UNIVERSITY
OF UDINE

Polytechnic Department
of Engineering and Architecture (DPIA)

PH.D. THESIS IN
INDUSTRIAL AND INFORMATION ENGINEERING

DEEP CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE SUPER-RESOLUTION

PHD CANDIDATE:
Rao Muhammad Umer

SUPERVISOR:
Prof. Christian Micheloni

PhD Cycle XXXIII — A.Y. 2020-2021

Abstract

Recently, image super-resolution methods have attained impressive performance by using deep convolutional neural networks (DCNNs). In this work, we describe the algorithmic advances and results obtained by the proposed methods in the image Super-Resolution field. First, we propose a series of Single Image Super-Resolution (SISR) methods for effective and efficient super-resolution tasks. We initially propose a deep feed-forward CNNs method that follows the realistic degradation model by handling the blur kernels of different sizes and different noise levels in an unified network. Next, we propose a deep iterative residual CNNs method that follows the image observation (physical) model by exploiting the powerful image regularization and large-scale optimization techniques with the residual learning. Then, we propose an efficient deep iterative CNNs method that solves the SR task by cascading the deep residual denoiser networks.

Second, we propose SR approaches for the Real-World super-resolution problem. For this purpose, we first propose a deep residual GAN-based SR approach with an adversarial training the network for the pixel-wise supervision of the generated realistic LR/HR pairs. Next, we propose a deep cyclic GAN-based SR method by translating the LR to HR domain and vice versa in an end-to-end manner. After that, we incorporate the learnable adaptive sinusoidal non-linearities into the LR and SR network, whose parameters are optimized during the network training.

Finally, we explore the multi image super-resolution (MISR) problems. We first

propose a deep star GAN-based SR method by training the network with a single model to super-resolve the LR images for the multiple LR degradation domains. Next, we propose a deep iterative burst SR method that adopts the burst photography pipeline by following the image observation (physical) model. Lastly, we discuss the broader impact, limitations of the current research work, and possible future research dimensions in the image super-resolution field.

List of Publications

Main author:

Works with the main contribution by the author:

1. **Rao Muhammad Umer**, and Christian Micheloni. "*RBSRICNN: Raw Burst Super-Resolution through Iterative Convolutional Neural Network*". Advances in Neural Information Processing Systems (NeurIPS) Workshops, December 06–14, 2021, Australia. [CORE2021 rating = A*]

2. **Rao Muhammad Umer**, Asad Munir, and Christian Micheloni. "*A Deep Residual Star Generative Adversarial Network for multi-domain Image Super-Resolution*". In proceedings of the 6th International Conference on Smart and Sustainable Technologies, Sept. 08–11, 2021, Croatia. [CORE2021 rating = B]

3. **Rao Muhammad Umer**, Gian Luca Foresti, and Christian Micheloni. "*Deep Iterative Residual Convolutional Network for Single Image Super-Resolution*". In proceedings of the IEEE International Conference on Pattern Recognition (ICPR), Jan 10–15, 2021, Italy. [CORE2021 rating = B]

4. **Rao Muhammad Umer**, Gian Luca Foresti, and Christian Micheloni. "*Deep Generative Adversarial Residual Convolutional Networks for Real-World Super-Resolution*". In proceedings of the IEEE/CVF Conference on Computer Vision

and Pattern Recognition (CVPR) Workshops, June 14–19, 2020, USA. [CORE2021 rating = A*]

5. **Rao Muhammad Umer**, and Christian Micheloni. "*Deep Cyclic Generative Adversarial Residual Convolutional Networks for Real Image Super-Resolution*". In proceedings of European Conference on Computer Vision (ECCV) Workshops, August 24–28, 2020, UK. [CORE2021 rating = A*]

6. **Rao Muhammad Umer**, Gian Luca Foresti, and Christian Micheloni. "*Deep Super-Resolution Network for Single Image Super-Resolution with Realistic Degradations*". In 13^{th} International Conference on Distributed Smart Cameras (ICDSC), Sept. 9—11, 2019, Italy. [CORE2021 rating = B]

Co-author:

In the following works, the author contributed to specific parts:

1. Goutam Bhat, Martin Danelljan, Radu Timofte, and others. "*NTIRE 2021 Challenge on Burst Super-Resolution: Methods and Results*". In proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 19–25, 2021, USA. [CORE2021 rating = A*]

2. Andreas Lugmayr, Martin Danelljan, Radu Timofte, and others. "*NTIRE 2020 Challenge on Real-World Image Super-Resolution: Methods and Results*". In proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 14–19, 2020, USA. [CORE2021 rating = A*]

3. Pengxu Wei, Hannan Lu, Radu Timofte, Liang Lin, Wangmeng Zuo, and others. "*AIM 2020 Challenge on Real Image Super-Resolution: Methods and Results*". In proceedings of European Conference on Computer Vision (ECCV) Workshops, August 24–28, 2020, UK. [CORE2021 rating = A*]

4. Kai Zhang, Martin Danelljan, Yawei Li, Radu Timofte and others. "*AIM 2020 Challenge on Efficient Super-Resolution: Methods and Results*". In proceedings of European Conference on Computer Vision (ECCV) Workshops, August 24–28, 2020, UK. [CORE2021 rating = A*]

Acknowledgements

After gratitude and thanks to God (Allah) Almighty, I must first thank my PhD supervisor Professor Christian Micheloni, who guided me through many emails, meetings, brain-storming sessions, and discussions over the three years of PhD. I am happy that I had a chance to learn and collaborate with him. His continuous guidance made me progress through my research work smoothly and his support was always there whenever I stuck somewhere in finding the relevant technical help. I do hope to continue to have him as a mentor and friend in the future as well.

I am also thankful to my PhD collaborators, especially Asad Munir for fruitful discussions and suggestions with him during the three-year long journey.

I am also grateful to the Machine Learning and Perception Lab, University of Udine for creating a unique environment for research and learning.

Finally, I would like to express my gratitude for my family and this work could not be completed without my family support.

Lastly, I would like to thank the PhD scholarship from the University of Udine for their funding for my PhD studies. I am grateful for this scholarship for the university fees, monthly stipends, learning resource allowances, and conference funding that it provided for my research work to be possible. I also thank the university PhD office for their administrative support.

Contents

1 Introduction **1**
1.1 Image Super-Resolution . . . 1
1.2 Motivation . . . 2
1.3 Categorization . . . 4
1.3.1 Single Image Super-Resolution (SISR) . . . 4
1.3.2 Multi-Image Super-Resolution (MISR) . . . 5
1.4 Overview . . . 6

2 Super-Resolution Background **8**
2.1 Classical SR Methods . . . 8
2.1.1 Nearest-Neighbor . . . 8
2.1.2 Bilinear . . . 10
2.1.3 Bicubic . . . 10
2.2 Deep Learning based SR methods . . . 10
2.3 Real-World SR methods . . . 11
2.4 Video SR methods . . . 12

3 Single Image Super-Resolution **13**
3.1 Deep Feed-Forward CNNs for SISR . . . 13
3.1.1 Proposed Methodology . . . 16
3.1.2 Network Architecture . . . 19
3.1.3 Experimental Setup . . . 21
3.1.4 Limitations . . . 27
3.2 Deep Iterative CNNs for SISR . . . 27
3.2.1 Proposed Method . . . 30
3.2.2 Experiments . . . 35
3.3 Deep Efficient CNNs for SISR . . . 40
3.3.1 Proposed method . . . 40
3.3.2 Experiments . . . 42
3.3.3 Discussion and Limitations . . . 44

4 Real-World Super-Resolution **45**
4.1 Deep Generative Adversarial Residual Convolutional Networks for Real-world SR . . . 45
4.1.1 Proposed Methodology . . . 49
4.1.2 Domain Learning . . . 53
4.1.3 Super-Resolution Learning . . . 54
4.1.4 Experiments . . . 57
4.2 Deep Cyclic Generative Adversarial Residual Convolutional Networks for Real-Image SR . . . 64
4.2.1 Proposed SR Learning Approach . . . 66
4.2.2 Experiments . . . 70
4.3 Real Image Super-Resolution using GAN with deep adaptive Sinusoidal Non-linearities . . . 77
4.3.1 Proposed Methodology . . . 80
4.3.2 Experimental Results . . . 84

5 Multi-Image Super-Resolution **90**
5.1 A Deep Residual Star Generative Adversarial Network for multi-domain Image SR . . . 90
5.1.1 Proposed Method . . . 92
5.1.2 Experiments . . . 95
5.2 Deep Iterative Convolutional Neural Networks for Raw Burst Super-Resolution . . . 99
5.2.1 Proposed Method . . . 102
5.2.2 Proposed Burst SR Network (BSRICNN) . . . 105
5.2.3 Experiments . . . 107
5.2.4 Discussion and Limitations . . . 111

6 Conclusion **113**
6.1 Broader Impact . . . 113
6.2 Future Works . . . 114
6.3 Conclusion . . . 114

List of Tables

3.1 The Average PSNR/SSIM SR results comparison of our method with the others on the test benchmark datasets, *i.e.*, Set5, Set14, and Urban100. The best performance is shown in red and the second best performance is shown in blue. 24

3.2 Computational time (Unit : seconds) comparison of our method with the other SISR methods. The best performance is shown in red and the second best performance is shown in blue. 25

3.3 The settings of input LR and corresponding HR patch sizes during the network training. 35

3.4 The impact of iterative (K) and feedback (FB) steps on ISRResCNet on the scale factor $\times 4$. The average PSNR/SSIM values are evaluated on Set5 testset. The best performance is shown in red. 38

3.5 Average PSNR/SSIM values for the scale factors $\times 2$, $\times 3$, and $\times 4$ with the bicubic degradation model. The best performance is shown in red and the second best performance is shown in blue. 38

3.6 The quantitative SR results ($\times 4$ upscale) comparison of our method with the others over the DIV2K validation set (100 LR images) and AIM 2020 Efficient SR challenge testset [117] (100 LR images). The best performance is shown in red. 43

4.1 Top section: $\times 4$ SR quantitative results comparison of our method over the DIV2K validation-set (100 images) with added two known degradation *i.e.*, sensor noise ($\sigma = 8$) and JPEG compression ($quality = 30$) artifacts. Middle section: $\times 4$ SR results with the unknown corruptions in the RWSR challenge track-1 (validation-set) [77]. Bottom section: $\times 4$ SR comparison with the unknown corruptions in the RWSR challenge series [76, 77]. The arrows indicate if high $\uparrow$ or low $\downarrow$ values are desired. The best performance is shown in red. 59

4.2 Final testset results for the RWSR challenge Track-1. The top section in the table contains ours (**MLP-SR**) with other methods that are ranked in the challenge. The middle section contains participating approaches that deviated from the challenge rules, whose results are reported for reference but not ranked. The bottom section contains baseline approaches. Participating methods are ranked according to their Mean Opinion Score (MOS). 62
4.3 This table reports the quantitative results of our method over the DIV2K validation set (100 images) with unknown degradation for our ablation study. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red. 62
4.4 The ×4 SR quantitative results comparison of our method with others over the DIV2K validation-set (100 images). Top section: SR results comparison with added sensor noise ($\sigma = 8$) and compression artifacts ($quality = 30$) in the validation-set. Middle section: SR results with the unknown corruptions (e.g., sensor noise, compression artifacts, etc.) in the validation-set provided in the RWSR challenge series [76, 77]. Bottom section: SR results with the real image corruptions in the validation-set and testset provided in the AIM 2020 Real Image SR challenge [108] for the track-3. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red and the second best performance is shown in blue. 71
4.5 Final Testset results for the Real Image SR (×4) challenge Track-3 [108]. The table contains ours (**MLP_SR**) with other methods that are ranked in the challenge. The participating methods are ranked according to their weighted Score of the PSNR and SSIM given in the AIM 2020 Real Image SR Challenge [108]. 74
4.6 This table reports the quantitative results of our method over the DIV2K validation set (100 images) with unknown degradation for our ablation study. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red. 76
4.7 ×4 SR quantitative results comparison of our method over the DIV2K validation-set (100 images) with added two known degradations *i.e.*, sensor noise ($\sigma = 8$) and JPEG compression ($quality = 30$) artifacts. Bottom section: ×4 SR results comparison with the unknown corruptions in the RWSR challenge series (validation-set) [77]. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red and the second best performance is shown in blue. 85
4.8 The table reports the quantitative results of our method over the DIV2K validation set (100 images) with unknown degradation for our ablation study. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red. 88

5.1 ×4 SR quantitative results comparison of our method with others over the DIV2K (100 images of validation-set) and RealSR (93 images of testset) that are total 393 images of the testset with the four LR degradation. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red and the second best performance is shown in blue. . . . 96
5.2 This table reports the quantitative SR results of our method over the DIV2K and RealSR validation-set (20 images, not used during the training phase) with the four LR domains (*i.e.*, Bicubic, Bilinear, Nearest, Real) for our ablation study. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red. 98
5.3 We compare our method with the common evaluation metrics (PSNR / SSIM / LPIPS). The quantitative SR results (×4 upscale) are shown over the synthetic and real Burst SR test sets. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red and the second best performance is shown in blue. 108
5.4 The participating methods results on the synthetic test set from Track-1 in the Burst SR Challenge, in terms of PSNR, SSIM, and LPIPS. 109
5.5 The participating methods results on BurstSR test set from Track-2 in the Burst SR Challenge. The PSNR, SSIM, and LPIPS scores are computed after spatial and color alignment of the network prediction to the ground truth. 109
5.6 Impact of different number of input burst frames (B) and number of iterative steps (K). The quantitative results are reported on the synthetic burst testset. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red. 111

List of Figures

1.1 SISR problem. The LR image is a small, blurred and noisy, while SR image is a large and sharp. Traditional upscaling methods fail to recover the sharp image. 2
1.2 Image SR categorization. The SR is divided into the Single Image (SISR) and Multi-Image (MISR). The SISR is further categorized into the effective and efficient SISR. The MISR is also further categorized into the burst and explorable SR. 4

2.1 Visual comparison of the classical SR methods on the $\times 4$ upscaling factor. The Nearest-Neighbor upscaling causes aliasing artifacts especially along edges, the Bilinear upscaling produces a smoothed/blurred image, and the Bicubic upscaling (usually opted) produces some sharpening artifacts or invalid values along edges. The $\downarrow$ and $\uparrow$ represent the downscaling and upscaling process. 9

3.1 SRWDNet architecture. The proposed network takes the input LR image, blur kernel $\mathbf{k}$ (top right corner in the LR input), noise sigma σ, and upscaling factor $\mathbf{s}$, and reconstructs the output SR image. The LR image has $W \times H \times C$ dimensions, while the SR image has $\mathbf{s}W \times \mathbf{s}H \times C$, where C is the number of channels of the input image, and $\mathbf{s}$ is the upscaling factor. 19
3.2 10 randomly generated blur kernels for the (a) training and (b) testing phase according to [11]. 23
3.3 The visual comparison of other SISR methods with ours for the scale factor $\times 2$ on Set5. The blur kernel is shown on the upper-right corner of the LR image. 25
3.4 The visual comparison of other SISR methods with ours for the scale factor $\times 3$ on Set14. The blur kernel is shown on the upper-right corner of the LR image. 26
3.5 The visual comparison of other SISR methods with ours for the scale factor $\times 4$ on Set14. The blur kernel is shown on the upper-right corner of the LR image. 26

3.6 The proposed iterative SISR approach as described in Algorithm 1. Given an LR image ($\mathbf{y}$) and an initial estimate ($\mathbf{x}^0$), each network's stage *ERD* (Encoder-Resnet-Decoder) produces a new estimate $\mathbf{x}^{(k+1)}$ from the previous step estimate $\mathbf{x}^{(k)}$. A single optimizer is used for all network stages with shared structures and parameters by K steps. 28
3.7 The architecture of ERD (Encoder-Resnet-Decoder) blocks used in the proposed ISRResCNet. The $\mathbf{z}^k$ is the LR noisy observation, refer to the Eq. (3.30). The $\mathcal{R}(.)$ corresponds to the regularizer learning part, refer to the Eq. (3.30). The *Prox* layer inside the *Decoder* computes the proximal map with the noise standard deviation σ, refer to the Eq. (3.31). The $\mathbf{x}^{(k+1)}$ is the new solution SR estimate. 34
3.8 Average PSNR/SSIM performance (Set5 on $\times 4$) of proposed ISRResCNet and ISRResCNet+ after each iterative step (K). 37
3.9 Visual comparison of our method with the other state-of-art methods on the $\times 4$ upscaling factor. 39
3.10 The structure of our proposed iterative SR approach. We cascade the ResDNet to the K stages. The first stage takes the LR ($\mathbf{y}$) image and the intermediate stages refine the estimated SR solution until the last stage. The loss $\mathcal{L}(.)$ function is jointly minimized with respect to all network stages. 40
3.11 The architecture of ResDNet. Each ResDNet block contains the *Upsample* layer, the *Non-linearities (RBFs)* is sandwitched between the *Conv* and *TConv* layers which have shared parameters, the *Proj* layer computes the proximal map with noise sigma σ, finally the *clipping* layer is applied to enforce the pixel values between 0 and 255. 41

4.1 $\times 4$ Super-resolution comparison of the proposed SRResCGAN method with the ESRGAN [106] and ESRGAN-FS [31] by the unknown artifacts for the '0815' image (DIV2K validation-set). Our method has better results to handle sensor noise and other artifacts, while the others have failed to remove these artifacts. 47
4.2 The structure of the proposed SR approach setup. In the Domain Learning part, we learn the domain distribution corruptions in the source domain ($\mathbf{x}$) by the network $\mathbf{G}_d$, where our goal is to map images from $\mathbf{z}$ to $\mathbf{x}$, while preserving the image content. Here $\mathbf{B}$ denotes the bicubic downscaling operator which is applied on the clean HR target domain ($\mathbf{y}$) images. In the SR Learning part, we trained the network $\mathbf{G}_{SR}$ in a GAN framework by using generated LR ($\hat{\mathbf{x}}$) images from the $\mathbf{G}_d$ network with their corresponding HR images. 48
4.3 The architectures of Generator and Discriminator networks. The k, c, s denote the kernel size, number of filters, and stride size. 56
4.4 Visual comparison of our method with other state-of-art methods on the NTIRE2020 RWSR (track-1) validation set at the $\times 4$ super-resolution. . 60

4.5 Visual comparison of our method with other state-of-art methods on the NTIRE2020 RWSR (track-2: Smartphone Images) test set [77] at the ×4 super-resolution. 61
4.6 Visual comparison of our method with other state-of-art methods on the NTIRE2020 RWSR (track-1: Image Processing Artifacts) testset [77] at the ×4 super-resolution. 63
4.7 The super-resolution results at the ×4 upscaling factor of the state-of-art–ESRGAN, the proposed SRResCycGAN+ with respect to the ground-truth images. SRResCycGAN+ has successfully remove the visible artifacts, while the ESRGAN has still artifacts due to data bias between the training and testing images. 65
4.8 The structure of our proposed SR approach setup. We trained the network $\mathbf{G}_{SR}$ in a GAN framework, where our goal is to map images from the LR ($\mathbf{y}$) to the HR ($\mathbf{x}$), while maintaining the domain consistency between the LR and HR images. 66
4.9 Visual comparison of our method with the other state-of-art methods on the DIV2K validation set at the ×4 super-resolution. 73
4.10 Visual comparison of our method with the other state-of-art methods on the AIM 2020 Real Image SR (track-3) validation set at the ×4 super-resolution upscaling factor. 75
4.11 Visual comparison of our method with the other state-of-art methods on the AIM 2020 Real Image SR (track-3) test set at the ×4 super-resolution upscaling factor. 76
4.12 The structure of our proposed SR approach setup. In the LR Learning part, we learn the degradation/corruptions in the source domain data ($\mathbf{x}$) by the network $\mathbf{G}_{LR}$ in a GAN framework, where our goal is to map images from clean domain $\mathbf{y}$ to corrupted domain $\mathbf{x}$, while preserving the input image content. In the SR Learning part, we trained the network $\mathbf{G}_{SR}$ in a GAN framework by using generated LR ($\hat{\mathbf{x}}$) images from the $\mathbf{G}_{LR}$ network with their corresponding HR images to super resolve the LR images. 79
4.13 The structure of proposed LR learning architecture. The *Sine* denotes sinusoidal activation layer with 64 output feature maps. 81
4.14 The structure of proposed SR learning architecture. The *Sine* denotes sinusoidal activation layer with 64 output feature maps. 82
4.15 Visual SR comparison of our method with the other state-of-art methods on the DIV2K validation set at the ×4 upscaling factor. 86

5.1 The Multi-domain SR proposed scheme, where a single generator $\mathbf{G}$ learns the mappings among multiple domains *i.e.*, LR/HR to LR/HR. 91

5.2 The structure of our proposed SR^2 *GAN setup. **G** takes an input as both the source image ($\mathbf{y}$) and target domain label and generates a fake target image ($\hat{\mathbf{y}}$). **G** tries to reconstruct the fake source image ($\mathbf{y}'$) from the fake target image ($\hat{\mathbf{y}}$) given the source domain label. **D** learns to discriminate between real and fake target images and classify the real target images to its corresponding domain. In this way, the **G** tries to generate fake target images indistinguishable from real target images and classifiable as target domain images by the **D**. 93
5.3 Visual comparison of our method with other state-of-art methods on ×4 super-resolution. 97
5.4 An image from the Zurich RAW to RGB Dataset [44] (testset), where we present (a) the ground truth HR reference image of size $384 \times 384 \times 3$, (b) the input LR bursts of size ($W \times H \times C \times B$) $48 \times 48 \times 4 \times 14$, and (c) the Burst SR output of size $384 \times 384 \times 3$ of our network (BSRICNN). All images are converted from raw sensor space to sRGB for visualization purpose. 101
5.5 The structure of our proposed iterative Burst SR scheme. Given an LR burst images ($\mathbf{y}_i$), each network's stage produces a new estimate $\mathbf{x}^{(k+1)}$ from the previous step estimate $\mathbf{x}^{(k)}$. A single optimizer is used for all network stages with shared structures and parameters. 101
5.6 SR visual comparison of the proposed BSRICNN method with the existing Burst SR methods on the real-world BurstSR testset at the ×4 upscaling factor. All images are converted from the raw sensor space to sRGB for visualization purpose. 110
5.7 Imprecise warp matrix. All images are converted from raw sensor space to sRGB for visualization purpose. 111

List of Abbreviations

AWGN	Additive White Gaussian Noise
ADMM	Alternating Direction Method of Multipliers minimization method
BPTT	Back-Propagation Through Time
BN	Batch Normalization
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CT	Computerized Tomography
ConvNets	Convolutional Neural Networks
CFA	Color Filter Array
DNN	Deep Neural Network
DCNNs	Deep Convolutional Neural Networks
DCT	Discrete Cosine Transform
DA	Data Augmentation
DSLR	Digital Single-Lens Reflex, High resolution camera
FLOPs	Floating Point Operations Per second
FB	Feedback
GAN	Generative Adversarial Network
GT	Ground Truth
GPU	Graphics Processing Unit
HR	High Resolution

HD	High Definition
HQS	Half Quadratic Splitting minimization method
JPEG	Image Compression technique
LR	Low Resolution
LPIPS	Learned Perceptual Image Patch Similarity Measure
LReLU	Leaky Rectified Linear Unit
MISR	Multi Image Super Resolution
MRI	Magnetic Resonance Imaging
MSE	Mean Squared Error
MOS	Mean Opinion Score
MOA	Mixture Of Augmentation
MFSR	Multi-Frame Super-Resolution
MM	Majorization-Minimization optimization scheme
NPU	Neural Processing Unit
PSNR	Peak Signal-to-Noise Ratio Measure
PGM	Proximal Gradient Descent Method
PReLU	Parametrized Rectified Linear Unit
PET	Positron Emission Tomography
ResNet	Residual Network
RGB	Red-Green-Blue, channels in a digital representation of an image
ReLU	Rectified Linear Unit, a popular activation function
RU	Residual Unit
RBFs	Radial Basis Functions
Re-ID	Re-Identification
RWSR	Real-World Super-Resolution
RAM	Random Access Memory
SISR	Single Image Super Resolution

SSIM	Structural Similarity Measure
SPECT	Single Photon Emission Computed Tomography
TBPTT	Truncated Back-Propagation Through Time
VSR	Video Super-Resolution
YCbCr	Color space, one luminance component and two chroma components

1

Introduction

1.1 Image Super-Resolution

The goal of image super-resolution (SR) is to restore a high-resolution (HR) image from its low-resolution (LR) counterpart by adding the lost high frequencies and rich texture details. The LR image is small, blurred and noisy, while the SR image is large and has sharp image details as shown in Fig. 1.1. The traditional upscaling methods fail to recover the sharp image.

Mathematically, the image SR problem is described as a linear forward observation model with the following image degradation process:

$$\mathbf{y} = (\mathbf{H} * \mathbf{x}) \downarrow_{\mathbf{s}} + \eta, \tag{1.1}$$

where $\mathbf{y}$ is an observed LR image, $\mathbf{H}$ is a *down-sampling operator* that convolves $(*)$ with a latent HR image $\mathbf{x}$ and resizes it by a scaling factor $\mathbf{s}$, and η is considered as an i.i.d additive white Gaussian noise of variance σ^2, *i.e.*, $\eta \sim \mathcal{N}\left(0, \sigma^2\right)$. The recovery of latent HR image $(\mathbf{x})$ from the observed LR image $(\mathbf{y})$, is also indicated as a *Inverse Problem*

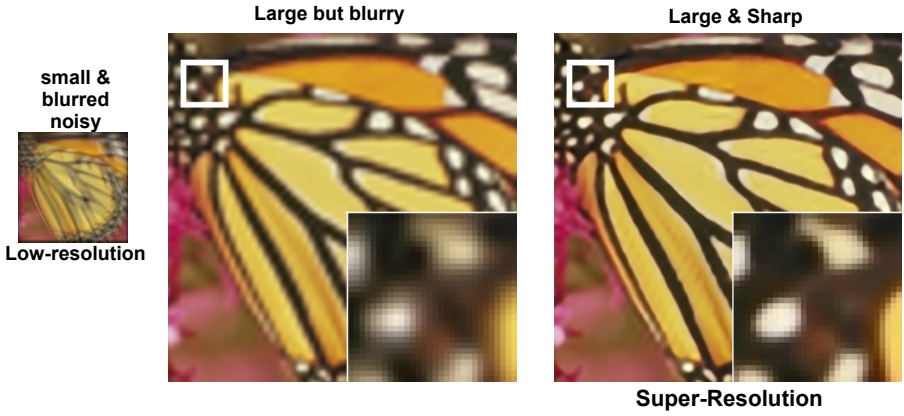


Figure 1.1: SISR problem. The LR image is a small, blurred and noisy, while SR image is a large and sharp. Traditional upscaling methods fail to recover the sharp image.

in Imaging. The operator **H** is usually ill-conditioned or singular due to the presence of noise, whose exact realization is unknown. Moreover, there could exist multiple possible HR images resulting into the same downscaled LR image that makes the image SR part of a highly ill-posed nature of the inverse problem.

1.2 Motivation

Image SR problem is a fundamental low-level computer vision and image processing problem with various important applications in *e.g.*, satellite imaging, medical imaging, astronomy, remote sensing, surveillance, image compression, environment and climate change monitoring, mobile photography, image / video enhancement, and security and surveillance imaging, etc. With the increasing amount of HR image / video data on the internet, there is a great demand for storing, transferring, and sharing such large sized data with low cost of storage and bandwidth resources. Moreover, the HR images are usually downscaled to easily fit into display screens with different resolution, while retaining visually plausible information. The downscaled LR counterpart of the HR can efficiently utilize lower bandwidth, save storage, and easily fit to various digital dis-

plays. However, some details are lost and sometimes visible artifacts appear when users downscale and upscale the digital contents. Modern computing and algorithm advances bring computational photography new modes of capture, post-processing, storage, and sharing in an effective and efficient manner.

