

Visual results. In Figure 3.5, we show visual comparisons on scale $\times 4$. For image "img_004", we observe that most of the compared methods cannot recover the lattices and would suffer from blurring artifacts. In contrast, our RCAN can alleviate the blurring artifacts better and recover more details. For image "img_073", most of the compared methods produce blurring artifacts along the horizontal lines. What's worse, for the right parts of the cropped images, FSRCNN cannot recover lines. Other methods would generate some lines with wrong directions. Only our RCAN produces more faithful results. For image "YumeiroCooking", the cropped part is full of textures. As we can see, all the compared methods suffer from heavy blurring artifacts, failing to recover more details. While, our RCAN can recover them obviously, being more faithful to the ground truth. Such obvious comparisons demonstrate that networks with more powerful representational ability can extract more sophisticated features from the LR space.

To further illustrate the analyses above, we show visual comparisons for $8\times$ SR in Figure 3.6. For image "img_040", due to very large scaling factor, the result by Bicubic would lose the structures and produce different structures. This wrong pre-scaling result would also lead some state-of-the-art methods (e.g., SRCNN, VDSR, and MemNet) to generate totally wrong structures. Even starting from the original LR input, other methods cannot recover the right structure either. While, our RCAN can recover them correctly. For smaller details, like the net in image "TaiyouNiSmash", the tiny lines can be lost in the LR image. When the scaling factor is very large (e.g., 8), LR images contain very limited information for SR. Losing most high-frequency information makes it

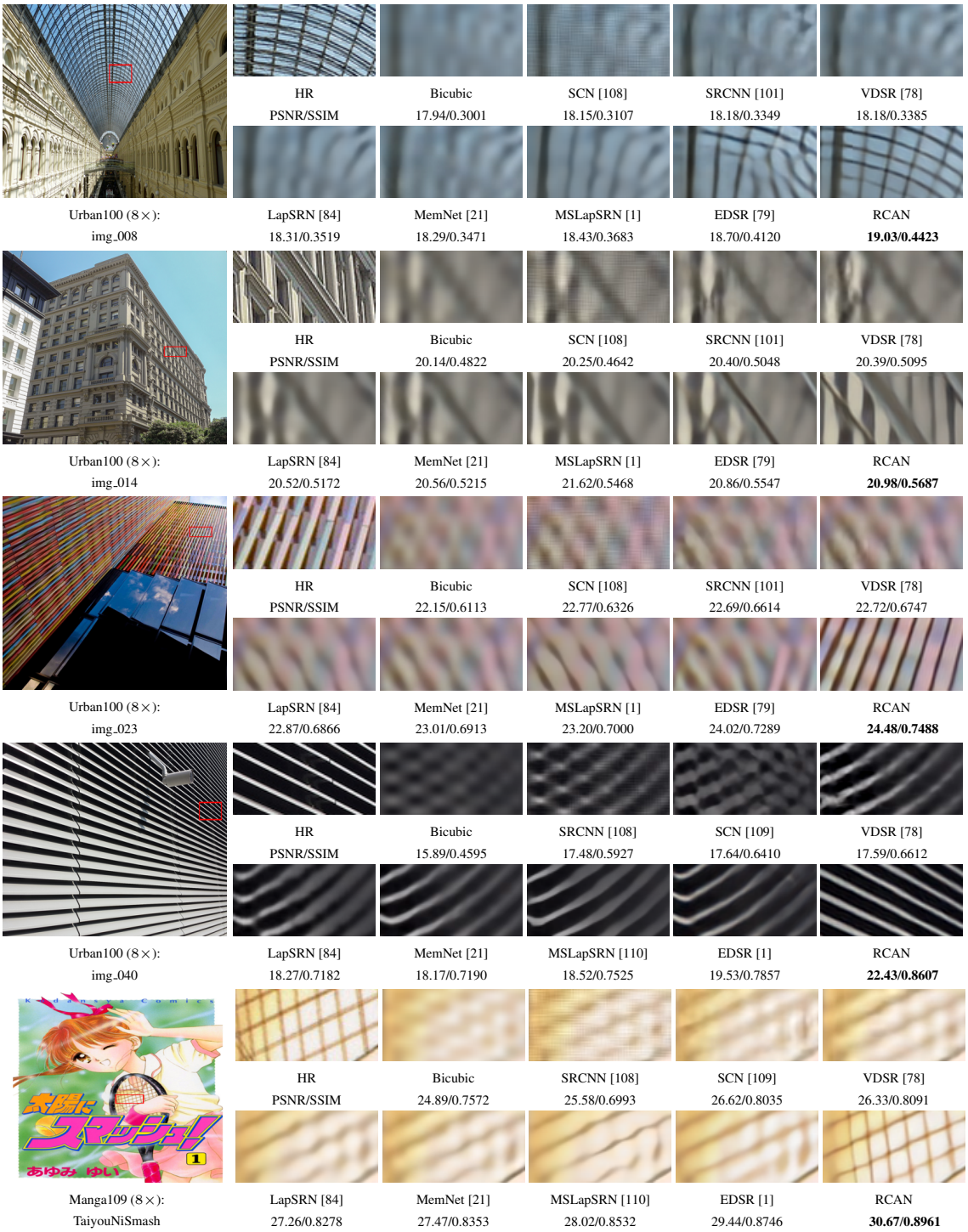


Figure 3.6: Visual comparison for 8× SR with BI model on Urban100 and Manga109 datasets. The best results are **highlighted**.

Table 3.3: Quantitative results with BD degradation model. Best and second best results are **highlighted** and underlined.

Method	Scale	Set5		Set14		B100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	×3	28.78	0.8308	26.38	0.7271	26.33	0.6918	23.52	0.6862	25.46	0.8149
SPMSR [75]	×3	32.21	0.9001	28.89	0.8105	28.13	0.7740	25.84	0.7856	29.64	0.9003
SRCNN [108]	×3	32.05	0.8944	28.80	0.8074	28.13	0.7736	25.70	0.7770	29.47	0.8924
FSRCNN [101]	×3	26.23	0.8124	24.44	0.7106	24.86	0.6832	22.04	0.6745	23.04	0.7927
VDSR [78]	×3	33.25	0.9150	29.46	0.8244	28.57	0.7893	26.61	0.8136	31.06	0.9234
IRCNN [37]	×3	33.38	0.9182	29.63	0.8281	28.65	0.7922	26.77	0.8154	31.15	0.9245
SRMDNF [79]	×3	34.01	0.9242	30.11	0.8364	28.98	0.8009	27.50	0.8370	32.97	0.9391
RDN [86]	×3	34.58	0.9280	30.53	0.8447	29.23	0.8079	28.46	0.8582	33.97	0.9465
RCAN (ours)	×3	34.70	0.9288	30.63	0.8462	29.32	0.8093	28.81	0.8647	34.38	0.9483
RCAN+ (ours)	×3	**34.83**	**0.9296**	**30.76**	**0.8479**	**29.39**	**0.8106**	**29.04**	**0.8682**	**34.76**	**0.9502**

very difficult for SR methods to reconstruct informative results. Most of compared methods cannot achieve this goal and produce serious blurring artifacts. However, our RCAN can obtain more useful information and produce finer results.

As we have discussed above, in BI degradation model, the reconstruction of high-frequency information is very important and difficult, especially with large scaling factor (e.g., 8). Our proposed RIR structure makes the main network learn residual information. Channel attention (CA) is further used to enhance the representational ability of the network by adaptively rescaling channel-wise features.

3.4.4 Results with Blur-downscale (BD) Degradation Model

We further apply our method to super-resolve images with blur-down (BD) degradation model, which is also commonly used recently [37, 79, 86].

Quantitative results by PSNR/SSIM. Here, we compare 3× SR results with 7 state-of-the-art methods: SPMSR [75], SRCNN [108], FSRCNN [101], VDSR [78], IRCNN [37], SRMDNF [79], and RDN [86]. As shown in Table 3.3, RDN has achieved very high performance on each dataset. While, our RCAN can obtain notable gains over RDN. Using self-ensemble, RCAN+ achieves even better results. Compared with fully using hierarchical features in RDN, a much deeper network with channel attention in RCAN achieves better performance. This comparison also indicates that there has promising potential to investigate much deeper networks for image SR.

Visual Results. We also show visual comparisons in Figure 3.7. For challenging details in images "img_062" and "img_078", most methods suffer from heavy blurring artifacts. RDN alleviates

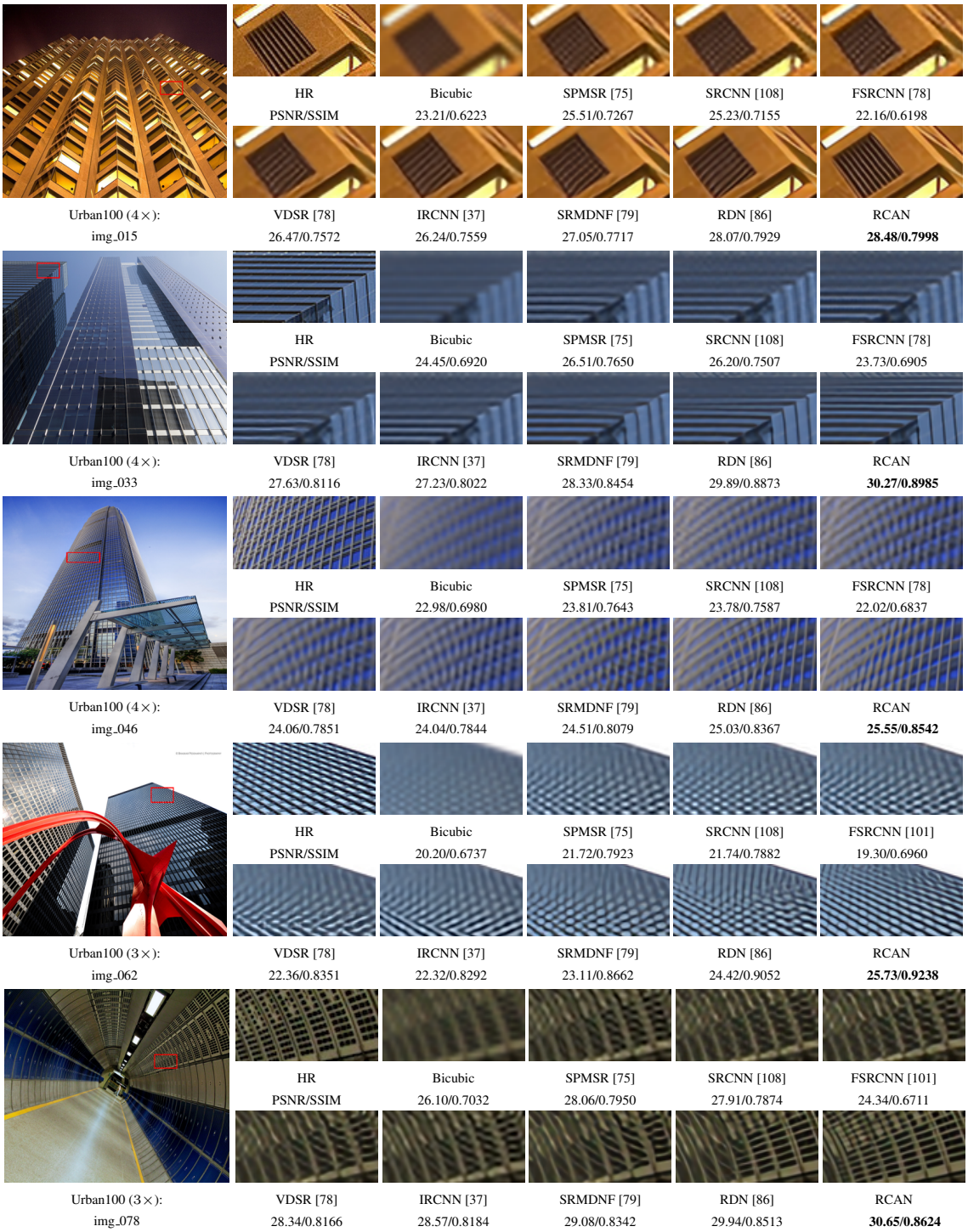
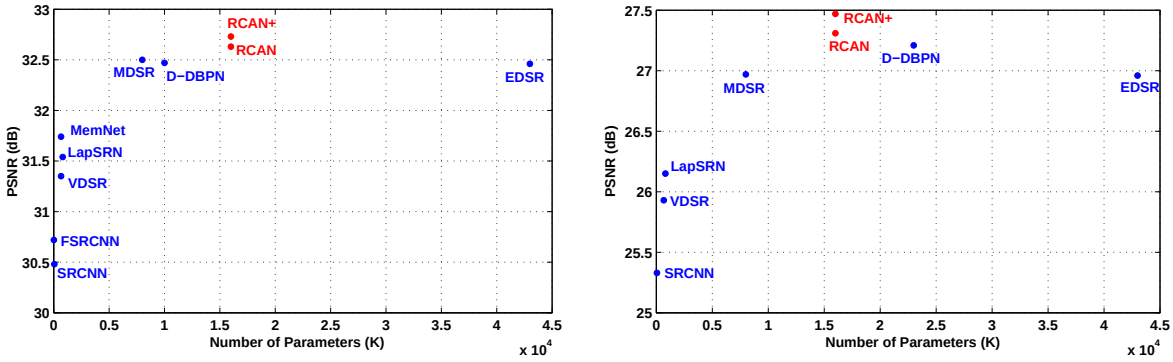


Figure 3.7: Visual comparison for 3× SR with BD model on Urban100 dataset. The best results are **highlighted**.

Table 3.4: ResNet object recognition performance on the first 1,000 images from ImageNet CLS-LOC validation dataset. The best results are **highlighted**

Evaluation	Bicubic	DRCN [82]	FSRCNN [101]	PSyCo [136]	ENet-E [125]	RCAN (ours)	Baseline
Top-1 error	0.506	0.477	0.437	0.454	0.449	**0.393**	0.260
Top-5 error	0.266	0.242	0.196	0.224	0.214	**0.167**	0.072



(a) Results on Set5 (4×)

(b) Results on Set5 (8×)

Figure 3.8: Performance and number of parameters. Results are evaluated on Set5.

it to some degree and can recover more details. In contrast, our RCAN obtains much better results by recovering more informative components. These comparisons indicate that very deep channel attention guided network would alleviate the blurring artifacts. It also demonstrates the strong ability of RCAN for BD degradation model.

3.4.5 Object Recognition Performance

Image SR also serves as pre-processing step for high-level visual tasks (e.g., object recognition). We evaluate the object recognition performance to further demonstrate the effectiveness of our RCAN.

Here we use the same settings as ENet [125]. We use ResNet-50 [19] as the evaluation model and use the first 1,000 images from ImageNet CLS-LOC validation dataset for evaluation. The original cropped 224×224 images are used for baseline and downscaled to 56×56 for SR methods. We use 4 stat-of-the-art methods (e.g., DRCN [82], FSRCNN [101], PSyCo [136], and ENet-E [125]) to upscale the LR images and then calculate their accuracies. As shown in Table 3.4, our RCAN achieves the lowest top-1 and top-5 errors. These comparisons further demonstrate the

highly powerful representational ability of our RCAN.

3.4.6 Model Size Analyses

We show comparisons about model size and performance in Figure 3.8. Although our RCAN is the deepest network, it has less parameter number than that of EDSR and RDN. Our RCAN and RCAN+ achieve higher performance, having a better tradeoff between model size and performance. It also indicates that deeper networks may be easier to achieve better performance than wider networks.

3.5 Summary

We propose very deep residual channel attention networks (RCAN) for highly accurate image SR. Specifically, the residual in residual (RIR) structure allows RCAN to reach very large depth with LSC and SSC. Meanwhile, RIR allows abundant low-frequency information to be bypassed through multiple skip connections, making the main network focus on learning high-frequency information. Furthermore, to improve ability of the network, we propose channel attention (CA) mechanism to adaptively rescale channel-wise features by considering interdependencies among channels. Extensive experiments on SR with BI and BD models demonstrate the effectiveness of our proposed RCAN. RCAN also shows promissing results for object recognition.

Chapter 4

Residual Non-local Attention Networks for Image Restoration

4.1 Background and Motivation

Image restoration aims to recover high-quality (HQ) images from their corrupted low-quality (LQ) observations and plays a fundamental role in various high-level vision tasks. It is a typical ill-posed problem due to the irreversible nature of the image degradation process. Some most widely studied image restoration tasks include image denoising, demosaicing, and compression artifacts reduction. By distinctively modelling the restoration process from LQ observations to HQ objectives, i.e., without assumption for a specific restoration task when modelling, these tasks can be uniformly addressed in the same framework. Recently, deep convolutional neural network (CNN) has shown extraordinary capability of modelling various vision problems, ranging from low-level (e.g., image denoising [40], compression artifacts reduction [38], and image super-resolution [78, 84, 21, 1, 79, 92, 137, 11, 96]) to high-level (e.g., image recognition [19]) vision applications.

Stacked denoising auto-encoder [138] is one of the best well-known CNN based model for image restoration. Dong *et al*. proposed SRCNN [76] for image super-resolution and ARCNN [38] for image compression artifacts reduction. Both SRCNN and ARCNN achieved superior performance against previous works. By introducing residual learning to ease the training difficulty for deeper network, Zhang *et al*. proposed DnCNN [40] for image denoising and compression artifacts reduction. The denoiser prior was lately introduced in IRCNN [37] for fast image restoration. Mao *et al*. proposed a very deep fully convolutional encoding-decoding framework with symmetric skip

connections for image restoration [39]. Tai *et al*. later proposed a very deep end-to-end persistent memory network (MemNet) for image restoration [21] and achieved promising results. These CNN based methods have demonstrated the great ability of CNN for image restoration tasks.

However, there are mainly three issues in the existing CNN based methods above. **First**, the receptive field size of these networks is relatively small. Most of them extract features in a local way with convolutional operation, which fails to capture the long-range dependencies between pixels in the whole image. A larger receptive field size allows to make better use of training inputs and more context information. This would be very helpful to capture the latent degradation model of LQ images, especially when the images suffer from heavy corruptions. **Second**, distinctive ability of these networks is also limited. Let's take image denoising as an example. For a noisy image, the noise may appear in both the plain and textural regions. Noise removal would be easier in the plain area than that in the textural one. It is desired to make the denoising model focus on textual area more. However, most previous denoising methods neglect to consider different contents in the noisy input and treat them equally. This would result in over-smoothed outputs and some textural details would also fail to be recovered. **Third**, all channel-wise features are treated equally in those networks. This naive treatment lacks flexibility in dealing with different types of information (e.g., low- and high-frequency information). For a set of features, some contain more information related to HQ image and the others may contain more information related to corruptions. The interdependencies among channels should be considered for more accurate image restoration.

To address the above issues, we propose the very deep residual non-local attention networks (RNAN) for high-quality image restoration. We design residual local and non-local attention blocks as the basic building modules for the very deep network. Each attention block consists of trunk and mask branches. We introduce residual block [19, 1] for trunk branch and extract hierarchical features. For mask branch, we conduct feature downscaling and upscaling with large-stride convolution and deconvolution to enlarge receptive field size. Furthermore, we incorporate non-local block in the mask branch to obtain residual non-local mixed attention. We apply RNAN for various restoration tasks, including image denoising, demosaicing, and compression artifacts reduction. Extensive experiments show that our proposed RNAN achieves state-of-the-art results compared with other recent leading methods in all tasks. To the best of our knowledge, this is the first time to consider residual non-local attention for image restoration problems.

The main contributions of this work are three-fold:

- We propose the very deep residual non-local networks for high-quality image restoration. The

powerful networks are based on our proposed residual local and non-local attention blocks, which consist of trunk and mask branches. The network obtains non-local mixed attention with non-local block in the mask branch. Such attention mechanis helps to learn local and non-local information from the hierarchical features.

- We propose residual non-local attention learning to train very deep networks by preserving more low-level features, being more suitable for image restoration. Using non-local low-level and high-level attention from the very deep network, we can pursue better network representational ability and finally obtain high-quality image restoration results.

- We demonstrate with extensive experiments that our RNAN is powerful for various image restoration tasks. RNAN achieves superior results over leading methods for image denoising, demosaicing, compression artifacts reduction, and super-resolution. In addition, RNAN achieves superior performance with moderate model size and performs very fast.

4.2 Related Work

Non-local prior. As a classical filtering algorithm, non-local means [139] is computed as weighted mean of all pixels of an image. Such operation allows distant pixels to contribute to the response of a position at a time. It was lately introduced in BM3D [118] for image denoising. Recently, [32] proposed non-local neural network by incorporating non-local operations in deep neural network for video classification. We can see that those methods mainly introduce non-local information in the trunk pipeline. [99] proposed non-local recurrent network for image restoration. However, in this paper, we mainly focus on learning non-local attention to better guide feature extraction in trunk branch.

Attention mechanisms. Generally, attention can be viewed as a guidance to bias the allocation of available processing resources towards the most informative components of an input [31]. Recently, tentative works have been proposed to apply attention into deep neural networks [128, 31]. It's usually combined with a gating function (e.g., sigmoid) to rescale the feature maps. [128] proposed residual attention network for image classification with a trunk-and-mask attention mechanism. [31] proposed squeeze-and-excitation (SE) block to model channel-wise relationships to obtain significant performance improvement for image classification. In all, these works mainly aim to guide the network pay more attention to the regions of interested. However, few works have been

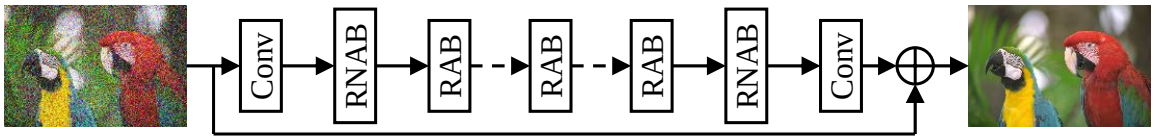


Figure 4.1: The framework of our proposed residual non-local attention network for image restoration. 'Conv', 'RNAB', and 'RAB' denote convolutional layer, residual non-local attention block, and residual local attention block respectively. Here, we take image denoising as a task of interest.

proposed to investigate the effect of attention for image restoration tasks. Here, we want to enhances the network with distinguished power for noise and image content.