In the last decade, most of the photos are taken using built-in smartphone cameras, where the resulting LR image is inevitable and undesirable due to their physical limitations. It is of great interest to restore sharp HR images because some captured moments are difficult to reproduce. Moreover, modern computational photography aims to generate DSLR like images with smartphone cameras.

In the security and surveillance field, particular in the case of distributed cameras networks [90], the possibility to transfer low resolution images is an essential feature that allows to share visual content for detection [30], classification [88], analysis [32] and network management [22]. In the person re-identification (Re-ID) [37], it is difficult to match the LR probes with the HR gallery images. To solve the resolution mismatch problem, the existing Re-ID methods typically recover missing HR details for low-resolution probes by super-resolution methods that greatly benefit the identification task.

In the medical imaging field, the inherent noise from the imaging device / environment lies in the different modality of medical imaging. The image SR provides high-quality clear images to facilitate intelligent data analysis tasks for other sub-problems like classification, detection, and segmentation. A broad spectrum of the image SR applications exist in *e.g.*, PET/SPECT, X-ray, CT [79], MRI [6], Ultrasound [6], Microscopy Imaging [87], Medical Pathology [3], etc. In medical pathology [3], using the SR image as the proxy for its plausibility, one can infer a low chance of pathology due the emerging artifacts when forcing the pathological form.

In the environmental field, the image SR is an important task for satellite monitoring (*i.e.*, depends upon reliable imagery) of human impacts on the planet like deforestation

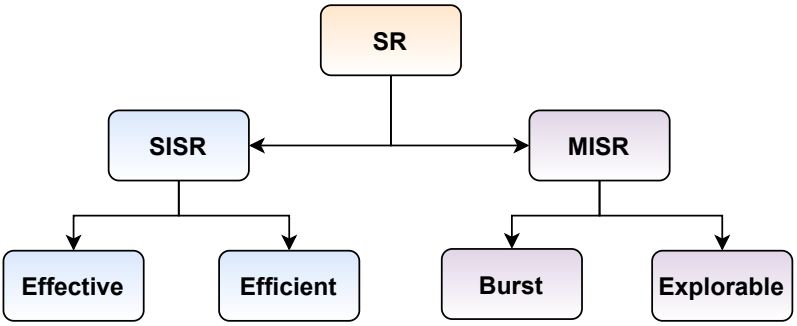


Figure 1.2: Image SR categorization. The SR is divided into the Single Image (SISR) and Multi-Image (MISR). The SISR is further categorized into the effective and efficient SISR. The MISR is also further categorized into the burst and explorable SR.

to monitor climate change and environment.

1.3 Categorization

The Image SR methods can be classified into the main categories shown in Fig. 1.2. The SISR methods are further divided into two branches depending upon their effectiveness and efficiency performance. The MISR methods are also divided into two categories depending upon the one-to-many mapping (*i.e.*, LR to HR) and vice-versa.

1.3.1 Single Image Super-Resolution (SISR)

1.3.1.1 Effective SISR

The goal of effective SISR is to improve PSNR/SSIM or perceptual quality for known degradation (usually bicubic). It accounts the development of non-blind SISR methods for realistic degradation, blind SISR methods with kernel estimation, designing a practical degradation model for blind SISR, and unpaired learning methods for real-world super-resolution. Since the SISR is normally solved by learning from examples *i.e.*, pairs of HR patches and the corresponding LR image patches, the performance of the SR method largely relies on the *quality* of the image patches collected. It is required

to design *active learning* paradigm to find the most informative / useful image patches to be used as the examples for the SISR.

1.3.1.2 Efficient SISR

The goal of efficient SISR is to super-resolve the LR image to the HR image by the scale factor $\mathbf{s}$ (*i.e.*, $\times 2, \times 3, \times 4$, or higher) with a deep network that reduces one or several factors such as runtime, parameters, FLOPs, activations and memory usage, while at least maintaining PSNR/SSIM of a certain baseline model.

1.3.2 Multi-Image Super-Resolution (MISR)

1.3.2.1 Explorable SR

The goal of explorable SR is to learn a stochastic mapping (one-to-many) that is capable of sampling from the space of plausible HR images given an LR image. Due to the ill-posed nature of the image SR problem, infinitely many HR images can be downsampled to the same LR image. The explorable SR method must be able to generate an arbitrary number of SR images with meaningful diversity, and each individual SR prediction should be consistent with the input LR image. The development of SR span learning methods enables us to explore the abundance of plausible solutions that can be applied to other computer vision tasks *e.g.*, image decompression, deblurring, dehazing, etc.

1.3.2.2 Burst SR

The Burst SR is the task of fusing several LR frames to produce a single HR image. The existing SISR methods have limited performance to recover high frequency details through learned single image priors. The multi-frame super-resolution (MFSR) aims to recover the latent HR image using multiple LR frames by exploiting the additional signal information.

1.4 Overview

In this dissertation, we propose the methods for image super-resolution, in particular the SISR, real-world SR, and multi-image SR tasks.

In **Chapter 2**, we provide the relevant background for the image super-resolution and describe commonly used classical methods, deep learning based methods, and real-world methods for image and video super-resolution problems.

In **Chapter 3**, we address the problem of single-image SR. We first propose the SRWDNet that follows the realistic degradation model. Next, we design the iterative SISR network by exploiting the powerful image regularization and large-scale optimization techniques. After that, we propose an efficient SISR approach that solves the SR task as a sub-solver of the image denoising by cascading residual denoiser networks.

In **Chapter 4**, we specifically address the problem of real-world SR. In this regard, we propose the SRResCGAN approach by solving the problem into two stages, domain-learning and SR-learning. In the domain-learning stage, we generate the realistic LR/HR pairs. In the SR-learning stage, we train the SR network by feeding the generated realistic LR data with their corresponding HR images. Next, we propose the SRResCycGAN approach by eliminating the domain-learning stage and train the LR and SR network in an end-to-end fashion by translating the LR to HR domain and vice-versa. Finally, we incorporate learnable adaptive sinusoidal non-linearties into the LR and SR network to further enhance the SR performance.

In **Chapter 5**, we propose the SR approaches for multi-image SR tasks. In this regard, we propose SR^2 *GAN scheme to super-resolve the LR images from the multiple LR degradation domains. Furthermore, we design the BSRICNN network that follows the physical model of burst photography pipeline to learn the image priors from the input LR bursts.

Finally, in **Chapter 6**, we conclude our proposed works of the image super-resolution.

We discuss the broader impact of image SR in a wide range of applications, especially mobile computational photography. We also identify the remaining research challenges and discuss the future research work in the existing SR problems.

2

Super-Resolution Background

2.1 Classical SR Methods

The classical SR methods include bicubic, bilinear, and nearest-neighbor image upscaling. Since these methods are interpretable and easy to implement, some of them are still widely used in image-related applications. These traditional methods have pros and cons depending on their image interpolation process.

2.1.1 Nearest-Neighbor

In the Nearest-Neighbor interpolation algorithm, the value of a pixel is selected by the distance between the pixel and its nearest neighbor point, and it does not consider the values of neighboring points at all, thus yielding a piecewise-constant value. It is very simple to implement and is usually used in real-time 3D rendering for selecting the color values of a textured surface. It can lead to several artifacts such as aliasing and stair-case effect, especially along the edges, as shown in the Fig. 2.1.

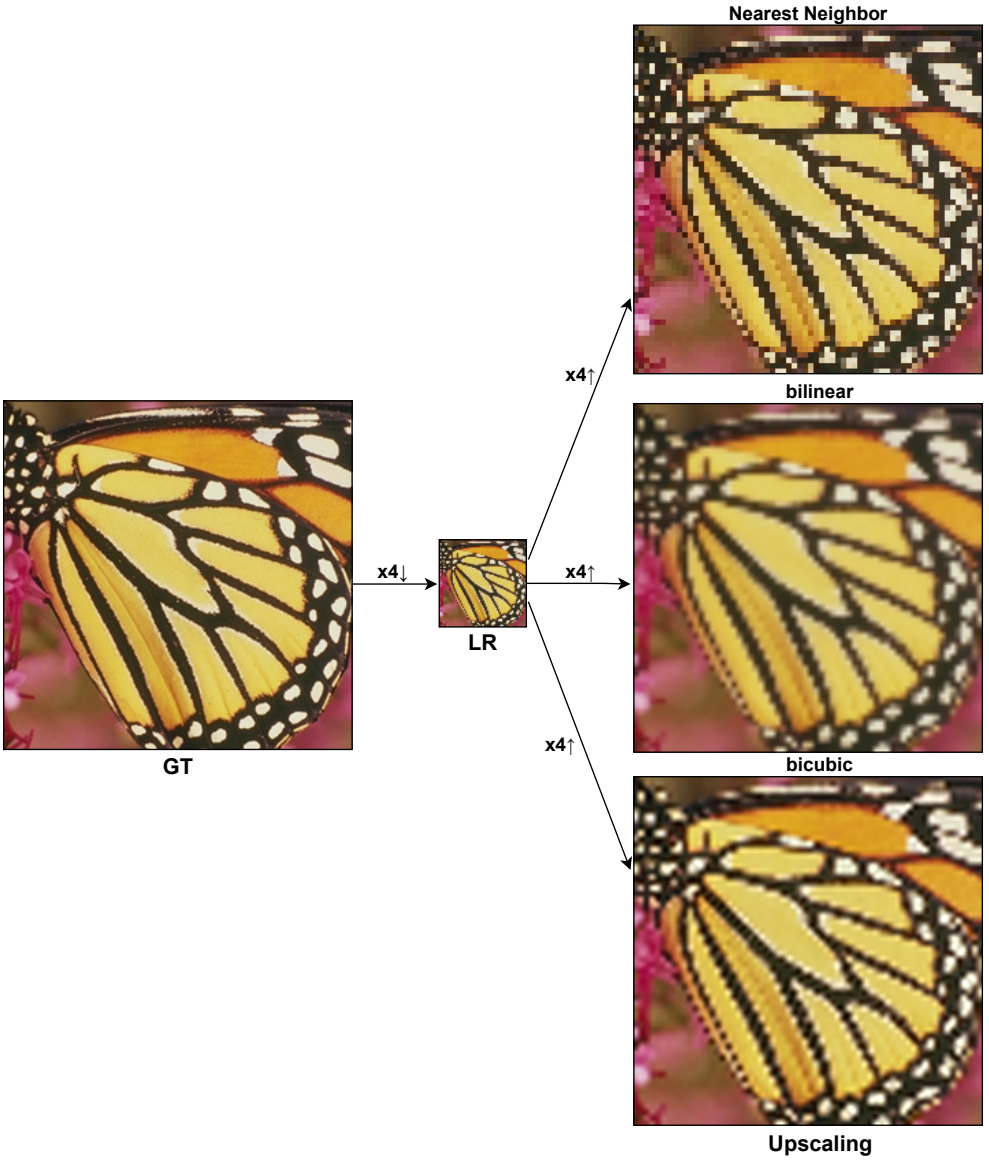


Figure 2.1: Visual comparison of the classical SR methods on the $\times 4$ upscaling factor. The Nearest-Neighbor upscaling causes aliasing artifacts especially along edges, the Bilinear upscaling produces a smoothed/blurred image, and the Bicubic upscaling (usually opted) produces some sharpening artifacts or invalid values along edges. The $\downarrow$ and $\uparrow$ represent the downscaling and upscaling process.

2.1.2 Bilinear

In the bilinear interpolation algorithm, the value of a pixel is the linear interpolation first in one direction, and then again in the other direction. The output is a weighted average of all pixels within the support. It does not suffer from aliasing as the nearest-neighbor interpolation, but it can cause blurring artifacts along the edges as shown in the Fig. 2.1.

2.1.3 Bicubic

In image-related applications, the bicubic interpolation is usually opted over the bilinear or nearest-neighbor in image resampling process. In contrast to the bilinear interpolation, which only considers 4 pixels (2×2) into account, the bicubic interpolation takes 16 pixels (4×4) into account. It produces sharper images than the bilinear interpolation as further pixels get lower weights, but it is computationally more expensive and can cause sharpening artifacts along the edges due to the characteristics of the weighting function as shown in the Fig. 2.1.

2.2 Deep Learning based SR methods

In recent years, various deep learning-based SR methods have been introduced into the image super-resolution field. Initially, learning-based upscaling methods are developed in the image SR field. First, *transposed convolution* layer performs the transformation opposite than a normal convolution layer and is used to upscale the image resolution. Next, *Pixelshuffle* [92] layer is another learnable upscaling layer, which generates a plurality of channels by convolution at first and then reshaping them, and is used to many existing deep learning based SR methods to upscale the feature maps in the reconstruction phase. Besides that, the preliminary deep learning based method (SRCNN) [23] is proposed to solve the SISR task by shallow CNN-based network and adopts the

bicubic downsampling for the LR/HR pairs. By following that, various SISR methods have been proposed such as a very deep network (VDSR) [51] with residual learning approach, an efficient sub-pixel convolutional network (ESPCNN) [92], an enhanced deep SR (EDSR) [69] network that modifies ResNet for enhancement, an iterative residual convolutional network (IRCNN) [120] that solves the SISR problem by using a plug-and-play framework, a deep CNN-based super-resolution with multiple degradation (SRMD) [122], a feedback network (SRFBN) [68] based on feedback connections and recurrent neural network like structure. These methods mostly rely on the PSNR-based metric with blurry results, while they do not preserve the visual quality with respect to human perception. Moreover, the above mentioned methods are deeper or wider CNN networks to learn non-linear mapping from LR to HR with a large number of training samples, while neglecting the real-world settings.

2.3 Real-World SR methods

Another category of the image SR is to obtain or generate training data (*i.e.*, LR/HR pairs) close enough to real data and then train a unified network in a GAN-framework [36] to produce realistic SR images. A preliminary attempt was made by Ledig *et al.*, who proposed SRGAN [106] method to produce more perceptually plausible SR results. Following which, several real-world SISR approaches have been proposed, such as ESRGAN [106] that achieves the state-of-art perceptual performance to further improve the SRGAN architecture, ESRGAN-FT [75] that describes the effects of bicubic downsampling and separate degradation learning for super-resolution, and ESRGAN-FS [31] that uses the DSGAN to learn degradation by training the network in an unsupervised way, and also modified the ESRGAN training scheme. Moreover, the real-world challenge series [76, 77, 108] have been introduced that propose a benchmark protocol for the real-world image corruptions. However, the above methods still suffer unpleasant

artifacts, do not generalize well to other real images captured by different cameras, inaccurate degradation estimations, or mismatch between the degradation model and imaging device.

2.4 Video SR methods

Video Super-Resolution (VSR) aims to restore a high-resolution video from its corresponding low-resolution version. It has been successfully applied in many computer vision applications, such as video surveillance and HD-television. Since the significant improvements of the image super-resolution task over the past few years thanks to deep learning, it encourages the research community to further attempt the more challenging video super-resolution problems. Initially, the video SR task is thought as a simple extension of the image SR problem, but recent studies [13, 95, 89, 105, 45, 96, 16, 14, 74] address the problem by following pipeline as (1) feature extraction, (2) feature alignment, (3) feature fusion, and (4) feature reconstruction. The challenges arise in the design of the feature alignment and fusion stage, when the video frames contain large motion, occlusion, and severe blurring. To get the SR video results, one has to accurately align multiple video frames in the feature space and then effectively fuse the aligned features for the reconstruction.

3

Single Image Super-Resolution

3.1 Deep Feed-Forward CNNs for SISR

Most of the existing CNNs-based SISR methods usually take an assumption that a LR image is only a bicubicly downsampled version of an HR image. However, the true degradation (*i.e.*, the LR image is a bicubicly downsampled, blurred, and noisy version of an HR image) of the LR image goes beyond the widely used bicubic assumption, which is an highly ill-posed nature of inverse problems. To address this issue, we propose the Super-Resolution Wiener deblurring Network (SRWDNet) that works for blur kernels of different sizes and different noise levels in an unified residual CNN-based denoiser network, which significantly improves a practical CNN-based super-resolver for real applications. The proposed method uses the more realistic degradation model in contrast to the existing methods. We split the SISR problem into joint deblurring, denoising, and super-resolution tasks, and then solve it by training the end-to-end network with proximal gradient descent optimization in an iterative manner. Extensive experimental results show that the proposed method is feasible for the more realistic degradations and outperforms the existing SISR methods in terms of PSNR/SSIM as well as the

computational cost.

The preliminary attempt was made to solve the SISR task with CNN-based SR network in SRCNN [23], where a three-layer convolutional neural network was proposed. In the extension of the SRCNN, Kim *et al.*proposed a very deep super-resolution (VDSR) [51] network with the residual learning approach. To improve the efficiency of CNN-based networks, the efficient subpixel convolutional network (ESPCNN) [92] was proposed to take bicubicly LR input and introduced an efficient subpixel convolution layer to upscale the LR feature maps to an HR image at the end of the network. While achieving the fair performance, the above methods take into account of the simpler degradation model (*i.e.*, usually bicubic, refers to Eq. (3.1)), which hinders the efficiency of practical SR applications due to the mismatch of image degradation models.

Beyond the widely used bicubic degradation models in the CNN-based methods, there is an interesting approach to solve SISR task by using model-based optimization frameworks [17, 62, 63, 120]. Besides that, an accurate estimate of the blur kernel plays a more important role than the sophisticated image priors, pointed in [25]. Since then, several methods have been proposed to tackle LR images that go beyond bicubic degradation to solve the energy function induced by the Eq. (3.2). Zhang *et al.*proposed an iterative residual convolutional network (IRCNN) [120] to solve SISR problem by using a plug-and-play framework. Zhang *et al.*proposed a deep CNN-based SR network with multiple degradation (SRMD) [122], which takes two degradation parameters (*i.e.*, blur kernel $\mathbf{k}$, and σ) as the inputs to the network, but they only consider Gaussian blur kernels with fixed kernel width. The above SISR methods have three main limitations to drop their performance in the realistic scenario. First, (1) they have difficulty in complex (e.g. motion) blur kernel estimation with arbitrary dimensions. Second, (2) they are usually designed for Gaussian blur kernels with fixed kernel dimension and thus cannot tackle the severely blurred LR image effectively. Third, (3) they do not handle the different blur kernels and multiple noise levels within a unified network by training

an end-to-end fashion.

The most widely used image forward observation model with the following degradation is given as:

$$\mathbf{y} = \mathbf{x} \downarrow_s, \tag{3.1}$$

where the LR image $\mathbf{y}$ is degraded bicubicly from a clean HR image. However, this simple degradation gives inferior results in many practical SR applications. Another more realistic degradation model in which the LR image $\mathbf{y}$ is mathematically described as a blur kernel $\mathbf{k}$ convolved with the latent sharp HR image $\mathbf{x}$, then the subsequent downsampling operation is applied on the blurred image, and further corrupted by an additive white Gaussian noise. The image forward observation model for this degradation process is given as:

$$\mathbf{y} = (\mathbf{k} * \mathbf{x}) \downarrow_s + \mathbf{n}, \tag{3.2}$$

where $*$ denotes the convolution operator, $\downarrow_s$ is a down-sampling operator with scale factor s, and $\mathbf{n} \in \mathcal{N}\left(0, \sigma^2\right)$ denotes an i.i.d. additive white Gaussian noise (AWGN) term with known standard deviation σ (*i.e.*, noise level). The Eq. (3.2) refers to a general degradation model for the SISR problem. The common choices of the blur kernels ($\mathbf{k}$) are an isotropic or anisotropic Gaussian blur kernel with standard deviation of the fixed kernel width [122]. The more realistic choice is the motion blur kernels with arbitrary size. The most popular choice of the downsampling is to use the bicubic downscaling operator. Since the LR images also contain noise, where the simple case is to take the assumption of AWGN with non-blind noise levels σ, but the more complex case is to consider AWGN with blind noise levels σ. Due to the unknown noise levels and the loss of high-frequency details, this makes the SISR problem an highly ill-posed nature of the inverse problem, and therefore it is an active and challenging research topic in low-level image processing, computer vision, mobile vision, and computational photography.

The contributions of our proposed SRWDNet method are as follows:

1. We follow the more realistic degradation model (refers to the Eq. (3.2)) than simple bicubic degradation model for SISR, which also considers blur kernels of arbitrary sizes and blind noise level to take the advantage of the existing deblurring methods for blur kernel estimation and denoising.

2. The SRWDNet is proposed to solve the SISR task which goes beyond bicubic degradation model and restores the HR image from the LR with the complex degradation.

3. The proposed SRWDNet is well designed as an iterative optimization scheme which aims to solve the forward observation model by minimization of the energy function.

3.1.1 Proposed Methodology

3.1.1.1 Problem Formulation

The modified degradation model of the Eq. (3.2) can be written as:

$$\mathbf{y} = \mathbf{k} * (\mathbf{x} \downarrow_s) + \mathbf{n}, \tag{3.3}$$

where, $\downarrow_s$ is the bicubic downsampler with the scaling factor s. The Eq. (3.3) corresponds to a deblurring problem followed by the SR with general degradation. This model has distinctive advantages over the general degradation model (3.2) as it estimates the blur kernel efficiently from the existing deblurring methods and holds the degradation assumptions of the general model.

After finalizing the suitable degradation model, we formally define the objective function according to the variational framework of Eq. (3.3), and it is given as follows:

$$\hat{\mathbf{x}} = \arg\min_{x} \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{k} * (\mathbf{x} \downarrow_s)\|_2^2 + \lambda\varphi(\mathbf{x}), \tag{3.4}$$

where, $\frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{k} * (\mathbf{x} \downarrow_s)\|_2^2$ is the data fidelity (log-likelihood) term that quantifies the proximity of the solution to the observations, while $\varphi(\mathbf{x})$ is the regularization term associated with the image prior, σ is the unknown noise level *i.e.*, belongs to AWGN, and λ is the trade-off parameter between the data fidelity and regularization term. The CNN-based inference models usually correspond to the energy function for the discriminative learning, where the degradation model is defined by the training LR/HR pairs. It demonstrates that the existing CNN-based SISR trained network with the bicubic degradation (refers to Eq. (3.1)) has limited performance for real SR applications.

3.1.1.2 Optimization Strategy

In this section, we briefly give an overview of the optimization strategy for our network training. By referring to the Eq. (3.4), we want to recover the underlying image $\mathbf{x}$ as the minimizer of the objective function as:

$$\hat{\mathbf{x}} = \arg\min_{x} \mathbf{E}(\mathbf{x}), \tag{3.5}$$

As the energy function $\mathbf{E}(\cdot)$ consists of the data fidelity term and regularizer term, which is given as:

$$\hat{\mathbf{x}} = \arg\min_{x} \mathbf{D}(\mathbf{x}; \mathbf{k}, \mathbf{y}, \downarrow_{\mathbf{s}}) + \lambda\varphi(\mathbf{x}), \tag{3.6}$$

The overall objective function (3.4) can be formally rewritten as a constrained optimization form:

$$\hat{\mathbf{x}} = \underbrace{\arg \min_{a \leqslant x \leqslant b} \frac{1}{2\sigma^2}\|\mathbf{y} - \mathbf{k} * (\mathbf{x} \downarrow_s)\|_2^2 + \lambda\varphi(\mathbf{x})}_{\mathbf{f}(\mathbf{x})}, \tag{3.7}$$

To solve the Eq. (3.7), there are several modern convex optimization schemes for large-scale problems such as Split-Bregman [35], HQS method [33], ADMM [12], Primal-dual algorithms [15], and Proximal methods [85]. In this work, we solve the Eq. (3.7) by using the Proximal Gradient Method (PGM) [85], which is a generalization of gradient

descent algorithm. The PGM [85] deals with the optimization of a function that is not fully differentiable, but it can be split into a smooth and a non-smooth part. To do so, we first rewrite the Eq. (3.7) as:

$$\hat{\mathbf{x}} = \arg\min_{x} \mathbf{f}(\mathbf{x}) + \mathbf{i}_c(\mathbf{x}), \tag{3.8}$$

where, $\mathbf{i_c}$ is the indicator function of the convex set $\mathbf{C} \in \{\mathbf{x} \in \mathbb{R}^m : \mathbf{a} \leqslant \mathbf{x}_k \leqslant \mathbf{b}, \forall k\}$. The gradient of $\mathbf{f}(\mathbf{x})$ is computed as:

$$\nabla_{\mathbf{x}}\mathbf{f}(\mathbf{x}) = \frac{1}{\sigma^2}\mathbf{K}^T(\mathbf{K}(\mathbf{x}\downarrow_s) - \mathbf{y}) + \lambda\Psi(\mathbf{x}), \tag{3.9}$$

where, $\mathbf{K}$ is the matrix version (*i.e.*, usually convolution matrix) of degradation blur kernel $\mathbf{k}$ and $\mathbf{K}^T$ is the transpose convolution of the $\mathbf{K}$. Thus, the solution of the Eq. (3.8) is computed in an iterative fashion by using the following update rule:

$$\mathbf{x}_t\downarrow_s = \mathrm{Prox}_{\gamma^t\mathbf{i_c}}\left(\mathbf{x}_{(t-1)}\downarrow_s - \gamma^t\nabla_{\mathbf{x}}\mathbf{f}(\mathbf{x}_{(t-1)})\right), \tag{3.10}$$

where γ^t is the step-size and $\mathrm{Prox}_{\gamma^t\mathbf{i_c}}$ is the proximal operator [85] related to the indicator function $\mathbf{i_c}$, which can be defined as:

$$\mathrm{Prox}_h(\mathbf{z}) = \arg\min_{x\in\mathbf{C}} \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|_2^2 + h(\mathbf{x}), \tag{3.11}$$

Since proximal map $\mathrm{Prox}_{\gamma\sigma^2}$ gives the regularized solution of a Gaussian denoising problem, so finally we have the following form of our solution:

$$\mathbf{x}_t = \left(\mathrm{Prox}_{\gamma^t\sigma^2}\left((1-\gamma^t\mathbf{K}^T\mathbf{K})(\mathbf{x}_{(t-1)})\downarrow_s + \gamma^t\mathbf{K}^T\mathbf{y} - \lambda\gamma^t\Psi(\mathbf{x}_{t-1})\right)\right)\uparrow_s, \tag{3.12}$$

where $\uparrow_s$ is the upscaling operator. Thus, we design the network by unrolling $\mathbf{S}$ stages of Eq. 3.12 between the LR input and the SR output. For the proposed network, the

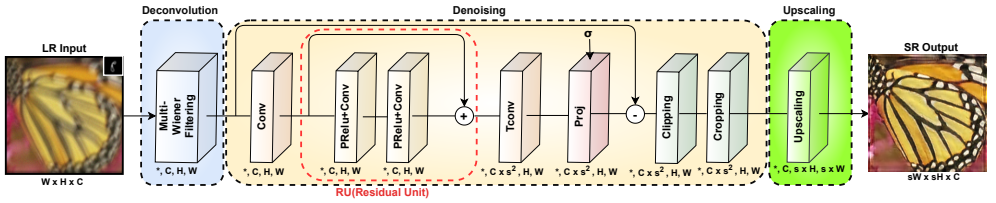


Figure 3.1: SRWDNet architecture. The proposed network takes the input LR image, blur kernel $\mathbf{k}$ (top right corner in the LR input), noise sigma σ, and up-scaling factor $\mathbf{s}$, and reconstructs the output SR image. The LR image has $W \times H \times C$ dimensions, while the SR image has $\mathbf{s}W \times \mathbf{s}H \times C$, where C is the number of channels of the input image, and $\mathbf{s}$ is the upscaling factor.

objective function is minimized by discriminative learning as:

$$\begin{cases} \arg\min_{\Theta} \mathcal{L}(\Theta) = \sum_{s=1}^{S} \frac{1}{2} \|\hat{\mathbf{x}}_T^s - \mathbf{x}_{gt}^s\|_2^2 \\ \text{s.t.} \begin{cases} \mathbf{x}_0^s = \mathbf{I}_0^s \\ \textit{update } \mathbf{x}_t^s \textit{ according to Eq. (3.12)}, \\ t = 1 \dots T \end{cases} \end{cases} \tag{3.13}$$

where, $\Theta = \{\Theta\}_{t=1}^{t=T}$, and $\mathbf{I}_0$ is the initial value of the regularizer term. It can be noted that the above loss function only depends upon the final iteration $\mathbf{T}$, where the network parameters in all stages $\mathbf{S}$ are optimized simultaneously. This minimization training strategy is usually called *joint training*, similar to as done in [63, 17, 91].

3.1.2 Network Architecture

The proposed network architecture for the non-blind SISR is shown in the Fig. 3.1. The input of our network is LR image $\mathbf{y}$ with the corresponding blur kernel $\mathbf{k}$, noise sigma σ, and scaling factor $\mathbf{s}$. Our network first applies deconvolution operation to the LR blurry and noisy input via the deconvolution module, estimates the noise variance by the denoising module, and finally reconstructs the HR image by the upscaling module.

3.1.2.1 Deconvolution module

In our proposed network, the deconvolution module is the learnable Wiener Filtering layer as shown in the Fig. 3.1. In Wiener filtering layer, we formulate the following objective function as:

$$\hat{\mathbf{x}} = \arg\min_{x} \frac{1}{2}\|\mathbf{y} - \mathbf{K}\mathbf{x}\|_2^2 + \frac{\alpha}{2}\|\mathbf{G}\mathbf{x}\|_2^2, \tag{3.14}$$

where, $\mathbf{y}$ is the LR observed image, $\mathbf{K}$ is the blur kernel, and $\mathbf{G}$ is the regularization kernel, and both (*i.e.*, $\mathbf{K}$ and $\mathbf{G}$) are considered as the circulant matrices. In case of multiple regularization kernels, the equation 3.14 can be written as:

$$\hat{\mathbf{x}} = \underbrace{\arg\min_{x} \frac{1}{2}\|\mathbf{y} - \mathbf{K}\mathbf{x}\|_2^2 + \frac{\alpha}{2}\sum_{i=1}^{d}\|\mathbf{G}_i\mathbf{x}\|_2^2}_{\mathbf{f}(\mathbf{x})}, \tag{3.15}$$

where, G_i plays the role of multiple regularization filters and the closed-form solution of the Eq. 3.15 is computed by the Wiener deconvolution technique [110]. So, we learn the Eq. (3.15) as following form in the Wiener filtering layer:

$$\hat{\boldsymbol{x}} = \mathcal{F}(\boldsymbol{y}, \boldsymbol{k}, \sigma; \Theta), \tag{3.16}$$

Where Θ denotes the trainable regularization kernel weights by gradient descent update rule in the network. Here, we compute the gradient of $\mathbf{f}(\mathbf{x})$ as:

$$\nabla_{\mathbf{x}}\mathbf{f}(\mathbf{x}) = \mathbf{K}^T(\mathbf{K}\mathbf{x} - \mathbf{y}) + \alpha\sum_{i=1}^{d}\mathbf{G}_i^T\mathbf{G}_i\mathbf{x} \tag{3.17}$$

After rearranging the Eq. (3.17), we have the following closed-form solution as:

$$\hat{\mathbf{x}} = (\mathbf{K}^T\mathbf{K} + \alpha\sum_{i=1}^{d}\mathbf{G}_i^T\mathbf{G}_i)^{-1}\mathbf{K}^T\mathbf{y}, \tag{3.18}$$

where, we take $\alpha \leftarrow e^{\alpha}$ (*i.e.*, [0.0001, 0.01]) to ensure the positivity of the regularization weights. The weights of Wiener Convolution layer (*i.e.*, Θ) are 24 output features map with kernel size 5×5 by initializing the discrete cosine transform (DCT) basis, which are updated according to PGM (refers to the Eq. (3.12)).

3.1.2.2 Denoising module

Since there are many image denoising neural networks such as DnCNN [119], IRCNN [120], and UDNet [63], we use UDNet [63] as a residual CNN denoiser, which helps to efficiently approximate the proximal map of our final solution (3.12). In the Fig. 3.1, the architecture of the denoising module consists of the *Conv* and *TConv* layers that have 64 feature maps by 7×7 kernel size with $C \times H \times W$ tensors, where C is the number of channels of the input image $\mathbf{y}$. We use five Residual Unit (RU) blocks, which are sandwich by *Conv* and *TConv* layers. Each RU block has two convolution layers, each of 64 feature maps by kernel support of 3×3 filter size, and each convolution layer is preceded by the parametrized rectified linear unit (PReLU) [38]. In our proposed network, the denoising module can be replaced by the other CNN-based denoiser networks, which exhibits the similar characteristics like UDNet [63].

3.1.2.3 Upscaling module

Finally, in the upscaling module, we use the efficient sub-pixel convolution [92] layer to convert multiple latent features of dimension $s^2C \times H \times W$ to the single SR image of size $sW \times sH \times C$.

3.1.3 Experimental Setup

The experimental performance of our proposed network is measured by the peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) measure. In the further subsections, we provide the details of our network training parameters setting, trainset,

testset, comparison with other SISR methods, and computational cost of our proposed method.

3.1.3.1 Network training parameters setting

To train the network, the image patch size is set to 256×256 by center cropping the LR image. We train the network for 50 epochs. We use the ADAM [52] optimizer with a single batch size for training on the loss function as described in the section 3.1.3.2. We set the learning rate as 10^{-3} and ADAM optimizer parameters as $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We set the weight decay to 10^{-4}, and the *amsgrad* flag as *True* for the optimizer.

3.1.3.2 Loss function

In the training phase, we opt the following loss function which consists of *content loss* and *gradient loss* as:

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_{grad}, \tag{3.19}$$

where, $\mathcal{L}_c$ is the mean squared error (MSE) between the GT image ($\mathbf{x}$) and the estimated SR image ($\hat{\mathbf{x}}$), and is computed as:

$$\mathcal{L}_{\mathrm{c}}(\mathbf{x}_i, \hat{\mathbf{x}}_i; \Theta) = \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2, \tag{3.20}$$

The $\mathcal{L}_{grad}$ is to minimize the gradient discrepancy between the GT image and the estimated SR image, and is computed as:

$$\mathcal{L}_{\mathrm{grad}}(\mathbf{x}_i, \hat{\mathbf{x}}_i; \Theta) = \|\nabla_v \hat{\mathbf{x}}_i - \nabla_v \mathbf{x}_i\|_2^2 + \|\nabla_h \hat{\mathbf{x}}_i - \nabla_h \mathbf{x}_i\|_2^2, \tag{3.21}$$

where, ∇_v and ∇_h denote the operators calculating the image directional derivatives in the horizontal and vertical directions, respectively. The $\mathcal{L}_{grad}$ helps to produce sharp

Figure 3.2: 10 randomly generated blur kernels for the (a) training and (b) testing phase according to [11].

image details.

3.1.3.3 Training dataset

We use BSDS500 [2] dataset for the network training. We split the training dataset into 400 HR GT images for training and 100 HR GT images for the validation. The Fig. 3.2 shows the 10 randomly generated motion blurred kernels for training and testing according to [11], whose blur kernel size ranges from 11×11 to 31×31 pixels. To generate the downsampled, blurred, and noisy LR image image patches, we bicubicly downsample the GT images with scaling factors $\mathbf{s}$ (*i.e.*, $\times2, \times3, \times4$), then convolve the downsampled images with the motion blur kernels $\mathbf{k}$, and add Gaussian noises with 1%, 2%, 3%, and 5% noise standard deviation. We uniformly sample kernel size from the range $[11, 13, 15, 17, 19, 21, 23, 27, 29, 31]$ and noise levels from the interval $[1\%, 2\%, 3\%, 5\%]$, which helps to learn a more versatile model to handle diverse data.

3.1.3.4 Testing datasets

We evaluate the proposed method on the well-known SISR benchmark testing datasets, *i.e.*, Set5 [100], Set14 [100], and Urban100 [39], that are independent of the training set. The LR images are generated according to subsection 3.1.3.3. We generate 50 LR images (*i.e.*, 5 GT images $\times$ 10 blur kernels) for the Set5, 140 LR images (*i.e.*, 14 GT images $\times$ 10 blur kernels) for the Set14, and 1000 LR images (*i.e.*, 100 GT images $\times$ 10

Table 3.1: The Average PSNR/SSIM SR results comparison of our method with the others on the test benchmark datasets, *i.e.*, Set5, Set14, and Urban100. The best performance is shown in red and the second best performance is shown in blue.

Dataset	Degradation Settings				Bicubic	VDSR(CVPR) [51]	TNRD(TPAMI) [17]	IRCNN(CVPR) [120]	SRMD(CVPR) [122]	SRWDNet(Ours)
	Scale Factor	Kernel size	Down-sampler	Noise Level	Average PSNR / SSIM					
Set5	×2	11×11 to 31×31	Bicubic	1%	19.30 / 0.5070	19.24 / 0.4767	19.41 / 0.4937	19.00 / 0.4545	17.94 / 0.4414	23.13 / 0.5870
	×3	11×11 to 31×31	Bicubic	1%	17.90 / 0.4668	17.86 / 0.4431	17.90 / 0.4765	17.63 / 0.4171	17.40 / 0.4311	21.00 / 0.5025
	×4	11×11 to 31×31	Bicubic	1%	17.01 / 0.4496	16.97 / 0.4296	17.21 / 0.4609	16.74 / 0.4053	16.72 / 0.4263	20.58 / 0.5036
Set14	×2	11×11 to 31×31	Bicubic	1%	18.85 / 0.4419	18.80 / 0.4147	18.99 / 0.4453	18.59 / 0.3981	17.15 / 0.3772	21.28 / 0.5120
	×3	11×11 to 31×31	Bicubic	1%	17.74 / 0.4127	17.70 / 0.3900	17.52 / 0.4726	17.49 / 0.3722	17.24 / 0.3858	19.25 / 0.4042
	×4	11×11 to 31×31	Bicubic	1%	16.99 / 0.4012	16.97 / 0.3818	17.10 / 0.4509	16.75 / 0.3651	16.73 / 0.3842	19.10 / 0.4109
Urban100	×2	11×11 to 31×31	Bicubic	1%	17.30 / 0.4007	17.25 / 0.3729	17.58 / 0.4336	17.01 / 0.4235	15.23 / 0.3357	19.81 / 0.4914
	×3	11×11 to 31×31	Bicublic	1%	16.44 / 0.3773	16.41 / 0.3539	16.45 / 0.4802	16.14 / 0.3523	15.85 / 0.3538	17.98 / 0.3810
	×4	11×11 to 31×31	Bicubic	1%	15.89 / 0.3694	15.87 / 0.3491	16.23 / 0.4608	15.95 / 0.3478	15.65 / 0.3601	17.65 /0.3744

blur kernels) for the Urban100.

3.1.3.5 Comparison with the SISR methods

We compare our proposed method with the traditional bicubic upscaling method (*i.e.*, *imresize* Matlab function) and other CNN-based SISR methods including VDSR [51], TNRD [17], IRCNN [120], and SRMD [122]. The IRCNN [120] and SRMD [122] can take degraded image $\mathbf{y}$, blur kernel $\mathbf{k}$, and noise level σ as the input, while VDSR [51] and TNRD [17] can take degraded image $\mathbf{y}$ and noise level σ as the input to the network. The various degradation settings have been considered under the same experimental situations. We run all source codes of the comparison methods with the default parameter settings for all experiments.

The Table 3.1 reports the results in terms of Average PSNR and SSIM on the testing benchmark datasets. Our method outperforms against the others SISR methods. Our method gets much cleaner and HR images with fine texture details without blur artifacts, while the others methods suffer from oversmoothed images and unpleasant artifacts. Fig. 3.3 shows the visual comparison of our method with others for $\times 2$ upscaling factor of the LR image with motion blur kernel. The VDSR produces unpleasant blurred results due to the bicubic degradation assumption which deviates from the true

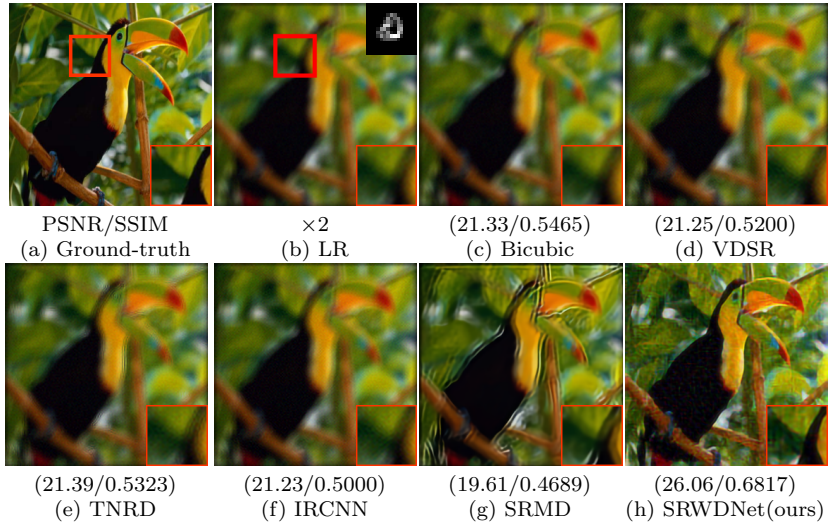


Figure 3.3: The visual comparison of other SISR methods with ours for the scale factor ×2 on Set5. The blur kernel is shown on the upper-right corner of the LR image.

one. The TNRDS also produce unpleasant results due to mismatch of realistic degradation model. Since the IRCNN and SRMD follow the true degradation assumption, the SRMD produce more visually pleasant results than IRCNN. The SRMD has still blurring artifacts due to opt simple Gaussian blur kernel with fixed width. Even though the input LR image is severely degraded by the complex degradation, our method achieves higher performance in both quantitatively and qualitatively than other methods due to obeying the more realistic degradation model.

We also provide the additional results comparison on test benchmark datasets for the upscaling factors ×3 and ×4. The Fig. 3.4 and 3.5 shows the visual comparison of our method with others. Our method gets much cleaner images by preserving fine texture details on the higher scale factors as well.

Table 3.2: Computational time (Unit : seconds) comparison of our method with the other SISR methods. The best performance is shown in red and the second best performance is shown in blue.

Degradation Scenario	VDSR	TNRD	IRCNN	SRMD	SRWDNet(Ours)
image size: 500 × 480, motion blur kernel: 31 × 31, σ= 1%, upscaling factor = ×4	1.573	19.573	30.561	0.305	0.593

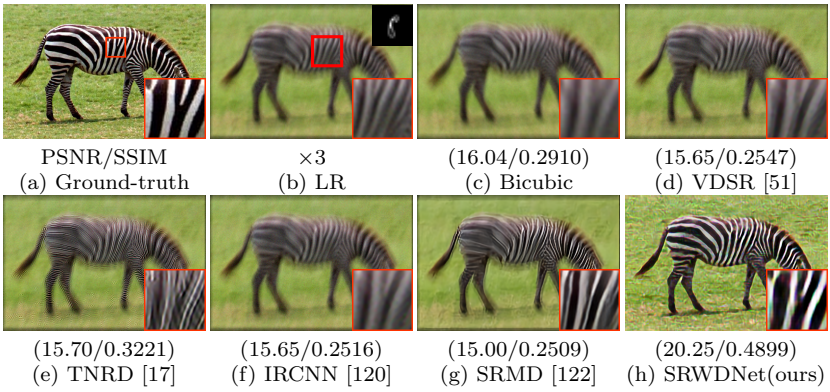


Figure 3.4: The visual comparison of other SISR methods with ours for the scale factor ×3 on Set14. The blur kernel is shown on the upper-right corner of the LR image.

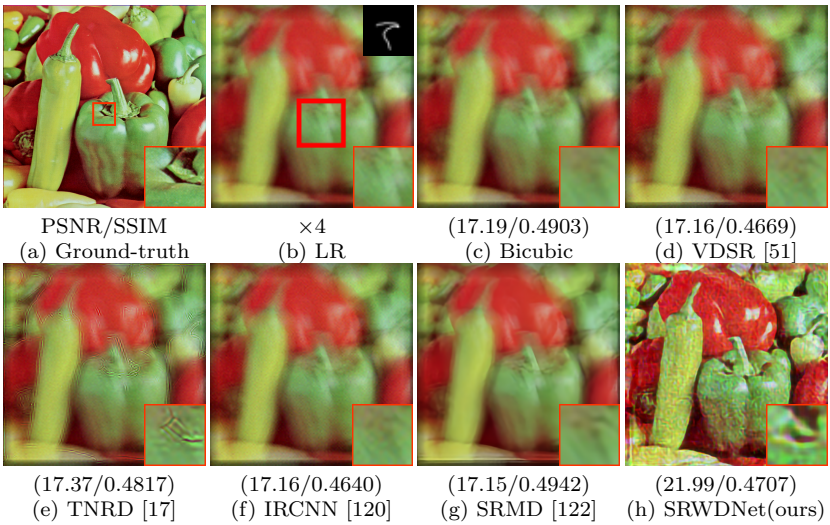


Figure 3.5: The visual comparison of other SISR methods with ours for the scale factor ×4 on Set14. The blur kernel is shown on the upper-right corner of the LR image.

3.1.3.6 Running time

Our proposed method performs well in terms of computational cost efficiency with other SISR methods, which is favorable for practical SR applications. Table 3.2 shows the test execution time of all SR methods with specific image degradation scenario, measured on hardware environment with the following specifications as CPU memory: 32GB, GPU

memory: 8GB Nvidia Quadro. The testing time of all methods is measured on GPU.

3.1.4 Limitations

Our method is capable of producing the HR images from severely degraded noisy LR images with complex degradation. However, the main limitation of our method is the unpleasant results when there is a strong presence of noise *i.e.*, 3%, 5%, or more. Moreover, we train different networks with their respective scaling factors, which limits the performance of our model with the other scaling factor. MDSR [70] approach is one possible solution to tackle multiple scaling factors within the same network, but it has not tackled multiple degradations.

3.2 Deep Iterative CNNs for SISR

The deep CNNs-based SISR methods focus on designing deeper or wider models to learn the nonlinear mapping between the LR inputs and the HR outputs. These existing SISR methods do not take into account the image observation (physical) model and thus require a large number of network's trainable parameters with a great volume of training data. To address these issues, we propose a deep Iterative Super-Resolution Residual Convolutional Network (ISRResCNet) that exploits the powerful image regularization and large-scale optimization techniques by training the network in an iterative manner with a residual learning approach. Our model requires few trainable parameters in comparison to other competing methods. Our method achieves excellent SR results in terms of PSNR/SSIM and visual quality by following the real-world settings for limited memory storage and cpu power requirements for the mobile/embedded deployment.

By referring to the general degradation model (3.2), the SISR is described as a linear

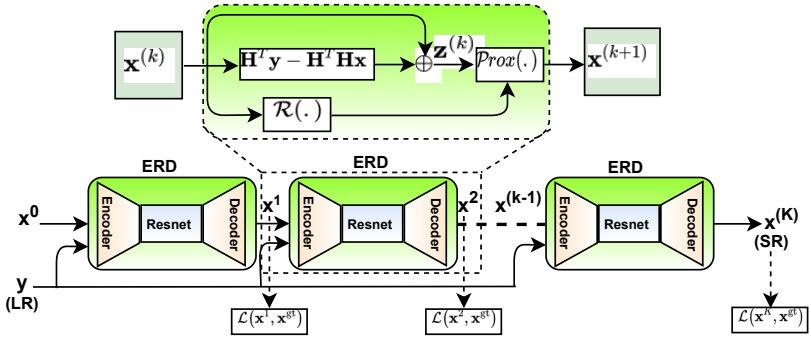


Figure 3.6: The proposed iterative SISR approach as described in Algorithm 1. Given an LR image ($\mathbf{y}$) and an initial estimate ($\mathbf{x}^0$), each network's stage *ERD* (Encoder-Resnet-Decoder) produces a new estimate $\mathbf{x}^{(k+1)}$ from the previous step estimate $\mathbf{x}^{(k)}$. A single optimizer is used for all network stages with shared structures and parameters by K steps.

forward observation model with the following image degradation process:

$$\mathbf{y} = \mathbf{H}\tilde{\mathbf{x}} + \eta, \tag{3.22}$$

where $\mathbf{y} \in \mathbb{R}^{N/s^2}$ is an observed LR image (here $N = m \times n$ is typically the total number of pixels in an image), $\mathbf{H} \in \mathbb{R}^{N/s \times N/s}$ is a *down-sampling operator* (usually a bicubic, circulant matrix) that resizes an HR image $\tilde{\mathbf{x}} \in \mathbb{R}^N$ by a scaling factor s and η is considered as an additive white Gaussian noise (AWGN) with standard deviation σ. The operator $\mathbf{H}$ is usually ill-conditioned or singular due to the presence of unknown noise (η) that makes the SISR of a highly ill-posed nature of inverse problems. Since, due to ill-posed nature, there are many possible solutions, regularization is required to select the most plausible ones.

Generally, SISR methods can be classified into three main categories, *i.e.*, interpolation based methods, model-based optimization methods, and discriminative learning methods. Interpolation-based methods *i.e.*, nearest-neighbor, bilinear, and bicubic interpolators are efficient and simple, but have very limited reconstruction image quality (refer to Chapter 2 for more details). Model-based optimization [24] methods have

powerful image priors to reconstruct high-quality clean images, but require hundreds of iterations to achieve acceptable performance, thus making these methods computationally expensive. Model-based optimization [121, 102] methods with the integration of deep CNNs priors can improve efficiency, but due to hand-crafted parameters, they are not suitable for end-to-end deep learning methods. On the other hand, discriminative learning [51, 69, 114, 47, 73, 68, 80] methods have attracted significant attentions due to their effectiveness and efficiency for SISR performance by using deep CNNs. This work is inspired by discriminative and residual learning approaches with powerful image priors and large-scale optimization schemes in an iterative manner for an end-to-end deep CNNs to solve SISR problem.

The visualization of our proposed iterative SISR approach is shown in Figure 3.6, where the LR input ($\mathbf{y}$) is given to the network and then the network reconstructs the SR output. A single optimizer is used for all network stages with shared structures and parameters. Our contributions in this section are in three-fold as follows:

1. We propose an end-to-end deep iterative Residual CNNs for image super-resolution. In contrast to the existing deep SISR networks, our proposed method strictly follows the image observation (physical) model (refers to Eq. (3.22)), and thus it is able to achieve better reconstruction results even with few network's trainable parameters (refers to Table 3.4).

2. A deep SISR network is proposed to solve image super-resolution in an iterative manner by minimizing the discriminative loss function with a residual learning approach.

3. The proposed ISRResCNet is inspired by powerful image regularization and large-scale optimization techniques that have been successfully used to solve general inverse problems in the past.

3.2.1 Proposed Method

3.2.1.1 Problem Formulation

By referencing to equation (3.22), the recovery of $\mathbf{x}$ from $\mathbf{y}$ mostly relies on the variational approach for combining the observation and prior knowledge, and is given as the following objective function:

$$\mathbf{J}(\mathbf{x}) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda\mathcal{R}(\mathbf{x}), \tag{3.23}$$

where $\frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2$ is the data fidelity (also known as log-likelihood) term that measures the proximity of the solution to the observations, $\mathcal{R}(\mathbf{x})$ is the regularization term that is associated with image priors, and λ is the trade-off parameter that governs the compromise between the data fidelity and the regularizer term. Interestingly, the variational approach has a direct link to the Bayesian approach and the derived solutions can be described by either as penalized maximum likelihood or as maximum a posteriori (MAP) estimates [7, 28]. Thanks to the recent advances of deep learning, the regularizer (*i.e.*, $\mathcal{R}(\mathbf{x})$) is employed by deep convolutional neural networks (ConvNets) [101] that have powerful image priors capabilities.

3.2.1.2 Objective Function Minimization Strategy

Besides the proper selection of the regularizer and formulation of the objective function, another important aspect of the variational approach is the minimization strategy that will be used to get the required solution. In the literature, there are several modern convex optimization schemes for large-scale machine learning problems, such as Split-Bregman [35], HQS method [33], ADMM [12], Primal-dual algorithms [15], etc. In our work, we solve the under study problem (3.23) by using the Majorization-Minimization (MM) framework [42] because $\mathbf{J}(\mathbf{x})$ is too complicated to manipulate (*i.e.*, convex func-

tion but possibly non-differentiable). In MM [42, 29, 65] approach, an iterative algorithm for solving the minimization problem

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \mathbf{J}(\mathbf{x}), \tag{3.24}$$

takes the form

$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}} \mathbf{Q}(\mathbf{x}; \mathbf{x}^{(k)}), \tag{3.25}$$

where, $\mathbf{Q}(\mathbf{x}; \mathbf{x}^{(k)})$ is the majorizer of the function $\mathbf{J}(\mathbf{x})$ at a fixed point $\mathbf{x}^{(k)}$ by satisfying the following two conditions:

$$\mathbf{Q}(\mathbf{x}; \mathbf{x}^{(k)}) > \mathbf{J}(\mathbf{x}),\ \forall \mathbf{x} \neq \mathbf{x}^{(k)} \text{ and } \mathbf{Q}(\mathbf{x}^{(k)}; \mathbf{x}^{(k)}) = \mathbf{J}(\mathbf{x}^{(k)}). \tag{3.26}$$

Here, we want to upper-bound the $\mathbf{J}(\mathbf{x})$ by a suitable majorizer $\mathbf{Q}(\mathbf{x}; \mathbf{x}^{(k)})$, and instead of minimizing the actual objective function (3.24) due to its complexity, we minimize the majorizer $\mathbf{Q}(.)$ to produce the next estimate $\mathbf{x}^{(k+1)}$. By satisfying the properties of the majorizer given in Eq. (3.26), iteratively minimizing $\mathbf{Q}(.; \mathbf{x}^{(k)})$ also decreases the actual objective function $\mathbf{J}(.)$ [42]. Thus, we can write a quadratic majorizer for the complete objective function (3.23) as the following form:

$$\mathbf{Q}(\mathbf{x}; \mathbf{x}^{(k)}) = \frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda \mathbf{Q}_{\mathcal{R}}(\mathbf{x}; \mathbf{x}^{(k)}), \tag{3.27}$$

To start an initial estimate $\mathbf{x}_0$, we have:

$$\mathbf{Q}_{\mathcal{R}}(\mathbf{x}; \mathbf{x}_0) = \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T[\alpha\mathbf{I} - \mathbf{H}^T\mathbf{H}](\mathbf{x} - \mathbf{x}_0), \tag{3.28}$$

where $\mathbf{Q}_{\mathcal{R}}(.)$ is a distance function between $\mathbf{x}$ and $\mathbf{x}_0$. In order to get a valid majorizer $\mathbf{Q}_{\mathcal{R}}(.)$, we need to satisfy two conditions in Eq. (3.26) as $\mathbf{Q}_{\mathcal{R}}(\mathbf{x}; \mathbf{x}_0) > 0$, $\forall \mathbf{x} \neq \mathbf{x}_0$ and $\mathbf{Q}_{\mathcal{R}}(\mathbf{x}; \mathbf{x}_0) = 0$. This suggests that $\alpha\mathbf{I} - \mathbf{H}^T\mathbf{H}$ must be a positive definite matrix,

which only holds if $\alpha > \|\mathbf{H}^T\mathbf{H}\|_2$. The parameter α depends upon the largest eigenvalue of $\mathbf{H}^T\mathbf{H}$, but, in most image restoration cases [29] such as inpainting, deblurring, demosaicking[55], and super-resolution, it approximately equals to one ($\alpha \approx 1$). Based on the above discussion, we can write the overall majorizer as:

$$\mathbf{Q}(\mathbf{x};\mathbf{x}_0) = \frac{1}{2/\alpha}\|\mathbf{x}-\mathbf{z}\|_2^2 + \lambda\mathcal{R}(\mathbf{x}) + const., \tag{3.29}$$

where $\mathbf{z} = \mathbf{x}_0 + \frac{1}{\alpha}\mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{x}_0)$, and the *constant* does not depend on $\mathbf{x}$ and thus it is irrelevant to the optimization task.