Image restoration architectures. Stacked denoising auto-encoder [138] is one of the most well-known CNN-based image restoration method. [38] proposed ARCNN for image compression artifact reduction with several stacked convolutional layers. With the help of residual learning and batch normalization [41], Zhang *et al*. proposed DnCNN [40] for accurate image restoration and denoiser priors for image restoration in IRCNN [37]. Recently, great progresses have been made in image restoration community, where Timofte *et al*. [91], Ancuti *et al*. [93], and Blau *et al*. [94] lead the main competitions recently and achieved new research status and records. For example, Wang *et al*. [137] proposed a fully progressive image SR approach. However, most methods are plain networks and neglect to use non-local information.

4.3 Residual Non-local Attention Network for Image Restoration

4.3.1 Framework

The framework of our proposed residual non-local attention network (RNAN) is shown in Figure 4.1. Let's denote I_L and I_H as the low-quality (e.g., noisy, blurred, or compressed images) and high-quality images. The reconstructed image I_R can be obtained by

$$I_R = H_{RNAN}\left(I_L\right), \tag{4.1}$$

where H_{RNAN} denotes the function of our proposed RNAN. With the usage of global residual learning in pixel space, the main part of our network can concentrate on learning the degradation components (e.g., noisy, blurring, or compressed artifacts).

The first and last convolutional layers are shallow feature extractor and reconstruction layer respectively. We propose residual local and non-local attention blocks to extract hierarchical attention-aware features. In addition to making the main network learn degradation components,

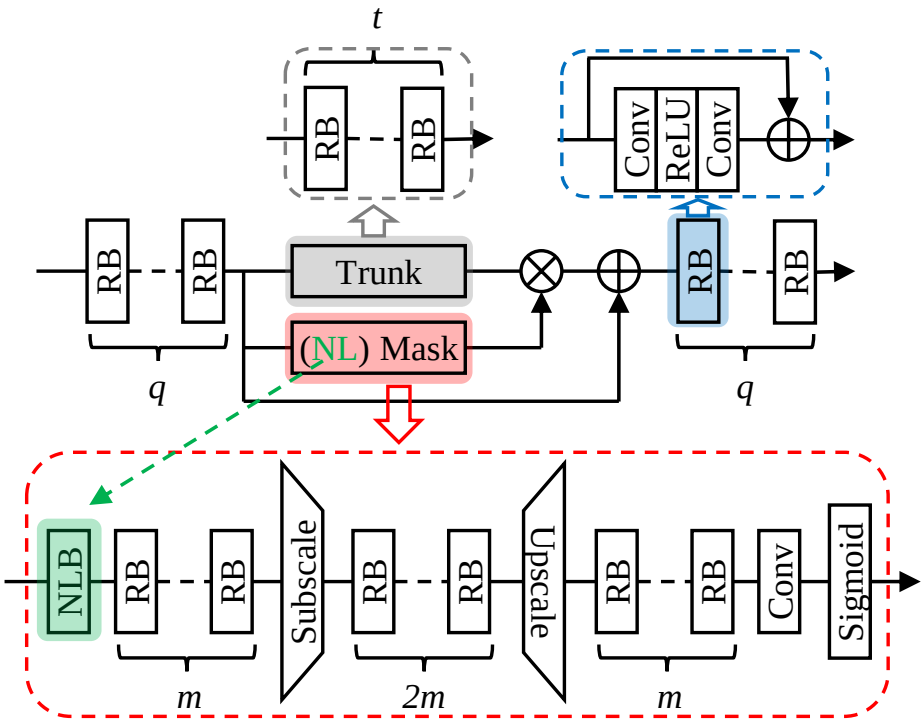


Figure 4.2: Residual (non-)local attention block. It mainly consists of trunk branch (labelled with gray dashed) and mask branch (labelled with red dashed). The trunk branch consists of t RBs. The mask branch is used to learning mixed attention maps in channel- and spatial-wise simultaneously.

we further concentrate on more challenging areas by using local and non-local attention. We only incorporate residual non-local attention block in low-level and high-level feature space. This is mainly because a few non-local modules can well offer non-local ability to the network for image restoration.

Then RNAN is optimized with loss function. Several loss functions have been investigated, such as L_2 [39, 40, 21, 37], L_1 [1, 86], perceptual and adversarial losses [103]. To show the effectiveness of our RNAN, we choose to optimize the same loss function (e.g., L_2 loss function) as previous works. Given a training set $\left\{I_L^i, I_H^i\right\}_{i=1}^N$, which contains N low-quality inputs and their high-quality counterparts. The goal of training RNAN is to minimize the L_2 loss function

$$L\left(\Theta\right) = \frac{1}{N}\sum_{i=1}^{N}\left\|H_{RNAN}\left(I_L^i\right) - I_H^i\right\|_2, \tag{4.2}$$

where $\|\cdot\|_2$ denotes l_2 norm. As detailed in Section 4.4, we use the same loss function as that in other compared methods. Such choice makes it clearer and more fair to see the effectiveness of our proposed RNAN. Then we give more details to residual local and non-local attention blocks.

4.3.2 Residual Non-local Attention Block

Our residual non-local attention network is constructed by stacking several residual local and non-local attention blocks shown in Figure 4.2. Each attention block is divided into two parts: q residual blocks (RBs) in the beginning and end of attention block. Two branches in the middle part: trunk branch and mask branch. For non-local attention block, we incorporate non-local block (NLB) in the mask branch, resulting non-local attention. Then we give more details to those components.

4.3.2.1 Trunk Branch

As shown in Figure 4.2, the trunk branch includes t residual blocks (RBs). Different from the original residual block in ResNet [19], we adopt the simplified RB from [1]. The simplified RB (labelled with blue dashed) only consists of two convolutional layers and one ReLU [134], omitting unnecessary components, such as maxpooling and batch normalization [41] layers. We find that such simplified RB not only contributes to image super-resolution [1], but also helps to construct very deep network for other image restoration tasks.

Feature maps from trunk branch of different depths serve as hierarchical features. If attention mechanism is not considered, the proposed network would become a simplified ResNet. With mask branch, we can take channel and spatial attention to adaptively rescale hierarchical features. Then we give more details about local and non-local attention.

4.3.2.2 Mask Branch

As labelled with red dashed in Figure 4.2, the mask branches used in our network include local and non-local ones. Here, we mainly focus on local mask branch, which can become a non-local one by using non-local block (NLB, labelled with green dashed arrow).

The key point in mask branch is how to grasp information of larger scope, namely larger receptive field size, so that it's possible to obtain more sophisticated attention map. One possible solution is to perform maxpooling several times, as used in [128] for image classification. However, more pixel-level accurate results are desired in image restoration. Maxpooling would lose lots of details of the image, resulting in bad performance. To alleviate such drawbacks, we choose to use large-stride convolution and deconvolution to enlarge receptive field size. Another way is considering non-local information across the whole inputs, which will be discussed in the next subsection.

From the input, large-stride (stride ≥ 2) convolutional layer increases the receptive field size after m RBs. After additional $2m$ RBs, the downscaled feature maps are then expanded by a

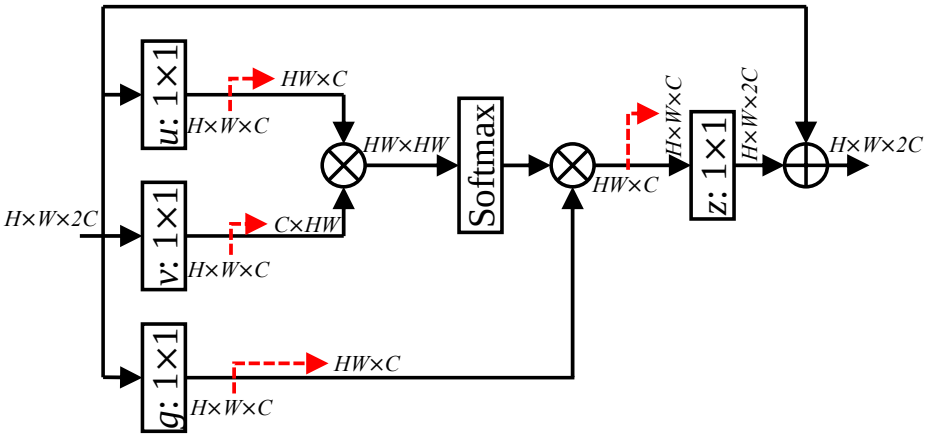


Figure 4.3: Non-local block. Red dashed line denotes matrix reshaping. $H \times W \times C$ means C features with height H and width W. $\otimes$ denotes matrix multiplication. $\bigoplus$ denotes element-wise addition.

deconvolutional layer (also known as transposed convolutional layer). The upscaled features are further forwarded through m RBs and one 1×1 convolutional layer. Then a sigmoid layer normalizes the output values, ranging in $[0, 1]$. Although the receptive field size of the mask branch is much larger than that of the trunk branch, it cannot cover the whole features at a time. This can be achieved by using non-local block (NLB), resulting in non-local mixed attention.

4.3.2.3 Non-local Mixed Attention

As discussed above, convolution operation processes one local neighbourhood at a time. In order to obtain better attention maps, here we seek to take all the positions into consideration at a time. Inspired by classical non-local means method [139] and non-local neural networks [32], we incorporate non-local block (NLB) into the mask branch to obtain non-local mixed attention (shown in Figure 4.3). The non-local operation can be defined as

$$y_i = \left(\sum_{\forall j} f(x_i, x_j) g(x_j) \right) / \sum_{\forall j} f(x_i, x_j), \tag{4.3}$$

where i is the output feature position index and j is the index that enumerates all possible positions. x and y are the input and output of non-local operation. The pairwise function $f(x_i, x_j)$ computes relationship between x_i and x_j. The function $g(x_j)$ computes a representation of the input at the position j.

As shown in Figure 4.3, we use embedded Gaussian function to evaluate the pairwise

relationship

$$f(x_i, x_j) = exp(u(x_i)^T v(x_j)) = exp((W_u x_i)^T W_v x_j), \tag{4.4}$$

where W_u and W_v are weight matrices. As investigated in [32], there are several versions of f, such as Gaussian function, dot product similarity, and feature concatenation. We also consider a linear embedding for g: $g(x_j) = W_g x_j$ with weight matrix W_g. Then the output z at position i of non-local block (NLB) is calculated as

$$z_i = W_z y_i + x_i = W_z softmax((W_u x_i)^T W_v x_j) g(x_j) + x_i, \tag{4.5}$$

where W_z is a weight matrix. For a given i, $\sum_{\forall j} f(x_i, x_j) / \sum_{\forall j} f(x_i, x_j)$ in Eq. 4.3 becomes the *softmax* computation along dimension j. The residual connection allows us to insert the NLB into pretrained networks [32] by initializing W_z as zero.

With non-local and local attention computation, feature maps in the mask branch are finally mapped by sigmoid function

$$f_{mix}(x_{i,c}) = \frac{1}{1 + exp(-x_{i,c})}, \tag{4.6}$$

where i ranges over spatial positions and c ranges over feature channel positions. Such simple sigmoid operation is applied to each channel and spatial position, resulting mixed attention [128]. As a result, the mask branch with non-local block can produce non-local mixed attention. However, simple multiplication between features from trunk and mask branches is not powerful enough or proper to form very deep trainable network. We propose residual non-local attention learning to solve those problems.

4.3.3 Residual Non-local Attention Learning

How to train deep image restoration network with non-local mixed attention remains unclear. Here we only consider the trunk and mask branches, and residual connection with them (Figure 4.2). We focus on obtaining non-local attention information from the input feature x. It should be noted that one form of attention residual learning was proposed in [128], whose formulation is

$$H_{RNA}(x) = H_{trunk}(x)(H_{mask}(x) + 1). \tag{4.7}$$

We find that this form of attention learning is not suitable for image restoration tasks. This is mainly because Eq. 4.7 is more suitable for high-level vision tasks (e.g., image classification), where

low-level features are not preserved too much. However, low-level features are more important for image restoration. As a result, we propose a simple yet more suitable residual attention learning method by introducing input feature x directly. We compute its output $H_{RNA}(x)$ as

$$H_{RNA}(x) = H_{trunk}(x)H_{mask}(x) + x, \tag{4.8}$$

where $H_{trunk}(x)$ and $H_{mask}(x)$ denote the functions of trunk and mask branches respectively. Such residual learning tends to preserve more low-level features and allows us to form very deep networks for high-quality image restoration tasks with stronger representation ability.

4.3.4 Implementation Details

Now, we specify the implementation details of our proposed RNAN. We use 10 residual local and non-local attention blocks (2 non-local one). In each residual (non-)local block, we set $q, t, m = 2, 2, 1$. We set 3×3 as the size of all convolutional layers except for those in non-local block and convolutional layer before sigmoid function, where the kernel size is 1×1. Features in RBs have 64 filters, except for that in the non-local block (see Figure 4.3), where $C = 32$. In each training batch, 16 low-quality (LQ) patches with the size of 48×48 are extracted as inputs. Our model is trained by ADAM optimizer [116] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The initial learning rate is set to 10^{-4} and then decreases to half every 2×10^5 iterations of back-propagation. We use PyTorch [135] to implement our models with a Titan Xp GPU.

4.4 Experiments

We apply our proposed RNAN to three classical image restoration tasks: image denoising, demosaicing, and compression artifacts reduction. For image denoising and demosaicing, we follow the same setting as IRCNN [37]. For image compression artifacts reduction, we follow the same setting as ARCNN [38]. We use 800 training images in DIV2K [91, 140] to train all of our models. For each task, we use commonly used dataset for testing and report PSNR and/or SSIM [114] to evaluate the results of each method.

4.4.1 Ablation Study

We show ablation study in Table 4.1 to investigate the effects of different components in RNAN.

Table 4.1: Ablation study of different components. PSNR values are based on Urban100 (σ=30).

Case Index	1	2	3	4	5	6	7	8
Mask Branch	✗	✓	✗	✓	✓	✓	✓	✓
Non-local Block	✗	✗	✓	✓	✓	✓	✓	✓
RAB Number	7	7	5	5	1	2	5	8
RNAB Number	0	0	2	2	1	2	1	2
PSNR (dB)	30.96	31.17	31.20	31.32	30.78	30.99	31.27	31.50

Table 4.2: Quantitative results about **color** image denoising. Best results are **highlighted**.

Method	Kodak24				BSD68				Urban100			
	10	30	50	70	10	30	50	70	10	30	50	70
CBM3D	36.57	30.89	28.63	27.27	35.91	29.73	27.38	26.00	36.00	30.36	27.94	26.31
TNRD	34.33	28.83	27.17	24.94	33.36	27.64	25.96	23.83	33.60	27.40	25.52	22.63
RED	34.91	29.71	27.62	26.36	33.89	28.46	26.35	25.09	34.59	29.02	26.40	24.74
DnCNN	36.98	31.39	29.16	27.64	36.31	30.40	28.01	26.56	36.21	30.28	28.16	26.17
MemNet	N/A	29.67	27.65	26.40	N/A	28.39	26.33	25.08	N/A	28.93	26.53	24.93
IRCNN	36.70	31.24	28.93	N/A	36.06	30.22	27.86	N/A	35.81	30.28	27.69	N/A
FFDNet	36.81	31.39	29.10	27.68	36.14	30.31	27.96	26.53	35.77	30.53	28.05	26.39
RNAN (ours)	**37.24**	**31.86**	**29.58**	**28.16**	**36.43**	**30.63**	**28.27**	**26.83**	**36.59**	**31.50**	**29.08**	**27.45**

Non-local Mixed Attention. In cases 1, all the mask branches and non-local blocks are removed. In case 4, we enable non-local mixed attention with same block number as in case 1. The positive effect of non-local mixed attention is demonstrated by its obvious performance improvement.

Mask branch. We also learn that mask branch contributes to performance improvement, no matter non-local blocks are used (cases 3 and 4) or not (cases 1 and 2). This's mainly because mask branch provides informative attention to the network, gaining better representational ability.

Non-local block. Non-local block also contributes to the network ability obviously, no matter mask branches are used (cases 2 and 4) or not (cases 1 and 3). With non-local information from low-level and high-level features, RNAN performs better image restoration.

Block Number. When comparing cases 2, 4, and 7, we learn that more non-local blocks achieve better results. However, the introduction of non-local block consumes much time. So we use 2 non-local blocks by considering low- and high-level features. When RNAB number is fixed in cases 5 and 7 or cases 6 and 8, performance also benefits from more RABs.

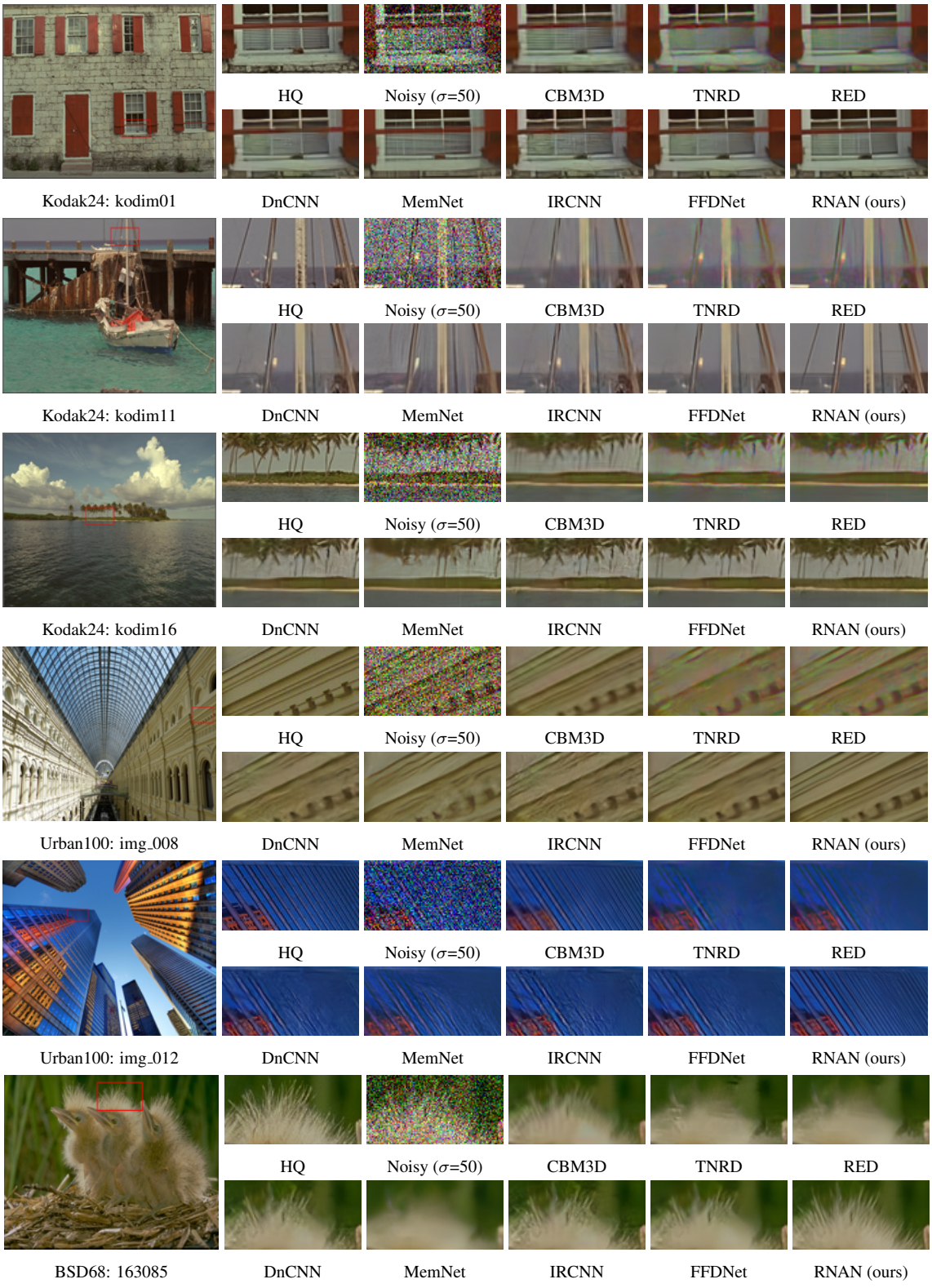


Figure 4.4: **Color** image denoising results with noise level $\sigma = 50$.