Finally, we proceed with the MM optimization scheme to iteratively minimize the quadratic majorizer function $\mathbf{Q}(.)$ by the following formulation:

$$\begin{aligned}
\hat{\mathbf{x}}^{(k)} &= \arg\min_{\mathbf{x}} \mathbf{Q}(\mathbf{x};\mathbf{x}^k) \\
&= \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y}-\mathbf{H}\mathbf{x}\|_2^2 + \lambda\mathbf{Q}_{\mathcal{R}}(\mathbf{x};\mathbf{x}^k) \\
&= \arg\min_{\mathbf{x}} \frac{1}{2/\alpha}\|\mathbf{x}-\mathbf{z}^k\|_2^2 + \lambda\mathcal{R}(\mathbf{x}) \\
&= \mathrm{Prox}_{(\lambda/\alpha)\mathcal{R}(.)}(\mathbf{z}^k)
\end{aligned} \tag{3.30}$$

where $\mathbf{z}^k = \mathbf{z}^k + \mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{z}^k)$ and $\mathrm{Prox}_{(.)}$ is the proximal operator [85], which is defined as:

$$\mathbf{P}_{\mathbb{C}}(\mathbf{z}) = \arg\min_{\mathbf{x}\in\mathbb{C}} \frac{1}{2\sigma^2}\|\mathbf{x}-\mathbf{z}\|_2^2 + \frac{\lambda}{\alpha}\mathcal{R}(\mathbf{x}). \tag{3.31}$$

It can be noted that the above Eq. (3.31) is treated as the objective function of a denoising problem, where $\mathbf{z}$ is the noisy observation with noise level σ. In this way, we heavily rely on employing a deep denoising neural network to get the required estimate $\hat{\mathbf{x}}^{(k)}$ by unrolling the MM scheme as K finite steps. Another thing to notice in Eq. (3.30), is that we decouple the degradation operator $\mathbf{H}$ from $\mathbf{x}$ and now we need to tackle it with a less complex denoising problem. However, obtaining the resulting solution $\hat{\mathbf{x}}^{(k)}$ from

Algorithm 1: The proposed SISR iterative approach. The ERD structure and parameters are shared across all iterative steps.

Input : $\mathbf{y}$: LR input, $\mathbf{H}$: Down-sampling operator, $\mathbf{H}^T$: Up-sampling operator, K: iterative steps, $\mathbf{w} \in \mathbb{R}^K$: extrapolation weights, σ: estimated noise, λ, α: projection parameters

Initialization: $\mathbf{x}^{(0)} = \mathbf{H}^T\mathbf{y}$, $\mathbf{H}^T$: Bilinear kernel;

$\mathbf{z}^{(1)} = \mathbf{x}^{(0)} + \mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{x}^{(0)})$;

for $k \leftarrow 1$ **to** K **do**

  Extrapolation step: $\mathbf{z}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{w}^{(\mathbf{k})}(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)})$;

  Proximal step (ERD-block): $\hat{\mathbf{x}}^{(k)} = \text{Prox}_{(\lambda/\alpha)\mathcal{R}(.)}(\mathbf{z}^k + \mathbf{H}^T(\mathbf{y} - \mathbf{H}\mathbf{z}^k))$;

end

Output: $\mathbf{x}^K$: SR output

Eq. (3.30) can be computationally expensive since it demands K times the parameters of the employed denoiser and can exhibit the slow convergence [4, 54]. To avoid this hurdles, we adopt the similar strategy as done in [55], where the trainable extrapolation weights $\mathbf{w}^{(\mathbf{k})}$ are learnt directly from the training data instead of the fixed ones [67]. Moreover, the convergence of our proposed method is sped up by adopting the continuation strategy [71]. Our overall proposed method is shown in Fig. 3.6 and also described in the Algorithm 1, where the input settings, initialization, extrapolation steps, and proximal steps are defined. Our proposed Algorithm 1 has a close connection with other proximal algorithms such as ISTA [19] and FISTA [5] that require the exact form of the employed regularizer such as Total Variation / Hessian Schatten-norm [65]. However, in our case, the regularizer is learned implicitly from the training data (*i.e.*, non-convex form), and therefore our algorithm acts as an inexact form of proximal gradient descent steps.

3.2.1.3 Network Architecture

The proposed network architecture for super-resolution is shown in Fig. 3.6. Given an LR image ($\mathbf{y}$) and an initial estimate ($\mathbf{x}^0$), each network's stage *ERD* (Encoder-Resnet-Decoder) produces a new estimate $\mathbf{x}^{(k+1)}$ from the previous step estimate $\mathbf{x}^{(k)}$. The Algorithm 1 describes the inputs, initial conditions, and desired updates for each

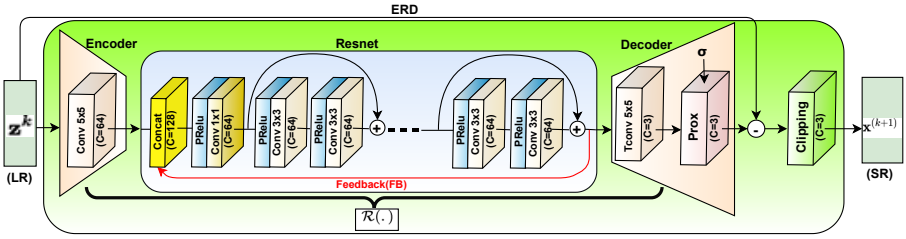


Figure 3.7: The architecture of ERD (Encoder-Resnet-Decoder) blocks used in the proposed ISRResCNet. The $\mathbf{z}^k$ is the LR noisy observation, refer to the Eq. (3.30). The $\mathcal{R}(.)$ corresponds to the regularizer learning part, refer to the Eq. (3.30). The *Prox* layer inside the *Decoder* computes the proximal map with the noise standard deviation σ, refer to the Eq. (3.31). The $\mathbf{x}^{(k+1)}$ is the new solution SR estimate.

network stage. The ERD structure and parameters are shared across all iterative steps. Finally, a single optimizer is used to minimize the ℓ_1-Loss between the estimated latent SR image ($\mathbf{x}^{(k)}$) and ground-truth (GT) ($\mathbf{x}^{(gt)}$) after k-steps as:

$$\arg\min_{\Theta} \mathcal{L}(\mathbf{\Theta}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{x}_n^k - \mathbf{x}_n^{gt}\|_1 \tag{3.32}$$

where N is the mini-batch size and $\mathbf{\Theta}$ are the trainable parameters of our network. Fig. 3.7 shows the *ERD* block used in the network. In ERD network, both *Encoder (Conv)* and *Decoder (TConv)* layers have 64 feature maps of 5×5 kernel size with $C \times H \times W$ tensors, where C is the number of channels of the input image. *Resnet* consists of 5 residual blocks with two Pre-activation *Conv* layers, each of 64 feature maps with kernels support 3×3, and the pre-activation is the parametrized rectified linear unit (PReLU)[38] with 64 out feature channels. The *Resnet* also contains the Feedback (FB) path after 5 resblocks with an initial concatenation pre-activation *Conv* layer by 1×1 kernel support that maps 128 features channels to 64 to feed into resblocks. The *Prox* layer [64] inside the *Decoder* computes the proximal map for Eq. (3.31) with given noise standard deviation σ and handle the data fidelity and prior terms. The noise realization is estimated in the intermediate *Resnet* that is sandwiched between the *Encoder* and *Decoder*. The estimated residual image after *Decoder* is subtracted from the LR input

image. Finally, the clipping layer incorporates our prior knowledge about the valid range of image intensities and enforces the pixel values of the reconstructed image to lie in the range $[0, 255]$. Reflection padding is also used before all *Conv* layers to ensure slowly-varying changes at the boundaries of the input images. Our ERD structure can also be described as the generalization of one stage TNRD [17] and UDNet [64] that have good reconstruction performance for the image denoising problem.

3.2.1.4 Network Training via TBPTT

Due to the iterative nature of our SISR approach, the network parameters are updated using back-propagation through time (BPTT) algorithm by unrolling K steps to train the network, which is previously used in recurrent neural networks training such as LSTMs. However, it is computationally expensive by increasing the number of iterative steps K, so both K and mini-batch (N) size are upper-bound on the GPU memory. Therefore, to tackle this problem, we use the Truncated Backpropagation Through Time (TBPTT) algorithm as done in [55] to train our network, where the sequence is unrolled into a small number of k-steps out of total K and then the back-propagation is performed on the small k-steps. Furthermore, we compute the ℓ_1-Loss with respect to GT images after k iterative steps according to Eq. (3.32).

Table 3.3: The settings of input LR and corresponding HR patch sizes during the network training.

Scale factor	LR Patch size	HR Patch size
$\times 2$	60×60	120×120
$\times 3$	50×50	150×150
$\times 4$	40×40	160×160

3.2.2 Experiments

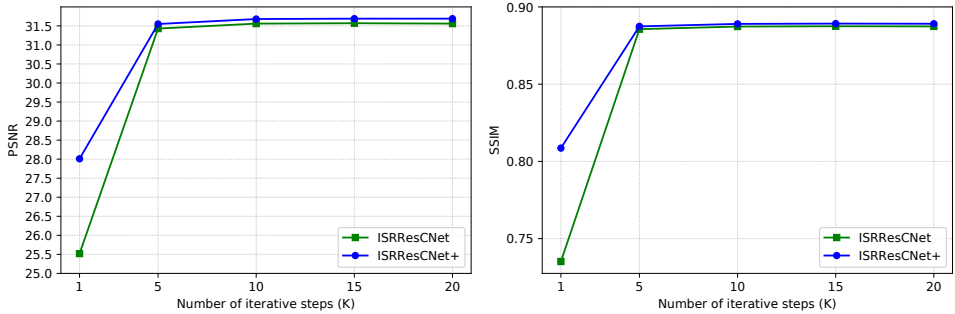
3.2.2.1 Data augmentation

We use DIV2K[1] dataset that contains 800 HR images for training. We take the input LR image patches as a bicubic downsample (*i.e.*, regarded as a standard degradation)

with their corresponding HR image patches. We augment the training data with random vertical and horizontal flipping, and 90° rotations. Moreover, we also consider another effective data augmentation technique, called *MixUp* [116]. In *Mixup*, we take randomly two samples $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ in the training HR/LR set $(\tilde{\mathbf{X}}, \mathbf{Y})$ and then form a new sample $(\tilde{\mathbf{x}}, \mathbf{y})$ by interpolation of the pair samples by following the same degradation model (3.22) as done in [27]. This simple technique encourages our network to support linear behavior among training samples.

3.2.2.2 Technical details

We use the RGB input LR and corresponding HR patches with different patch sizes according to the upscaling factor as listed in Table 3.3. We train the network for 300 epochs with a batch size of 4 using the Adam optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ without weight decay to minimize the ℓ_1-Loss (3.32). We use the method of Kaiming He [38] to initial the *Conv* weights and bias to zero. The learning rate is initially set to 10^{-3} for the first 100 epochs and then multiplies by 0.5 for every 50 epochs. We set the number of iterative steps (K) to 20 and feedback steps (FB) to 4 for our method. The extrapolation weights $\mathbf{w} \in \mathbb{R}^K$ are initialized with $\mathbf{w}_t = \frac{t^k - 1}{t^k + 2}, \forall 1 \leq t \leq K$, and then further fine-tuned on the training data as done in [55]. The projection layer parameter σ is estimated according to [72] from the input LR image. We initialize the projection layer parameter α on a log-scale value from $\alpha_{max} = 2$ to $\alpha_{min} = 1$ and then further fine-tuned during the training via back-propagation. To further enhance the performance of our network, we use a self-ensemble strategy [99] (denoted as ISRResCNet+), where the LR inputs are flipped/rotated and the SR results are aligned and averaged for enhanced prediction.



(a) PSNR vs. K (b) SSIM vs. K

Figure 3.8: Average PSNR/SSIM performance (Set5 on $\times 4$) of proposed ISRResCNet and ISRResCNet+ after each iterative step (K).

3.2.2.3 Evaluation metrics and SR benchmarks

We evaluate the trained model under the Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity (SSIM) metrics on four benchmark datasets: Set5 [8], Set14 [115], B100 [78], and Urban100 [40]. In order to keep a fair comparison with existing networks, the quantitative SR results are only evaluated on Y (luminance) channel of the transformed $YCbCr$ color space.

3.2.2.4 Ablation study of iterative (K) and feedback (FB) steps

For our ablation study, we evaluate our proposed ISRResCNet and ISRResCNet+ performance on Set5 benchmark dataset at $\times 4$ upscaling factor. Table 3.4 shows the average PSNR/SSIM performance after iterative steps (K) and feedback (FB) steps. Our trained model achieves better performance (PSNR/SSIM) by increasing the number of iterative steps 1 to 20 with the shared network parameters (*i.e.*, 380K) without using FB steps (see in Fig. 3.8 and Table 3.4). When the FB connections introduce into our network, the model converges in the less number of iterative steps (*i.e.*, 10) with better reconstruction results by requiring a few additional parameters (*i.e.*, +8K) because these error feedback connections [68] after residual blocks provide strong early recon-

Table 3.4: The impact of iterative (K) and feedback (FB) steps on ISRResCNet on the scale factor ×4. The average PSNR/SSIM values are evaluated on Set5 testset. The best performance is shown in red.

Feedback steps (FB)	Iterative steps (K)	#Params ($\times 10^3$)	ResBlocks (D)	Feature-Maps (F)	ISRResCNet	ISRResCNet+
✗	10	380	5	64	31.44 / 0.8855	31.59 / 0.8876
✗	20	380	5	64	31.56 / 0.8874	31.69 / 0.8891
✓	10	388	5	64	31.63 / 0.8890	31.77 / 0.8908

Table 3.5: Average PSNR/SSIM values for the scale factors ×2, ×3, and ×4 with the bicubic degradation model. The best performance is shown in red and the second best performance is shown in blue.

Dataset	Scale	Bicubic	SRCNN [23] (ECCV-2014)	VDSR [51] (CVPR-2016)	EDSR-baseline [69] (CVPR-2017)	RISR [47] (ICPR-2018)	SRFBN-S [68] (CVPR-2019)	ISRResCNet (Ours)	ISRResCNet+ (Ours)
Set5	×2	33.55 / 0.9304	36.16 / 0.9509	37.30 / 0.9573	37.59 / 0.9605	37.63 / 0.9590	37.39 / 0.9597	37.67 / 0.9596	37.79 / 0.9600
	×3	30.35 / 0.8686	32.28 / 0.9020	33.50 / 0.9197	34.18 / 0.9270	33.91 / 0.9234	33.99 / 0.9252	34.08 / 0.9251	34.20 / 0.9258
	×4	28.39 / 0.8109	29.99 / 0.8519	31.20 / 0.8818	31.89 / 0.8932	31.58 / 0.8870	31.76 / 0.8914	31.63 / 0.8890	31.77 / 0.8908
Set14	×2	30.05 / 0.8701	31.81 / 0.9033	32.84 / 0.9121	33.21 / 0.9177	33.16 / 0.9133	33.04 / 0.9157	32.89 / 0.9144	33.06 / 0.9155
	×3	27.40 / 0.7763	28.70 / 0.8151	29.54 / 0.8323	29.91 / 0.8421	29.91 / 0.8338	29.72 / 0.8376	29.63 / 0.8365	29.76 / 0.8381
	×4	25.86 / 0.7056	26.92 / 0.7427	27.75 / 0.7688	28.20 / 0.7820	28.19 / 0.7707	28.05 / 0.7785	27.99 / 0.7757	28.08 / 0.7776
B100	×2	29.51 / 0.8439	31.07 / 0.8838	31.83 / 0.8949	32.03 / 0.8996	32.01 / 0.8968	31.87 / 0.8972	31.98 / 0.8974	32.03 / 0.8980
	×3	27.19 / 0.7399	28.17 / 0.7799	28.80 / 0.7971	29.03 / 0.8056	28.92 / 0.7996	28.90 / 0.8015	28.91 / 0.8014	28.96 / 0.8024
	×4	25.96 / 0.6698	26.70 / 0.7029	27.27 / 0.7252	27.53 / 0.7365	27.37 / 0.7270	27.41 / 0.7321	27.40 / 0.7301	27.44 / 0.7313
Urban100	×2	26.84 / 0.8409	29.01 / 0.8885	30.67 / 0.9129	31.81 / 0.9271	31.06 / 0.9168	31.27 / 0.9208	31.29 / 0.9205	31.45 / 0.9220
	×3	24.44 / 0.7359	25.82 / 0.7874	27.09 / 0.8271	28.05 / 0.8524	27.41 / 0.8338	27.60 / 0.8418	27.57 / 0.8409	27.70 / 0.8432
	×4	23.13 / 0.6593	24.11 / 0.7051	25.14 / 0.7522	25.98 / 0.7850	25.41 / 0.7595	25.66 / 0.7725	25.56 / 0.7682	25.65 / 0.7705

struction ability. Since these error feedbacks are beneficial on the higher scale (×4), so we report the quantitative results in the Table 3.5 with feedback steps at ×4 upscaling factor, while the others (×2, ×3) are without feedback steps with 20 iterative steps. It can also be noted (see Fig. 3.8) that a few iterative steps (e.g. 5) are enough to obtain excellent SR results with the performance trade-off between quantitative results and the computation time of our method.

3.2.2.5 Comparison with the state-of-art methods

We compare our method with other state-of-art SISR methods including SRCNN [23], VDSR [51], EDSR [69], RISR [47], and SRFBN [68], whose source codes are available online except for RISR method for which the quantitative results are directly taken from the paper. We run all source codes with default parameters settings through all experiments. We report the quantitative results of our method with others in the Table 3.5. Our method exhibits better improvement in PSNR and SSIM compared to other methods, except the EDSR. Since the EDSR has a much deeper network containing

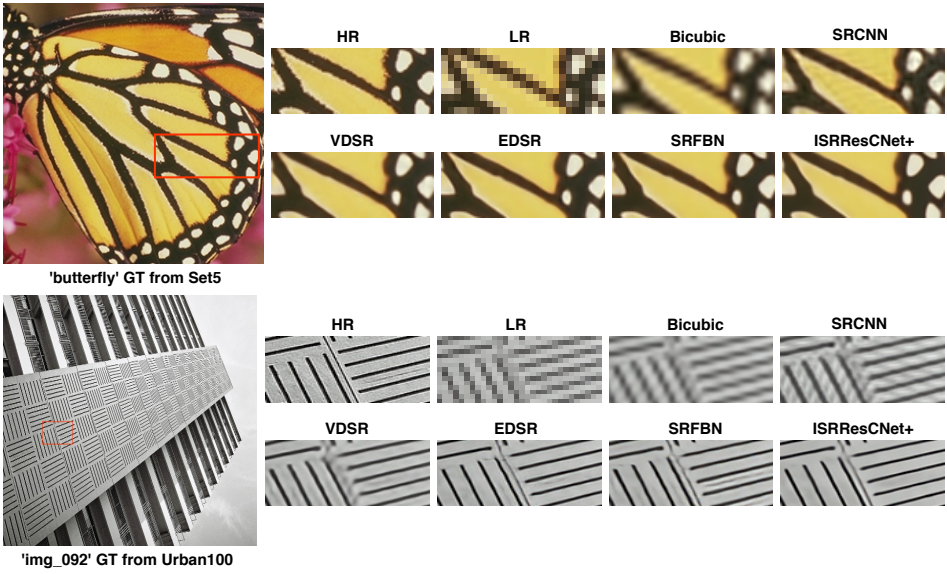


Figure 3.9: Visual comparison of our method with the other state-of-art methods on the $\times 4$ upscaling factor.

16 residual blocks with $1.5M$ parameters, while our model contains 5 residual blocks with $380K$ parameters, which is a much lighter model than EDSR with slightly performance difference in the PSNR (*i.e.*, $+0.12dB$ on Set5) at $\times 4$ upscaling factor. Despite that, the parameters of the proposed network are much less than the other state-of-art SISR networks, which makes it suitable for deployment in mobile devices where memory storage and cpu power are limited as well as good image reconstruction quality (see section 3.2.2.4).

Regarding the visual quality, Fig. 3.9 shows the visual comparison of our method with other SR methods for a high ($\times 4$) upscaling factor. The proposed method successfully reconstructs the good textures regions, sharp edges, and finer details of SR image compared to the other methods.

3.3 Deep Efficient CNNs for SISR

We propose a light-weight deep iterative SR learning method (ISRResDNet) that solves the SR task as a sub-solver of the image denoising by the residual denoiser networks. The proposed method super-resolved the LR image to the SR image by the high upscaling factor $\times 4$ which reduces the several factors such as #parameters, #Conv (depths of the network), and FLOPs, while getting the PSNR/SSIM close to the baseline model, refer to the Table 3.6. It is inspired by the powerful inherent non-local self-similarity property of natural images and the efficient proximal gradient optimization technique to solve general inverse problems.

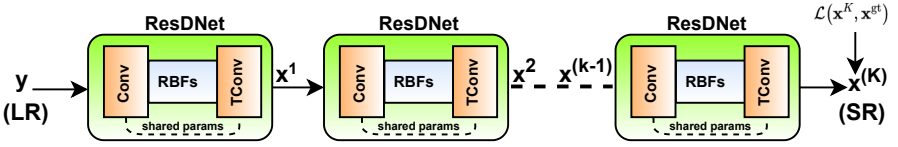


Figure 3.10: The structure of our proposed iterative SR approach. We cascade the ResDNet to the K stages. The first stage takes the LR ($\mathbf{y}$) image and the intermediate stages refine the estimated SR solution until the last stage. The loss $\mathcal{L}(.)$ function is jointly minimized with respect to all network stages.

3.3.1 Proposed method

The proposed iterative SR approach is shown in the Fig. 3.10. We unroll the ResDNet into K stages, where each stage performs the PGM [85] updates. Here, $\mathbf{y}$ is an input LR image, $\mathbf{x}^K$ is a final estimated SR image, and $\mathbf{x}^{gt}$ is the corresponding ground truth image. We learn the different parameters in each stage by jointly minimizing the loss $\mathcal{L}(\Theta)$ function with respect to all network parameters Θ. However, in each ResDNet stage, the convolution layer (*i.e.*, *Conv*) and its transposed convolution layer (*i.e.*, *TConv*) share the same parameters.

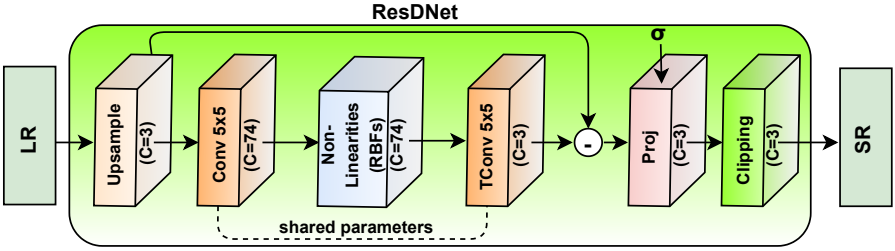


Figure 3.11: The architecture of ResDNet. Each ResDNet block contains the *Upsample* layer, the *Non-linearities (RBFs)* is sandwitched between the *Conv* and *TConv* layers which have shared parameters, the *Proj* layer computes the proximal map with noise sigma σ, finally the *clipping* layer is applied to enforce the pixel values between 0 and 255.

3.3.1.1 Network Architecture

The Fig. 3.11 shows the network architecture of ResDNet, which is the modified form of the UDNet [64] denoiser network. The LR image ($\mathbf{y}$) is upsampled by the Bilinear kernel with *Upsample* layer, where the choice of the upsampling kernel is arbitrary. Both the *Conv* and *TConv* layers have 74 feature maps of 5×5 kernel size with $C \times H \times W$ tensors, where C is the number of channels of the input image. The *RBFs* [64] layer contains the RBF-mixture with truncated Gaussian basis functions and serves as the powerful approximator for high accuracy arbitrary non-linear functions. The trainable projection layer [64] computes the proximal map with the estimated noise standard deviation σ and handles the data fidelity and prior terms. The noise realization is estimated in the intermediate *RBFs* that are sandwiched between *Conv* and *TConv*. The estimated residual features map is subtracted from the upsampled LR input image. Finally, the clipping layer incorporates our prior knowledge about the valid range of image intensities and enforces the pixel values of the reconstructed image to lie in the range $[0, 255]$. Symmetric padding is also used before *Conv* layer to ensure slowly varying changes at the boundaries of the input images. Our proposed ResDNet structure can also be described as the TNRD [17] and UDNet [64] that have good reconstruction performance for image denoising problem.

3.3.1.2 Network Losses

For the SR learning, we train the proposed ISRResDNet with the following training loss functions:

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_{tv} \tag{3.33}$$

where, these losses are defined as follows:

Content loss ($\mathcal{L}_1$): It preserves the image content and is defined as:

$$\mathcal{L}_1(\mathbf{x}^K, \mathbf{x}^{gt}) = \frac{1}{N}\sum_{i}^{N} \|\mathbf{x}_i^K - \mathbf{x}_i^{gt}\|_1 \tag{3.34}$$

where, N is represents the size of mini-batch and $\mathbf{x}^K$ is the final estimated SR image of the last stage of the network.

TV (total-variation) loss ($\mathcal{L}_{tv}$): It focuses to minimize the gradient discrepancy and produce sharpness in the output SR image and is defined as:

$$\mathcal{L}_{tv}(\mathbf{x}^K, \mathbf{x}^{gt}) = \frac{1}{N}\sum_{i}^{N} \left(\left\|\nabla_h \mathbf{x}_i^K - \nabla_h \mathbf{x}_i^{gt}\right\|_1 + \left\|\nabla_v \mathbf{x}_i^K - \nabla_v \mathbf{x}_i^{gt}\right\|_1\right) \tag{3.35}$$

Here, ∇_h and ∇_v denote the horizontal and vertical gradients of the images.

3.3.2 Experiments

3.3.2.1 Training Data

We use DIV2K [1] dataset that contains 800 HR images and Flickr2K [97] dataset that contains 2650 HR images for training. We take the input LR images as a bicubic downsampled to their corresponding HR images. We augment the training data with random vertical and horizontal flipping, and 90° rotations. The validation dataset contains 100 LR images with bicubic degradation from the DIV2K data. The test dataset contains 100 LR images provided in the AIM 2020 Efficient SR challenge [117].

Table 3.6: The quantitative SR results (×4 upscale) comparison of our method with the others over the DIV2K validation set (100 LR images) and AIM 2020 Efficient SR challenge testset [117] (100 LR images). The best performance is shown in red.

Efficient SR methods	PSNR (Val.)	PSNR (Test)	SSIM (Val.)	#Params (M)	depths (#Conv)	Runtime (Val.)(s)	#FLOPs (G)	#Activations (M)	Memory (M)
MSRResNet (Baseline)	29.00	28.70	0.8199	1.517	37	0.114	166.36	292.55	610
ISRResDNet (ours)	27.89	27.77	0.7898	0.047	10	1.313	50.66	351.27	1064

3.3.2.2 Training description

We use DIV2K dataset that contains 800 HR images with their corresponding LR images for training, provided in the AIM 2020 Challenge on Efficient Super-Resolution [117]. We further use the Flickr2K [98] dataset consisting of 2650 high-resolution images. At training time, we set the input LR patch sizes as 40×40 with their corresponding HR patch sizes as 160×160. We train the network for 300 epochs with a batch size of 16 using Adam optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ without weight decay to minimize the loss (4.15). The learning rate is initially set to 10^{-3} for the first 100 epochs and then multiplies by 0.5 for every 50 epochs. The projection layer parameter σ is estimated according to [72] from the input LR image. We unroll the network into K stages, where we set K as 5.

3.3.2.3 Quantitative Results

The Table 3.6 shows the quantitative results of our method over the DIV2K validation-set and testset (100 images) with bicubic degradation provided in the AIM 2020 Efficient SR challenge [117], and comparison to the baseline SR method.

3.3.2.4 Technical details

We implemented our method with Pytorch 1.5.0. The experiments are performed under Windows 10 with i7-8700 CPU with 32GB RAM and on NVIDIA Quadro P4000 GPU with 8GB memory. The running time at test time per image (on GPU in seconds) is shown in the Table 3.6.