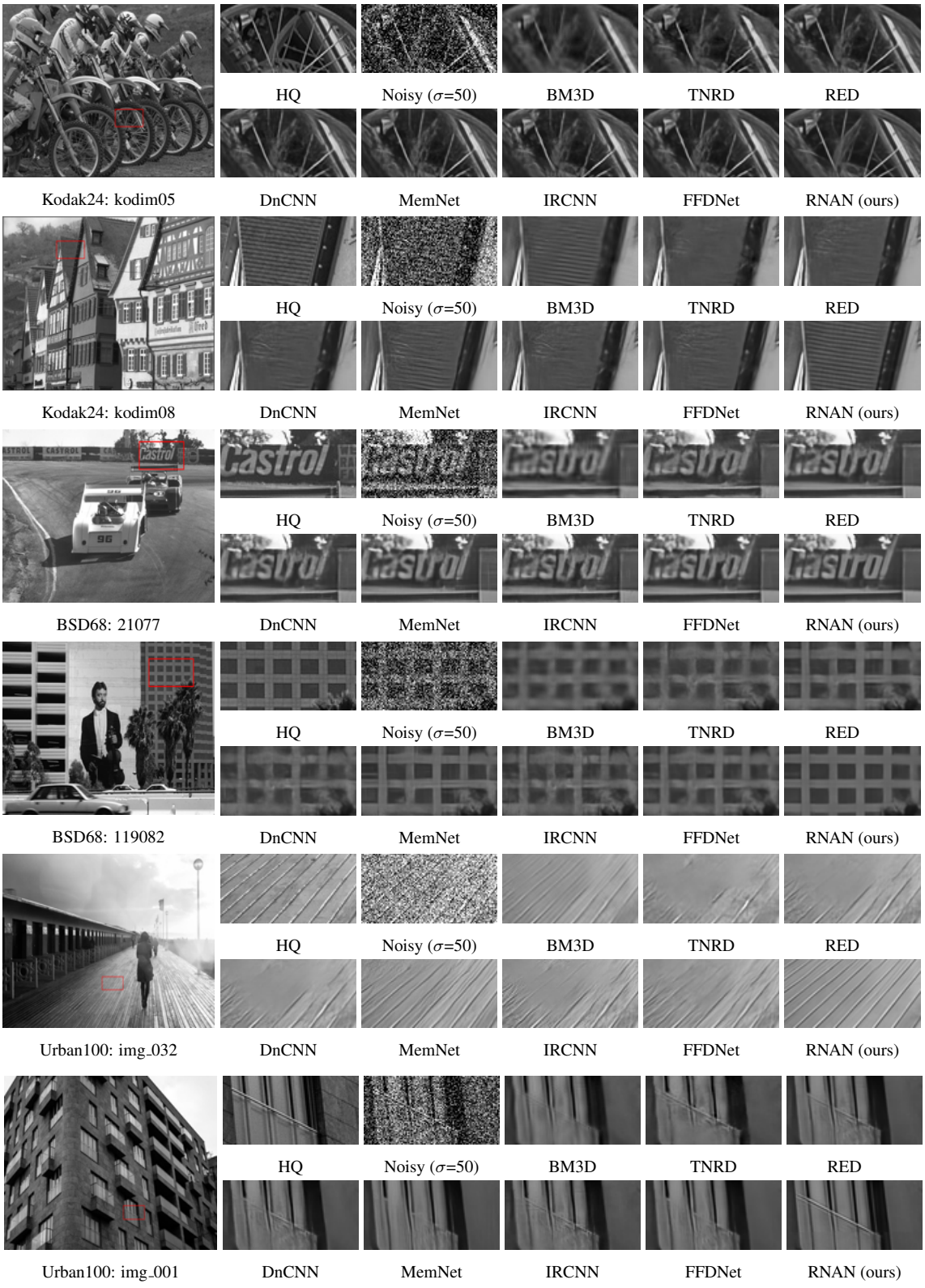


Figure 4.5: **Gray** image denoising results with noise level $\sigma = 50$.

Table 4.3: Quantitative results about **gray-scale** image denoising. Best results are **highlighted**.

Method	Kodak24				BSD68				Urban100			
	10	30	50	70	10	30	50	70	10	30	50	70
BM3D	34.39	29.13	26.99	25.73	33.31	27.76	25.62	24.44	34.47	28.75	25.94	24.27
TNRD	34.41	28.87	27.20	24.95	33.41	27.66	25.97	23.83	33.78	27.49	25.59	22.67
RED	35.02	29.77	27.66	26.39	33.99	28.50	26.37	25.10	34.91	29.18	26.51	24.82
DnCNN	34.90	29.62	27.51	26.08	33.88	28.36	26.23	24.90	34.73	28.88	26.28	24.36
MemNet	N/A	29.72	27.68	26.42	N/A	28.43	26.35	25.09	N/A	29.10	26.65	25.01
IRCNN	34.76	29.53	27.45	N/A	33.74	28.26	26.15	N/A	34.60	28.85	26.24	N/A
FFDNet	34.81	29.70	27.63	26.34	33.76	28.39	26.29	25.04	34.45	29.03	26.52	24.86
RNAN (ours)	**35.20**	**30.04**	**27.93**	**26.60**	**34.04**	**28.61**	**26.48**	**25.18**	**35.52**	**30.20**	**27.65**	**25.89**

4.4.2 Color and Gray Image Denoising

We compare our RNAN with state-of-the-art denoising methods: BM3D [118], CBM3D [119], TNRD [80], RED [39], DnCNN [40], MemNet [21], IRCNN [37], and FFDNet [81]. Kodak24 [1], BSD68 [107], and Urban100 [61] are used for color and gray-scale image denoising. AWGN noises of different levels (e.g., 10, 30, 50, and 70) are added to clean images.

Quantitative results are shown in Tables 4.2 and 4.3. As we can see that our proposed RNAN achieves the best results on all the datasets with all noise levels. Our proposed non-local attention covers the information from the whole image, which should be effective for heavy image denoising. To demonstrate this analysis, we take noise level $\sigma = 70$ as an example. We can see that our proposed RNAN achieves 0.48, 0.30, and 1.06 dB PSNR gains over the second best method FFDNet. This comparison strongly shows the effectiveness of our proposed non-local mixed attention.

We also show visual results in Figures 4.4 and 4.5. With the learned non-local mixed attention, RNAN treats different image parts distinctively, alleviating over-smoothing artifacts obviously.

4.4.3 Image Compression Artifacts Reduction

We further apply our RNAN to reduce image compression artifacts. We compare our RNAN with SA-DCT [113], ARCNN [38], TNRD [80], and DnCNN [40]. We apply the standard JPEG compression scheme to obtain the compressed images by following [38]. Four JPEG quality settings q = 10, 20, 30, 40 are used in Matlab JPEG encoder. Here, we only focus on the restoration of Y channel (in YCbCr space) to keep fair comparison with other methods. We use the same datasets

[1] http://r0k.us/graphics/kodak/

Figure 4.6: Image compression artifacts reduction results with JPEG quality q = 10.

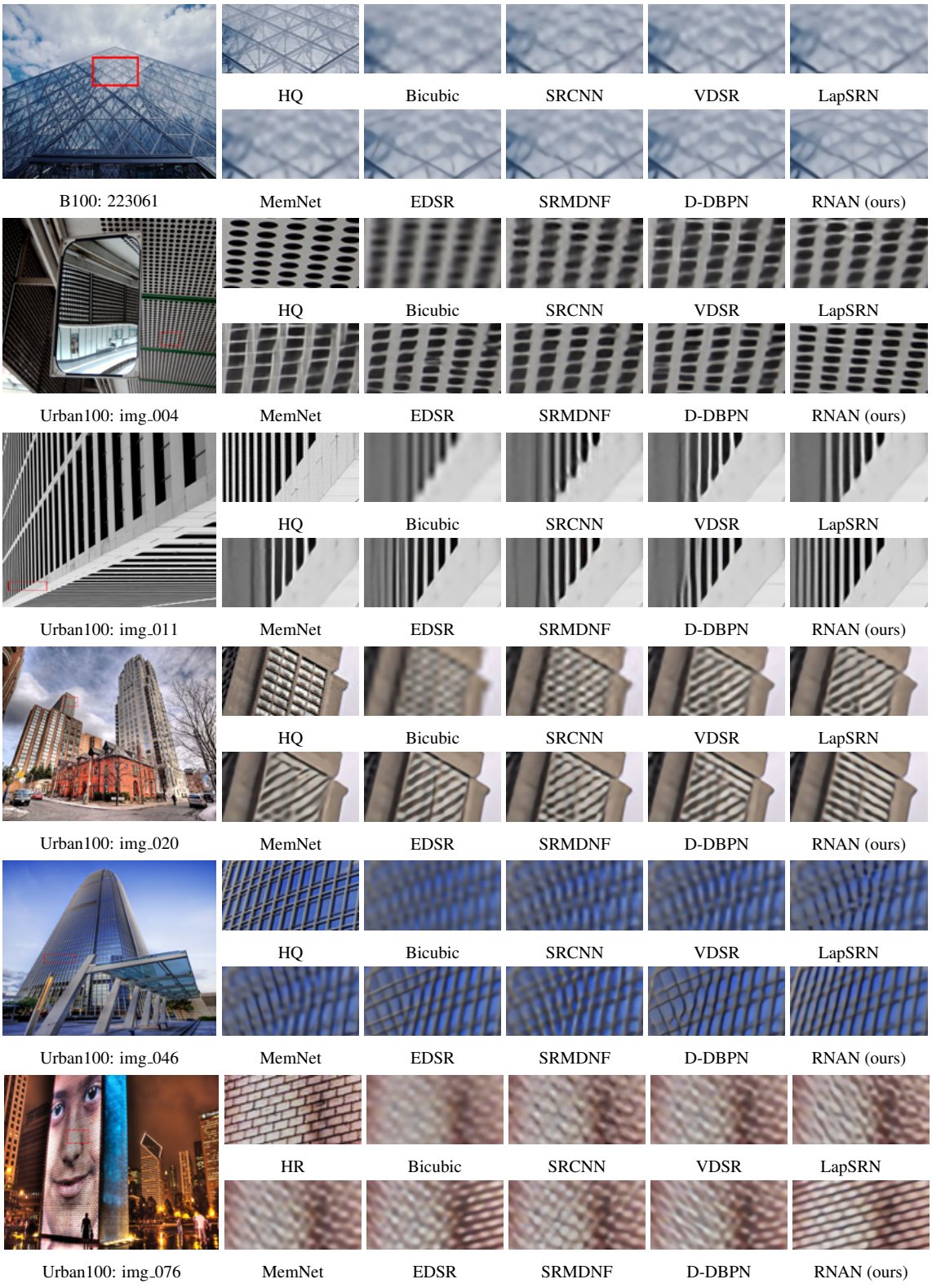


Figure 4.7: Image super-resolution results with scaling factor $s = 4$.

Table 4.4: Quantitative results about compression artifacts reduction. Best results are **highlighted**.

Dataset	q	JPEG		SA-DCT		ARCNN		TNRD		DnCNN		RNAN (ours)	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
LIVE1	10	27.77	0.7905	28.65	0.8093	28.98	0.8217	29.15	0.8111	29.19	0.8123	**29.63**	**0.8239**
	20	30.07	0.8683	30.81	0.8781	31.29	0.8871	31.46	0.8769	31.59	0.8802	**32.03**	**0.8877**
	30	31.41	0.9000	32.08	0.9078	32.69	0.9166	32.84	0.9059	32.98	0.9090	**33.45**	**0.9149**
	40	32.35	0.9173	32.99	0.9240	33.63	0.9306	N/A	N/A	33.96	0.9247	**34.47**	**0.9299**
Classic5	10	27.82	0.7800	28.88	0.8071	29.04	0.8111	29.28	0.7992	29.40	0.8026	**29.96**	**0.8178**
	20	30.12	0.8541	30.92	0.8663	31.16	0.8694	31.47	0.8576	31.63	0.8610	**32.11**	**0.8693**
	30	31.48	0.8844	32.14	0.8914	32.52	0.8967	32.78	0.8837	32.91	0.8861	**33.38**	**0.8924**
	40	32.43	0.9011	33.00	0.9055	33.34	0.9101	N/A	N/A	33.77	0.9003	**34.27**	**0.9061**

Table 4.5: Quantitative image SR results. Best and second best results are **highlighted** and underlined

Method	Scale	Set5		Set14		B100		Urban100		Manga109	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Bicubic	×2	33.66	0.9299	30.24	0.8688	29.56	0.8431	26.88	0.8403	30.80	0.9339
EDSR	×2	38.11	0.9602	33.92	0.9195	32.32	0.9013	32.93	0.9351	39.10	0.9773
SRMDNF	×2	37.79	0.9601	33.32	0.9159	32.05	0.8985	31.33	0.9204	38.07	0.9761
D-DBPN	×2	38.09	0.9600	33.85	0.9190	32.27	0.9000	32.55	0.9324	38.89	0.9775
RCAN	×2	**38.27**	**0.9614**	**34.12**	0.9216	**32.41**	**0.9027**	**33.34**	**0.9384**	**39.44**	0.9786
RNAN (ours)	×2	38.17	0.9611	33.87	0.9207	32.32	0.9014	32.73	0.9340	39.23	0.9785
RNAN+ (ours)	×2	38.22	0.9613	33.97	**0.9216**	32.36	0.9018	32.90	0.9351	39.41	**0.9789**
Bicubic	×4	28.42	0.8104	26.00	0.7027	25.96	0.6675	23.14	0.6577	24.89	0.7866
EDSR	×4	32.46	0.8968	28.80	0.7876	27.71	0.7420	26.64	0.8033	31.02	0.9148
SRMDNF	×4	31.96	0.8925	28.35	0.7787	27.49	0.7337	25.68	0.7731	30.09	0.9024
D-DBPN	×4	32.47	0.8980	28.82	0.7860	27.72	0.7400	26.38	0.7946	30.91	0.9137
RCAN	×4	**32.63**	**0.9002**	28.87	**0.7889**	**27.77**	**0.7436**	**26.82**	**0.8087**	31.22	0.9173
RNAN (ours)	×4	32.49	0.8982	28.83	0.7878	27.72	0.7421	26.61	0.8023	31.09	0.9149
RNAN+ (ours)	×4	32.56	0.8992	**28.90**	0.7883	27.77	0.7424	26.75	0.8052	**31.37**	**0.9175**

LIVE1 [112] and Classic5 [113] in ARCNN and report PSNR/SSIM values in Table 4.4. As we can see, our RNAN achieves the best PSNR and SSIM values on LIVE1 and Classic5 with all JPEG qualities.

We further shown visual comparisons in Figure 4.6. We provide comparisons under very low image quality (q=10). The blocking artifacts can be removed to some degree, but ARCNN, TNRD, and DnCNN would also over-smooth some structures. RNAN obtains more details with consistent structures by considering non-local mixed attention.

Table 4.6: Quantitative results about color image demosaicing. Best results are **highlighted**.

Method	McMaster18		Kodak24		BSD68		Urban100	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Mosaiced	9.17	0.1674	8.56	0.0682	8.43	0.0850	7.48	0.1195
IRCNN	37.47	0.9615	40.41	0.9807	39.96	0.9850	36.64	0.9743
RNAN (ours)	**39.71**	**0.9725**	**43.09**	**0.9902**	**42.50**	**0.9929**	**39.75**	**0.9848**

4.4.4 Image Super-Resolution

We further compare our RNAN with state-of-the-art SR methods: EDSR [1], SRMDNF [79], D-DBPN [92], and RCAN [11]. Similar to [1, 86], we also introduce self-ensemble strategy to further improve our RNAN and denote the self-ensembled one as RNAN+.

As shown in Table 4.5, our RNAN+ achieves the second best performance among benchmark datasets: Set5 [105], Set14 [106], B100 [107], Urban100 [61], and Manga109 [111]. Even without self-ensemble, our RNAN achieves third best results in most cases. Such improvements are notable, because the parameter number of RNAN is 7.5 M, far smaller than 43 M in EDSR and 16 M in RCAN. The network depth of our RNAN (about 120 convolutional layers) is also far shallower than that of RCAN, which has about 400 convolutional layers. It indicates that non-local attention make better use of main network, saving much network parameter.

In Figure 4.7, we conduct image SR ($\times 4$) with several state-of-the-art methods. We can see that our RNAN obtains better visually pleasing results with finer structures. These comparisons further demonstrate the effectiveness of our proposed RNAN with the usage of non-local mixed attention.

4.4.5 Image Demosaicing

Following the same setting in IRCNN [37], we compare image demosaicing results with those of IRCNN on McMaster [37], Kodak24, BSD68, and Urban100. Since IRCNN has been one of the best methods for image demosaicing and limited space, we only compare with IRCNN in Table 4.6. As we can see, mosaiced images have very poor quality, resulting in very low PSNR and SSIM values. IRCNN can enhance the low-quality images and achieve relatively high values of PSNR and SSIM. Our RNAN can still make significant improvements over IRCNN. Using local and non-local attention, our RNAN can better handle the degradation situation.

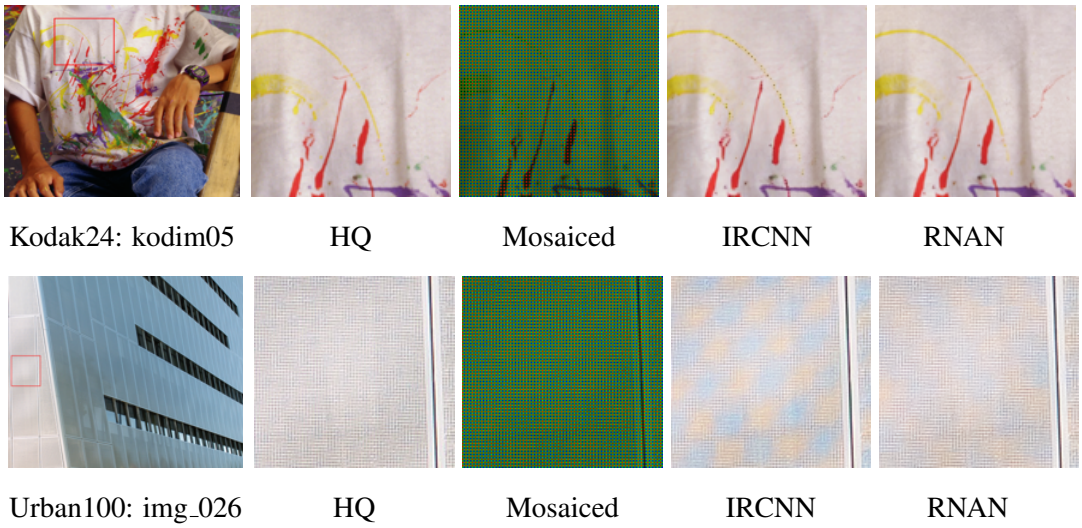


Figure 4.8: Image demosaicing results.

Table 4.7: Parameter and time comparisons. '*' means being applied channel-wise for color images.

Methods	RED*	DnCNN	MemNet*	RNAN (1LB+1NLB)	RNAN (8LBs+2NLBs)
Parameter Number	4,131 K	672K	677K	1,494k	7,409K
PSNR (dB)	26.40	28.16	26.53	28.36	29.15
Time (s) (Platform)	58.7 (Caffe)	16.5 (Matlab+GPU Array)	239.0 (Caffe)	2.6 (PyTorch)	6.5 (PyTorch)

Visual results are shown in Figure 4.8. Although IRCNN can remove mosaicing effect greatly, there're still some artifacts in its results (e.g., blocking artifacts in 'img_026'). However, RNAN recovers more faithful color and alliciates blocking artifacts.

4.4.6 Parameters and Running Time Analyses

We also compare parameters, running time, and performance based on color image denoising in Table 4.7. PSNR values are tested on Urban100 (σ=50). RNAN with 10 blocks achieves the best performance with the highest parameter number, which can be reduced to only 2 blocks and obtains second best performance. Here, we report running time for reference, because the time is related to implementation platform and code.

4.5 Summary

In this chapter, we propose residual non-local attention networks for high-quality image restoration. The networks are built by stacking local and non-local attention blocks, which extract local and non-local attention-aware features and consist of trunk and (non-)local mask branches. They're used to extract hierarchical features and adaptively rescale hierarchical features with soft weights. We further generate non-local attention by considering the whole feature map. Furthermore, we propose residual local and non-local attention learning to train very deep networks. We introduce the input feature into attention computation, being more suitable for image restoration. RNAN achieves state-of-the-art image restoration results with moderate model size and running time.

Chapter 5

Multimodal Style Transfer

5.1 Background and Motivation

Image style transfer is the process of rendering a content image with characteristics of a style image. Usually, it would take a long time for a diligent artist to create a stylized image with particular style. Recently, it draws a lot of interests [46, 50, 53, 141, 3, 4, 7, 8, 56, 142, 143, 5] since Gatys *et al.* [46] discovered that the correlations between convolutional features of deep networks can represent image styles, which would have been hard for traditional patch-based methods to deal with. These neural style transfer methods either use an iterative optimization scheme [46] or feed-forward networks [50, 53, 141, 3, 4, 8, 5] to synthesize the stylizations. Most of them are applicable for arbitrary style transfer with a pre-determined model. These universal style transfer methods [3, 4, 8, 5] inherently assume that the style can be represented by the global statistics of deep features such as gram matrix [46] and its approximates [3, 4]. Although these neural style transfer methods can preserve the content well and match the overall style of the reference style images, they will also distort the local style patterns, resulting unpleasing visual artifacts.

Let's start with some examples in Figure **??**. In the first row, where the style image consists of complex textures and strokes, these methods cannot tell them apart and neglect to match style patterns to content structures adaptively. This would introduce some less desired strokes in smooth content areas, e.g., the sky. In the second row, the style image has clear spatial patterns (e.g., large uniform background and blue/red hand). AdaIN, WCT, and LST failed to maintain the content structures and suffered from wash-out artifacts. This is mainly because the unified style background occupies a large proportion in the style image, resulting its domination in the global statistics of style features. These observations indicate that it may not be sufficient to represent style features as at may

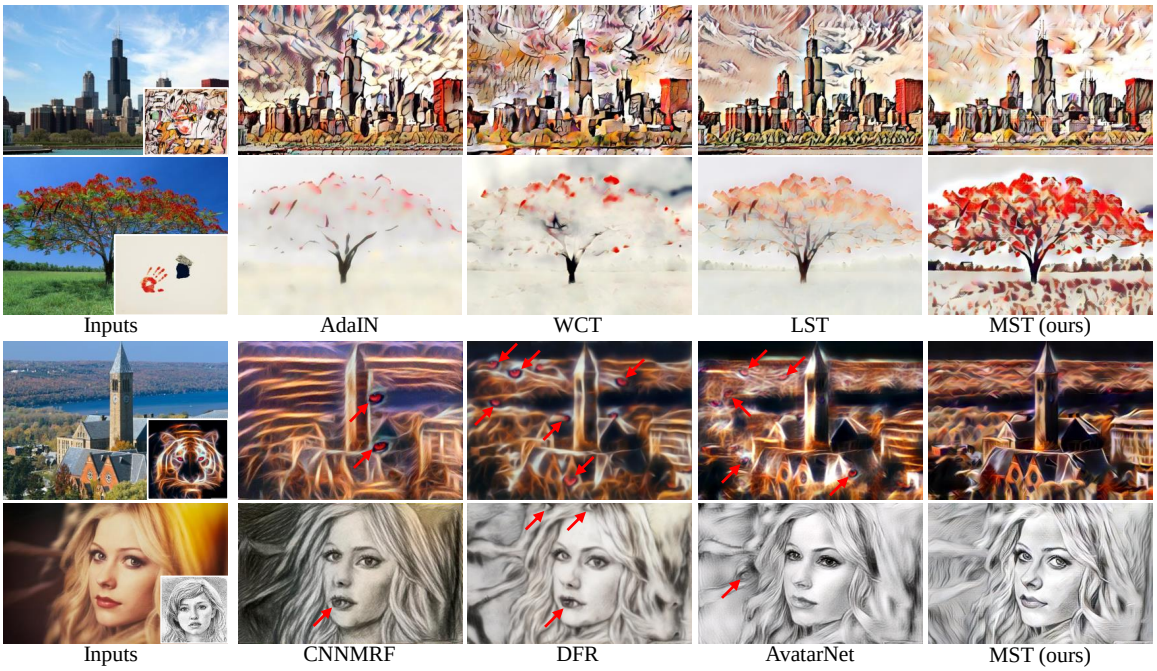


Figure 5.1: Gram matrix based style transfer methods (AdaIN [3], WCT [4], and LST [5]) may fail to distinguish style patterns (1st and 2nd rows). Patch-swap based methods (CNNMRF [6], DFR [7], and AvatarNet [8]) may copy some less desired style patterns (labeled with red arrows) to the results (3rd and 4th rows). Our MST alleviates all these limitations.

not be sufficient to represent style features as a unimodal distribution such as a Gram or covariance matrix. An ideal style representation should respect to the spatially-distributed style patterns.

Inherited from traditional patch-based methods, these neural patch-based algorithms could generate visually pleasing results when content and style images have similar structures. However, the greedy example matching usually employed by these methods will introduce less desired style patterns to the outputs. This is illustrated by the bottom two examples in Figure **??**, where some salient patterns in the style images, e.g., the eyes and lips, are improperly copied to the buildings and landscape. Moreover, the last row of Figure **??** also illustrates shape distortion problem of these methods; e.g., the appearance for the girl has changed. This phenomenon apparently limits the choice of style images for these methods.

To address these issues, we propose the multimodal style transfer (MST), a more flexible and general style transfer method that seeks for a sweet spot between parametric (gram matrix based) and non-parametric (patch based) approaches. Specifically, instead of representing the style with a unimodal distribution, we propose a multimodal style representation with graph based style matching mechanism to adaptively match the style patterns to a content image.

Figure 5.2: t-SNE [9] visualization for style features. The original high-dimension style features are extracted at layer Conv_4_1 in VGG-19 [10] and are reduced to 3 dimensions via t-SNE. We can see the feature distributions tend to fit as multimodal distributions rather than single-modal ones.

Our main contributions are summarized as follows:

- We analyze the feature distributions of different style images (see Figure **??**) and propose a multimodal style representation that better models the style feature distribution. This multimodal representation consists of a mixture of clusters, each of which represents a particular style pattern. It also allows users to mix-and-match different styles to render diverse stylized results.
- We formulate style-content matching as an energy minimization problem with a graph and solve it via graph cuts. Style clusters are adapted to content features with respect to the content spatial configuration.
- We demonstrate the strength of MST by extensive comparison with several state-of-the-art style transfer methods. The robustness and flexibility of MST is shown with different sub-style numbers and multi-style mixtures. The general idea of MST can be extended to improve other existing stylization methods.

5.2 Related Work

Style Transfer. Originating from non-realistic rendering [42], image style transfer is closely related to texture synthesis [43, 44, 45]. Gatys *et al.* [46] were the first to formulate style transfer as the matching of multi-level deep features extracted from a pre-trained deep neural network, which has been widely used in various tasks [47, 48, 49]. Lots of improvements have been proposed based on the works of Gatys *et al.* [46]. Johnson *et al.* [50] trained feed-forward style-specific

network and produced one stylization with one model. Sanakoyeu *et al.* [51] further proposed a style-aware content loss for high-resolution style transfer. Jing *et al.* [52] proposed a StrokePyramid module to enable controllable stroke with adaptive receptive fields. However, these methods are either time consuming or have to re-train new models for new styles.

The first arbitrary style transfer was proposed by Chen and Schmidt [53], who matched each content patch to the most similar style patch and swapped them. Luan *et al.* [54] proposed deep photo style transfer by adding a regularization term to the optimization function. Based on markov random field (MRF), Li and Wand [6] proposed CNNMRF to enforce local patterns in deep feature space. Ruder *et al.* [55] improved video stylization with temporal coherence. Although their visual stylizations for arbitrary style are appealing, the results are not stable [55].

Recently, Huang *et al.* [3] proposed real-time style transfer by matching the mean-variance statistics between content and style features. Li *et al.* [4] further introduced whitening and coloring (WCT) by matching the covariance matrices. Li *et al.* boosted style transfer with linear style transfer (LST) [5]. Gu *et al.* [7] proposed deep feature reshuffle (DFR), which connects both local and global style losses used in parametric and non-parametric methods. Sheng *et al.* [8] proposed AvatarNet to enable multi-scale transfer for arbitrary style. Shen *et al.* [56] built meta networks by taking style images as inputs and generating corresponding image transformation networks directly. Mechrez *et al.* [57] proposed contextual loss for image transformation. However, these methods fail to treat the style patterns distinctively and neglect to adaptively match style patterns with content semantic information. For more neural style transfer works, readers can refer to the survey [58].

Graph Cuts based Matching. Many problems that arose in early vision can be naturally expressed in terms of energy minimization. For example, a large number of computer vision problems attempt to assign labels to pixels based on noisy measurements. Graph cuts is a powerful method to solve such discrete optimization problems. Greig *et al.* [144] were firstly successful solving graph cuts by using powerful min-cut/max-flow algorithms from combinatorial optimization. Roy and Cox [145] were the first to use these techniques for multi-camera stereo computation. Later, a growing number of researches in computer vision use graph-based energy minimization for a wide range of applications, which includes stereo [146], texture synthesis [147], image segmentation [148], object recognition [149], and others. In this paper, we formulate the matching between content and style features as an energy minimization problem. We approximate its global minimum via efficient graph cuts algorithms. To the best of our knowledge, we are the first to formulate style matching as energy minimization problem and solve it via graph cuts.

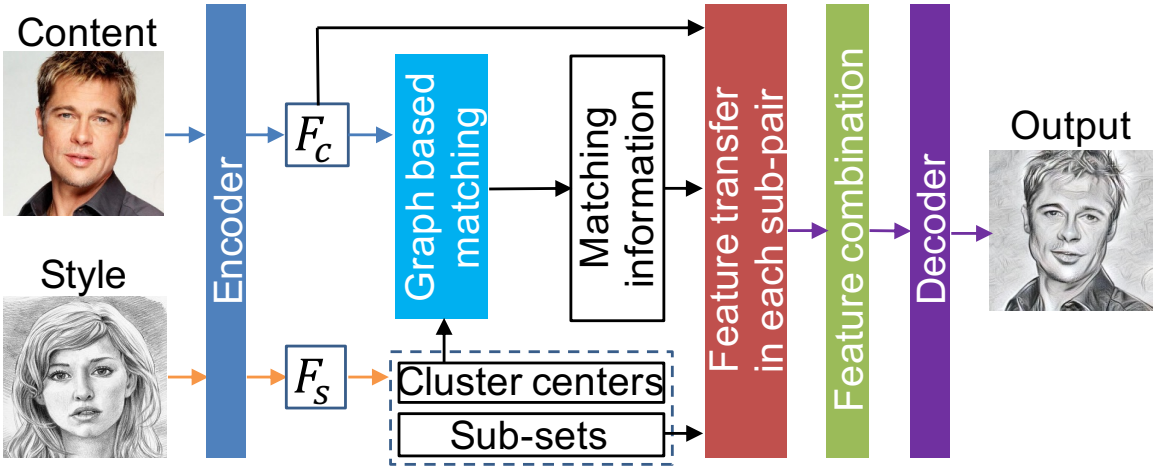


Figure 5.3: An overview of our MST algorithm.

5.3 Multimodal Style Transfer via Graph Cuts

We show the pipeline of our proposed MST in Figure **??**.

5.3.1 Multimodal Style Representation

In previous CNN-based image style transfer works, there are two main ways to represent style. One is to use the features from the whole image and assume that they are in the same distribution (e.g., AdaIN [3] and WCT [4]). The other one treats the style patterns as individual style patches (e.g., Deep Feature Reshuffle [7]). Equal treatments to different style patterns lack flexibility in the real cases, where there has several distributions among style features. Let's see the t-SNE [9] visualization for style features in Figure **??**, where the style features are clustered to multiple groups. Therefore, if a cluster dominates the feature space, e.g., second example of Figure **??**, the Gram matrix based methods [4, 5, 3] will fail to capture the the overall style patterns. On the other hand, patch-based methods which treat each sub-patch distinctly would suffer from copying multiple same style patterns to the results directly. For example, in Figure **??**, the eyes in the style images are copied multiple times, causing unpleasing stylization results.

Based on the observations and analyses above, we argue that neither a global statics of deep features nor local neural patches could be a suitable way to represent the complex real-world cases. As a result, we propose multimodal style representation, a more efficient and flexible way to represent different style patterns.

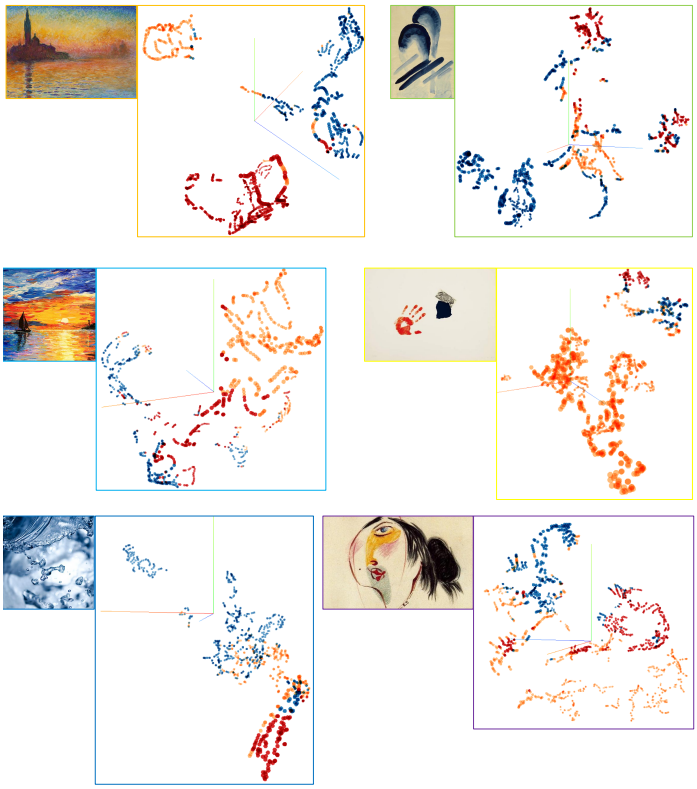


Figure 5.4: t-SNE [9] visualization for style features with cluster labels. For each style-visualization pair, we set $K = 3$ and label each style feature point with its corresponding cluster label. We can see that features in similar group have similar cluster labels. This observation indicates that clustering style features is a proper way for multimodal style representation.

For a given style image I_s, we can extract its deep features $F_s \in \Re^{C \times H_s W_s}$ via a pre-trained encoder $E_{\theta_{enc}}(\cdot)$, like VGG-19 [10]. H_s and W_s are the height and width of the style feature. To achieve multimodal representation in high-dimension feature space, we target to segment the style patterns into multiple subsets. Technically, we simply apply K-means to cluster all the style feature points into K clusters without considering spatial style information

$$F_s = F_s^{l_1} \cup F_s^{l_2} \cup \cdots \cup F_s^{l_k} \cup \cdots \cup F_s^{l_K}, \tag{5.1}$$

where $F_s^{l_k} \in \Re^{C \times N_k}$ is the k-th cluster with N_k features and we assign this cluster a label l_k. In the clustered space, features in the same cluster have similar visual properties and are likely drawn from the same distribution (resembling Gaussian Mixture Model [150]). This process helps us obtain a multimodal representation of style.

We visualize multimodal style representation in Figure **??**. For each style image, we extract its VGG feature (at layer Conv_4_1 in VGG-19) and cluster it into $K = 3$ clusters. Then, we conduct

t-SNE [9] visualization with the cluster labels. As shown in Figure **??**, clustering results match our assumption of multimodal style representation well. Nearby feature points tend to be in the same cluster. These observation not only shows the multimodal style distribution, but also demonstrates that clustering is a proper way to model such a multimodal distribution.

5.3.2 Graph Based Style Matching

Like style feature extraction, we extract deep content features $F_c \in \Re^{C \times H_c W_c}$ from a content image I_c. H_c and W_c are the height and width of the content feature. Distance measurement is the first step before matching. To reach a good distance metric, we should consider the scale difference between the content and style features. Computation complexity should also be taken into consideration, since all the content features will be used to match. Based on above analysis, we calculate the cosine distance between content feature $F_{c,p} \in \Re^{C \times 1}$ and style cluster center $F_{s,l_k} \in \Re^{C \times 1}$ as follows

$$D\left(F_{c,p}, F_{s,l_k}\right) = 1 - \frac{{F_{c,p}}^T F_{s,l_k}}{\|F_{c,p}\| \, \|F_{s,l_k}\|}, \tag{5.2}$$

where $(\cdot)^T$ is transpose operation and $\|\cdot\|$ is magnitude of the feature vector.

Then we target to find a labeling f that assigns each content feature $F_{c,p}$ with a style cluster center label $f_p \in \{l_1, l_2, \cdots, l_K\}$. We formulate the disagreement between f and content features as follows

$$E_{data}\left(f\right) = \sum_{p=1}^{HcWc} D\left(F_{c,p}, F_{s,f_p}\right), \tag{5.3}$$

where we name E_{data} as data energy. Minimizing E_{data} encourages f to be consistent with content features.

However, the spatial content information here is not considered, failing to preserve discontinuity and producing some unpleasing structures in the stylized results. Instead, we hope pixels in the same content local region have same labels. Namely, we want f to be piecewise smooth and discontinuity preserving. So, we further introduce another smooth term $E_{smooth}\left(f\right)$ as follows

$$E_{smooth}\left(f\right) = \sum_{\{p,q\} \in \Omega} V_{p,q}\left(f_p, f_q\right), \tag{5.4}$$

where Ω is the position set of direct interacting pairs of content features. $V_{p,q}$ denotes the distinct penalty for each position pair of features $\{p, q\}$. This has been investigated to be important in various

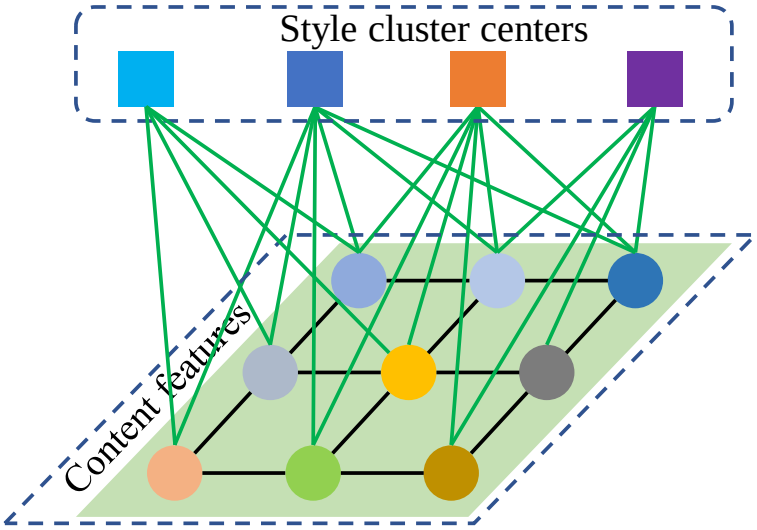


Figure 5.5: Graph based style matching. Example of the graph containing content features and style cluster centers. We match content features with style cluster in pixel level.

computer vision applications [151]. Also, various forms of energy functions have been investigated before. Here, we take the discontinuity preserving function given by the Potts model

$$V_{p,q}\left(f_p, f_q\right) = \lambda \cdot T\left(f_p \neq f_q\right), \tag{5.5}$$

where $T\left(\cdot\right)$ is 1 if its argument is true, and otherwise 0. λ is a smooth constant. This model encourages the labeling f to pursue several regions, where content features in the same region have same style cluster labels.

By taking Eqs. (**??**) and (**??**) into consideration, we naturally formulate the style matching problem as a minimization of the following energy function:

$$E\left(f\right) = E_{data}\left(f\right) + E_{smooth}\left(f\right). \tag{5.6}$$

The whole energy $E\left(f\right)$ measures not only the disagreement between f and content features, but also the extent to which f is not piecewise smooth. However, the global minimization of such an energy function is NP-hard even in the simplest discontinuity-preserving case [151].

To solve the energy minimization problem in Eq. (**??**), we propose to build a graph by regarding content features as p-vertices and style cluster centers as l-vertices (shown in Figure **??**). Then the energy minimization is equal to min-cut/max-flow problem, which can be efficiently solved via graph cuts [151]. After finding a local minimum, the whole content features can be re-organized

Figure 5.6: Visualization of style matching. Here, we cluster style features into $K = 2$ subsets for better understanding.

as follows

$$F_c = F_c^{l_1} \cup F_c^{l_2} \cup \cdots \cup F_c^{l_k} \cup \cdots \cup F_c^{l_K}, \tag{5.7}$$

where $F_c^{l_k}$ demotes the sub-set whose content features are matched with the same style label l_k.

We show visualization details about graph based style matching in Figure **??**. We extract style and content features from Conv_4_1 layer in VGG-19. Due to several downsampling modules in VGG-19, the spatial resolution of the features is much smaller than that of the inputs. We label the spacial style feature pixels with their corresponding cluster labels and obtain the style cluster maps. According to the style cluster maps in Figure **??**, we find that style feature clustering grasps semantic information from style images.

After style matching in pixel level, we get the content-style matching map, which also reflects the semantic information, matching the content structures adaptively. Such an adaptive matching alleviates the wash-out artifacts, when the style is very simple or has large area of unified background. Then, we are able to conduct feature transform in each content-style pair group.

5.3.3 Multimodal Style Transfer

For each content-style pair group $\left\{F_c^{l_k}, F_s^{l_k}\right\}$, we first center them by subtracting their mean vectors $\mu\left(F_c^{l_k}\right)$ and $\mu\left(F_s^{l_k}\right)$ respectively. We conduct feature whitening and coloring as used in WCT [4].

$$F_{cs}^{l_k} = C_s W_c F_c^{l_k} + \mu\left(F_s^{l_k}\right), \tag{5.8}$$

where $W_c = E_c^{l_k}\left(D_c^{l_k}\right)^{-\frac{1}{2}}\left(E_c^{l_k}\right)^T$ is a whitening matrix and $C_s = E_s^{l_k}\left(D_s^{l_k}\right)^{\frac{1}{2}}\left(E_s^{l_k}\right)^T$ is a coloring matrix. $E_c^{l_k}$ and $D_c^{l_k}$ are diagonal matrix of eigenvalues and the orthogonal matrix of eigenvectors of the covariance matrix $F_c^{l_k}\left(F_c^{l_k}\right)^T$. For style covariance matrix $F_s^{l_k}\left(F_s^{l_k}\right)^T$, the corresponding matrices are $E_s^{l_k}$ and $D_s^{l_k}$. The reasons why we choose WCT to transfer features is its robustness and efficiency [4, 5]. More details about whitening and coloring are introduced in [4].

After feature transformation, we may also want to blend transferred features with content features as did in previous works (e.g., AdaIN [3] and WCT [4]). Most previous works have to blend the whole transferred features with a unified content-style trade-off, which treats different content parts equally and is not flexible to the real-world cases. Instead, our multimodal style representation and matching make it possible to adaptively blend features. Namely, for each content-style pair group, we blend them via

$$F_{cs}^{l_k} = \alpha_k F_{cs}^{l_k} + (1 - \alpha_k) F_c^{l_k}, \tag{5.9}$$

where $\alpha_k \in [0, 1]$ is a content-style trade-off for specific labeled content features. After blending all the features, we obtain the whole transferred features

$$F_{cs} = F_{cs}^{l_1} \cup F_{cs}^{l_2} \cup \cdots \cup F_{cs}^{l_k} \cup \cdots \cup F_{cs}^{l_K}. \tag{5.10}$$

F_{cs} is then fed into the decoder $D_{\theta_{dec}}(\cdot)$ to reconstruct the final output I_{cs}.