3.3.3 Discussion and Limitations

We have higher the number of activations (*i.e.*, a better correlation to the Memory and the Runtime as described in [117]) than the Baseline method due the K iterative stages of our method, while we have better other factors, *i.e.*, the number of parameters, the depth of the network, and the number of FLOPs. Since the automatic differentiation [86] requires more memory than available on commercial graphical processing units (GPUs), vanilla automatic differentiation can be replaced with memory-efficient techniques [49] to solve the memory usage issue.

4

Real-World Super-Resolution

4.1 Deep Generative Adversarial Residual Convolutional Networks for Real-world SR

Most current deep learning based SISR methods focus on designing deeper / wider models to learn the nonlinear mapping between LR inputs and HR outputs from a large number of paired (LR/HR) training data. They usually take as assumption that the LR image is a bicubic down-sampled version of the HR image. However, such degradation process is not available in real-world settings *i.e.*, inherent sensor noise, stochastic noise, compression artifacts, possible mismatch between image degradation process and camera device. It reduces significantly the performance of current SISR methods due to real-world image corruptions. To address these problems, we propose a deep Super-Resolution Residual Convolutional Generative Adversarial Network (SRResCGAN) to follow the real-world degradation settings by adversarial training the model with pixel-wise supervision in the HR domain from its generated LR counterpart. The proposed network exploits the residual learning by minimizing the energy-based ob-

jective function with powerful image regularization and convex optimization techniques. We demonstrate our proposed approach in quantitative and qualitative experiments that generalize robustly to real inputs and it is easy to deploy for other down-scaling operators and mobile/embedded devices.

By referring to the general degradation model (3.2), the SISR is described as a linear forward observation model by the following image degradation process:

$$\mathbf{y} = \mathbf{H}\tilde{\mathbf{x}} + \eta, \tag{4.1}$$

where, $\mathbf{y}$ is an observed LR image, $\mathbf{H}$ is a *down-sampling operator* (usually bicubic, circulant matrix) that resizes an HR image $\tilde{\mathbf{x}}$ by a scaling factor s and η is considered as an additive white Gaussian noise with standard deviation σ. However, in real-world settings, η also accounts for all possible errors during the image acquisition process that include inherent sensor noise, stochastic noise, compression artifacts, and the possible mismatch between the forward observation model and the camera device. The operator $\mathbf{H}$ is usually ill-conditioned or singular due to the presence of unknown noises (η) that makes the SISR to the highly ill-posed nature of inverse problems.

Recently, numerous works have addressed the task of SISR using deep CNNs for their powerful feature representation capabilities. The designed SISR methods [23, 51, 92, 69, 69, 120, 122, 68, 102] mostly rely on the PSNR-based metric by optimizing the $\mathcal{L}_1/\mathcal{L}_2$ losses with blurry results, while they do not preserve the visual quality with respect to human perception. Moreover, the above mentioned methods are deeper or wider CNN networks to learn non-linear mapping from LR to HR with a large number of training samples, while neglecting the real-world settings.

For the perception SR task, a preliminary attempt was made by Ledig *et al.* [106], who proposed the SRGAN method to produce perceptually more pleasant results. To further enhance the performance of the SRGAN, Wang *et al.* [106] proposed the ESR-

Figure 4.1: ×4 Super-resolution comparison of the proposed SRResCGAN method with the ESRGAN [106] and ESRGAN-FS [31] by the unknown artifacts for the '0815' image (DIV2K validation-set). Our method has better results to handle sensor noise and other artifacts, while the others have failed to remove these artifacts.

GAN model to achieve the state-of-art perceptual performance. Despite their success, the previously mentioned methods are trained with HR/LR image pairs using the bicubic down-sampling and thus limited performance int real-world settings. More recently, Lugmayr *et al.* [75] proposed a benchmark protocol for the real-world image corruptions and introduced the real-world challenge series [76] that described the effects of bicubic downsampling and separate degradation learning for super-resolution. Later on, Fritsche *et al.* [31] proposed the DSGAN to learn degradation by training the network in an unsupervised way, and also modified the ESRGAN as ESRGAN-FS to further enhance it performance in real-world settings. However, the above methods still suffer unpleasant artifacts as shown in Fig. 4.1. Our approach takes into account the real-world settings by increasing its applicability.

Since there are many visible corruptions in the real-world images, the current state-of-the-art SISR methods often fail to produce convincing SR results as shown in the Fig. 4.1. Most of the existing SR methods rely on the known degradation operators

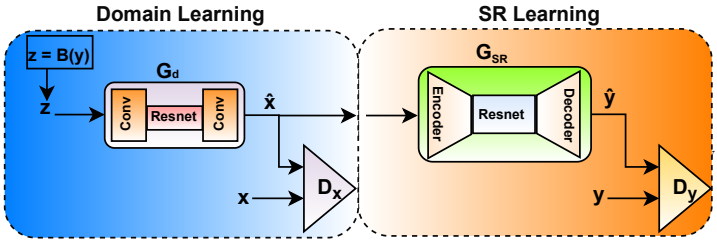


Figure 4.2: The structure of the proposed SR approach setup. In the Domain Learning part, we learn the domain distribution corruptions in the source domain ($\mathbf{x}$) by the network $\mathbf{G}_d$, where our goal is to map images from $\mathbf{z}$ to $\mathbf{x}$, while preserving the image content. Here $\mathbf{B}$ denotes the bicubic downscaling operator which is applied on the clean HR target domain ($\mathbf{y}$) images. In the SR Learning part, we trained the network $\mathbf{G}_{SR}$ in a GAN framework by using generated LR ($\hat{\mathbf{x}}$) images from the $\mathbf{G}_d$ network with their corresponding HR images.

such as bicubic with paired LR and HR images in the supervised training, while other methods do not follow the image observation (physical) model (refers to Eq. (4.1)). Three major problems arise in the existing SR methods: **(1)** the first is to train the deeper/wider (lots of model's parameters) networks from a huge volume of training data, **(2)** the second is not to generalize well for natural image characteristics due to follow the known bicubic down-sampling degradation, and **(3)** it is not easy to deploy to the current generation of smartphone cameras due to lots of network parameters and memory footprints. Therefore, we focus on a robust SISR method that is useful to improve the quality of images in such real-world settings.

In this work, we propose SR learning method (SRResCGAN) that strictly follows the image observation (physical) model (refers to Eq. (4.1)) to overcome the challenges of real-world super-resolution and is inspired by powerful image regularization and large-scale optimization techniques to solve general inverse problems (*i.e.*, easy to deployable for other downscaling operators). The visualization of our proposed SISR approach setup is shown in the Fig. 4.2. Due to the unavailability of the paired (LR/HR) data, we train firstly the domain learning network ($\mathbf{G}_d$) to generate the LR images with same characteristics as the corrupted source domain ($\mathbf{x}$). We aim to learn the distribution (real-world) mapping from bicubically down-sampled images ($\mathbf{z}$) of HR images ($\mathbf{y}$) to

the source domain images ($\mathbf{x}$), while preserving the image content. In the second part, the SR network ($\mathbf{G}_{SR}$) is trained in a GAN framework [36] by using generated LR ($\hat{\mathbf{x}}$) images with their corresponding HR images with pixel-wise supervision in the clean HR target domain ($\mathbf{y}$).

We evaluate our proposed SR method on multiple datasets with synthetic and natural image corruptions. We use the Real-World Super-resolution (RWSR) dataset [77] to show the effectiveness of our method through quantitative and qualitative experiments. Finally, we also participated in the NTIRE2020 RWSR challenges (track-1 and track-2) associated with the CVPR 2020 workshops. Table 4.2 shows the final testset results of the track-1 of our method (**MLP-SR**) with others, while we only provide the visual comparison of the track-2 since no ground truth (GT) is available (refers to Fig. 4.5), and the quantitative results of the track-2 are in the challenge report [77].

4.1.1 Proposed Methodology

4.1.1.1 Problem Formulation

By referencing to the equation (4.1), the recovery of $\mathbf{x}$ from $\mathbf{y}$ mostly relies on the variational approach for combining the observation and prior knowledge, and is given as the following objective function:

$$\hat{\mathbf{E}}(\mathbf{x}) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda \mathbf{R}_W(\mathbf{x}), \tag{4.2}$$

where $\frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2$ is the data fidelity term, $\mathbf{R}_W(\mathbf{x})$ is regularization term, and λ is the trade-off parameter that governs the compromise between data fidelity and regularizer terms.

A generic form of the regularizers in the literature [17, 64, 102, 83] is given as below:

$$\mathbf{R}_W(\mathbf{x}) = \sum_{k=1}^{K} \rho_k(\mathbf{L}_k\mathbf{x}), \tag{4.3}$$

where $\mathbf{L}$ corresponds to the first or higher-order differential linear operators such as gradient, while $\rho(.)$ denotes a potential functions such as ℓ_p vector or matrix norms that acts on the filtered outputs [58]. Thanks to the recent advances of deep learning, the regularizer (*i.e.*, $\mathbf{R}_W(\mathbf{x})$) is employed by deep convolutional neural networks (ConvNets), whose parameters are denoted by $\mathbf{W}$, that have the powerful image priors capabilities.

Besides the proper selection of the regularizer and formulation of the objective function, another important aspect of the variational approach is the minimization strategy that will be used to get the required solution. In the next subsection 4.1.1.2, we briefly explain the minimization approach to get the solution of the objective function (4.2).

4.1.1.2 Objective Function Minimization Strategy

The proper optimization strategy is employed to find $\mathbf{W}$ that minimizes the energy-based objective function to get the required latent HR image. So, we want to recover the underlying image $\mathbf{x}$ as the minimizer of the objective function in Eq. (4.2) as:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\tilde{\mathbf{x}}} \hat{\mathbf{E}}(\mathbf{x}), \tag{4.4}$$

By referencing the Eqs. (4.2) and (4.3), we can write it as:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda \sum_{k=1}^{K} \rho_k(\mathbf{L_k}\mathbf{x}), \tag{4.5}$$

Since it is reasonable to require constraints on the image intensities such as non-negativity values (*i.e.*, $\alpha = 0, \beta = +\infty$) that arise in the natural images, Eq. (4.5) can be rewritten in a constrained optimization form:

$$\hat{\mathbf{x}} = \arg\min_{\alpha\leq\mathbf{x}\leq\beta} \frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda \sum_{k=1}^{K} \rho_k(\mathbf{L_k}\mathbf{x}), \tag{4.6}$$

To solve the Eq. (4.6), there are several modern convex optimization schemes for large-scale machine learning problems, such as HQS method [33], ADMM [12], and Proximal methods [85]. In our work, we solve the under study problem in (4.6) by using the Proximal Gradient Method (PGM) [85], which is a generalization of the gradient descent algorithm. PGM deals with the optimization of a function that is not fully differentiable, but it can be split into a smooth and a non-smooth part. To do so, we rewrite the problem in (4.6) as:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \underbrace{\frac{1}{2}\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda \sum_{k=1}^{K} \rho_k(\mathbf{L_k}\mathbf{x})}_{\mathbf{F}(\mathbf{x})} + \iota_c(\mathbf{x}), \tag{4.7}$$

where, ι_c is the indicator function on the convex set $\mathbb{C} \in \{\mathbf{x} \in \mathbb{R}^{N\times N} : \forall k, \alpha \leqslant \mathbf{x}_k \leqslant \beta\}$. In [64], Lefkimmiatis proposed a trainable projection layer that computes the proximal map for the indicator function as:

$$\iota_c(\mathbf{x},\ \varepsilon) = \begin{cases} 0 \quad , & if \quad \|\mathbf{x}\|_2 \leqslant \varepsilon \\ +\infty \ , & otherwise \end{cases} \tag{4.8}$$

where, $\epsilon = e^{\alpha}\sigma\sqrt{C \times H \times W - 1}$ is the parametrized threshold, in which α is a trainable parameter, σ is the noise level, and $C \times H \times W$ is the total number of pixels in the image.

Thus, the solution of the problem in (4.7) is given by the PGM by the following update rule:

$$\mathbf{x}^t = \mathrm{Prox}_{\gamma^t \iota_c}\left(\mathbf{x}^{(t-1)} - \gamma^t \nabla_{\mathbf{x}} \mathbf{F}(\mathbf{x}^{(t-1)})\right), \tag{4.9}$$

where, γ^t is a step-size and $\mathrm{Prox}_{\gamma^t \iota_c}$ is the proximal operator [85], related to the indicator function ι_c, that is defined as:

$$\mathbf{P}_{\mathbb{C}}(\mathbf{z}) = \arg\min_{\mathbf{x}\in\mathbf{C}} \frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{z}\|_2^2 + \iota_c(\mathbf{x}), \tag{4.10}$$

The gradient of the $\mathbf{F}(\mathbf{x})$ is computed as:

$$\nabla_{\mathbf{x}}\mathbf{F}(\mathbf{x}) = \mathbf{H}^T(\mathbf{H}\mathbf{x} - \mathbf{y}) + \lambda \sum_{k=1}^{K} \mathbf{L_k}^T \phi_k(\mathbf{L_k}\mathbf{x}), \tag{4.11}$$

where, $\phi_k(.)$ is the gradient of the potential function (ρ_k). By combining the Eqs. (4.9), (4.10) and (4.11), we have the final form of our solution as:

$$\mathbf{x}^t = \mathbf{P}_{\mathbb{C}} \left((1 - \gamma^t \mathbf{H}^T\mathbf{H})\mathbf{x}^{(t-1)} + \ \gamma^t \mathbf{H}^T\mathbf{y} - \lambda\gamma^t \sum_{k=1}^{K} \mathbf{L_k}^T \phi_k(\mathbf{L_k}\mathbf{x}^{(\mathbf{t-1})}) \right), \tag{4.12}$$

The formulation in Eq. (4.12) can be thought as performing one proximal gradient descent inference step at starting points $\mathbf{y}$ and $\mathbf{x}^{(0)} = 0$, which is given by:

$$\mathbf{x} = \mathbf{P}_{\mathbb{C}} \left(\mathbf{H}^T\mathbf{y} - \alpha \sum_{k=1}^{K} \mathbf{L_k}^T \phi_k(\mathbf{L_k}\mathbf{y}) \right), \tag{4.13}$$

where, $\alpha = \lambda\gamma$ corresponds to the projection layer trainable parameter, $\mathbf{L_k}^T$ is the adjoint filter (*i.e.*, transpose convolution) of $\mathbf{L_k}$, and $\mathbf{H}^T$ represents the up-scaling operation.

Thus, we design the generator network ($\mathbf{G}_{SR}$, refers to Fig. 4.3-(a)) according to Eq. (4.13), where $\phi_k(.)$ corresponds to a point-wise non-linearity (*i.e.*, *PReLU*) applied to convolution feature maps. It can be noted that most of the parameters in Eq. (4.13) are derived from the prior term of Eq. (4.2), which leads to the proposed generator network representing most of its parameters as image priors. To learn the valid weights of regularization parameters, the weights should be zero-mean and fixed-scale constraints. To tackle this, we use the same parametrization technique proposed in [64]. Our generator network structure can also be described as the generalization of one stage TNRD [17], UDNet [64], and SRWDNet [102] that have good reconstruction performance for the image denoising problem.

4.1.2 Domain Learning

To learn the domain distribution corruptions from the source domain ($\mathbf{x}$), we train the network $\mathbf{G}_d$ (see in the Fig. 4.2) in a GAN framework [36] as done in DSGAN [31] with the following loss function:

$$\mathcal{L}_{\mathbf{G}_d} = \mathcal{L}_{color} + 0.005 \cdot \mathcal{L}_{tex} + 0.01 \cdot \mathcal{L}_{per} \tag{4.14}$$

where, $\mathcal{L}_{color}, \mathcal{L}_{tex}, \mathcal{L}_{per}$ denote the color loss (*i.e.*, $\mathcal{L}_1$ loss focuses on the low frequency of the image), texture/GAN loss (*i.e.*, focus on the high frequencies of the image), and perceptual loss (*i.e.*, VGG-based loss, refer to section 4.1.3.1 for more details), respectively.

4.1.2.1 Network architectures

The generator network ($\mathbf{G}_d$) consists of 8 *Resnet* blocks (two *Conv* layers and PReLU activations in between) that are sandwiched between two *Conv* layers. All *Conv* layers have 3×3 kernel support with 64 feature maps. Finally, *sigmoid* non-linearity is applied on the output of the $\mathbf{G}_d$ network. While, the discriminator network ($\mathbf{D}_{\mathbf{x}}$) consists of a three-layer convolutional network that operates on a patch level [46, 66]. All *Conv* layers have 5×5 kernel support with feature maps from 64 to 256 and also applied Batch Normalization (BN) and Leaky ReLU (LReLU) activations after each *Conv* layer except the last *Conv* layer that maps 256 to 1 features.

4.1.2.2 Training description

We train the $\mathbf{G}_d$ network with image patches 512×512, which are bicubically downsampled with MATLAB *imresize* function. We randomly crop source domain images ($\mathbf{x}$) by 128×128 as done in [31]. We train the network for 300 epochs with a batch size of 16 using Adam optimizer [52] with parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ without

weight decay for both generator and discriminator to minimize the loss in (4.14). The learning rate is initially set to 2×10^{-4} for first 150 epochs and then linearly decay to zero after the remaining (*i.e.*, 150) epochs as done in [31].

4.1.3 Super-Resolution Learning

4.1.3.1 Network Losses

To learn the super-resolution for the target domain, we train the proposed ($\mathbf{G}_{SR}$) network in a GAN framework [36] with the following loss functions:

$$\mathcal{L}_{G_{SR}} = \mathcal{L}_{\text{per}} + \mathcal{L}_{\text{GAN}} + \mathcal{L}_{tv} + 10 \cdot \mathcal{L}_1 \tag{4.15}$$

where these loss functions are defined as follows:

Perceptual loss ($\mathcal{L}_{\text{per}}$): It focuses on the perceptual quality of the output image and is defined as:

$$\mathcal{L}_{\text{per}} = \frac{1}{N} \sum_{i}^{N} \mathcal{L}_{\text{VGG}} = \frac{1}{N} \sum_{i}^{N} \|\phi(\mathbf{G}_{SR}(\hat{\mathbf{x}}_i)) - \phi(\mathbf{y}_i)\|_1 \tag{4.16}$$

where, ϕ is the feature extracted from the pretrained VGG-19 network at the same depth as ESRGAN [106].

Texture loss ($\mathcal{L}_{\text{GAN}}$): It focuses on the high frequencies of the output image and is defined as:

$$\begin{aligned} \mathcal{L}_{\text{GAN}} = \mathcal{L}_{\text{RaGAN}} = &- \mathbb{E}_{\mathbf{y}} \left[\log \left(1 - \mathbf{D}_{\mathbf{y}}(\mathbf{y}, \mathbf{G}_{SR}(\hat{\mathbf{x}}))\right)\right] \\ &- \mathbb{E}_{\hat{\mathbf{y}}} \left[\log \left(\mathbf{D}_{\mathbf{y}}(\mathbf{G}_{SR}(\hat{\mathbf{x}}), \mathbf{y})\right)\right] \end{aligned} \tag{4.17}$$

where, $\mathbb{E}_{\mathbf{y}}$ and $\mathbb{E}_{\hat{\mathbf{y}}}$ represent the operations of taking average for all real ($\mathbf{y}$) and fake ($\hat{\mathbf{y}}$) data in the mini-batches respectively. We employed the relativistic discriminator used in the ESRGAN [106] that provides the relative GAN score of real HR and fake

SR image patches and is defined as:

$$\mathbf{D}_{\mathbf{y}}(\mathbf{y}, \hat{\mathbf{y}})(C) = \sigma(C(\mathbf{y}) - \mathbb{E}[C(\hat{\mathbf{y}})]) \tag{4.18}$$

where, C is the raw discriminator output (see in the Fig. 4.3-(b)) and σ is the sigmoid function.

Content loss ($\mathcal{L}_1$): It is defined as:

$$\mathcal{L}_1 = \frac{1}{N} \sum_{i}^{N} \|\mathbf{G}_{SR}(\hat{\mathbf{x}}_i) - \mathbf{y}_i\|_1 \tag{4.19}$$

where, N is represents the size of mini-batch.

TV (total-variation) loss ($\mathcal{L}_{tv}$): It focuses to minimize the gradient discrepancy and produce sharpness in the output image and is defined as:

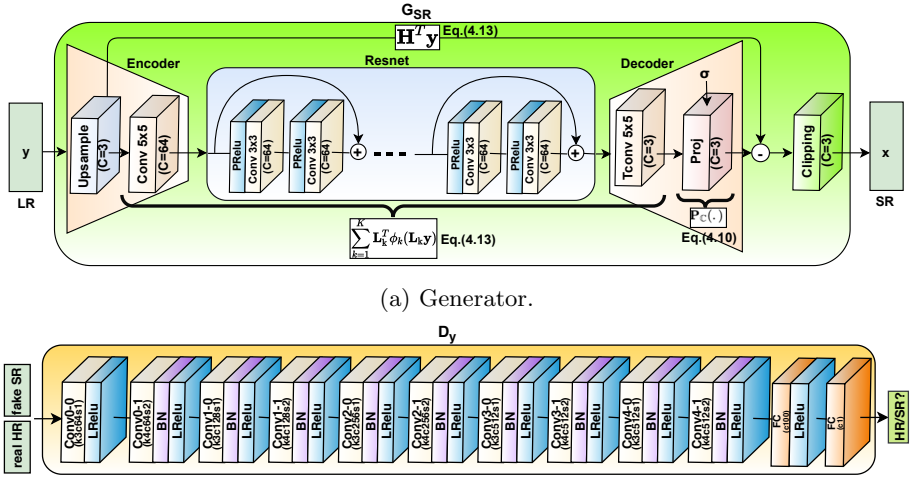
$$\begin{aligned} \mathcal{L}_{tv} = \frac{1}{N} \sum_{i}^{N} (\|\nabla_h \mathbf{G}_{SR}(\hat{\mathbf{x}}_i) - \nabla_h(\mathbf{y}_i)\|_1 + \\ \|\nabla_v \mathbf{G}_{SR}(\hat{\mathbf{x}}_i) - \nabla_v(\mathbf{y}_i)\|_1) \end{aligned} \tag{4.20}$$

where, ∇_h and ∇_v denote the horizontal and vertical gradients of the images, and N is the size of mini-batch.

4.1.3.2 Network Architectures

Figure 4.3 shows the network architectures of both Generator ($\mathbf{G}_{SR}$) and Discriminator ($\mathbf{D}_{\mathbf{y}}$).

Generator ($\mathbf{G}_{SR}$): We design the generator network according to the optimization update formula in (4.13). In the $\mathbf{G}_{SR}$ network (refers to Fig. 4.3-(a)), both *Encoder (Conv, refers to* $\mathbf{L_k}$ *filters)* and *Decoder (TConv, refers to* ${\mathbf{L_k}}^T$ *filters)* layers have 64 feature maps of 5×5 kernel size with $C \times H \times W$ tensors, where C is the number of channels of the input image. Inside the *Encoder*, LR image ($\mathbf{y}$) is upsampled by the Bilinear



(a) Generator.

(b) Discriminator.

Figure 4.3: The architectures of Generator and Discriminator networks. The k, c, s denote the kernel size, number of filters, and stride size.

kernel with *Upsample* layer (refers to the operation $\mathbf{H}^T\mathbf{y}$), where the choice of the up-sampling kernel is arbitrary. *Resnet* consists of 5 residual blocks with two Pre-activation *Conv* layers, each of 64 feature maps with kernels support 3×3. The pre-activations (refers to the learnable non-linearity functions $\phi_k(.)$) are the parametrized rectified linear unit (PReLU) with 64 out feature channels support. The trainable projection (*Proj*) layer [64] (refers to the proximal operator $\mathbf{P}_{\mathbb{C}}$) inside *Decoder* computes the proximal map with the estimated noise standard deviation σ and handles the data fidelity and prior terms. Moreover, the *Proj* layer parameter α is fine-tuned during the training via a back-propagation. The noise realization is estimated in the intermediate *Resnet* that is sandwiched between *Encoder* and *Decoder*. The estimated residual image after *Decoder* is subtracted from the LR input image. Finally, the clipping layer incorporates our prior knowledge about the valid range of image intensities and enforces the pixel values of the reconstructed image to lie in the range $[0, 255]$. Reflection padding is also used before all *Conv* layers to ensure slowly-varying changes at the boundaries of the input images.

Discriminator ($\mathbf{D_y}$): The Figure 4.3-(b) shows the architecture of discriminator network that is trained to discriminate real HR images from generated fake SR images. The raw discriminator network contains 10 convolutional layers with kernel support 3×3 and 4×4 of increasing feature maps from 64 to 512 followed by Batch Normalization (BN) and leaky ReLU as done in SRGAN [60].

4.1.3.3 Training description

At the training time, we set the input LR patch size as 32×32. We train the network for 51000 training iterations with a batch size of 16 using Adam optimizer [52] with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ without weight decay for both generator and discriminator to minimize the loss in (4.15). The learning rate is initially set to 10^{-4} and then multiplies by 0.5 after 5K, 10K, 20K, and 30K iterations. The projection layer parameter σ is estimated according to [72] from the input LR image. We initialize the projection layer parameter α on a log-scale values from $\alpha_{max} = 2$ to $\alpha_{min} = 1$ and then further fine-tune during the training via a back-propagation.

4.1.4 Experiments

4.1.4.1 Training data

We use the source domain data ($\mathbf{x}$: 2650 HR images) that are corrupted with unknown degradation e.g. sensor noise, compression artifacts, etc. and target domain data ($\mathbf{y}$: 800 clean HR images) provided in the NTIRE2020 Real-World Super-resolution (RWSR) Challenge track1 [77]. We use the source and target domain data for training the $\mathbf{G}_d$ network to learn the domain corruptions, while due to unavailability of paired LR/HR data, we train the $\mathbf{G}_{SR}$ network (refers to section-4.3.1.3) with generated LR data ($\hat{x}$) from the $\mathbf{G}_d$ network (refers to section-4.1.2) with their corresponding HR target ($\mathbf{y}$) images.

4.1.4.2 Data augmentation

We take the input LR image patches as generated by the domain learning $\mathbf{G}_d$ network (refers to section 4.1.2) with their corresponding HR image patches. Due to the network training efficiency, we take the LR/HR patches and also we assume that the patch based degradation is same as in the whole image. We augment the training data with random vertical and horizontal flipping, and 90° rotations. Moreover, we also consider another effective data augmentation technique, called *MixUp* [116]. In *Mixup*, we take randomly two samples $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ in the training LR/HR set $(\tilde{\mathbf{X}}, \mathbf{Y})$ and then form a new sample $(\tilde{\mathbf{x}}, \mathbf{y})$ by interpolations of the pair samples by following the same degradation model (3.22) as done in [27]. This simple technique encourages our network to support linear behavior among training samples.

4.1.4.3 Technical details

We implemented our method with Pytorch. The experiments are performed under Windows 10 with i7-8750H CPU with 16GB RAM and on NVIDIA RTX-2070 GPU with 8GB memory. It takes about 28.57 hours to train the model. The run time per image (on GPU) is 0.1289 seconds for the testset. To further enhance the fidelity, we use a self-ensemble strategy [99] (denoted as SRResCGAN+) at the test time, where the LR inputs are flipped/rotated and the SR results are aligned and averaged for enhanced prediction.

4.1.4.4 Evaluation metrics

We evaluate the trained model under the Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), and LPIPS [124] metrics. The PSNR and SSIM are distortion-based measures that correlate poorly with actual perceived similarity, while LPIPS better correlates with human perception than the distortion-based/handcrafted measures. As LPIPS is based on the features of pretrained neural networks, so we use it for the

Table 4.1: Top section: $\times 4$ SR quantitative results comparison of our method over the DIV2K validation-set (100 images) with added two known degradation *i.e.*, sensor noise ($\sigma = 8$) and JPEG compression ($quality = 30$) artifacts. Middle section: $\times 4$ SR results with the unknown corruptions in the RWSR challenge track-1 (validation-set) [77]. Bottom section: $\times 4$ SR comparison with the unknown corruptions in the RWSR challenge series [76, 77]. The arrows indicate if high $\uparrow$ or low $\downarrow$ values are desired. The best performance is shown in red.