5.3.4 Implementation Details

Now, we specify the implementation details about our proposed MST. Similar to some previous works (e.g., AdaIN, WCT, DFR), we incorporate the pre-trained VGG-19 (up to Conv_4_1) [10]

as the encoder $E_{\theta_{enc}}(\cdot)$. We obtain decoder $D_{\theta_{dec}}(\cdot)$ by mirroring the encoder, whose pooling layers are replaced by nearest up-scaling layers.

To train the decoder, we use the pre-trained VGG-19 [10] to compute perceptual loss $l_{total} = l_c + \gamma l_s$, which combines content loss l_c and style loss l_s. We simply set the weighting constant as $\gamma = 10^{-2}$. Inspired by the loss designations in [50, 152, 3], we formulate content loss l_c as

$$l_c = \left\| \phi_{4\_1}(I_c) - \phi_{4\_1}(I_{cs}) \right\|_2, \tag{5.11}$$

where $\phi_{4\_1}(\cdot)$ extracts features at layer Conv_4_1 in VGG-19. We then formulate the style loss l_s as

$$\begin{aligned} l_s = &\sum_{i=1}^{4} \left(\left\| \mu\left(\phi_{i\_1}(I_s)\right) - \mu\left(\phi_{i\_1}(I_{cs})\right) \right\|_2 \right) \\ &+ \sum_{i=1}^{4} \left(\left\| \sigma\left(\phi_{i\_1}(I_s)\right) - \sigma\left(\phi_{i\_1}(I_{cs})\right) \right\|_2 \right), \end{aligned} \tag{5.12}$$

where $\phi_{i\_1}(\cdot)$ extracts features at layer Conv_i_1 in VGG-19. We use $\mu(\cdot)$ and $\sigma(\cdot)$ to compute the mean and standard deviation of the content and style features.

We train our network by using images from MS-COCO [153] and WikiArt [154] as content and style data respectively. Each dataset contains about 80,000 images. In each training batch, we randomly crop one pair of content and style images with the size of 256×256 as input. We implement our model with TensorFlow and apply Adam optimizer [116] with learning rate of 10^{-4}.

5.4 Discussions

To better position MST among the whole body of style transfer works, we further discuss and clarify the relationship between MST and some representative works.

Differences to CNNMRF. CNNMRF [6] extracts a pool of neural patches from style images, with which patch matching is used to match content. MST clusters style features into multiple sub-sets and matches style cluster centers with content feature points via graph cuts. CNNMRF uses smoothness prior for reconstruction, while MST uses it for style matching only. CNNMRF minimizes energy function to synthesize the results. MST generates stylization results with a decoder.

Differences to MT-Net. Both color and luminance are treated as a mixture of modalities in MT-Net [141]. MST obtains multimodal representation from style features via clustering. It should

Figure 5.7: Distance measurement investigation.

also be noted that MT-Net has to train new models for new style images. While, MST is designed for arbitrary style transfer with a single model.

Differences to WCT. In WCT [4], the decoder is trained by using only content data and loss. MST introduces additional style images for training. WCT uses multiple layers of VGG features and conducts multi-level coarse-to-fine stylization, which costs much more time and sometimes distorts structures. While, MST only transfers single-level content and style features. Consequently, even we set $K = 1$ in MST, we achieve more efficient stylizations.

5.5 Experiments

We conduct extensive experiments to validate the contributions of each component in our method, the effectiveness of our method, and the flexibility for user control.

5.5.1 Ablation Study

Distance Measurement. We first investigate the choice of the distance measurement, as it is critical for graph building. Here, we mainly investigate Euclidean distance and cosine distance (shown in Eq. (**??**)). As shown in Figure **??**, MST with Euclidean distance is affected by the huge background and may fail to transfer desired style patterns, leading to wash-out artifacts. This is mainly because there is no normalization for the deep features. As a result, the weight of style cluster center is proportional to its spatial proportion, weakening its semantic meaning. Instead, MST with cosine distance performs much better.

Discontinuity Preservation. In Figure **??**, we show the effectiveness of the smooth term $E_{smooth}(f)$ in Eq. (**??**). Specifically, we set λ as 0, 0.1, and 1 respectively. In real-world style transfer, people would like to smooth the facial area, as they do in the real photos. Here, we select one portrait to investigate how λ affects smoothness. When we set $\lambda = 0$, the energy function in

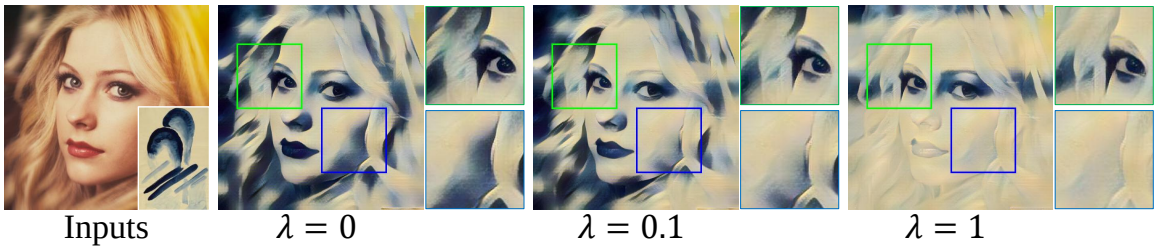


Figure 5.8: Discontinuity preservation investigation.

Figure 5.9: Feature transformation investigation.

Eq. (**??**) is minimized by only considering the data term. This would introduce some unpleasing artifacts in the facial area near edges and demonstrate the necessity of smooth term. However, large smooth term (e.g., $\lambda = 1$) would over-smooth the stylization results, decreasing the style diversity. A proper value of λ would not only keep better smoothness, but also preserve style diversity. We empirically set $\lambda = 0.1$ through the whole experiments.

Feature Transform. Here we set $K = 1$ in MST and compare with AdaIN [3] to show the effectiveness of WCT for feature transfer. As shown in Figure **??**, AdaIN produces some stroke artifacts in the smooth area. These artifacts may make the cloud unnatural. This is mainly because AdaIN uses the mean and variance of the whole content/style features. Instead, by using whitening and coloring in a more optimized way, our MST-1 achieves more natural stylization and cleaner smoothed area. As a result, we introduce whitening and coloring for feature transform.

5.5.2 Comparisons with Prior Arts

After investigating the effects of each component in our method, we turn to validate the effectiveness of our proposed MST. We compare with 7 state-of-the-art methods: method by Gatys *et al.* [46], CNNMRF [6], AdaIN [3], WCT [4], Deep feature reshuffle (DFR) [7], AvatarNet [8], and LST [5]. We obtain results using their official codes and default parameters, except for Gatys *et al.*[1].

[1]We use code from https://github.com/jcjohnson/neural-style

Figure 5.10: Visual comparison. MST ($K = 3$) and all compared methods use default parameters.

Qualitative Comparisons. We show extensive comparisons [2] in Figure **??**. Gatys *et al.* [46] transfer style with iterative optimization, being likely to falling in local minimum (e.g., 1st and 3rd columns). AdaIN [3] often produces less desired artifacts in the smooth area and some halation around the edges (e.g., 1st, 5th, and 6th columns). CNNMRF [6] may suffer from distortion effects and not preserve content structure well. Due to the usage of higher-level deep feature (e.g., Conv_5_1), WCT [4] would generate distorted results, failing to preserve the main content structures (e.g., 1st and 2nd columns). DRF [7] reconstructs the results by using the style patches, which could also distort the content structure (e.g., 1st and 3rd columns). In some cases (e.g., 5th, 6th, and

[2]The fifth content is from https://www.mordeo.org

Table 5.1: Percentage of the votes that each method received.

Method	Gatys	AdaIN	WCT	DFR	AvatarNet	MST
Perc./%	21.41	11.31	12.67	11.55	9.61	**33.45**

7th columns), some tiny style patterns (e.g., the eyes in the flowers and tree) would be copied to the results, leading unpleasing stylizations. AvatarNet [8] would introduce some less desired style patterns in the smooth area (e.g., 1st column) and also copy some style patterns in the results (e.g., 6th and 7th columns). LST [5] could generate very good result in some cases (e.g., 6th column). However, it may suffer from wash-out artifacts (e.g., 3rd and 4th columns) and halation around the edges (e.g., 5th column). These compared methods mainly treat the style patterns as a whole, lacking distinctive ability to style patterns.

Instead, we treat style features as multimodal presentations in high-dimension space. We match each content feature to its most related style cluster and adaptively transfer features according to the content semantic information. These advantages help explain why MST generates clearer results (e.g., 1st, 3rd, 5th, and 7th columns), performs more semantic matching with style patterns (e.g., 2nd column), and alleviates wash-out artifacts (e.g., 4th column). Such superior results demonstrate the effectiveness of our MST.

User Study. To further evaluate the 6 methods shown in Figure **??**, we conduct a user study like [4]. We use 15 content images and 30 style images. For each method, we use the released codes and default parameters to generate 450 results. 20 content-style pairs are randomly selected for each user. For each style-content pair, we display the stylized results of 6 methods on a web-page in random order. Each user is asked to vote the one that he/she like the most. Finally, we collect 2,000 votes from 100 users and calculate the percentage of votes that each method received. The results are shown in Table **??**, where our MST (K=3) obtains 33.45% of the total votes. It's much higher than that of Gatys *et al.* [46], whose stylization results are usually thought to be high-quality. This user study result is consistent with the visual comparisons (in Figure **??**) and further demonstrate the superior performance of our MST.

Efficiency. We further compare the running time of our methods with previous ones [46, 3, 4, 7, 8]. Table **??** gives the average time of each method on 100 image pairs with size of 512×512. All the methods are tested on a PC with an Intel i7-6850K 3.6 GHz CPU and a Titan Xp GPU. Our MST with different K performs relatively faster than methods by Gatys *et al.* [46] and DFR [7]. Even using SVD in CPU, MST-1 is faster than AvatarNet [8] and WCT [4]. It should be noted that WCT

Table 5.2: Running time (s) comparisons.

Method	Gatys	AdaIN	WCT	DFR	AvatarNet
Time (s)	116.46	0.09	0.92	54.32	0.33
Method	MST-1	MST-2	MST-3	MST-4	MST-5
Time (s)	0.20	1.10	1.40	1.97	2.27

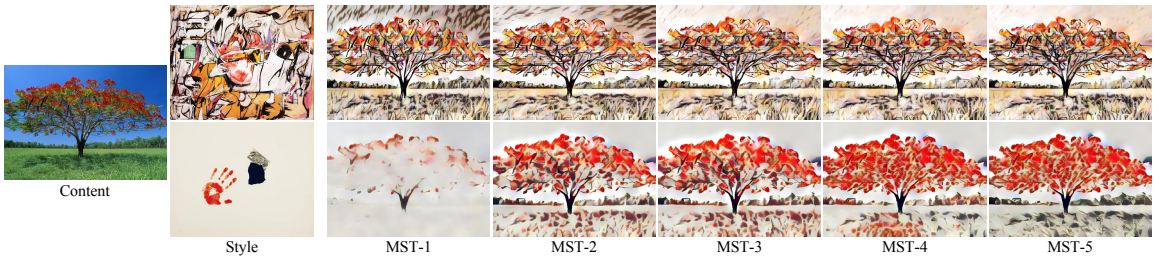


Figure 5.11: Style cluster number investigation. Same content image with complex and simple style images.

conducts multi-level stylization, which costs much more time than that of MST-1. MST-K ($K > 1$) becomes much slower with larger K. This is mainly because our cluster operation is executed in CPU and consumes much more time. On the other hand, although MST with larger K would consume more time, its stylized results would be very robust. So, in general, we don't have to choose very large K, of which we give more details about the effects later.

5.5.3 Style Cluster Number

We investigate how style cluster number K affects the stylization in Figure **??**. When $K = 1$, our MST performs style transfer by taking the whole style features equally, resulting in either very complex (1st row) or simple (2nd row) stylizations. These results are not consistent with the content structures and lack flexibility, leading to unpleasing feelings to users. Instead, we can produce multiple results with different K. When we enlarge K with multimodal style representation, stylization results would either throw unnecessary style patterns (1st row) or introduce more matched style patterns (2nd row). The stylizations become more matched with the content structures. This is mainly because multimodal style representation allows distinctive and adaptive treatment for the style patterns. More important, MST reconstructs several stylization results with different K, providing multiple selections for the users.

Figure 5.12: Multi-style transfer. MST treats patterns from different style images distinctively and transfers them adaptively.

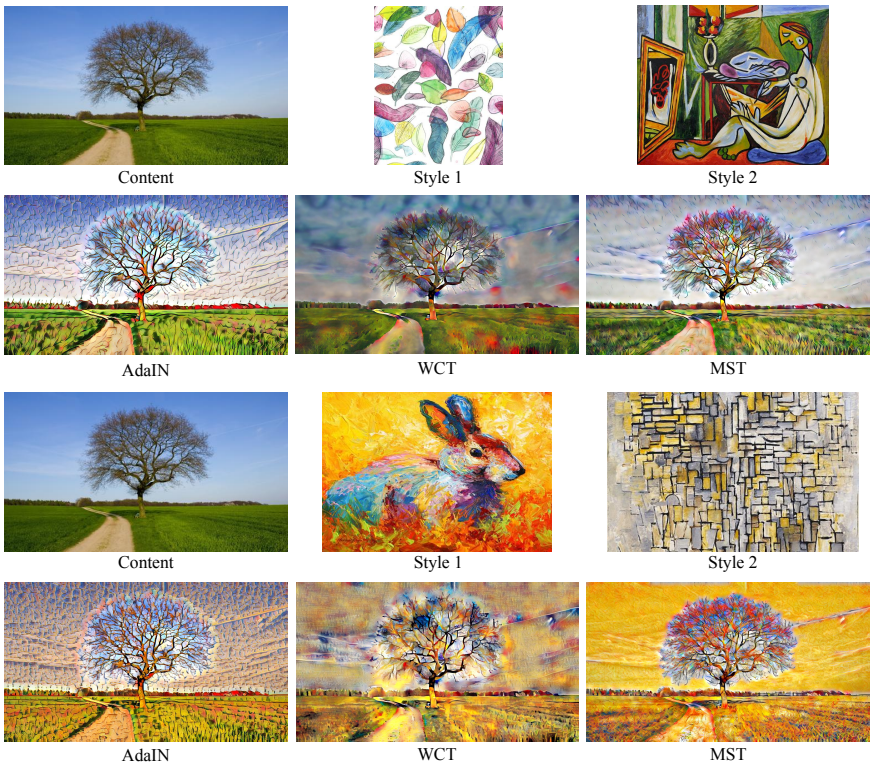


Figure 5.13: Multi-style transfer. We set $K = 3$ in MST. Our MST treats patterns from different style images distinctively and transfer them adaptively according to the specific content structures.

5.5.4 Adaptive Multi-Style Transfer

Most previous style transfer methods enable style interpolation, which blends the content image with a set of weighted stylizations. However, we don't fix the weights for each style image, but adaptively interpolate the style patterns to the content. As shown in Figure **??**, the content image [3] is stylized by two style images simultaneously. We use AdaIN [3] and WCT [4] for reference (because it's not strictly fair comparisons) by setting equal weight for each style image. In Figure **??**, AdaIN and WCT suffer from wash-out artifacts. While, our MST preserves the content structures well. MST transfers more portrait hair style to the cat body and more cloud style to the cat eyes and green leaves.

[3] It's from https://wallpaperstream.com

Figure 5.14: Generalization of MST to AdaIN [3].

Our adaptive multi-style transfer is also similar to spatial control in previous methods [3, 4]. But, they need additional manually designed mask as input, consuming more user efforts. Instead, MST automatically allows good matching between content and style features.

We show other multi-style transfer examples in Figure **??**. Our adaptive multi-style transfer is also similar to spatial control in previous methods [3, 4]. But, they need additional manually designed mask as input, consuming more user effort and lacks flexibility to specific content image. Instead, MST automatically allows good matching between content and style features. More analyses are given in the figure captions.

5.5.5 Generalization of MST

We further investigate the generalization of our proposed MST to improve some existing style transfer methods. Here, we take the popular AdaIN [3] as an example. We apply style clustering and graph based style matching to AdaIN, which is then denoted as "AdaIN + MST-K". As shown in Figure **??**, AdaIn may distort some content structures (e.g., mouth) by switching the global mean and standard deviation between style and content features. When we cluster the style feature into K sub-sets and match them with content features via graph cuts, such a phenomenon can be obviously alleviated (see 3rd and 4th columns in Figure **??**). According to these observations and analyses, we can learn that our MST can be generalized and will benefit to some other existing style transfer methods.

5.6 Summary

We propose multimodal style representation to model the complex style distribution. We then formulate the style matching problem as an energy minimization one and solve it using

our proposed graph based style matching. As a result, we propose multimodal style transfer to transform features in a multimodal way. We treat the style patterns distinctively, and also consider the semantic content structure and its matching with style patterns. We also investigate that MST can be generalized to some existing style transfer methods. We conduct extensive experiments to validate the effectiveness, robustness, and flexibility of MST.

Chapter 6

Large-Factor Painting Texture Hallucination

6.1 Background and Motivation

Image super-resolution (SR) aims to reconstruct high-resolution (HR) output with details from its low-resolution (LR) counterpart. Super-resolution for digital painting images has important values in both culture and research aspects. Many historical masterpieces were damaged and their digital replications are in low-resolution (LR), low-quality due to technological limitation in old days. Recovery of their fine details is crucial for maintaining and protecting human heritage. It is also a valuable research problem for computer scientists to restore high-resolution (HR) painting due to the rich content and texture of paintings in varying scales. A straightforward way to solve this problem is to borrow some knowledge from natural image SR [76, 11, 12], which however is not enough.

Super-resolving painting images is particularly challenging as vary large upscaling factors ($8\times$, $16\times$, or even larger) are required to recover the brush and canvas details of artworks, so that a viewer can fully appreciate the aesthetics as from the original painting. One state-of-the-art (SOTA) single image super-resolution (SISR) method RCAN [11] can upscale input with large scales with high PSNR values. But, it would suffer from over-smoothing artifacts, because most high-frequency components (e.g., textures) have been lost in the input. It's hard to recover high-frequency information from LR input directly. Some reference-based SR (Ref-SR) methods try to transfer high-quality textures from another reference image. One SOTA Ref-SR method SRNTT [12] matches features between input and reference. Then, feature swapping is conducted in a multi-level way. SRNTT

performs well in the texture transfer. However, the results of SRNTT could be affected by the reference obviously. Also, it's hard for SRNTT to transfer high-quality textures when scaling factor becomes larger.

Based on the analyses above, we try to transfer detailed textures from reference images and also tackle with large scaling factors. Fortunately, there is a big abundance of existing artworks scanned in high-resolution, which provide the references for the common texture details shared among most paintings. Therefore, in this paper, we formulate a Ref-SR problem setting for digital paintings with large upscaling factors.

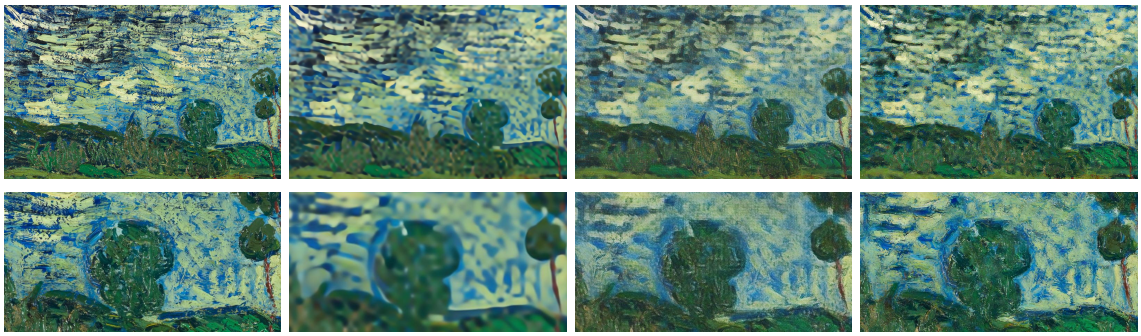
To this end, we collect a large-scale high-quality dataset PaintHD for oil painting images with diverse contents and styles. We explore new deep network architectures with efficient texture transfer (i.e., match feature in smaller scale and swap feature in fine scale) for large upscaling factors. We also design wavelet-based texture loss and degradation loss to achieve high perceptual quality and fidelity at the same time. The network architecture helps tackle large scaling factors better. The wavelet-based texture loss and degradation loss contribute to achieve better visual results. Our proposed method can hallucinate realistic details based on the given reference images, which is especially desired for large factor image upscaling. Compared to the previous SOTA SISR and Ref-SR methods, our proposed method achieves significantly improved quantitative (perceptual index (PI) [94]) and visual results, which are further verified in our human subjective evaluation (i.e., user study). In Figure 5.1, we compare with other state-of-the-art methods for large scaling factor (e.g., $16\times$). We can see our method can transfer more vivid and faithful textures.

In summary, the main contributions of this work are:

- We proposed a reference-based image super-resolution framework for large upscaling factors (e.g., $8\times$ and $16\times$) with novel training objectives. Specifically, we proposed wavelet texture loss and degradation loss, which allow to transfer more detailed and vivid textures.

- We collected a new digital painting dataset PaintHD with high-quality images and detailed meta information, by considering both physical and resolution sizes. Such a high-resolution dataset is suitable for painting SR.

- We achieved significantly improved quantitative and visual results over previous single image super-resolution (SISR) and reference based SR (Ref-SR) state-of-the-arts. A new technical direction is opened for Ref-SR with large upscaling factor on painting images.