Dataset (HR/LR pairs)	SR methods	#Params	PSNR↑	SSIM↑	LPIPS↓	Artifacts
Bicubic	EDSR [69]	$43M$	24.48	0.53	0.6800	Sensor noise ($\sigma = 8$)
Bicubic	EDSR [69]	$43M$	23.75	0.62	0.5400	JPEG compression (quality=30)
Bicubic	ESRGAN [106]	$16.7M$	17.39	0.19	0.9400	Sensor noise ($\sigma = 8$)
Bicubic	ESRGAN [106]	$16.7M$	22.43	0.58	0.5300	JPEG compression (quality=30)
CycleGAN [75]	ESRGAN-FT [75]	$16.7M$	22.42	0.55	0.3645	Sensor noise ($\sigma = 8$)
CycleGAN [75]	ESRGAN-FT [75]	$16.7M$	22.80	0.57	0.3729	JPEG compression (quality=30)
DSGAN [31]	ESRGAN-FS [31]	$16.7M$	22.52	0.52	0.3300	Sensor noise ($\sigma = 8$)
DSGAN [31]	ESRGAN-FS [31]	$16.7M$	20.39	0.50	0.4200	JPEG compression (quality=30)
DSGAN [31]	SRResCGAN (ours)	$380K$	25.46	0.67	0.3604	Sensor noise ($\sigma = 8$)
DSGAN [31]	SRResCGAN (ours)	$380K$	23.34	0.59	0.4431	JPEG compression (quality=30)
DSGAN [31]	SRResCGAN+ (ours)	$380K$	26.01	0.71	0.3871	Sensor noise ($\sigma = 8$)
DSGAN [31]	SRResCGAN+ (ours)	$380K$	23.69	0.62	0.4663	JPEG compression (quality=30)
DSGAN [31]	SRResCGAN (ours)	$380K$	25.05	0.67	0.3357	unknown (validset) [77]
DSGAN [31]	SRResCGAN+ (ours)	$380K$	25.96	0.71	0.3401	unknown (validset) [77]
DSGAN [31]	ESRGAN-FS [31]	$16.7M$	20.72	0.52	0.4000	unknown (testset) [76]
DSGAN [31]	SRResCGAN (ours)	$380K$	24.87	0.68	0.3250	unknown (testset) [77]

quantitative evaluation with features of AlexNet [124]. The quantitative SR results are evaluated with the RGB color space.

4.1.4.5 Comparison with the state-of-art methods

We compare our method with other state-of-art SR methods including EDSR [69], ESRGAN [106], ESRGAN-FT [75], and ESRGAN-FS [31]. Table 4.1 shows the quantitative results comparison of our method over the DIV2K validation-set (100 images) with two known degradation (*i.e.*, sensor noise, JPEG compression) as well as unknown degradation in the RWSR challenge series [76, 77]. Our method results outperform in term of PSNR and SSIM compared to other methods, while in the case of LPIPS, we are slightly behind the ESRGAN-FS (*i.e.*, sensor noise ($\sigma = 8$), JPEG compression ($quality = 30$)), but ESRGAN-FS has the worst PSNR and SSIM values. We have much better LPIPS (+0.08) than the ESRGAN-FS (winner of AIM2019 RWSR challenge [76]) with unknown artifacts. The ESRGAN-FT has a good LPIPS value, but it achieved the worst PSNR and SSIM scores. Despite that, the parameters of the proposed $\mathbf{G}_{SR}$ network are much less (*i.e.*, $\times 44$) than the other state-of-art SISR networks, which makes it suitable

for deployment in mobile/embedded devices where memory storage and CPU power are limited as well as good image reconstruction quality.

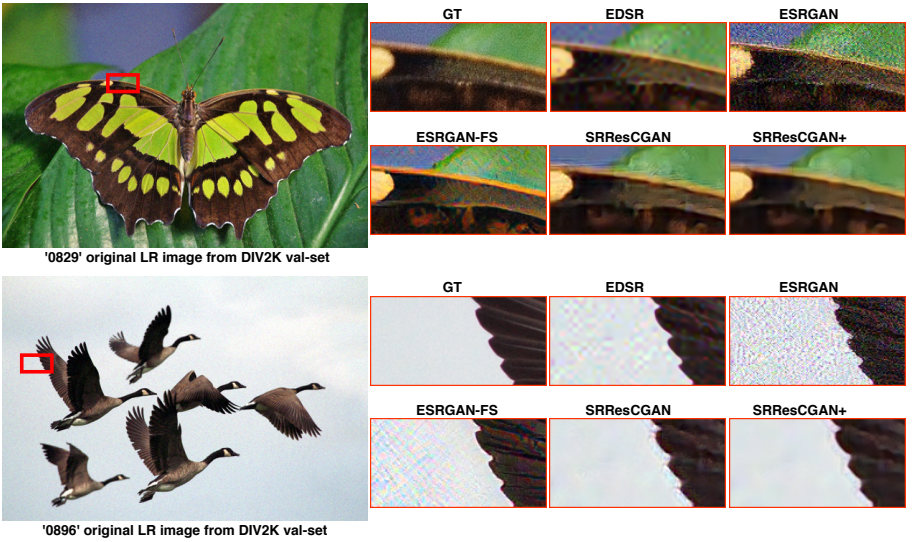


Figure 4.4: Visual comparison of our method with other state-of-art methods on the NTIRE2020 RWSR (track-1) validation set at the $\times 4$ super-resolution.

Regarding the visual quality, Fig. 4.4 shows the qualitative comparison of our method with other SR methods on the $\times 4$ upscaling factor (validation-set). In contrast to the existing state-of-art methods, our proposed method produces very good SR results that are reflected in the PSNR/SSIM/LPIPS values, as well as the visual quality of the reconstructed images with almost no visible corruptions.

4.1.4.6 Visual comparison on the Real-World smartphone images

We also evaluate our proposed method on the real-world images captured from the smartphone provided in the RWSR challenge track-2 [77] (testset). We use the our pretrained model (refers to section-4.3.1.3) without any fine-tuning from the source domain data of the smartphone images for getting the SR results. Since there are no GT images available, we only compare the visual comparison as shown in the Fig. 4.5. ESRGAN still produces strong noise presence artifacts, while the EDSR produce less

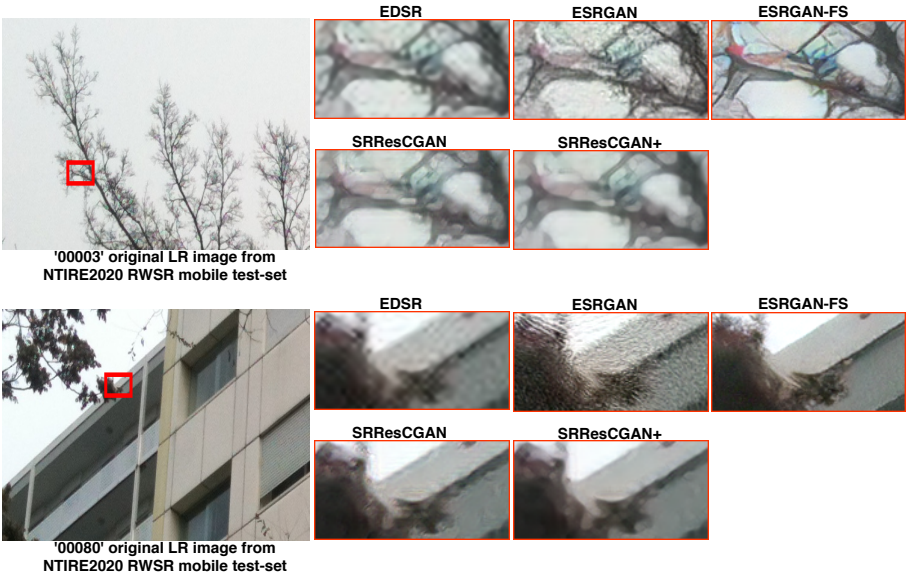


Figure 4.5: Visual comparison of our method with other state-of-art methods on the NTIRE2020 RWSR (track-2: Smartphone Images) test set [77] at the $\times 4$ super-resolution.

noisy, but more blurry results due the PSNR-based metric. ESRGAN-FS produces sharp images and less amount of corruptions due to fine-tuning of the source domain images (*i.e.*, extra training efforts). In contract, our method has still produced satisfying results by reducing the visible corruptions without any extra fine-tuning effort.

4.1.4.7 The NTIRE2020 RWSR Challenge

We also participated in the NTIRE2020 Real-World Super-Resolution (RWSR) Challenge [77] associated with the CVPR 2020 workshops. The goal of this challenge is to super-resolve ($\times 4$) images from the Source Domain (corrupted) to the Target Domain (clean). We train firstly the domain learning model ($\mathbf{G}_d$) on the corrupted source domain dataset to learn visible corruptions, and after that train the SR learning model ($\mathbf{G}_{SR}$) on the clean target domain dataset with their correspond generated LR pairs from the ($\mathbf{G}_d$) model (refers to the sections-4.1.2 and 4.3.1.3 for more details). Ta-

Table 4.2: Final testset results for the RWSR challenge Track-1. The top section in the table contains ours (**MLP-SR**) with other methods that are ranked in the challenge. The middle section contains participating approaches that deviated from the challenge rules, whose results are reported for reference but not ranked. The bottom section contains baseline approaches. Participating methods are ranked according to their Mean Opinion Score (MOS).

Team	PSNR↑	SSIM↑	LPIPS↓	MOS↓
Impressionism	24.67 (16)	0.683 (13)	0.232 (1)	2.195 (1)
Samsung-SLSI-MSL	25.59 (12)	0.727 (9)	0.252 (2)	2.425 (2)
BOE-IOT-AIBD	26.71 (4)	0.761 (4)	0.280 (4)	2.495 (3)
MSMers	23.20 (18)	0.651 (17)	0.272 (3)	2.530 (4)
KU-ISPL	26.23 (6)	0.747 (7)	0.327 (8)	2.695 (5)
InnoPeak-SR	26.54 (5)	0.746 (8)	0.302 (5)	2.740 (6)
ITS425	27.08 (2)	0.779 (1)	0.325 (6)	2.770 (7)
MLP-SR	24.87 (15)	0.681 (14)	0.325 (7)	2.905 (8)
Webbzhou	26.10 (9)	0.764 (3)	0.341 (9)	-
SR-DL	25.67 (11)	0.718 (10)	0.364 (10)	-
TeamAY	27.09 (1)	0.773 (2)	0.369 (11)	-
BIGFEATURE-CAMERA	26.18 (7)	0.750 (6)	0.372 (12)	-
BMIPL-UNIST-YH-1	26.73 (3)	0.752 (5)	0.379 (13)	-
SVNIT1-A	21.22 (19)	0.576 (19)	0.397 (14)	-
KU-ISPL2	25.27 (14)	0.680 (15)	0.460 (15)	-
SuperT	25.79 (10)	0.699 (12)	0.469 (16)	-
GDUT-wp	26.11 (8)	0.706 (11)	0.496 (17)	-
SVNIT1-B	24.21 (17)	0.617 (18)	0.562 (18)	-
SVNIT2	25.39 (13)	0.674 (16)	0.615 (19)	-
AITA-Noah-A	24.65 (-)	0.699 (-)	0.222 (-)	2.245
AITA-Noah-B	25.72 (-)	0.737 (-)	0.223 (-)	2.285
Bicubic	25.48 (-)	0.680 (-)	0.612 (-)	3.050
ESRGAN Supervised	24.74 (-)	0.695 (-)	0.207 (-)	2.300

ble 4.2 provides the final $\times 4$ SR results for track-1 (testset) of our method (**MLP-SR**) with others. The final ranking is based on the Mean Opinion Score (MOS) [77]. Our method remains among the top 8 best solutions. We also provide the visual comparison of our method with others on the track-1 testset in the Fig. 4.6. Our method produces sharp images without any visible corruptions, while the others suffer image corruptions.

Table 4.3: This table reports the quantitative results of our method over the DIV2K validation set (100 images) with unknown degradation for our ablation study. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red.

SR methods	SR Generator Loss combinations ($\mathcal{L}_{G_{SR}}$)	unknown artifacts		
		PSNR↑	SSIM↑	LPIPS↓
SRResCGAN	$\mathcal{L}_{\text{per}} + \mathcal{L}_{\text{GAN}} + 10 \cdot \mathcal{L}_1$	25.48	0.69	0.3458
SRResCGAN	$\mathcal{L}_{\text{per}} + \mathcal{L}_{\text{GAN}} + \mathcal{L}_{tv} + 10 \cdot \mathcal{L}_1$	25.40	0.69	0.3452
SRResCGAN	$\mathcal{L}_{\text{per}} + \mathbf{w}_H * \mathcal{L}_{\text{GAN}} + \mathcal{L}_{tv} + 10 \cdot \mathcal{L}_1$	25.05	0.67	0.3357

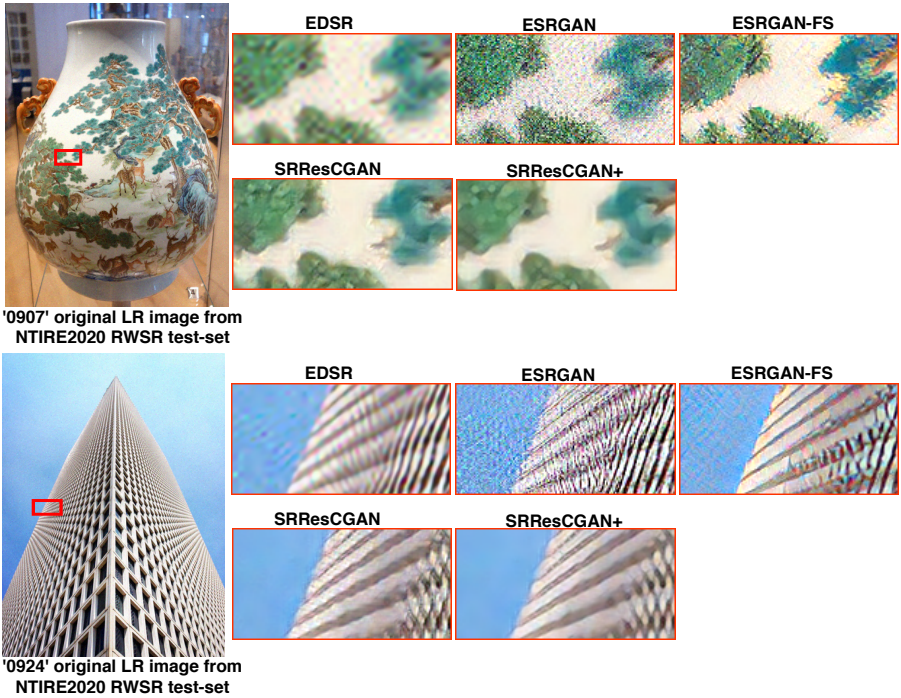


Figure 4.6: Visual comparison of our method with other state-of-art methods on the NTIRE2020 RWSR (track-1: Image Processing Artifacts) testset [77] at the $\times 4$ super-resolution.

4.1.4.8 Ablation Study

For our ablation study, we compare the different combinations of losses of the proposed SR learning model ($\mathbf{G}_{SR}$). We consider the LPIPS measure for its better visual correlation with the human perception. Table 4.3 shows the quantitative results of our method over the DIV2K validation-set (track-1) [77] with unknown degradation. We first train the SR model with combination of the losses ($\mathcal{L}_{\text{per}}, \mathcal{L}_{\text{GAN}}, \mathcal{L}_1$) similar to ESRGAN. After that, we add $\mathcal{L}_{tv}$ to the previous combinations, and train the model again, we obtain little a bit better LPIPS with sharp SR images. Finally, when we apply the high-pass filter ($\mathbf{w}_H$) weights to the output image to compute the GAN loss ($\mathcal{L}_{\text{GAN}}$) focus on the high-frequencies with the previous combinations during training the network, we get the best LPIPS value (*i.e.*, +0.01 improvement to the previous variants) with more realistic

SR images. Therefore, we opt the last one as the final combination of loss functions for our model ($\mathbf{G}_{SR}$) training and used for the evaluation in section 4.1.4.5.

4.2 Deep Cyclic Generative Adversarial Residual Convolutional Networks for Real-Image SR

The deep learning based SISR methods mostly train their models in a clean data domain where the LR and HR images come from noise-free settings (*i.e.*, same domain) due to the bicubic downsampling assumption. However, such degradation process is not available in real-world settings. We consider a deep cyclic network structure to maintain the domain consistency between the LR and HR data distributions, which is inspired by the recent success of CycleGAN in image-to-image translation applications. We propose the Super-Resolution Residual Cyclic Generative Adversarial Network (SRResCycGAN) by training with a generative adversarial network (GAN) framework for the LR to HR domain translation in an end-to-end manner. Our method achieves excellent SR results in terms of the PSNR/SSIM values as well as visual quality compared to the existing state-of-art methods.

Numerous works have been proposed towards the task of SISR that are based on deep CNNs either on PSNR values [51, 69, 120, 122, 113, 68, 123, 102, 83] or on visual quality [61, 106, 75, 31, 81]. The SR methods mostly rely on the known degradation operators such as bicubic (*i.e.*, noise-free) with paired LR and HR images (same clean domain) in the supervised training, while other methods do not follow the image observation (physical) model (refer to Eq. (4.1)). In the real-world settings, the input LR images suffer from different kinds of degradation or LR is different from the HR domain. Under such circumstances, these SR methods often fail to produce convincing SR results. In the Fig. 4.7, we show the results of the state-of-art deep learning method–ESRGAN on the noisy input image. The degraded ESRGAN SR result is due to the difference of

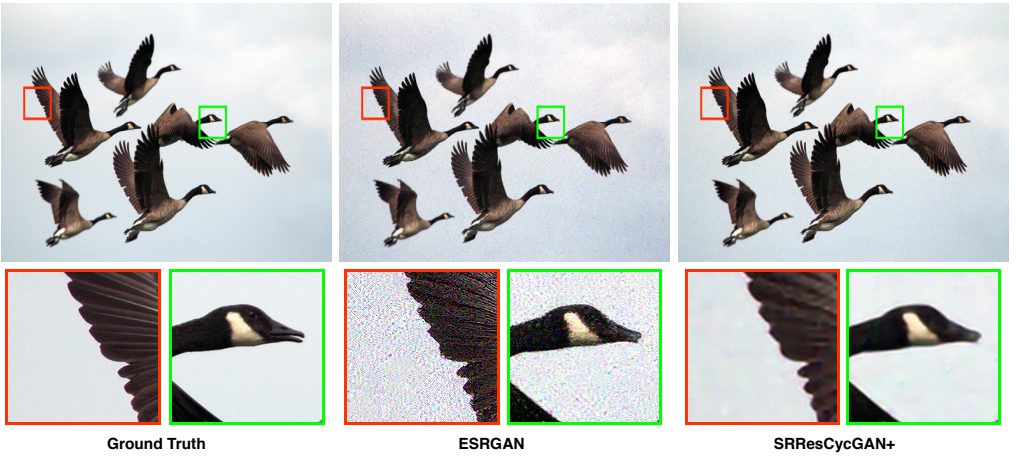


Figure 4.7: The super-resolution results at the $\times 4$ upscaling factor of the state-of-art– ESRGAN, the proposed SRResCycGAN+ with respect to the ground-truth images. SRResCycGAN+ has successfully remove the visible artifacts, while the ESRGAN has still artifacts due to data bias between the training and testing images.

training and testing data domains. The detailed analysis of the deep learning-based SR models on the real-world data can be found in the recent literature [75, 31].

In this work, we propose a SR learning method (SRResCycGAN) that overcomes the challenges of real image super-resolution. It is inspired by CycleGAN [126] structure which maintains the domain consistency between the LR and HR domains. It is also inspired by powerful image regularization and large-scale optimization techniques to solve general inverse problems in the past. The scheme of our proposed real image SR approach setup is shown in the Fig. 4.8. The $\mathbf{G}_{SR}$ network takes as input the LR image and produces the SR as output with the supervision of the SR discriminator network $\mathbf{D}_{\mathbf{x}}$. For the domain consistency between the LR and HR, the $\mathbf{G}_{LR}$ network reconstructs the LR image from the SR output with the supervision of the LR discriminator network $\mathbf{D}_{\mathbf{y}}$.

We evaluate our proposed SR method on multiple datasets with synthetic and natural image corruptions. We use the Real-World Super-resolution (RWSR) dataset [77] to show the effectiveness of our method through quantitative and qualitative experi-

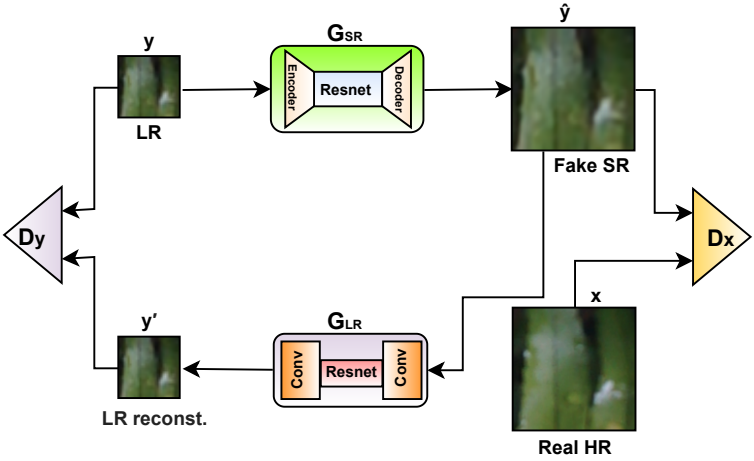


Figure 4.8: The structure of our proposed SR approach setup. We trained the network $\mathbf{G}_{SR}$ in a GAN framework, where our goal is to map images from the LR ($\mathbf{y}$) to the HR ($\mathbf{x}$), while maintaining the domain consistency between the LR and HR images.

ments. Finally, we also participated in the AIM2020 Real Image Super-resolution Challenge [108] for the Track-3 ($\times$4 upscaling) associated with the ECCV 2020 workshops. Table 4.5 shows the final testset SR results for the track-3 of our method (**MLP_SR**) with others as well as the visual comparison in the Fig. 4.10 and Fig. 4.11.

4.2.1 Proposed SR Learning Approach

The proposed Real Image SR approach setup is shown in the Fig. 4.8. The SR generator network $\mathbf{G}_{SR}$ borrowed from the SRResCGAN [82] is trained in a GAN [36] framework by using the LR ($\mathbf{y}$) images with their corresponding HR images with pixel-wise supervision in the clean HR target domain ($\mathbf{x}$), while maintaining the domain consistency between the LR and HR images. In the next coming sections 4.2.1.1, 4.2.1.2, and 4.2.1.3, we present the details of the network architectures, network losses, and training descriptions for the proposed SR setup.

4.2.1.1 Network Architectures

SR Generator ($\mathbf{G_{SR}}$): We use the SR generator $\mathbf{G_{SR}}$ network which is basically an *Encoder-Resnet-Decoder* like structure as done SRResCGAN [82]. In the $\mathbf{G_{SR}}$ network, both *Encoder* and *Decoder* layers have 64 convolutional feature maps of 5×5 kernel size with $C \times H \times W$ tensors, where C is the number of channels of the input image. Inside the *Encoder*, LR image is upsampled by the Bicubic kernel with *Upsample* layer, where the choice of the upsampling kernel is arbitrary. *Resnet* consists of 5 residual blocks with two Pre-activation *Conv* layers, each of 64 feature maps with kernel support 3×3, and the pre-activation is the parametrized rectified linear unit (PReLU) with 64 output feature channels. The trainable projection layer [64] inside the *Decoder* computes the proximal map with the estimated noise standard deviation σ and handles the data fidelity and prior terms. The noise realization is estimated in the intermediate *Resnet* that is sandwiched between *Encoder* and *Decoder*. The estimated residual image after *Decoder* is subtracted from the LR input image. Finally, the clipping layer incorporates our prior knowledge about the valid range of image intensities and enforces the pixel values of the reconstructed image to lie in the range $[0, 255]$. The reflection padding is also used before all the *Conv* layers to ensure slowly varying changes at the boundaries of the input images.

SR Discriminator ($\mathbf{D_x}$): The SR discriminator network is trained to discriminate the real HR images from the fake HR images generated by the $\mathbf{G_{SR}}$. The raw discriminator network contains 10 convolutional layers with kernel support 3×3 and 4×4 of increasing feature maps from 64 to 512 followed by Batch Normalization (BN) and leaky ReLU as done in SRGAN [61].

LR Generator ($\mathbf{G_{LR}}$): We adapt the similar architecture as done in [113] for the down-sampling which is basically a *Conv-Resnet-Conv* like structure. We use 6 residual blocks in the *Resnet* with 3 convolutional layers at the head and tail *Conv*, while the stride is set to 2 in the second and third head *Conv* layers for the down-sampling

purpose.

LR Discriminator ($\mathbf{D_y}$): The LR discriminator network consists of a three-layer convolutional network that operates on the patch level as done in PatchGAN [46, 66]. All the *Conv* layers have 5×5 kernel support with feature maps from 64 to 256 and also applied the Batch Normalization (BN) and Leaky ReLU (LReLU) activation after each *Conv* layer except the last *Conv* layer that maps 256 to 1 features.

4.2.1.2 Network Losses

To learn the image super-resolution, we train the proposed SRResCycGAN network with the following loss functions:

$$\mathcal{L}_{G_{SR}} = \mathcal{L}_{\text{per}} + \mathcal{L}_{\text{GAN}} + \mathcal{L}_{tv} + 10 \cdot \mathcal{L}_1 + 10 \cdot \mathcal{L}_{\text{cyc}} \tag{4.21}$$

where these losses are defined as follows:

Perceptual loss ($\mathcal{L}_{\text{per}}$): It focuses on the perceptual quality of the output image and is defined as:

$$\mathcal{L}_{\text{per}} = \frac{1}{N} \sum_{i}^{N} \mathcal{L}_{\text{VGG}} = \frac{1}{N} \sum_{i}^{N} \|\phi(\mathbf{G}_{SR}(\mathbf{y}_i)) - \phi(\mathbf{x}_i)\|_1 \tag{4.22}$$

where, ϕ is the feature extracted from the pretrained VGG-19 network at the same depth as ESRGAN [106].

Texture loss ($\mathcal{L}_{\text{GAN}}$): It focuses on the high frequencies of the output image and it is defined as:

$$\mathbf{D}_{\mathbf{x}}(\mathbf{x}, \hat{\mathbf{y}})(C) = \sigma(C(\mathbf{x}) - \mathbb{E}[C(\hat{\mathbf{y}})]) \tag{4.23}$$

Here, C is the raw discriminator output and σ is the sigmoid function. By using the relativistic discriminator [106], we have:

$$\begin{aligned}\mathcal{L}_{\text{GAN}} = \mathcal{L}_{\text{RaGAN}} = &- \mathbb{E}_{\mathbf{x}} \left[\log \left(1 - \mathbf{D}_{\mathbf{x}}(\mathbf{x}, \mathbf{G}_{SR}(\mathbf{y}))\right)\right] \\ &- \mathbb{E}_{\hat{\mathbf{y}}} \left[\log \left(\mathbf{D}_{\mathbf{x}}(\mathbf{G}_{SR}(\mathbf{y}), \mathbf{x})\right)\right]\end{aligned} \tag{4.24}$$

where, $\mathbb{E}_{\mathbf{x}}$ and $\mathbb{E}_{\hat{\mathbf{y}}}$ represent the operations of taking average for all real ($\mathbf{x}$) and fake ($\hat{\mathbf{y}}$) data in the mini-batches respectively.

Content loss ($\mathcal{L}_1$): It is defined as:

$$\mathcal{L}_1 = \frac{1}{N} \sum_{i}^{N} \|\mathbf{G}_{SR}(\mathbf{y}_i) - \mathbf{x}_i\|_1 \tag{4.25}$$

where, N represents the size of mini-batch.