HR RCAN [11] SRNTT [12] Ours

Figure 6.1: Visual comparisons for the scaling factor of 16× (the first row) and zoom-in patches (the second row). We compare with state-of-the-art SISR and Ref-SR methods.

6.2 Related Work

Recent work on deep-learning-based methods for image SR [1, 125, 103, 11, 86, 12] is clearly outperforming more traditional methods [124, 155, 65] in terms of either PSNR/SSIM or visual quality. Here, we focus on the former for conciseness.

Single Image Super-Resolution. Single image super-resolution (SISR) recovers a high-resolution image directly from its low-resolution (LR) counterpart. The pioneering SRCNN proposed by Dong *et al*. [76], made the breakthrough of introducing deep learning to SISR, achieving superior performance than traditional methods. Inspired by this seminal work, many representative works [109, 78, 82, 101, 102, 1, 11] were proposed to further explore the potential of deep learning and have continuously raised the baseline performance of SISR. In SRCNN and followups VDSR [78] and DRCN [82], the input LR image is upscaled to the target size through interpolation before fed into the network for recovery of details. Later works demonstrated that extracting features from LR directly and learning the upscaling process would improve both quality and efficiency. For example, Dong *et al*. [101] provide the LR image directly to the network and use a deconvolution for upscaling. Shi *et al*. [102] further speed up the upscaling process using sub-pixel convolutions, which became widely adopted in recent works. Current state-of-the-art performance is achieved by EDSR [1] and RCAN [11]. EDSR takes inspiration from ResNet [19], using long-skip and sub-pix convolutions to achieve stronger edge and finer texture. RCAN introduced channel attention to learn high-frequency information.

Once larger upscaling factors were achievable, e.g., 4×, 8×, many empirical studies [103, 125, 12] demonstrated that the commonly used quality measurements PSNR and SSIM proved to

be not representative of visual quality, i.e., higher visual quality may result in lower PSNR; a fact first investigated by Johnson *et al*. [50] and Ledig *et al*. [103]. The former investigated perceptual loss using VGG [10], while the later proposed SRGAN by introducing GAN [126] loss into SISR, which boosted significantly the visual quality compared to previous works. Based on SRGAN [103], Sajjadi *et al*. [125] further adopted texture loss to enhance textural reality. Along with higher visual quality, those GAN-based SR methods also introduce artifacts or new textures synthesized depending on the content, which would contribute to increased perceived fidelity.

Although SISR has been studied for decades, it is still limited by its ill-posed nature, making it difficult to recover fine texture detail for upscaling factors of $8\times$ or $16\times$. So, most existing SISR methods are limited to a maximum of $4\times$. Otherwise, they suffer serious degradation of quality. Works that attempted to achieve $8\times$ upscaling, e.g., LapSRN [84] and RCAN [11], found visual quality would degrade quadratically with the increase of upscaling factor.

Reference-based Super-Resolution. Different from SISR, reference-based SR (Ref-SR) methods attempt to utilize self or external information to enhance the texture. Freeman *et al*. [59] proposed the first work on Ref-SR, which replaced LR patches with fitting HR ones from a database/dictionary. [60, 61] considered the input LR image itself as the database, from which references were extracted to enhance textures. These methods benefit the most from repeated patterns with perspective transformation. Light field imaging is an area of interest for Ref-SR, where HR references can be captured along the LR light field, just with a small offset. Thus, making easier to align the reference to the LR input, facilitating the transfer of high-frequency information in [62, 63]. CrossNet [64] took advantage of deep learning to align the input and reference by estimating the flow between them and achieved SOTA performance.

A more generic scenario for Ref-SR is to relax the constraints on references, i.e., the references could present large spacial/color shift from the input. More extremely, references and inputs could contain unrelated content. Sun *et al*. [65] used global scene descriptors and internet-scale image databases to find similar scenes that provide ideal example textures. Yue *et al*. [66] proposed a similar idea, retrieving similar images from the web and performing global registration and local matching. Recent works [67, 12] leveraged deep models and significantly improved Ref-SR performance, e.g., visual quality and generalization capacity.

Our proposed method further extends the feasible scaling factor of previous Ref-SR methods from $4\times$ to $16\times$. More importantly, as oppose to the previous approach [12], which transfers the high-frequency information from reference as a style transfer task, we conduct texture transfer only in high-frequency band, which reduces the transfer effect on the low-frequency content.

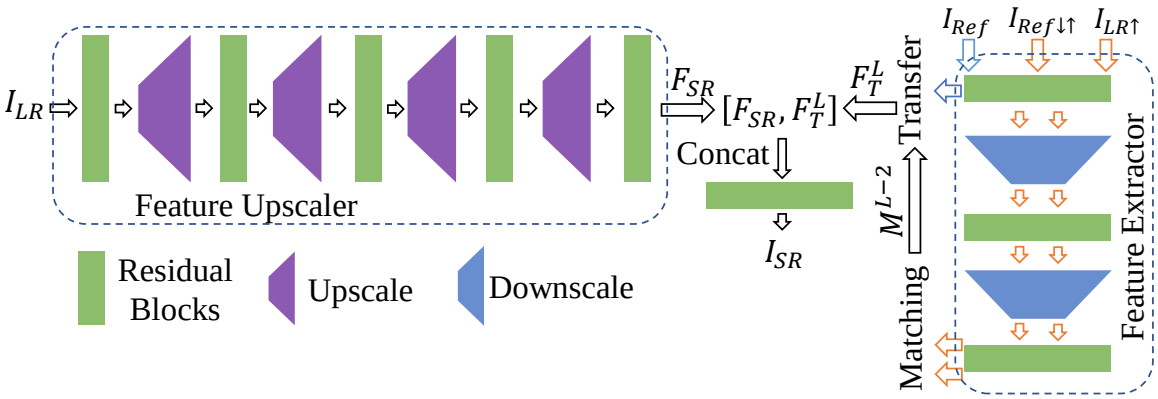


Figure 6.2: The pipeline of our proposed method.

6.3 Painting Texture Hallucination

We aim to hallucinate the SR image I_{SR} for large scaling factor s from its low-resolution (LR) input I_{LR} and transfer highly detailed textures from high-resolution (HR) reference I_{Ref}. However, most previous Ref-SR methods [64, 12] could mainly handle relatively small scaling factors (e.g., $\times 4$). To achieve visually pleasing I_{SR} with larger scaling factors, we firstly build a more compact network (see Figure 5.2) and then apply novel loss functions to the output.

6.3.1 Pipeline

We first define L levels according to scaling factor s, where $s = 2^L$. Inspired by SRNTT [12], we conduct texture swapping in the feature space to transfer highly detailed textures to the output (Figure 5.2). The feature upscaler acts as the mainstream of upscaling the input LR image. Meanwhile, the reference feature that carries richer texture is extracted by the deep feature extractor. At the finest layer (largest scale) the reference feature is transferred to the output.

As demonstrated in recent works [85, 11], the batch normalization (BN) layers commonly used in deep models for stabilizing the training process turns to degrade the SR performance. Therefore, we avoid BN layer in our feature upscaling model. More importantly, the GPU memory usage is largely reduced, as the BN layer consumes similar amount of GPU memory as convolutional layer [1].

To efficiently transfer high-frequency information from the reference, we swap features at the finest level L, where the reference features are swapped according to the local feature matching between the input and reference. Since patch matching is time-consuming, it is conducted at lower

level (small spatial size), i.e., we obtain feature matching information M^{L-2} in level $L-2$ via

$$M^{L-2} = H_M^{L-2}\left(\phi^{L-2}\left(I_{LR\uparrow}\right), \phi^{L-2}\left(I_{Ref\downarrow\uparrow}\right)\right), \tag{6.1}$$

where $H_M^{L-2}(\cdot)$ denotes feature matching operation in level $L-2$. $\phi^{L-2}(\cdot)$ is a neural feature extractor (e.g., VGG19 [10]) matching the same level. $I_{LR\uparrow}$ is upscaled by Bicubic interpolation with scaling factor s. To match the frequency band of $I_{LR\uparrow}$, we first downscale and then upscale it with scaling factor s. For each patch from $\phi^{L-2}\left(I_{LR\uparrow}\right)$, we could find its best matched patch from $\phi^{L-2}\left(I_{Ref\downarrow\uparrow}\right)$ with highest similarity.

Then, using the matching information M^{L-2}, we transfer features at level L and obtain the new feature F_T^L via

$$F_T^L = H_T^L\left(\phi^L\left(I_{Ref}\right), M^{L-2}\right), \tag{6.2}$$

where $H_T^L(\cdot)$ denotes feature transfer operation. $\phi^L\left(I_{Ref}\right)$ extracts neural feature from the high-resolution reference I_{Ref} at level L.

On the other hand, we also extract deep feature from the LR input I_{LR} and upscale it with scaling factor s. Let's denote the upscaled input feature as F_{SR} and the operation as $H_{FSR}(\cdot)$, namely $F_{SR} = H_{FSR}(I_{LR})$. To introduce the transferred feature F_T^L into the image hallucination, we fuse F_T^L and F_{SR} by using residual learning, and finally reconstruct the output I_{SR}. Such a process can be expressed as follows

$$I_{SR} = H_{Rec}\left(H_{Res}\left(\left[F_{SR}, F_T^L\right]\right) + F_{SR}\right), \tag{6.3}$$

where $[F_{SR}, F_T^L]$ refers to channel-wise concatenation, $H_{Res}(\cdot)$ denotes several residual blocks, and $H_{Rec}(\cdot)$ denotes a reconstruction layer.

We can already achieve super-resolved results with larger scaling factors by using the above simplifications and improvements. The ablation study in Section 5.5.1 would demonstrate the effectiveness of the simplified pipeline. However, we still aim to transfer highly-detailed texture from reference even in such challenging cases (i.e., very large scaling factors). To achieve this goal, we further propose wavelet texture and degradation losses.

6.3.2 Wavelet Texture Loss

Motivation. Textures are mainly composed of high-frequency components. LR images contain less high-frequency components, when the scaling factor goes larger. If we apply the loss

functions (including texture loss) on the color image space, it's still hard to recover or transfer more high-frequency ones. However, if we pay more attention to the high-frequency components and relax the reconstruction of color image space, such an issue could be alleviated better. Specifically, we aim to transfer as many textures as possible from reference by applying texture loss on the high-frequency components. Wavelet is a proper way to decompose the signal into different bands with different frequency levels.

Haar wavelet. Inspired by the excellent WCT^2 [156], where a wavelet-corrected transfer was proposed, we firstly apply Haar wavelet to obtain different components. The Haar wavelet transformation has four kernels, $\left\{LL^T, LH^T, HL^T, HH^T\right\}$, where L^T and H^T denote the low and high pass filters,

$$L^T = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\end{bmatrix},\ H^T = \frac{1}{\sqrt{2}}\begin{bmatrix}-1 & 1\end{bmatrix}. \tag{6.4}$$

As a result, such a wavelet operation would split the signal into four channels, capturing low-frequency and high-frequency components. We denote the extraction operations for these four channels as $H_W^{LL}(\cdot)$, $H_W^{LH}(\cdot)$, $H_W^{HL}(\cdot)$, and $H_W^{HH}(\cdot)$ respectively. Then, we aim to pay more attention to the recovery of high-frequency components with the usage of wavelet texture loss.

Wavelet texture loss. As investigated in WCT^2 [156], in Haar wavelet, the low-pass filter can extract smooth surface and parts of texture and high-pass filters capture higher frequency components (e.g., horizontal, vertical, and diagonal edge like textures).

Ideally, it's a wise choice to apply texture loss on each channel split by Haar wavelet. However, as we calculate texture loss in different scales, such a choice would suffer from very heavy GPU memory usage and running time. Moreover, as it's very difficult for the network to transfer highly detailed texture with very large scaling factors, focusing on the reconstruction of more desired parts would be a better choice. Consequently, we propose a wavelet texture loss with HH kernel and formulate it as follows

$$\mathcal{L}_{tex} = \sum_l \lambda_l \left\| Gr\left(\phi^l\left(H_W^{HH}\left(I_{SR}\right)\right)\right) - Gr\left(F_T^l\right)\right\|_F, \tag{6.5}$$

where $H_W^{HH}(\cdot)$ extracts high-frequency component from the upscaled output I_{SR} with HH kernel. F_T^l is the transferred feature in feature map space of ϕ^l. $Gr(\cdot)$ calculates the Gram matrix for each level l, where λ_l is the corresponding normalization weight. $\|\cdot\|_F$ denotes Frobenius norm.

As shown in Eq. (5.5), we mainly focus on the texture reconstruction of higher frequency components, which would transfer more textures with somehow creative ability. Then, we further relax the reconstruction constraint by proposing a degradation loss.

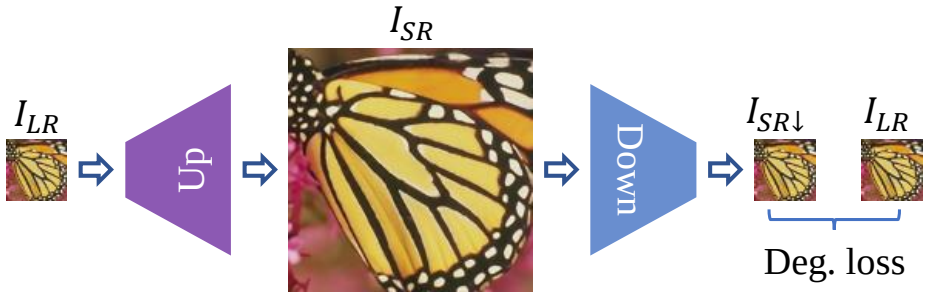


Figure 6.3: The illustration of our proposed degradation loss. We try to minimize the degradation loss $\mathcal{L}_{deg}$ between the downscaled output $I_{SR\downarrow}$ and the original input I_{LR}.

6.3.3 Degradation Loss

Motivation. Most previous single image SR methods (e.g., RCAN [11]) mainly concentrate on minimizing the loss between the upscaled image I_{SR} and ground truth I_{GT}. For small scaling factors (e.g., $\times 2$), those methods would achieve excellent results with very high PSNR values. However, when the scaling factor goes very large (e.g., $16\times$), the results of those methods would suffer from heavy smoothing artifacts and lack favorable textures (see Figure 5.1). On the other hand, as we try to transfer textures to the results as many as possible, emphasizing on the overall reconstruction in the upscaled image may also smooth some transferred textures. To alleviate such texture oversmoothing artifacts, we turn to additionally introduce the LR input I_{LR} into network optimization.

Degradation loss. Different from image SR, which is more challenging to obtain favorable results, image downscaling could be relatively easier. It's possible to learn a degradation network H_D, that maps the HR image to an LR one. We train such a network by using HR ground truth I_{GT} as input and try to minimize the loss between its output $H_D(I_{GT})$ and the LR counterpart I_{LR}.

With the degradation network H_D, we are able to mimic the degradation process from I_{GT} to I_{LR}, which can be a many-to-one case. Namely, there exists many upscaled images corresponding to the original LR image I_{LR}, which helps to relax the constraints on the reconstruction. To make use of this property, we try to narrow the gap between the downscaled output $I_{SR\downarrow}$ and the original LR input I_{LR}. As shown in Figure 5.3, we formulate it as a degradation loss

$$\mathcal{L}_{deg} = \|I_{SR\downarrow} - I_{LR}\|_1 = \|H_D(I_{SR}) - I_{LR}\|_1, \tag{6.6}$$

where $I_{SR\downarrow}$ denotes the downscaled image from I_{SR} with scaling factor s and $\|\cdot\|_1$ denotes ℓ_1-norm. With the proposed loss functions, we further give details about the implementation.

6.3.4 Differences with SRNTT

As our work is developed from SRNTT [12], we would like to highlight two main differences between SRNTT and ours. The first one is the design of the pipeline. Specifically, we remove all the BN layers, which helps reduce GPU memory usage and also improve performance (it's investigated in EDSR [1]). SRNTT conducts feature swap in multi-level, which may achieve higher PSNR/SSIM values, but hurt the perceptual quality. Instead, our method only conduct feature swap in the fine scale, which further saves running time and transfers textures more easily. The comparison between SRNTT and ours with $\mathcal{L}_{rec}$ only demonstrates the effectiveness of our pipeline. More details are given in Section 3.4.2. The third one is that new loss functions $\mathcal{L}_{tex}$ and $\mathcal{L}_{deg}$ are used in our method. Such loss functions help to transfer more textures from high-quality reference, resulting in better perceptual quality. Compared with SRNTT, our method achieves better quantitative performance (e.g., perceptual index (PI) and user study scores) and visual output (e.g., more faithful textures to the ground truth).

6.3.5 Implementation Details

Network details. An overview of the proposed network structure is illustrated in Figure 5.4. All batch normalization (BN) layers are removed from the residual blocks (RBs) since BN layers shrink range flexibility from networks by normalizing the features [157, 1]. Therefore, the structure of a residual block is Conv-ReLU-Conv with a short cut. The convolution kernel is set to be 3×3 for all convolutional (Conv) layers, and zero-padding is conducted to preserve the feature map size after convolution. Instead, the concatenation layer adopts a kernel size of 1×1. The activation function is ReLU (except for the output layer that uses tanh), and the number of channels at each intermedia convolutional layers is set to be 64. The upscaling layers employ the sub-pixel convolution [102]. The network difference between 8× and 16× upscaling is that the first upscaling layer will perform 2× and 4× upscaling, respectively.

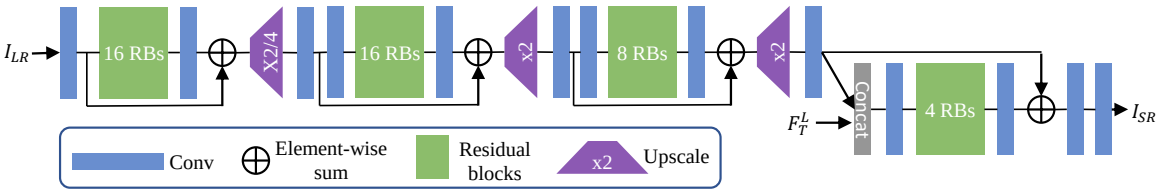
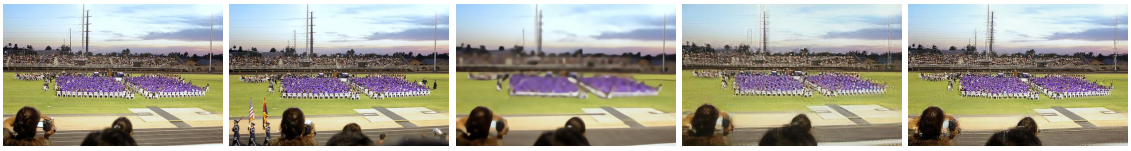


Figure 6.4: Illustration of the proposed network for large-scale painting super-resolution. "Conv" denotes convolutional layer, and "RBs" indicates residual blocks removed the batch normalization.

Loss functions. We also adopt another three common loss functions [50, 103, 125, 12]:



(a) HR (b) Reference (c) RCAN (d) SRNTT (e) Ours

Figure 6.5: Visual results (8×) of RCAN [11], SRNTT [12], and our method on CUFED5. Our result is visually more pleasing than others, and generates plausible texture details.

reconstruction ($\mathcal{L}_{rec}$), perceptual ($\mathcal{L}_{per}$), and adversarial ($\mathcal{L}_{adv}$) losses. We briefly introduce them as follows.

$$\mathcal{L}_{rec} = \left\| I_{SR} - I_{GT} \right\|_1 , \tag{6.7}$$

$$\mathcal{L}_{per} = \frac{1}{N_{5,1}} \sum_{i=1}^{N_{5,1}} \left\| \phi_i^{5,1} \left(I_{SR} \right) - \phi_i^{5,1} \left(I_{GT} \right) \right\|_F , \tag{6.8}$$

where $\phi^{5,1}$ extracts $N_{5,1}$ feature maps from 1-st convolutional layer before 5-th max-pooling layer of the VGG-19 [10] network. $\phi_i^{5,1}$ is the i-th feature map.

We also adopt WGAN-GP [158] for adversarial training [126], which can be expressed as follows

$$\min_G \max_D \mathbb{E}_{I_{GT} \sim \mathbb{P}_r} \left[D(I_{GT}) \right] - \mathbb{E}_{I_{SR} \sim \mathbb{P}_g} \left[D \left(I_{SR} \right) \right] , \tag{6.9}$$

where G and D denote generator and discriminator respectively, and $I_{SR} = G(I_{LR})$. $\mathbb{P}_r$ and $\mathbb{P}_g$ represent data and model distributions. For simplicity, here, we mainly focus on the adversarial loss for generator and show it as follows

$$\mathcal{L}_{adv} = -\mathbb{E}_{I_{SR} \sim \mathbb{P}_g} \left[D \left(I_{SR} \right) \right] . \tag{6.10}$$

Training. The weights for $\mathcal{L}_{rec}$, $\mathcal{L}_{tex}$, $\mathcal{L}_{deg}$, $\mathcal{L}_{per}$, and $\mathcal{L}_{adv}$ are 1, 10^{-4}, 1, 10^{-4}, and 10^{-6} respectively. To stabilize the training process, we pre-train the network for 2 epochs with $\mathcal{L}_{rec}$ and $\mathcal{L}_{tex}$. Then, all the losses are applied to train another 20 epochs. We implement our model with TensorFlow and apply Adam optimizer [116] with learning rate 10^{-4}.

6.4 Dataset

For large upscaling factors, e.g., 8× and 16×, input images with small size, e.g., 30×30, but with rich texture in its originally HR counterpart will significantly increase the arbitrariness/smoothness for texture recovery because fewer pixels result in looser content constraints

Figure 6.6: Examples from our collected PaintHD dataset.

on the texture recovery. Existing datasets for Ref-SR are unsuitable for such large upscaling factors (see Figure 5.5). Therefore, we collect a new dataset of high-resolution painting images that carry rich and diverse stroke and canvas texture.