TV (total-variation) loss ($\mathcal{L}_{tv}$): It focuses to minimize the gradient discrepancy and produces sharpness in the output SR image, and it is defined as:

$$\mathcal{L}_{tv} = \frac{1}{N} \sum_{i}^{N} \left(\|\nabla_h \mathbf{G}_{SR}\left(\mathbf{y}_i\right) - \nabla_h\left(\mathbf{x}_i\right)\|_1 + \|\nabla_v \mathbf{G}_{SR}\left(\mathbf{y}_i\right) - \nabla_v\left(\mathbf{x}_i\right)\|_1\right) \tag{4.26}$$

Here, ∇_h and ∇_v denote the horizontal and vertical gradients of the images.

Cyclic loss ($\mathcal{L}_{\text{cyc}}$): It focuses to maintain the cyclic consistency between LR and HR domain and it is defined as:

$$\mathcal{L}_{cyc} = \frac{1}{N} \sum_{i}^{N} \|\mathbf{G}_{LR}(\mathbf{G}_{SR}(\mathbf{y}_i)) - \mathbf{y}_i\|_1 \tag{4.27}$$

SSIM loss ($\mathcal{L}_{\text{ssim}}$): It incorporates the structure similarity [125] of the output image and is defined as:

$$\mathcal{L}_{ssim} = 1 - \frac{1}{N} \sum_{i}^{N} \mathbf{SSIM}(\hat{\mathbf{y}}_i, \mathbf{x}_i) \tag{4.28}$$

MSSSIM loss ($\mathcal{L}_{\text{msssim}}$): It incorporates the variations of image resolution and viewing conditions, and is defined as:

$$\mathcal{L}_{msssim} = 1 - \frac{1}{N}\sum_{i}^{N} \mathbf{MSSSIM}(\hat{\mathbf{y}}_i, \mathbf{x}_i) \tag{4.29}$$

where, MSSSIM is the Multi-Scale SSIM [125] loss.

4.2.1.3 Training description

In the training phase, we set the input LR patch size as 32×32 with their corresponding HR patches. We train the network in an end-to-end manner for 51000 training iterations with a batch size of 16 using Adam optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ without weight decay for generators ($\mathbf{G_{SR}}$ & $\mathbf{G_{LR}}$) and discriminators ($\mathbf{D_x}$ & $\mathbf{D_y}$) to minimize the loss in Eq. (4.21). The learning rate is initially set to 10^{-4} and then multiplies by 0.5 after 5K, 10K, 20K, and 30K iterations. The projection layer parameter σ is estimated according to [72] from the input LR image.

4.2.2 Experiments

4.2.2.1 Training data

We use the source domain data ($\tilde{\mathbf{y}}$: 2650 HR images) that are corrupted with two known degradation, e.g., sensor noise, compression artifacts as well as unknown degradation, and target domain data ($\mathbf{x}$: 800 clean HR images from the DIV2K [1]) provided in the NTIRE2020 Real-World Super-resolution (RWSR) Challenge [77] for the track-1. We use the source and target domain data for training the $\mathbf{G_{SR}}$ network under the different degradation scenarios. The LR data ($\mathbf{y}$) with similar corruption as in the source domain is generated from the down-sample GAN network (DSGAN) [31] with their corresponding HR target domain ($\mathbf{x}$) images. Furthermore, we use the training data (*i.e.*, $\mathbf{y}$: 19000 LR images, $\mathbf{x}$: 19000 HR images) provided in the AIM2020 Real

Table 4.4: The ×4 SR quantitative results comparison of our method with others over the DIV2K validation-set (100 images). Top section: SR results comparison with added sensor noise ($\sigma = 8$) and compression artifacts ($quality = 30$) in the validation-set. Middle section: SR results with the unknown corruptions (e.g., sensor noise, compression artifacts, etc.) in the validation-set provided in the RWSR challenge series [76, 77]. Bottom section: SR results with the real image corruptions in the validation-set and testset provided in the AIM 2020 Real Image SR challenge [108] for the track-3. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red and the second best performance is shown in blue.

SR methods	#Params	sensor noise ($\sigma = 8$)			compression artifacts ($q = 30$)		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
EDSR [69]	43.0M	24.48	0.53	0.6800	23.75	0.62	0.5400
ESRGAN [106]	16.7M	17.39	0.19	0.9400	22.43	0.58	0.5300
ESRGAN-FT [75]	16.7M	22.42	0.55	0.3645	22.80	0.57	0.3729
ESRGAN-FS [31]	16.7M	22.52	0.52	0.3300	20.39	0.50	0.4200
SRResCGAN [82]	380K	25.46	0.67	0.3604	23.34	0.59	0.4431
SRResCycGAN (ours)	380K	25.98	0.70	0.4167	23.96	0.63	0.4841
SRResCycGAN+ (ours)	380K	26.27	0.72	0.4542	24.05	0.64	0.5192
		unknown corruptions [77]					
SRResCGAN [82]	380K	25.05	0.67	0.3357			
SRResCycGAN (ours)	380K	26.13	0.71	0.3911			
SRResCycGAN+ (ours)	380K	26.39	0.73	0.4245			
		real image corruptions [108]					
SRResCycGAN (ours, valset)	380K	28.6239	0.8250	-			
SRResCycGAN (ours, testset)	380K	28.6185	0.8314	-			

Image SR Challenge [108] for the track-3 (×4 upscaling) for training the SRResCycGAN (refers to the section-4.2.2.4).

4.2.2.2 Technical details

We implemented our method in the Pytorch. The experiments are performed under Windows 10 with i7-8750H CPU with 16GB RAM and on the NVIDIA RTX-2070 GPU with 8GB memory. It takes about 25 hours to train the network. The run time per image (on the GPU) is 4.54 seconds for the AIM2020 Real Image SR testset. To further enhance the fidelity, we use a self-ensemble strategy [99] (denoted as SRResCycGAN+) at the test time, where the LR inputs are flipped/rotated and the SR results are aligned and averaged for enhanced prediction.

4.2.2.3 Comparison with the state-of-art methods

We compare our method with other state-of-art SR methods including EDSR [69], ESRGAN [106], ESRGAN-FT [75], ESRGAN-FS [31], and SRResCGAN [82], whose source

codes are available online. The two degradation settings (*i.e.*, sensor noise, JPEG compression) have been considered under the same experimental situations for all methods. We run all the original source codes and trained models by the default parameters settings for the comparison. The EDSR is trained without the perceptual loss (only $\mathcal{L}_1$) by a deep SR residual network using the bicubic supervision. The ESRGAN is trained with the $\mathcal{L}_{\text{perceptual}}$, $\mathcal{L}_{\text{GAN}}$, and $\mathcal{L}_1$ by a deep SR network using the bicubic supervision. The ESRGAN-FT and ESRGAN-FS apply the same SR architecture and perceptual losses as in the ESRGAN using the two known degradation supervisions. The SRResCGAN is trained with the similar losses combination as done in the ESRGAN using the two known degradation supervisions. We train the proposed SRResCycGAN with the similar losses combination as done in the ESRGAN and SRResCGAN with the additional cyclic loss by using the bicubic supervision.

We evaluate the trained model under the Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), and LPIPS [124] metrics, refer to the section 4.1.4.4 for more details of the evaluation metrics. Table 4.4 shows the quantitative results comparison of our method over the DIV2K validation-set (100 images) with two known degradation (*i.e.*, sensor noise, JPEG compression), the unknown degradation in the NTIRE2020 Real-World SR challenge series [77], and the validation-set and testset in the AIM2020 Real Image SR Challenge [108]. Our method results outperform in terms of PSNR and SSIM compared to the other methods, while in the case of LPIPS, we have comparable results to others. In the case of sensor noise ($\sigma = 8$) and JPEG compression ($q = 30$) in the top section of the Table 4.4, the ESRGAN has the worst performance in terms of the PSNR, SSIM, and LPIPS among all methods. It also depicts the visual quality in Fig. 4.9. The EDSR has better performance to the noisy input, but it produces more blurry results. These are due to the domain distribution difference by the bicubic down-sampling during the training phase. The ESRGAN-FT and ESRGAN-FS have much better performance due to overcoming the domain distribution shift prob-

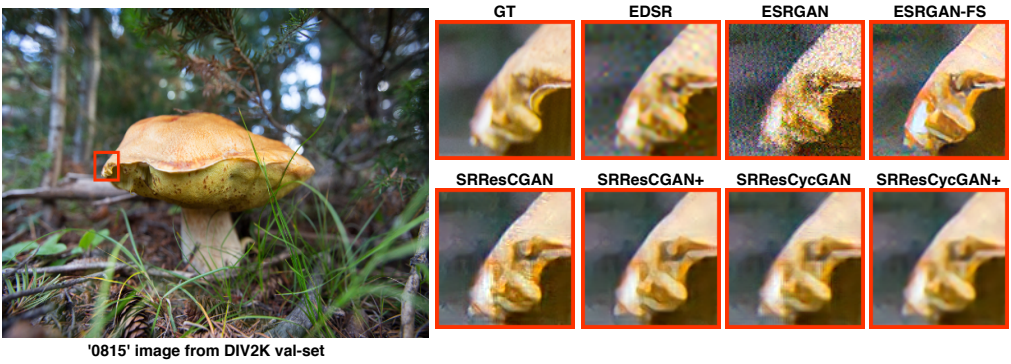


Figure 4.9: Visual comparison of our method with the other state-of-art methods on the DIV2K validation set at the $\times 4$ super-resolution.

lem, but they have still visible artifacts. The SRResCGAN has better robustness to the noisy input, but still has lower PSNR and SSIM due to lacking the domain consistency problem. The proposed method has successfully overcome the challenge of the domain distribution shift in both degradation settings, which depicts in both quantitative and qualitative results. In the middle section of the Table 4.4, for the unknown degradation in the NTIRE2020 Real-World SR challenge [77], the SRResCycGAN has much better PSNR/SSIM improvement, while the LPIPS is also comparable with the SRResCGAN. In the bottom section of the Table 4.4, we also report the validation-set and testset SR results in the AIM2020 Real Image SR Challenge [108] for the track-3. Despite that, the parameters of the proposed $\mathbf{G}_{SR}$ network are much less, which makes it suitable for deployment in mobile/embedded devices where memory storage and CPU power are limited as well as good image reconstruction quality.

Regarding the visual quality, Fig. 4.9 shows the qualitative comparison of our method with other SR methods at the $\times 4$ upscaling factor on the validation-set [77]. In contrast to the existing state-of-art methods, our proposed method produces excellent SR results that are reflected in the PSNR/SSIM values, as well as the visual quality of the reconstructed images with almost no visible corruptions.

Table 4.5: Final Testset results for the Real Image SR (×4) challenge Track-3 [108]. The table contains ours (**MLP_SR**) with other methods that are ranked in the challenge. The participating methods are ranked according to their weighted Score of the PSNR and SSIM given in the AIM 2020 Real Image SR Challenge [108].

Team Name	PSNR↑	SSIM↑	Weighed_score↑
Baidu	31.3960	0.8751	$0.7099_{(1)}$
ALONG	31.2369	0.8742	$0.7076_{(2)}$
CETC-CSKT	31.1226	0.8744	$0.7066_{(3)}$
SR-IM	31.1735	0.8728	0.7057
DeepBlueAI	30.9638	0.8737	0.7044
JNSR	30.9988	0.8722	0.7035
OPPO_CAMERA	30.8603	0.8736	0.7033
Kailos	30.8659	0.8734	0.7031
SR_DL	30.6045	0.8660	0.6944
Noah_TerminalVision	30.5870	0.8662	0.6944
Webbzhou	30.4174	0.8673	0.6936
TeamInception	30.3465	0.8681	0.6935
IyI	30.3191	0.8655	0.6911
MCML-Yonsei	30.4201	0.8637	0.6906
MoonCloud	30.2827	0.8644	0.6898
qwq	29.5878	0.8547	0.6748
SrDance	29.5952	0.8523	0.6729
MLP_SR	28.6185	0.8314	0.6457
RRDN_IITKGP	27.9708	0.8085	0.6201
congxiaofeng	26.3915	0.8258	0.6187

4.2.2.4 The AIM 2020 Real Image SR Challenge

We participated in the AIM2020 Real Image Super-Resolution Challenge [108] for the track-3 (×4 upscaling) associated with the ECCV 2020 workshops. The goal of this challenge is to learn a generic model to super-resolve LR images captured in practical scenarios with more complex degradation than bicubic downsampling. In that regard, we propose the SRResCycGAN to super-resolve the LR images with the real-world settings. We use the pretrained model $\mathbf{G_{SR}}$ taken from the SRResCGAN [82] (excellent perceptual quality) and further fine-tune it on the training data provided in the AIM 2020 Real Image SR challenge with the proposed SR scheme as shown in the Fig. 4.8 by using the following training losses:

$$\mathcal{L}_{G_{SR}} = \mathcal{L}_{\text{GAN}} + \mathcal{L}_{tv} + 10 \cdot \mathcal{L}_1 + \mathcal{L}_{ssim} + \mathcal{L}_{msssim} + 10 \cdot \mathcal{L}_{\text{cyc}} \tag{4.30}$$

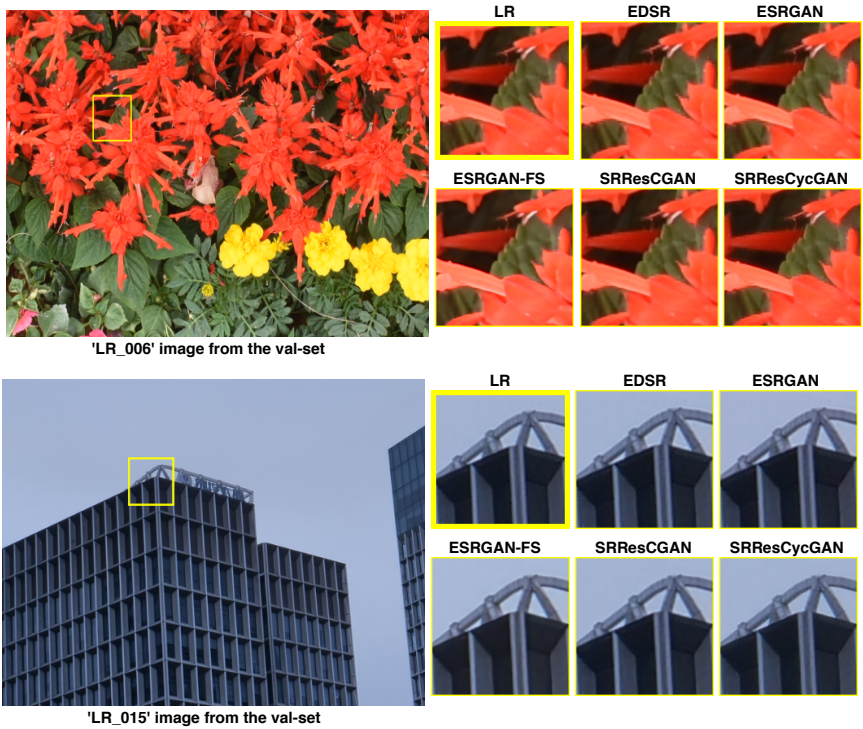


Figure 4.10: Visual comparison of our method with the other state-of-art methods on the AIM 2020 Real Image SR (track-3) validation set at the $\times 4$ super-resolution upscaling factor.

Since the final ranking is based on the weighted score of the PSNR and SSIM given in this challenge, we adopt the above losses combination where we neglect the $\mathcal{L}_{\text{per}}$ and use the $\mathcal{L}_{ssim}$ and $\mathcal{L}_{msssim}$, defined in the section 4.2.1.2, whose incorporate the structure similarity [107] as well as the variations of image resolution and viewing conditions for the output image. Table 4.5 provides the final $\times 4$ SR testset results for the track-3 of our method (**MLP_SR**) with others participants. Our proposed method (**MLP_SR**) is ranked in the top 20 solutions among other participants.

We also provide the visual comparison of our method with the state-of-art methods on the track-3 validation-set and testset in the Fig. 4.10 and Fig. 4.11. Our method produces sharp images without any visible corruptions and achieves comparable visual results with the other methods.

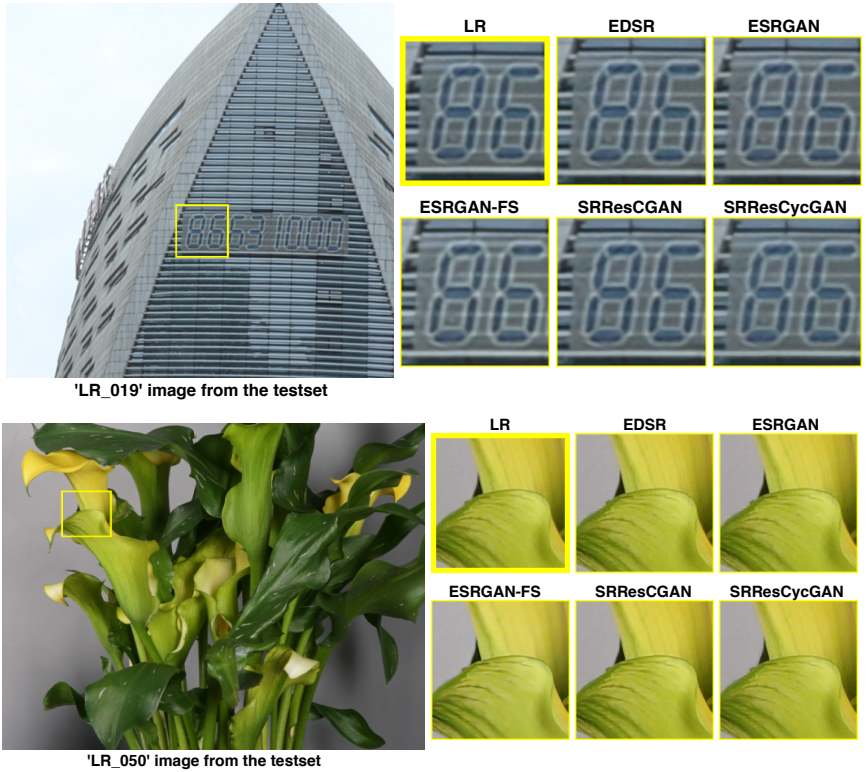


Figure 4.11: Visual comparison of our method with the other state-of-art methods on the AIM 2020 Real Image SR (track-3) test set at the ×4 super-resolution upscaling factor.

Table 4.6: This table reports the quantitative results of our method over the DIV2K validation set (100 images) with unknown degradation for our ablation study. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red.

SR method	Cyclic Path	Network structure	PSNR↑	SSIM↑	LPIPS↓
SRResCycGAN	✗	$\mathbf{y} \rightarrow \mathbf{G_{SR}} \rightarrow \hat{\mathbf{y}}$	25.05	0.67	0.3357
SRResCycGAN	✓	$\mathbf{y} \rightarrow \mathbf{G_{SR}} \rightarrow \hat{\mathbf{y}} \rightarrow \mathbf{G_{LR}} \rightarrow \mathbf{y}'$	26.13	0.71	0.3911
SRResCycGAN+	✓	$\mathbf{y} \rightarrow \mathbf{G_{SR}} \rightarrow \hat{\mathbf{y}} \rightarrow \mathbf{G_{LR}} \rightarrow \mathbf{y}'$	26.39	0.73	0.4245

4.2.2.5 Ablation Study

For our ablation study, we design two variants of the proposed network structure with cyclic path or not. The first network structure (*i.e.*, $\mathbf{y} \rightarrow \mathbf{G_{SR}} \rightarrow \hat{\mathbf{y}}$) takes the LR input to the $\mathbf{G_{SR}}$ and produces the SR output by the supervision of the SR discriminator network $\mathbf{D_x}$ without the cyclic path ($\mathbf{G_{LR}}$ & $\mathbf{D_y}$) as shown in the Fig. 4.8. Corre-

spondingly, we minimize the total loss in the Eq. (4.21) without the $\mathcal{L}_{\text{cyc}}$. The second network structure (*i.e.*, $\mathbf{y} \rightarrow \mathbf{G_{SR}} \rightarrow \hat{\mathbf{y}} \rightarrow \mathbf{G_{LR}} \rightarrow \mathbf{y}'$) takes the LR input to the $\mathbf{G_{SR}}$ and produces the SR output by the supervision of the SR discriminator network $\mathbf{D_x}$. After that, the SR output fed into the $\mathbf{G_{LR}}$ and reconstructs the LR output by the supervision of the LR discriminator network $\mathbf{D_y}$, refers to the Fig. 4.8. Accordingly, we minimize the total loss in the Eq. (4.21). Table 4.6 shows the quantitative results of our method over the DIV2K validation-set [77] with the unknown degradation. We found that in the presence of the cyclic path, we get the significant improvement of the PSNR/SSIM *i.e.*, $+1.34/+0.06$ for the first variant. It suggests that the cyclic structure gives the benefits to handle complex degradation such as noise, blurring, compression artifacts, etc., while the other structure lacks this due to the domain difference between LR and HR.

4.3 Real Image Super-Resolution using GAN with deep adaptive Sinusoidal Non-linearities

Most of the state-of-the-art SISR methods have attained excellent performance by using deep residual neural networks with ReLU / PReLU activation functions. The typical bicubic downscaling process imposes great challenges to restore the high resolution image details from the downscaled LR images due to the significant loss of high-frequency information. Moreover, the current deep network architectures with ReLU/PReLU activation functions are incapable of modeling the underlying signals with fine details for image restoration tasks, thus reducing the restoring performance of the current SISR methods in real-world settings. To address these issues, we propose the GAN-based SRResCSinGAN approach with learnable adaptive sinusoidal nonlinearities, whose parameters are optimized during the network training. The proposed scheme solves the problem by modeling the LR and HR processes with sinusoidal activations that robustly

fit the complicated signals such as natural images. We train the generalized SR model in a GAN framework by synthesizing the more realistic LR/HR data instead of the traditional bicubic or existing deep learning based downsampling method. Our method achieves better SR results in terms of PSNR/SSIM values and comparable LPIPS values as well as visual quality compared to the existing state-of-art methods.

The SR methods [51, 69, 120, 122, 113, 68, 123, 102, 83] mostly rely on the PSNR-based metric by optimizing the $\mathcal{L}_1/\mathcal{L}_2$ losses with blurry results in a supervised way, while they do not preserve the visual quality with respect to human perception. For the perception SR task, the GAN-based SR methods [61, 106, 75, 31, 81, 84] have been proposed to solve the real-world SR problem. However, the above methods still suffer unpleasant artifacts and under performed often fail to produce convincing SR results. Moreover, in the past decades, numerous works have been investigated, with a variety of possible activation functions, such as sigmoid, ReLU, Tanh, PReLU, RBF, and many more. The preferred choice that has emerged over the years is the ReLU activation unit due to promoting sparsity of the feature maps and the faster training of very deep networks. The continuous and piecewise linear functions have proven as a universal approximation of complex signals such as natural images. Recent works have demonstrated the potential to robustly outperform ReLU by using alternative activation functions for image reconstruction / restoration tasks, such as deep spline activations [104] and periodic nonlinearities like sinusoidal [93]. Motivated by the continuous and differentiable periodic nonlinearities (sinusoidal) that are capable of representing fine details in the signals better than ReLU, we explored deep residual networks with sinusoidal nonlinearities.

The scheme of the proposed SISR approach setup is shown in the Fig. 4.12. Due to the unavailability of realistic paired (LR/HR) data, we train firstly the LR network ($\mathbf{G}_{LR}$) to generate LR images with the same degradation/corruption as in the source domain ($\mathbf{x}$). We aim to learn the realistic data distribution mapping from bicubically

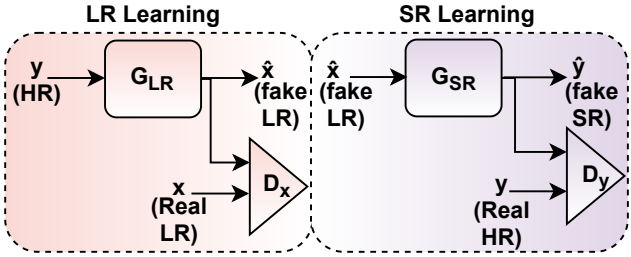


Figure 4.12: The structure of our proposed SR approach setup. In the LR Learning part, we learn the degradation/corruptions in the source domain data ($\mathbf{x}$) by the network $\mathbf{G}_{LR}$ in a GAN framework, where our goal is to map images from clean domain $\mathbf{y}$ to corrupted domain $\mathbf{x}$, while preserving the input image content. In the SR Learning part, we trained the network $\mathbf{G}_{SR}$ in a GAN framework by using generated LR ($\hat{\mathbf{x}}$) images from the $\mathbf{G}_{LR}$ network with their corresponding HR images to super resolve the LR images.

down-sampled images ($\mathbf{z}$) of HR images to the source domain images, while preserving the input image content. In the second part, the SR network ($\mathbf{G}_{SR}$) is trained in a GAN framework [36] by using generated LR ($\hat{\mathbf{x}}$) images from the $\mathbf{G}_{LR}$ network with their corresponding HR images to super-resolve the LR images.

We evaluate our proposed SR method on the Real-World Super-resolution (RWSR) dataset [77] to show the effectiveness of our approach through quantitative and qualitative experiments. In this work, we extend our work SRResCGAN [81] to significantly improve the reconstruction quality in terms of the PSNR and SSIM. In details, the extensions are:

1. We propose an end-to-end deep SRResCSinGAN for real image super-resolution. In contrast to [81] and the existing deep SISR networks, our generated LR/HR data pairs, instead of using traditional bicubic downsampling or an existing deep downsampling network like DSGAN, achieve better reconstruction results in terms of PSNR/SSIM/LPIPS (refer to Tables 4.7 and 4.8).

2. A modified version of the network (*i.e.*, both LR and SR learning models) that better models the underlying signals in the deep Resnet exploiting the sine non-linearities instead of the ReLU/PReLU activations.

4.3.1 Proposed Methodology

4.3.1.1 ResNet with Sinusoidal Non-linearities

We can describe the overall explicit compositional structure of the L-layer deep residual network (*ResNet*) with the following formulation:

$$\begin{aligned}\mathbf{f}_{ResNet}(\mathbf{x}) = ((f_L \circ \sigma_L \circ f_{L-1})(\mathbf{x}_{L-1}) + \mathbf{x}_{L-1}) \circ \cdots \circ \\ ((f_2 \circ \sigma_1 \circ f_1)(\mathbf{x}) + \mathbf{x}),\end{aligned} \tag{4.31}$$

Here, f is the affine transformation (*i.e.*, *Conv* layer) defined by the weight matrix $\mathbf{W}$ and the biases $\mathbf{b}$ applied to the input as:

$$f(\mathbf{x}) = \mathbf{W} * \mathbf{x} + \mathbf{b} \tag{4.32}$$

And followed by the sine nonlinearity [93] σ applied to the resulting vector f as:

$$\sigma(f) = sin(\omega_0 . \mathbf{W} f) \tag{4.33}$$

Where ω_0 is the scalar frequency factor, which is a hyperparameter. The derivative of the sine is a cosine (*i.e.*, the phase-shifted sine) for the backpropagation. The weights of the *Sine* layer are updated during the training via the stochastic gradient descent steps by minimization of the loss function. To initialize the weights ($\mathbf{W}$) of the *Sine* layer, we use the same initialization technique proposed in [93], where we draw the weights with $\mathbf{W}_i \sim \mathcal{U}(-\sqrt{6/n}, \sqrt{6/n})$ which ensures that the input to each sine activation is normal distributed with a unit standard deviation.

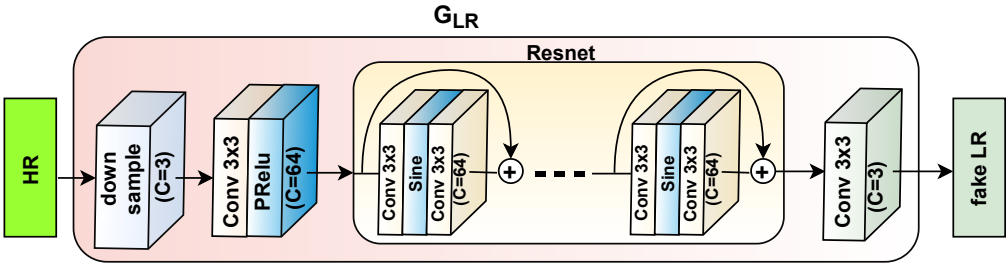


Figure 4.13: The structure of proposed LR learning architecture. The *Sine* denotes sinusoidal activation layer with 64 output feature maps.