The new dataset, named PaintHD, is sourced from the Google Art Project [159], which is a collection of very large zoom-able images. In total, we collected over 13,600 images, some of which achieve gigapixel. Intuitively, an image with more pixels does not necessarily present finer texture since the physical size of the corresponding painting may be large as well. To measure the richness of texture, the physical size of paintings is considered to calculate pixel per inch (PPI) for each image. Finally, we construct the training set consisting of 2,000 images and the testing set of 100 images with relatively higher PPI. Figure 5.6 shows some examples of PaintHD, which contains abundant textures.

To further evaluate the generalization capacity of the proposed method, we also test on the CUFED5 [12] dataset, which is designed specifically for Ref-SR validation. There are 126 groups of samples. Each group consists of one HR image and four references with different levels of similarity to the HR image. For simplicity, we adopt the most similar reference for each HR image to construct the testing pairs. The images in CUFED5 are of much lower resolution, e.g., 500×300, as compared to the proposed PaintHD dataset.

6.5 Experimental Results

We conduct extensive experiments to validate the contributions of each component in our method. We demonstrate the effectiveness of our method by comparing with other state-of-the-art methods quantitatively and visually. We also provide more details, results, and analyses in supplementary material.

(a) 8× (b) 16×

Figure 6.7: Visual comparisons between SRNTT and ours by using $\mathcal{L}_{rec}$ only.

6.5.1 Ablation Study

Effect of Our Pipeline. We firstly try to demonstrate the effectiveness of our simplified pipeline and also support our analyses about the difference with SRNTT [12] in Section 5.3.4. We re-train SRNTT and our model by using PaintHD and reconstruction loss $\mathcal{L}_{rec}$ only with scaling factors 8× and 16×. We show visual comparisons about 8× in Figure 5.7(a). We can see the color of the background by our method is more faithful to the ground truth. Furthermore, our method achieves sharper result than that of SRNTT. Such a observation can be much clearer, when the scaling factor becomes 16× (e.g., see Figure 5.7(b)). As a result, our method transfer more textures and achieve shaper results. We also provide quantitative results about 'SRNTT-$\mathcal{L}_{rec}$' and 'Ours–$\mathcal{L}_{rec}$' in Table 5.1, where we'll give more details and analyses. In summary, these comparisons demonstrate the effectiveness of our simplified pipeline.

Effect of Wavelet Texture Loss. The wavelet texture loss is imposed on the high-frequency band of the feature maps, rather than directly applying on raw features like SRNTT [12] and traditional style transfer [46]. Comparison between the wavelet texture loss and tradition texture loss is illustrated in Figure 5.8. To highlight the difference, weights on texture losses during training are increased by 100 times as compared to the default setting in Section 5.3.5. Let's compare Figures 5.8(c) and 5.8(d), the result without wavelet is significantly affected by the texture/color from the reference (Figure 5.8(b)), lost identity to the input content. By contrast, the result with wavelet still preserves similar texture and color to the ground truth (Figure 5.8(a)).

Effect of Degradation Loss. To demonstrate the effectiveness of our proposed degradation loss $\mathcal{L}_{deg}$, we train one of our models with $\mathcal{L}_{rec}$ only and another same model with $\mathcal{L}_{rec}$ and $\mathcal{L}_{deg}$

(a) HR (b) Reference (c) w/o Wavelet (d) w/ Wavelet

Figure 6.8: Comparison of super-resolved results with and without wavelet.

with scaling factor $8\times$. We show the visual comparison in Figure 5.9, where we can see result with $\mathcal{L}_{rec}$ only would suffer from some blurring artifacts (see Figure 5.9(c)). While, in Figure 5.9(d), $\mathcal{L}_{deg}$ helps suppress such artifacts to some degree. This is mainly because the degradation loss $\mathcal{L}_{deg}$ alleviates the training difficulty in the ill-posed image SR problem. Such observations not only demonstrate the effectiveness of $\mathcal{L}_{deg}$, but also are consistent with our analyses in Section 5.3.3.

6.5.2 Quantitative Results

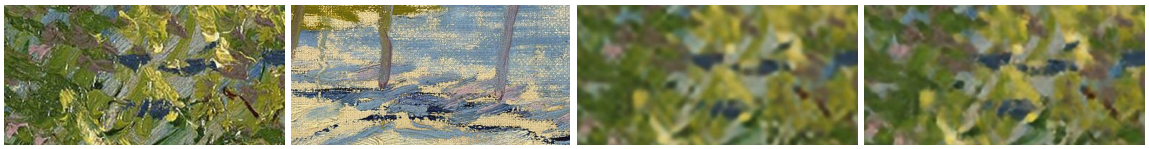
We compare our method with state-of-the-art SISR and Ref-SR methods[1]. The SISR methods are EDSR [1], RCAN [11], and SRGAN [103], where RCAN [11] achieved state-of-the-art performance in terms of PSNR (dB). Due to limited space, we only introduce the state-of-the-art Ref-SR method SRNTT [12] for comparison. However, most of those methods are not originally designed for very large scaling factors. Here, to make them suitable for $8\times$ and $16\times$ SR, we adopt them with some modifications. In $8\times$ case, we use RCAN [11] to first upscale the input I_{LR} by $2\times$. The upscaled intermediate result would be the input for EDSR and SRGAN, which then upscale the result by $4\times$. Analogically, in $16\times$ case, we use RCAN to first upscale I_{LR} by $4\times$. The intermediate result would be def into RCAN, EDSR, and SRGAN, which further upscale it by $4\times$. For SRNTT and our method, we would directly upscale the input by $8\times$ or $16\times$. SRNTT is re-trained with our PaintHD training data by its authors.

We not only compute the pixel-wise difference with PSNR and SSIM [114], but also evaluate perceptual quality with perceptual index (PI) [94] by considering Ma's score [160] and NIQE [161]. Specifically, PI = 0.5((10 - Ma) + NIQE). Lower PI value reflects better perceptual quality. We show quantitative results in Table 5.1, where we have some interesting and thought-

[1]We use implementations from
EDSR: https://github.com/thstkdgus35/EDSR-PyTorch
RCAN: https://github.com/yulunzhang/RCAN
SRGAN: https://github.com/tensorlayer/srgan
SRNTT: https://github.com/ZZUTK/SRNTT

Table 6.1: Quantitative results (PSNR/SSIM/PI) of different SR methods for $8\times$ and $16\times$ on two datasets: CUFED5 [12] and our collected PaintHD. The methods are grouped into two categories: SISR (top group) and Ref-SR (bottom). We highlight the best results for each case. 'Ours-$\mathcal{L}_{rec}$' denotes our method by using only $\mathcal{L}_{rec}$.

Data	CUFED5		PaintHD	
Scale	$8\times$	$16\times$	$8\times$	$16\times$
Bicubic	21.63/0.572/9.445	19.75/0.509/10.855	23.73/0.432/9.235	22.33/0.384/11.017
EDSR	23.02/0.653/7.098	20.70/0.548/8.249	24.42/0.477/7.648	22.90/0.405/8.943
RCAN	23.37/0.666/6.722	20.71/0.548/8.188	24.43/0.478/7.448	22.91/0.406/8.918
SRGAN	22.93/0.642/5.714	20.54/0.537/7.367	24.21/0.466/7.154	22.75/0.396/7.955
SRNTT-$\mathcal{L}_{rec}$	22.34/0.612/7.234	20.17/0.528/8.373	23.96/0.449/7.992	22.47/0.391/8.464
SRNTT	21.08/0.548/2.502	19.09/0.418/2.956	22.90/0.377/3.856	21.48/0.307/4.314
Ours-$\mathcal{L}_{rec}$	22.40/0.635/4.520	19.71/0.526/5.298	24.02/0.461/5.253	22.13/0.375/5.815
Ours	20.36/0.541/2.339	18.51/0.442/2.499	22.49/0.361/3.670	20.69/0.259/4.131



(a) HR (b) Reference (c) w/o $\mathcal{L}_{deg}$ (d) w/ $\mathcal{L}_{deg}$

Figure 6.9: Comparison of super-resolved results ($8\times$) with and without degradation loss.

provoking observations.

First, SISR methods would obtain higher PSNR and SSIM values than those of Ref-SR methods. This is reasonable because SISR methods mainly target to minimize MSE, which helps to pursue higher PSNR values. But, when the scaling factor goes to larger (e.g., $16\times$), the gap among SISR methods also becomes very smaller. It means that it would be difficult to distinguish the performance between different SISR methods by considering PSNR/SSIM.

Based on the observations and analyses above, we conclude that we should turn to other more visually-perceptual ways to evaluate the performance of SR methods, instead of only depending on PSNR/SSIM values. So, we further evaluate the PI values of each SR method. We can see 'Ours-$\mathcal{L}_{rec}$' achieves lower PI values than those of 'SRNTT-$\mathcal{L}_{rec}$', which is consistent with the analyses in Section 5.5.1. SRNTT [12] would achieve lower PI values than other SISR methods. It's predictable, as SRNTT transfers textures from high-quality reference. However, our method would

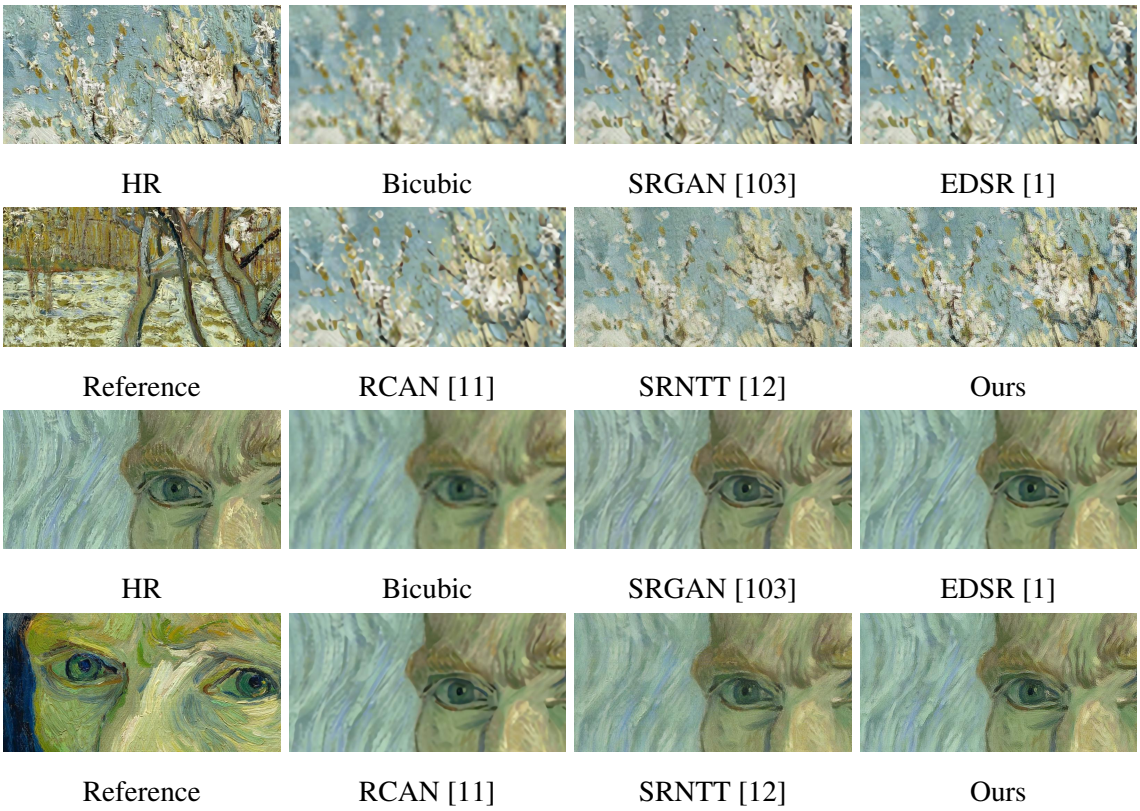


Figure 6.10: Visual results ($8\times$) of different SR methods on PaintHD.

achieve the lowest PI values among all the compared methods. Such quantitative results indicate that our method obtains outputs with better visual quality. To further support our analyses, we further conduct visual results comparisons and user study.

6.5.3 Visual Comparisons

As our PaintHD contains very high-resolution images with abundant textures, it's a practical way for us to show the zoom-in image patches for comparison. To better view the details of high-resolution image patches, it's hard for us to show image patches from too many methods. As a result, we only show visual comparison with state-of-the-art SISR and Ref-SR methods: SRGAN [103], EDSR [1], RCAN [11], and SRNTT [12].

We show visual comparisons in Figures 5.10 and 5.11 for $8\times$ and $16\times$ cases respectively. In the $8\times$ case, SISR methods (e.g., SRGAN, EDSR, and RCAN) could handle it to some degree, because the LR input has abundant details for reconstruction. But, SISR methods still suffer from some blurring artifacts due to use PSNR-oriented loss function (e.g., ℓ_1-norm loss). By transferring

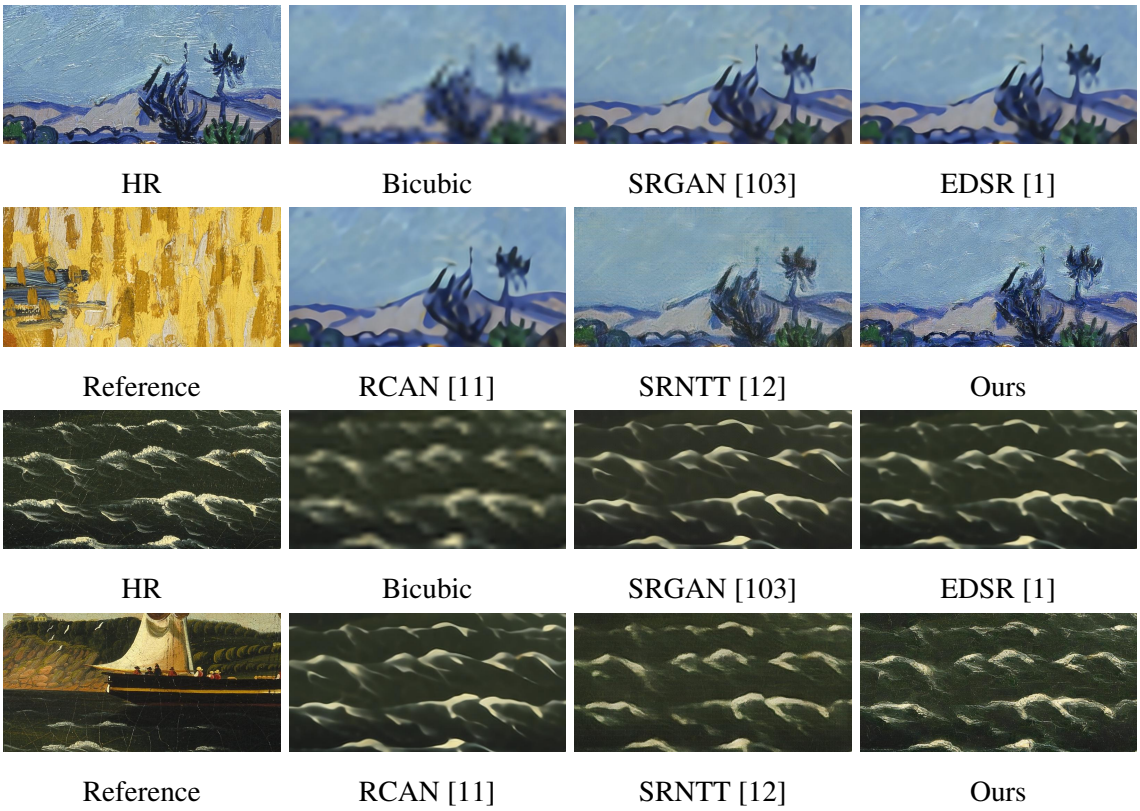


Figure 6.11: Visual results ($16\times$) of different SR methods on PaintHD.

textures from reference and using other loss functions (e.g., texture, perceptual, and adversarial losses), SRNTT [12] performs visually better than RCAN. But SRNTT still can hardly transfer more detailed textures. In contrast, our method would obviously address the blurring artifacts and can transfer more vivid textures.

When the case goes more challenging, namely $16\times$, both SISR methods and SRNTT would generate obvious over-smoothing artifacts (see Figure 5.11). The reasons for SISR methods are mainly that they aim to narrow the pixel-wise difference between the output I_{SR} and the ground truth I_{GT}. Such an optimization way would encourage more low-frequency components, but restrain the generation of high-frequency ones. SRNTT is originally designed for $4\times$ upscaling, which restricts its ability for larger scaling factors. SRNTT neglects to pay more attention to the recovery of high-frequency components. In contrast, our method establishes trainable network from the original LR input to the target HR output firstly. Moreover, we focus on the reconstruction of high-frequency components more with wavelet texture loss. We even further relax the constraint between the

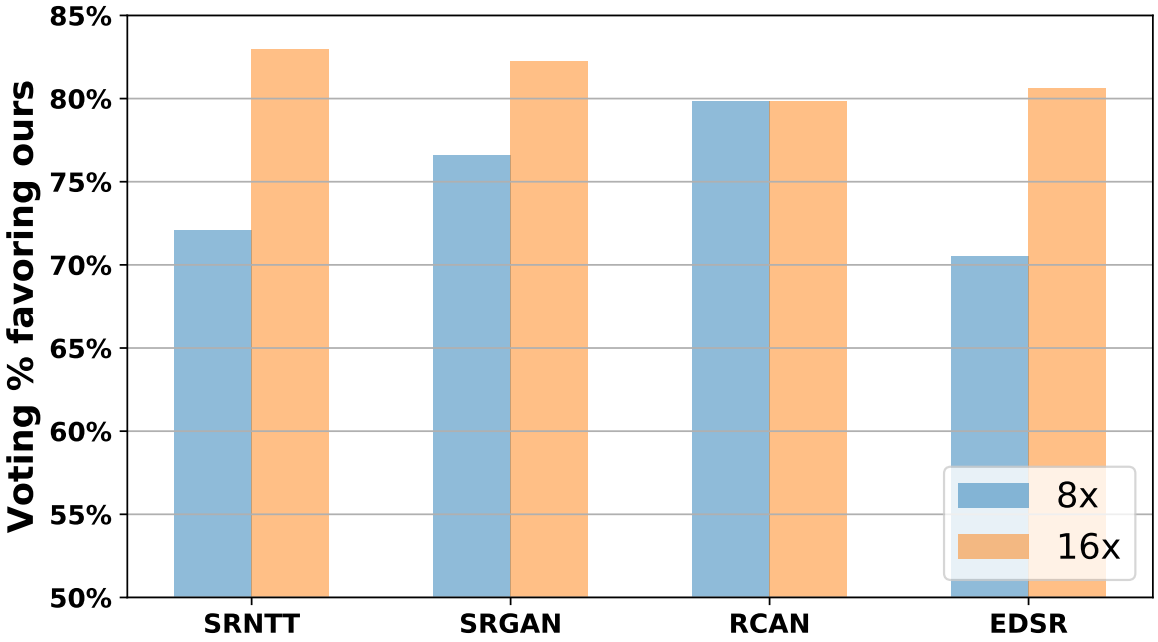


Figure 6.12: User study on the results of SRNTT, SRGAN, RCAN, EDSR, and ours on the PaintHD and CUFED5 datasets. The bar corresponding to each method indicates the percentage favoring ours as compared to the method.

output and ground truth by propose the degradation loss. As a result, our method alleviates the over-smoothing artifacts to some degree and recovers more detailed textures (see Figure 5.11).

6.5.4 User Study

Since the traditional metric PSNR and SSIM do not consistent to visual quality [50, 103, 125, 12], we conducted user study by following the setting of SRNTT [12] to compare our results with those from other methods, i.e., SRNTT [89], SRGAN [103], RCAN [11], and EDSR [1]. The EDSR and RCAN achieve state-of-the-art performance in terms of PSNR/SSIM, while SRGAN (SISR) and SRNTT (Ref-SR) focus more on visual quality. All methods are tested on a random subset of CUFED5 and PaintHD at the upscaling factor of $8\times$ and $16\times$. In each query, the user is asked to select the visually better one between two side-by-side images super-resolved from the same LR input, i.e., one from ours and the other from another method. In total, we collected 3,480 votes, and the results are shown in Figure 5.12. The height of a bar indicates the percentage of users who favor our results as compared to those from a corresponding method.

In general, our results achieve better visual quality at both upscaling scales, and the relative

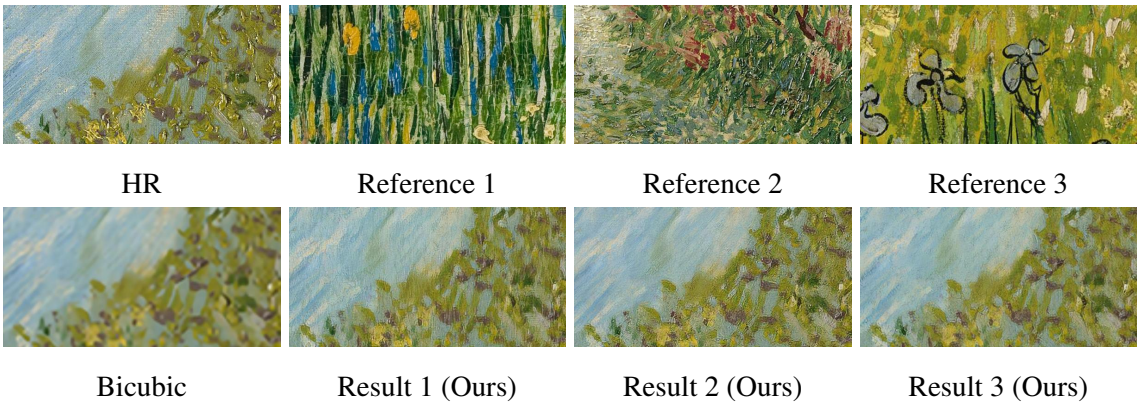


Figure 6.13: Visual results with scaling factor 8× using different reference images.

quality at 16× further outperforms the others. The main reason lies in the texture transfer from references. With the increase of the upscaling factor, more details are lost in the LR input, which is difficult to be recovered solely by the deep model. Thus, externally high-frequency information tends to be more important to texture recovery, which causes the gap from 5% to 10% between the results of 8× and 16×.

6.5.5 Effect of Different References

For Ref-SR methods, investigation on the effect from references is an interesting and opening problem, e.g., how the references affect SR results, how to control (i.e., utilize or suppress) such effect, etc. This section intends to explore the effect of references in the proposed Ref-SR method. As shown in Figure 5.13, the same LR input is super-resolved using different reference images, respectively. We can see that the results keep similar content structures as the input. If we give a further look at the details, we find each result has specific textures from the corresponding reference. It indicates that our method keeps the main structures to the LR input, but also adaptively transfers texture details from reference.

6.6 Summary

We aim to hallucinate painting images with very large upscaling factors and transfer high-resolution (HR) detailed textures from HR reference images. Such a task could be very challenging. The popular single image super-resolution (SISR) could hardly transfer textures from reference images. On the other hand, reference-based SR (Ref-SR) could transfer textures to some degree,

but could hardly handle very large scaling factors. We address this problem by first construct an efficient Ref-SR network, being suitable for very large scaling factor. To transfer more detailed textures, we propose a wavelet texture loss to focus on more high-frequency components. To alleviate the potential over-smoothing artifacts caused by reconstruction constraint, we further relax it by proposed a degradation loss. We collect high-quality painting dataset PaintHD, where we conduct extensive experiments and compare with other state-of-the-art methods. We achieved significantly improvements over both SISR and Ref-SR methods.

Chapter 7

Conclusion

Deep CNNs have been showing excellent performance in computer vision applications, including image restoration and synthesis. In this thesis, we consider several challenges in deep CNN based image restoration and synthesis, e.g., , deep hierarchical features, very deep network training, multimodal feature representation, and texture hallucination with very large scaling factor.

This dissertation studies deep CNN models for image restoration and synthesis, including image super-resolution (SR), denoising, deblurring, demosaicing, compression artifacts reduction, style transfer, and texture hallucination.

In chapter 2, based on our proposed residual dense block (RDB), we propose deep residual dense network (RDN) for image restoration. RDB allows direct connections from preceding RDB to each Conv layer of current RDB, which results in a contiguous memory (CM) mechanism. We proposed LFF to adaptively preserve the information from the current and previous RDBs. With the usage of LRL, the flow of gradient and information can be further improved and training wider network becomes more stable. Extensive benchmark and real-world evaluations well demonstrate that our RDN achieves superiority over state-of-the-art methods for several image restoration tasks.

In chapter 3, we propose a very deep residual channel attention network (RCAN) for highly accurate image SR. Specifically, the residual in residual (RIR) structure allows RCAN to reach very large depth with LSC and SSC. Meanwhile, RIR allows abundant low-frequency information to be bypassed through multiple skip connections, making the main network focus on learning high-frequency information. Furthermore, to improve ability of the network, we propose channel attention (CA) mechanism to adaptively rescale channel-wise features by considering interdependencies among channels. Extensive experiments on SR with BI and BD models demonstrate the effectiveness of our proposed RCAN. RCAN also shows promissing results for object recognition.

In chapter 4, we design residual non-local attention networks for high-quality image restoration. The networks are built by stacking local and non-local attention blocks, which extract local and non-local attention-aware features and consist of trunk and (non-)local mask branches. They're used to extract hierarchical features and adaptively rescale hierarchical features with soft weights. We further generate non-local attention by considering the whole feature map. Furthermore, we propose residual local and non-local attention learning to train very deep networks. We introduce the input feature into attention computation, being more suitable for image restoration. RNAN achieves state-of-the-art image restoration results with moderate model size and running time.

In chapter **??**, we propose multimodal style representation to model the complex style distribution. We then formulate the style matching problem as an energy minimization one and solve it using our proposed graph based style matching. As a result, we propose multimodal style transfer to transform features in a multimodal way. We treat the style patterns distinctively, and also consider the semantic content structure and its matching with style patterns. We also investigate that MST can be generalized to some existing style transfer methods. We conduct extensive experiments to validate the effectiveness, robustness, and flexibility of MST.

In chapter 5, we further investigate how to hallucinate painting images with very large upscaling factors and transfer high-resolution (HR) detailed textures from HR reference images. Such a task could be very challenging. The popular single image super-resolution (SISR) could hardly transfer textures from reference images. On the other hand, reference-based SR (Ref-SR) could transfer textures to some degree, but could hardly handle very large scaling factors. We address this problem by first construct an efficient Ref-SR network, being suitable for very large scaling factor. To transfer more detailed textures, we propose a wavelet texture loss to focus on more high-frequency components. To alleviate the potential over-smoothing artifacts caused by reconstruction constraint, we further relax it by proposed a degradation loss. We collect high-quality painting dataset PaintHD, where we conduct extensive experiments and compare with other state-of-the-art methods.

In addition, we apply deep CNNs for image enhancement in other domains, like biomedical images, by further considering domain specific knowledge. We propose squeeze and excitation reasoning attention networks (SERAN) for accurate magnetic resonance (MR) image SR [13]. We believe that our investigations about deep CNN based image restoration and synthesis would contribute to other related works [14, 15]. In the future, we would like to explore our efficient deep CNN models for other applications, such as biomedical image analysis, bioinformatics, and compressive imaging. Moreover, we'll investigate more efficient models, which have smaller model size and less computation operations by using model compression techniques, like network pruning.

Bibliography

[1] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *CVPRW*, 2017.

[2] T. Tong, G. Li, X. Liu, and Q. Gao, "Image super-resolution using dense skip connections," in *ICCV*, 2017.

[3] X. Huang and S. J. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization." in *ICCV*, 2017.

[4] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Universal style transfer via feature transforms," in *NeurIPS*, 2017.

[5] X. Li, S. Liu, J. Kautz, and M.-H. Yang, "Learning linear transformations for fast arbitrary style transfer," in *CVPR*, 2019.

[6] C. Li and M. Wand, "Combining markov random fields and convolutional neural networks for image synthesis," in *CVPR*, 2016.

[7] S. Gu, C. Chen, J. Liao, and L. Yuan, "Arbitrary style transfer with deep feature reshuffle," in *CVPR*, 2018.

[8] L. Sheng, Z. Lin, J. Shao, and X. Wang, "Avatar-net: Multi-scale zero-shot style transfer by feature decoration," in *CVPR*, 2018.

[9] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *JMLR*, 2008.

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

[11] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *ECCV*, 2018.

[12] Z. Zhang, Z. Wang, Z. Lin, and H. Qi, "Image super-resolution by neural texture transfer," in *CVPR*, 2019.

[13] Y. Zhang, K. Li, K. Li, and Y. Fu, "Mr image super-resolution with squeeze and excitation reasoning attention network," in *CVPR*, 2021.

[14] C. Qiao, D. Li, Y. Guo, C. Liu, T. Jiang, Q. Dai, and D. Li, "Evaluation and development of deep neural networks for image super-resolution in optical microscopy," *Nature Methods*, 2021.

[15] J. Chen, H. Sasaki, H. Lai, Y. Su, J. Liu, Y. Wu, A. Zhovmer, C. A. Combs, I. Rey-Suarez, H.-Y. Chang *et al.*, "Three-dimensional residual channel attention networks denoise and sharpen fluorescence microscopy image volumes," *Nature Methods*, 2021.

[16] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, 1989.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[20] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *CVPR*, 2017.

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

[22] A. M. Treisman and G. Gelade, "A feature-integration theory of attention," *Cognitive psychology*, 1980.

[23] F. Katsuki and C. Constantinidis, "Bottom-up and top-down attention: different processes and overlapping neural systems," *The Neuroscientist*, 2014.

[24] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in *CVPR*, 2018.

[25] K.-H. Lee, X. Chen, G. Hua, H. Hu, and X. He, "Stacked cross attention for image-text matching," in *ECCV*, 2018.

[26] K. Li, Y. Zhang, K. Li, Y. Li, and Y. Fu, "Visual semantic reasoning for image-text matching," in *ICCV*, 2019.

[27] T. Yao, Y. Pan, Y. Li, and T. Mei, "Exploring visual relationship for image captioning," in *ECCV*, 2018.

[28] R. Cadene, H. Ben-Younes, M. Cord, and N. Thome, "Murel: Multimodal relational reasoning for visual question answering," in *CVPR*, 2019.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019.

[31] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018.

[32] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *CVPR*, 2018.

[33] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *TIP*, 2006.

[34] W. Dong, L. Zhang, G. Shi, and X. Li, "Nonlocally centralized sparse representation for image restoration," *TIP*, 2012.

[35] W. Dong, G. Shi, Y. Ma, and X. Li, "Image restoration via simultaneous sparse coding: Where structured sparsity meets gaussian scale mixture," *IJCV*, 2015.

[36] W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, and X. Lu, "Denoising prior driven deep neural network for image restoration," *TPAMI*, 2019.

[37] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep cnn denoiser prior for image restoration," in *CVPR*, 2017.

[38] C. Dong, Y. Deng, C. Change Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *ICCV*, 2015.

[39] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *NeurIPS*, 2016.

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

[41] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[42] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg, "State of the" art: A taxonomy of artistic stylization techniques for images and video," *TVCG*, 2013.

[43] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *ICCV*, 1999.

[44] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *NeurIPS*, 2015.

[45] M. Elad and P. Milanfar, "Style transfer via texture synthesis." *TIP*, 2017.

[46] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *CVPR*, 2016.

[47] K. Li, M. R. Min, and Y. Fu, "Rethinking zero-shot learning: A conditional visual classification perspective," in *ICCV*, 2019.

[48] K. Li, Z. Wu, K.-C. Peng, J. Ernst, and Y. Fu, "Guided attention inference network," *TPAMI*, 2019.

[49] K. Li, Y. Zhang, K. Li, Y. Li, and Y. Fu, "Visual semantic reasoning for image-text matching," in *ICCV*, 2019.

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

[51] A. Sanakoyeu, D. Kotovenko, S. Lang, and B. Ommer, "A style-aware content loss for real-time hd style transfer," in *ECCV*, 2018.

[52] Y. Jing, Y. Liu, Y. Yang, Z. Feng, Y. Yu, D. Tao, and M. Song, "Stroke controllable fast style transfer with adaptive receptive fields," in *ECCV*, 2018.

[53] T. Q. Chen and M. Schmidt, "Fast patch-based style transfer of arbitrary style," in *NeurIPS Workshop*, 2016.

[54] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," in *CVPR*, 2017.

[55] M. Ruder, A. Dosovitskiy, and T. Brox, "Artistic style transfer for videos," in *German Conference on Pattern Recognition*, 2016.

[56] F. Shen, S. Yan, and G. Zeng, "Neural style transfer via meta networks," in *CVPR*, 2018.

[57] R. Mechrez, I. Talmi, and L. Zelnik-Manor, "The contextual loss for image transformation with non-aligned data," in *ECCV*, 2018.

[58] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural style transfer: A review," *TVCG*, 2017.

[59] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Computer Graphics and Applications*, 2002.

[60] G. Freedman and R. Fattal, "Image and video upscaling from local self-examples," *TOG*, 2011.

[61] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *CVPR*, 2015.

[62] V. Boominathan, K. Mitra, and A. Veeraraghavan, "Improving resolution and depth-of-field of light field cameras using a hybrid imaging system," in *ICCP*, 2014.

[63] H. Zheng, M. Guo, H. Wang, Y. Liu, and L. Fang, "Combining exemplar-based approach and learning-based approach for light field super-resolution using a hybrid imaging system," in *ICCV*, 2017.

[64] H. Zheng, M. Ji, H. Wang, Y. Liu, and L. Fang, "Crossnet: An end-to-end reference-based super resolution network using cross-scale warping," in *ECCV*, 2018.

[65] L. Sun and J. Hays, "Super-resolution from internet-scale scene matching," in *ICCP*, 2012.

[66] H. Yue, X. Sun, J. Yang, and F. Wu, "Landmark image super-resolution by retrieving web images," *TIP*, 2013.

[67] W. Yang, S. Xia, J. Liu, and Z. Guo, "Reference-guided deep super-resolution via manifold localized external compensation," *TCSVT*, 2018.

[68] W. W. Zou and P. C. Yuen, "Very low resolution face recognition problem," *TIP*, 2012.

[69] W. Shi, J. Caballero, C. Ledig, X. Zhuang, W. Bai, K. Bhatia, A. M. S. M. de Marvao, T. Dawes, D. ORegan, and D. Rueckert, "Cardiac image super-resolution with global correspondence using multi-atlas patchmatch," in *MICCAI*, 2013.

[70] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *ICLR*, 2018.

[71] L. Zhang and X. Wu, "An edge-guided image interpolation algorithm via directional filtering and data fusion," *TIP*, 2006.

[72] K. Zhang, X. Gao, D. Tao, and X. Li, "Single image super-resolution with non-local means and steering kernel regression," *TIP*, 2012.

[73] R. Timofte, V. De, and L. V. Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *ICCV*, 2013.

[74] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *ACCV*, 2014.

[75] T. Peleg and M. Elad, "A statistical prediction model based on sparse representations for single image super-resolution." *TIP*, 2014.

[76] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *ECCV*, 2014.

[77] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *CVPR*, 2015.

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

[79] K. Zhang, W. Zuo, and L. Zhang, "Learning a single convolutional super-resolution network for multiple degradations," in *CVPR*, 2018.

[80] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *TPAMI*, 2017.

[81] K. Zhang, W. Zuo, and L. Zhang, "Ffdnet: Toward a fast and flexible solution for cnn-based image denoising," *TIP*, 2018.

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

[83] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, 2017.

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

[85] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *AISTATS*, 2015.

[86] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," in *CVPR*, 2018.

[87] H. Zhang, V. Sindagi, and V. M. Patel, "Image de-raining using a conditional generative adversarial network," *TCSVT*, 2019.

[88] H. Zhang and V. M. Patel, "Density-aware single image de-raining using a multi-stream dense network," in *CVPR*, 2018.

[89] ——, "Densely connected pyramid dehazing network," in *CVPR*, 2018.

[90] K. Li, Z. Wu, K.-C. Peng, J. Ernst, and Y. Fu, "Tell me where to look: Guided attention inference network," in *CVPR*, 2018.

[91] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, L. Zhang, B. Lim, S. Son, H. Kim, S. Nah, K. M. Lee *et al.*, "Ntire 2017 challenge on single image super-resolution: Methods and results," in *CVPRW*, 2017.

[92] M. Haris, G. Shakhnarovich, and N. Ukita, "Deep back-projection networks for super-resolution," in *CVPR*, 2018.

[93] C. Ancuti, C. O. Ancuti, R. Timofte, L. Van Gool, L. Zhang, M.-H. Yang, V. M. Patel, H. Zhang, V. A. Sindagi, R. Zhao *et al.*, "Ntire 2018 challenge on image dehazing: Methods and results," in *CVPRW*, 2018.

[94] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli, and L. Zelnik-Manor, "2018 pirm challenge on perceptual image super-resolution," in *ECCVW*, 2018.

[95] K. Yu, C. Dong, L. Lin, and C. C. Loy, "Crafting a toolchain for image restoration by deep reinforcement learning," in *CVPR*, 2018.

[96] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao, and X. Tang, "Esrgan: Enhanced super-resolution generative adversarial networks," in *ECCVW*, 2018.

[97] P. Liu, H. Zhang, K. Zhang, L. Lin, and W. Zuo, "Multi-level wavelet-cnn for image restoration," in *CVPRW*, 2018.

[98] T. Plötz and S. Roth, "Neural nearest neighbors networks," in *NeurIPS*, 2018.

[99] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, "Non-local recurrent network for image restoration," in *NeurIPS*, 2018.

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

[101] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *ECCV*, 2016.

[102] 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 *CVPR*, 2016.

[103] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *CVPR*, 2017.

[104] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS*, 2011.

[105] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *BMVC*, 2012.

[106] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proc. 7th Int. Conf. Curves Surf.*, 2010.

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

[108] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *TPAMI*, 2016.

[109] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior," in *ICCV*, 2015.

[110] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Fast and accurate image super-resolution with deep laplacian pyramid networks," *TPAMI*, 2018.

[111] 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*, 2017.

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

[113] A. Foi, V. Katkovnik, and K. Egiazarian, "Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images," *TIP*, 2007.

[114] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *TIP*, 2004.

[115] J. Jancsary, S. Nowozin, and C. Rother, "Loss-specific training of non-parametric image restoration models: A new state of the art," in *ECCV*, 2012.

[116] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2014.

[117] R. Timofte, R. Rothe, and L. Van Gool, "Seven ways to improve example-based single image super resolution," in *CVPR*, 2016.

[118] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *TIP*, 2007.

[119] ——, "Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space," in *Proc. IEEE Int. Conf. Image Process.*, 2007.

[120] A. Levin, B. Nadler, F. Durand, and W. T. Freeman, "Patch complexity, finite pixel correlations and optimal denoising," in *ECCV*, 2012.

[121] J. Xu, L. Zhang, D. Zhang, and X. Feng, "Multi-channel weighted nuclear norm minimization for real color image denoising," in *ICCV*, 2017.

[122] G. Chen, F. Zhu, and P. Ann Heng, "An efficient statistical method for image noise level estimation," in *ICCV*, 2015.

[123] M. Lebrun, M. Colom, and J.-M. Morel, "The noise clinic: a blind image denoising algorithm," *Image Processing On Line*, 2015.

[124] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *IJCV*, 2000.

[125] M. S. Sajjadi, B. Schölkopf, and M. Hirsch, "Enhancenet: Single image super-resolution through automated texture synthesis," in *ICCV*, 2017.

[126] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014.

[127] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *NeurIPS*, 2014.

[128] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *CVPR*, 2017.

[129] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, W. Xu, D. Ramanan, and T. S. Huang, "Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks," in *ICCV*, 2015.

[130] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *NeurIPS*, 2015.

[131] T. Bluche, "Joint line segmentation and transcription for end-to-end handwritten paragraph recognition," in *NeurIPS*, 2016.

[132] A. Miech, I. Laptev, and J. Sivic, "Learnable pooling with context gating for video classification," *arXiv preprint arXiv:1706.06905*, 2017.

[133] V. Dumoulin, J. Shlens, and M. Kudlur, "A learned representation for artistic style," in *ICLR*, 2017.

[134] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, 2010.

[135] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[136] E. Pérez-Pellitero, J. Salvador, J. Ruiz-Hidalgo, and B. Rosenhahn, "Psyco: Manifold span reduction for super resolution," in *CVPR*, 2016.

[137] 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 *CVPRW*, 2018.

[138] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *ICML*, 2008.

[139] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *CVPR*, 2005.

[140] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *CVPRW*, 2017.

[141] X. Wang, G. Oxholm, D. Zhang, and Y.-F. Wang, "Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer," in *CVPR*, 2017.

[142] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, "Stereoscopic neural style transfer," in *CVPR*, 2018.

[143] Y. Zhang, Y. Zhang, and W. Cai, "Separating style and content for generalized style transfer," in *CVPR*, 2018.

[144] D. M. Greig, B. T. Porteous, and A. H. Seheult, "Exact maximum a posteriori estimation for binary images," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 271–279, 1989.

[145] S. Roy and I. J. Cox, "A maximum-flow formulation of the n-camera stereo correspondence problem," in *ICCV*, 1998.

[146] V. Kolmogorov and R. Zabih, "Multi-camera scene reconstruction via graph cuts," in *ECCV*, 2002.

[147] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: image and video synthesis using graph cuts," *TOG*, 2003.

[148] O. Veksler, "Image segmentation by nested cuts," in *CVPR*, 2000.

[149] Y. Boykov and D. P. Huttenlocher, "A new bayesian framework for object recognition," in *CVPR*, 1999.

[150] D. Reynolds, "Gaussian mixture models," *Encyclopedia of biometrics*, 2015.

[151] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *TPAMI*, 2001.

[152] Y. Li, N. Wang, J. Liu, and X. Hou, "Demystifying neural style transfer," *arXiv preprint arXiv:1701.01036*, 2017.

[153] 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 *ECCV*, 2014.

[154] K. Nichol, "Painter by numbers, wikiart," *https://www.kaggle.com/c/painter-by-numbers*, 2016.

[155] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *CVPR*, 2004.

[156] J. Yoo, Y. Uh, S. Chun, B. Kang, and J.-W. Ha, "Photorealistic style transfer via wavelet transforms," in *ICCV*, 2019.

[157] S. Nah, T. Hyun Kim, and K. Mu Lee, "Deep multi-scale convolutional neural network for dynamic scene deblurring," in *CVPR*, 2017.

[158] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *NeurIPS*, 2017.

[159] W. Commons, "Google art project," https://commons.wikimedia.org/wiki/Category:Google_Art_Project, 2018.

[160] C. Ma, C.-Y. Yang, X. Yang, and M.-H. Yang, "Learning a no-reference quality metric for single-image super-resolution," *CVIU*, 2017.

[161] A. Mittal, R. Soundararajan, and A. C. Bovik, "Making a completely blind image quality analyzer," *SPL*, 2012.