4.3.1.2 LR Learning Model

Network Architectures: By referencing to the section 4.3.1.1, we replace the PReLU activations with the sine nonlinearities in the *Resnet* structure. The modified LR generator network ($\mathbf{G}_{LR}$) (see Fig. 4.13) consists of 8 *Resnet* blocks (two *Conv* layers and *Sine* nonlinearities layer in between them) that are sandwiched between two *Conv* layers. All *Conv* layers have 3×3 kernel support with 64 feature maps. Finally, *sigmoid* nonlinearity is applied on the output of the $\mathbf{G}_d$ network. While, the discriminator network ($\mathbf{D}_\mathbf{x}$) consists of a three-layer convolutional network that operates on a patch level [46, 66]. All *Conv* layers have 5×5 kernel support with feature maps from 64 to 256 and also applied Batch Normalization (BN) and Leaky ReLU (LReLU) activations after each *Conv* layer except the last *Conv* layer that maps 256 to 1 features.

Network Losses: To learn the degradation/corruptions from the source domain ($\mathbf{x}$), we train the modified network $\mathbf{G}_{LR}$ in a GAN framework [36] as done in DSGAN [31] with the following loss functions:

$$\mathcal{L}_{\mathbf{G}_d} = \mathcal{L}_{color} + 0.005 \cdot \mathcal{L}_{tex} + 0.01 \cdot \mathcal{L}_{per} \tag{4.34}$$

where, $\mathcal{L}_{color}, \mathcal{L}_{tex}, \mathcal{L}_{per}$ denote the color loss (*i.e.*, $\mathcal{L}_1$ loss focuses on the low frequency of the image), texture/GAN loss (*i.e.*, focus on the high frequencies of the image), and perceptual (VGG-based) loss, respectively.

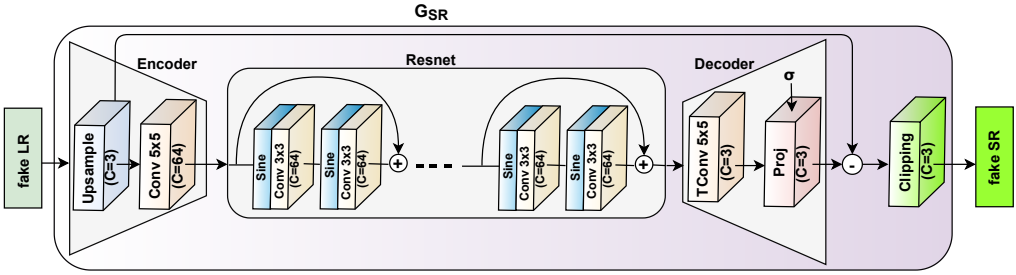


Figure 4.14: The structure of proposed SR learning architecture. The *Sine* denotes sinusoidal activation layer with 64 output feature maps.

Training description: We train the $\mathbf{G}_{LR}$ network with image patches 512×512, which are bicubically downsampled with MATLAB *imresize* function. We randomly crop the source domain images ($\mathbf{x}$) by 128×128 as done in [31]. We set the $\omega_0 = 30$ for the *Sine* layer. We train the network for 300 epochs with a batch size of 16 using Adam optimizer [52] with parameters $\beta_1 = 0.5$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ without weight decay for both generator and discriminator to minimize the loss in (4.34). The learning rate is initially set to 2.10^{-4} for the first 150 epochs and then linearly decay to zero after the remaining (*i.e.*, 150) epochs as done in [31].

4.3.1.3 SR Learning Model

Network Architectures: We use the SR generator $\mathbf{G_{SR}}$ network (see Fig. 4.14) which is an *Encoder-Resnet-Decoder* like structure as does SRResCGAN [81] with the modified *Resnet* structure by introducing the sine nonlinearities (refer to section 4.3.1.1). In the $\mathbf{G_{SR}}$ network, both *Encoder* and *Decoder* layers have 64 convolutional feature maps of 5×5 kernel size with $C \times H \times W$ tensors, where C is the number of channels of the input image. Inside the *Encoder*, LR image is upsampled by the Bicubic kernel with *Upsample* layer, where the choice of the upsampling kernel is arbitrary. *Resnet* consists of 5 residual blocks with two Pre-activation *Conv* layers, each of 64 feature maps with kernel support 3×3, and the preactivation is the *Sine* nonlinearities layer with 64 output feature channels. The trainable projection layer [64] inside the *Decoder* computes the

proximal map with the estimated noise standard deviation σ and handles the data fidelity and prior terms. The noise realization is estimated in the intermediate *Resnet* that is sandwiched between *Encoder* and *Decoder*. The estimated residual image after *Decoder* is subtracted from the LR input image. Finally, the clipping layer incorporates our prior knowledge about the valid range of image intensities and enforces the pixel values of the reconstructed image to lie in the range $[0, 255]$. The reflection padding is also used before all *Conv* layers to ensure slowly varying changes at the boundaries of the input images.

The SR discriminator network ($\mathbf{D_y}$) is trained to discriminate the real HR images from the fake HR images generated by the $\mathbf{G_{SR}}$. The raw discriminator network contains 10 convolutional layers with kernels that support 3×3 and 4×4 of increasing feature maps from 64 to 512 followed by Batch Normalization (BN) and leaky ReLU as done in SRGAN [61].

Network Losses: To learn the image super-resolution for the target domain ($\mathbf{y}$), we train the modified network $\mathbf{G}_{SR}$ in a GAN framework as done in SRResCGAN [81] with the following loss function:

$$\mathcal{L}_{G_{SR}} = \mathcal{L}_{\text{per}} + \mathcal{L}_{\text{GAN}} + \mathcal{L}_{tv} + 10 \cdot \mathcal{L}_1 \tag{4.35}$$

where $\mathcal{L}_{per}, \mathcal{L}_{GAN}, \mathcal{L}_{tv}, \mathcal{L}_1$ denote the perceptual (VGG-based) loss, the texture/GAN loss (*i.e.*, focus on the high frequencies of the image), the total variation loss (focus to minimize the gradient discrepancy and produce sharpness), and the content loss (*i.e.*, $\mathcal{L}_1$ loss focus on the low frequencies of the image), respectively.

Training description: At the training time, we set the input LR patch size as 32×32. We train the network for 51000 training iterations with a batch size of 16 using Adam optimizer [52] with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ without weight decay for both generator and discriminator to minimize the loss in (4.35). We set the $\omega_0 = 30$

for the *Sine* layer. The learning rate is initially set to 10^{-4} and then multiplies by 0.5 after 5K, 10K, 20K, and 30K iterations. The projection layer parameter σ is estimated according to [72] from the input LR image. We initialize the projection layer parameter α on a log-scale values from $\alpha_{max} = 2$ to $\alpha_{min} = 1$ and then further fine-tune during the training via back-propagation.

4.3.2 Experimental Results

4.3.2.1 Training data preparation

We use the source domain data ($\mathbf{x}$: 2650 HR images) that are corrupted with unknown degradation, e.g., sensor noise, compression artifacts, etc., and target domain data ($\mathbf{y}$: 800 clean HR images), provided in the NTIRE-2020 Real-World Super-resolution (RWSR) Challenge for track-1 [77]. The source domain data contains synthetic visible corruptions that are similar to the induced corruptions by the current camera devices. We use the source and target domain data to train the $\mathbf{G}_{LR}$ network and learn the degradation/corruptions, while due to the unavailability of paired LR/HR data, we train the $\mathbf{G}_{SR}$ network (refer to section-4.3.1.3) with the generated LR data ($\hat{x}$) from the $\mathbf{G}_{LR}$ network (refer to section-4.3.1.2) with their corresponding HR target ($\mathbf{y}$) images.

4.3.2.2 Data augmentation

We take the input LR image patches as generated by the LR learning $\mathbf{G}_{LR}$ network (refer to section 4.3.1.2) with their corresponding HR image patches. We augment the training data with random vertical and horizontal flipping, and 90° rotations. Moreover, we also consider another effective data augmentation technique, called mixture of augmentation (MOA) [112] strategy. In the MOA, a data augmentation (DA) method, among *i.e.*, Blend, RGB permutation, Mixup, Cutout, Cutmix, Cutmixup, and CutBlur is randomly selected and then applied on the inputs. This MOA technique encourages

Table 4.7: $\times 4$ SR quantitative results comparison of our method over the DIV2K validation-set (100 images) with added two known degradations *i.e.*, sensor noise ($\sigma = 8$) and JPEG compression ($quality = 30$) artifacts. Bottom section: $\times 4$ SR results comparison with the unknown corruptions in the RWSR challenge series (validation-set) [77]. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red and the second best performance is shown in blue.

Dataset (LR/HR pairs)	SR methods	#Params	PSNR↑	SSIM↑	LPIPS↓
			sensor noise ($\sigma = 8$)		
Bicubic	EDSR [69]	$43M$	24.48	0.53	0.6800
Bicubic	ESRGAN [106]	$16.7M$	17.39	0.19	0.9400
CycleGAN [75]	ESRGAN-FT [75]	$16.7M$	22.42	0.55	0.3645
DSGAN [31]	ESRGAN-FS [31]	$16.7M$	22.52	0.52	0.3300
DSGAN [31]	SRResCGAN [81]	$380K$	25.46	0.67	0.3604
DSSinGAN (ours)	SRResCSinGAN (ours)	$380K$	25.50	0.69	0.3750
			JPEG compression (quality=30)		
Bicubic	EDSR [69]	$43M$	23.75	0.62	0.5400
Bicubic	ESRGAN [106]	$16.7M$	22.43	0.58	0.5300
CycleGAN [75]	ESRGAN-FT [75]	$16.7M$	22.80	0.57	0.3729
DSGAN [31]	ESRGAN-FS [31]	$16.7M$	20.39	0.50	0.4200
DSGAN [31]	SRResCGAN [81]	$380K$	23.34	0.59	0.4431
DSSinGAN (ours)	SRResCSinGAN (ours)	$380K$	23.70	0.63	0.4258
			unknown (validset) [77]		
DSGAN [31]	SRResCGAN [81]	$380K$	25.05	0.67	0.3357
DSSinGAN (ours)	SRResCSinGAN (ours)	$380K$	25.58	0.69	0.3610
DSSinGAN (ours)	SRResCSinGAN+ (ours)	$380K$	25.89	0.71	0.3769

the network to acquire more generalization power by partially blocking or corrupting the training sample.

4.3.2.3 Technical details

We implemented our method with Pytorch. The experiments are performed under Windows 10 with i7-8750H CPU with 16GB RAM and on NVIDIA RTX-2070 GPU with 8GB memory. It takes about 28.57 hours to train the SR model. The run time per image (on GPU) is 0.1659 seconds on the validation set. To further enhance the fidelity, we use a self-ensemble strategy [99] (denoted as SRResCSinGAN+) at the test time, where the LR inputs are flipped/rotated and the SR results are aligned and averaged for enhanced prediction.

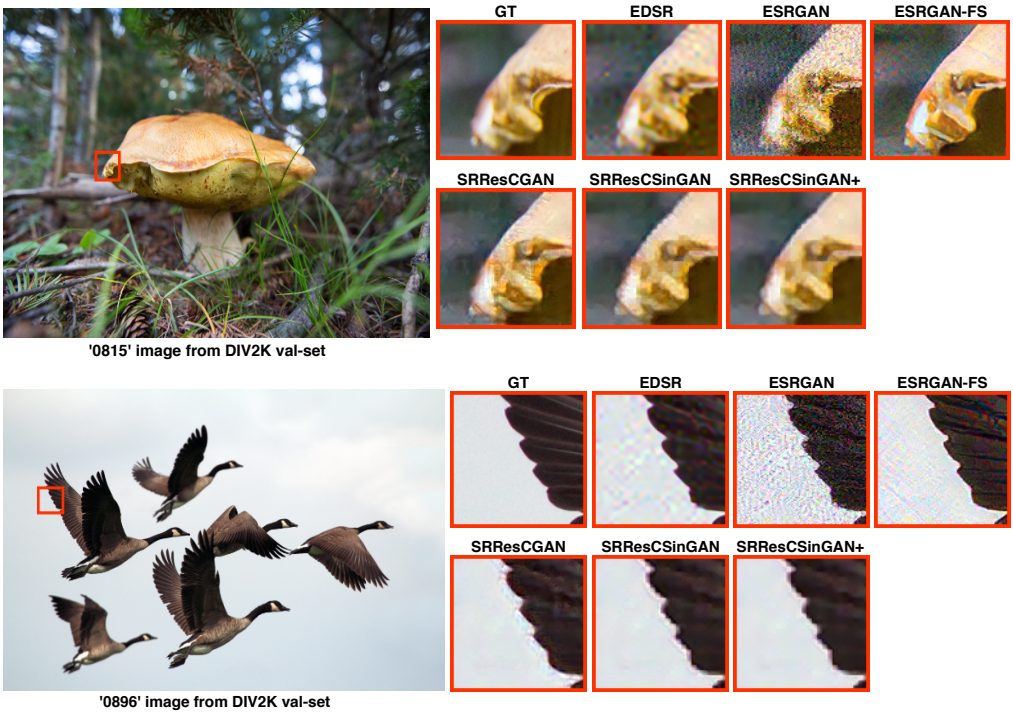


Figure 4.15: Visual SR comparison of our method with the other state-of-art methods on the DIV2K validation set at the $\times 4$ upscaling factor.

4.3.2.4 Comparison with state-of-the-art SR methods

We compare our method with other state-of-art SR methods including EDSR [69], ESRGAN [106], ESRGAN-FT [75], and ESRGAN-FS [31] and SRResCGAN [81], whose source codes are available online. The two degradation settings (*i.e.*, sensor noise, JPEG compression) have been considered under the same experimental situations for all methods. We run all original source codes and trained models by the default parameters settings for comparison. The EDSR is trained without perceptual loss (only $\mathcal{L}_1$) by a deep SR residual network using bicubic supervision. The ESRGAN is trained with the $\mathcal{L}_{\text{perceptual}}$, $\mathcal{L}_{\text{GAN}}$, and $\mathcal{L}_1$ by a deep SR network using bicubic supervision. The ESRGAN-FT and ESRGAN-FS apply the same SR architecture and perceptual losses as in the ESRGAN using the two known degradation supervisions. The SRResCGAN is

trained with the similar loss combination as done in the ESRGAN using the two known degradation supervisions. We train the proposed SRResCSinGAN with the similar loss combination as done in the ESRGAN and SRResCGAN with the modified Resnet structure by the sine nonlinearities.

We evaluate the trained model under the Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), and LPIPS [124] metrics, refer to the section 4.1.4.4 for more details of the evaluation metrics. Table 4.7 shows the quantitative results comparison of our method over the DIV2K validation-set (100 images) with two known degradations (*i.e.*, sensor noise, JPEG compression) as well as the unknown degradation in the RWSR challenge dataset [77]. In the case of sensor noise, our method has better PSNR/SSIM values compared to all existing SR methods, while we have comparable LPIPS value. Since these are the contradicted measures (PSNR/SSIM vs. LPIPS), our objective is to achieve a good PSNR/SSIM score, while getting the satisfactory LPIPS value. In the case of jpeg compression artifacts, our proposed method has better PSNR/SSIM values except the EDSR, which is slightly better PSNR, but low LPIPS value, and it has a very deep network with $43M$ parameters, while our model has only $380K$ parameters. Finally, in the case of unknown corruptions, our method has improved SR results in terms of PSNR and SSIM with the modified network of our previous work *i.e.*, SRResCGAN. Despite that, the parameters of the proposed $\mathbf{G}_{SR}$ network are much less than the other state-of-the-art SR models.

Regarding the visual quality, Fig. 4.15 shows the qualitative comparison of our method with the other SR methods on $\times 4$ upscaling factor (validation-set). In contrast to the existing state-of-art methods, our proposed method produces excellent SR results that are reflected in the PSNR/SSIM/LPIPS values, as well as the visual quality of the reconstructed images with almost no visible corruptions.

Table 4.8: The table reports the quantitative results of our method over the DIV2K validation set (100 images) with unknown degradation for our ablation study. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red.

Dataset (LR/HR pairs)	SR method	PSNR↑	SSIM↑	LPIPS↓
Bicubic	SRResCGAN	24.13	0.57	0.4853
Bicubic	SRResCSinGAN	24.78	0.62	0.4365
DSGAN	SRResCGAN	25.05	0.67	0.3357
DSGAN	SRResCSinGAN	25.53	0.69	0.3792
DSSinGAN	SRResCSinGAN	25.58	0.69	0.3610
DSSinGAN	SRResCSinGAN+	25.89	0.71	0.3769

4.3.2.5 Ablation Study

For our ablation study, we generated different LR/HR pair data to train the SR models. We reached the better PSNR/SSIM score, while achieving good LPIPS for its better visual correlation with human perception. Table 4.8 shows the quantitative results of our method over the DIV2K validation-set (100 images) with unknown degradation [77]. In the top section of the table, we trained the SRResCGAN [81] method with and without sine nonlinearities with the bicubic downsampled data (refer to section 4.3.1.3 for the SR learning training). The SRResCGAN with sine non-linearities (*i.e.*, SRResCSinGAN) has achieved better results in terms of PSNR, SSIM, and LPIPS. In the middle section, we generate the LR data from the DSGAN [31] as done in SRResCGAN [81] and then trained the two variants of our SR model on the generated LR/HR pairs. The SRResCSinGAN has better SR results in terms of PSNR and SSIM, while satisfactory LPIPS value compared to the SRResCGAN. In the bottom section, we generate the LR data from the DSGAN with sine nonlinearities (denoted as DSSinGAN, refer to section 4.3.1.2 for the LR learning) and then finally train our proposed SRResCSinGAN method with the generated LR/HR pairs. The SRResCSinGAN has better PSNR and LPIPS values, while the same SSIM value. To further enhance the performance, we used the self-ensemble strategy [99] at the test time, denoted as SRResCSinGAN+. It suggests that better generation of the LR images instead of the traditional bicubic downscaling gives the better performance gain and also incorporating the sinusoidal non-linearites

instead of ReLU/PReLU activation in the resnet structure gives the improvement in the reconstruction quality.

5

Multi-Image Super-Resolution

5.1 A Deep Residual Star Generative Adversarial Network for multi-domain Image SR

The existing SISR methods have performance drop due to a fixed degradation settings, *i.e.*, usually a bicubic downscaling of low-resolution (LR) image. However, in real-world settings, the LR degradation process is unknown which can be bicubic LR, bilinear LR, nearest-neighbor LR, or real LR. Therefore, the most SR methods are ineffective and inefficient in handling more than one degradation setting within a single network. To handle the multiple degradation, *i.e.*, refers to multi-domain image super-resolution, we propose a GAN-based SR^2 *GAN method, a novel and scalable approach that super-resolves the LR images for the multiple LR domains using only a single model. The proposed scheme is trained in a StarGAN like network topology with a single generator and discriminator networks. Our method achieves excellent SR results in terms of the PSNR/SSIM values compared to the existing SR methods.

By referencing to the the general degradation model (3.2), in the most of existing SR

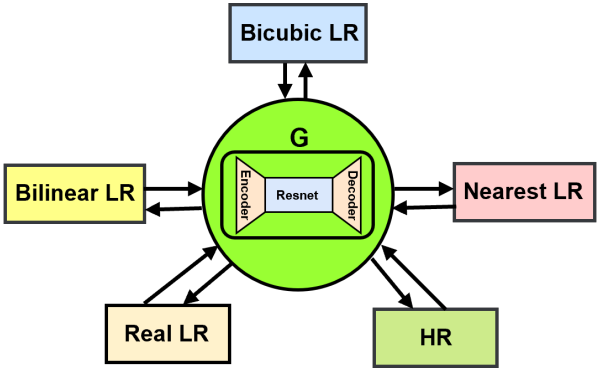


Figure 5.1: The Multi-domain SR proposed scheme, where a single generator **G** learns the mappings among multiple domains *i.e.*, LR/HR to LR/HR.

methods, the operator **H** is usually known / fixed *i.e.*, bicubic. However, in real-world settings, the operator **H** is unknown that can be bicubic, bilinear, nearest-neighbor, and real degradation kernel. In the recent years, numerous works have been addressed toward the task of SISR that are based on DCNNs for their powerful feature representation capabilities either on PSNR values [51, 69, 120, 122, 113, 68, 123] or on perceptual quality [61, 106, 75, 31, 81, 84]. The SR methods mostly rely on the single degradation (*i.e.*, usually bicubic) with paired LR and HR images in the supervised training. In the real-world settings, the input LR image contains more complex degradation. Since the bicubic, bilinear, and nearest-neighbor LR degradation are rarely suitable for the real LR images, these degradations can be used for data augmentation and are indeed a good choice for clean and sharp image super-resolution. Moreover, the *Real LR* domain in the proposed method contains the LR/HR image pairs that follow the realistic physical image model instead of artificial ones. Due to such different degradation settings, the most SR methods often fail to produce convincing SR results or train their model independently for every pair of image domains.

Instead of learning a fixed degradation setting (*e.g.*, bicubic LR), the proposed SR^2 *GAN takes as inputs both an image and a domain label, and learns to generate the image into the corresponding domain (LR/HR). The domain labels are encoded

in binary or one-hot vector to represent domain information. Fig. 5.1 illustrates an overview of our proposed SISR approach, where a single network **G** learns the mappings among the multiple domains *i.e.*, Bicubic LR, Bilinear LR, Nearest-Neighbor LR, Real LR, and HR. During the training phase, we translate the source domain images into the target domain images by randomly generate target domain labels. At the testing phase, we fix the target domain as HR domain to generate the SR images from any blind LR domain. The proposed scheme is inspired by the recent success of StarGAN [18] for multi-domain image-to-image translation applications. The proposed approach (refers to the section 5.1.1 for more details) allows us to simultaneously train for multiple domains in a unified network.

Our contributions in this section are as follows:

1. We propose SR^2 *GAN for multi-domain image super-resolution task. In contrast to the existing deep SISR methods, our method learns the mappings among multiple domains within a single model.

2. The proposed scheme requires a single generator and discriminator networks to train efficiently for the images of multiple domains.

5.1.1 Proposed Method

5.1.1.1 Multi-domain training strategy

Our goal is to train a single network **G** that super-resolves the LR images from multiple LR domains. To achieve this objective, we train the generator **G** to translate an input source image $\mathbf{y}$ into an output target image $\hat{\mathbf{y}}$ conditioned on the target domain labels ℓ as $\mathbf{G}(\mathbf{y}, \ell) \rightarrow \hat{\mathbf{y}}$. The domain labels ℓ are encoded in binary or one-hot vector to represent domain information. We randomly generated the target domain labels ℓ so that the network **G** flexibly translates the source domain images into the target domain images. The discriminator network **D** learns to distinguish between real/fake images and

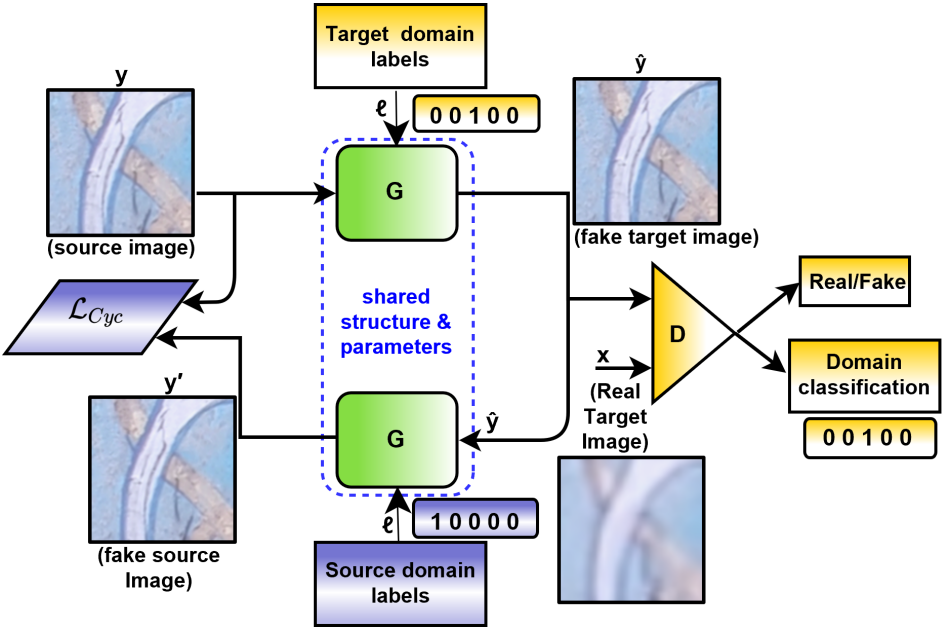


Figure 5.2: The structure of our proposed SR^2 *GAN setup. **G** takes an input as both the source image ($\mathbf{y}$) and target domain label and generates a fake target image ($\hat{\mathbf{y}}$). **G** tries to reconstruct the fake source image ($\mathbf{y}'$) from the fake target image ($\hat{\mathbf{y}}$) given the source domain label. **D** learns to discriminate between real and fake target images and classify the real target images to its corresponding domain. In this way, the **G** tries to generate fake target images indistinguishable from real target images and classifiable as target domain images by the **D**.

also tries to minimize the domain classification error only associated with the known domain label. The discriminator **D** (*i.e.*, adapted from StarGAN [18]) produces the probability distributions over both target images and target domain labels, $\mathbf{D} : \mathbf{x} \rightarrow \{\mathbf{D}_{trg}(\mathbf{x}), \mathbf{D}_{cls}(\mathbf{x})\}$. Fig. 5.2 illustrates the training strategy of the proposed SR^2 *GAN.

5.1.1.2 Network Architectures

Generator (G): We use the generator **G** network as a *Encoder-Resnet-Decoder* like structure, adapted from SRResCGAN [81] which strictly follows the degradation process (3.22). In the **G** network, both *Encoder* and *Decoder* layers have 64 convolutional feature maps of 5×5 kernel size with $C \times H \times W$ tensors, where C is the number of

channels of the input image. *Resnet* consists of 5 residual blocks with two Pre-activation *Conv* layers, each of 64 feature maps with kernel support 3×3, and the preactivation is the *Sine* nonlinearities layer with 64 output feature channels. The trainable projection layer [64] inside the *Decoder* computes the proximal map with the estimated noise standard deviation σ and handles the data fidelity and prior terms.

Discriminator (D): The discriminator network (**D**) learns to distinguish between real and fake images and classify the real images to its corresponding domain. The discriminator network contains 6 convolutional layers with kernels that support 4×4 of increasing feature maps from 64 to 2048 followed by the leaky ReLU as done in StarGAN [18].

5.1.1.3 Network Losses

To train the generator **G** and discriminator **D**, we use the following respective objective loss functions:

$$\mathcal{L}_G = \mathcal{L}_{\text{per}} + \mathcal{L}_{\text{GAN}} + \mathcal{L}_{tv} + \mathcal{L}_{cls}^{f} + 10 \cdot \mathcal{L}_1 + 10 \cdot \mathcal{L}_{\text{cyc}}, \tag{5.1}$$

$$\mathcal{L}_D = \mathcal{L}_{GAN} + \mathcal{L}_{cls}^{r}, \tag{5.2}$$

where $\mathcal{L}_{per}$, $\mathcal{L}_{GAN}$, $\mathcal{L}_{tv}$, $\mathcal{L}_{cls}^{r/f}$, $\mathcal{L}_1$, and $\mathcal{L}_{cyc}$ denote the perceptual (VGG-based) loss, texture/GAN loss, total variation loss, real/fake classification loss, content loss, and cyclic loss respectively.

5.1.2 Experiments

5.1.2.1 Training data

For the training, we use DIV2K [1], Flickr2K [97], and RealSR [109] datasets that jointly contain 22,430 high-quality HR images for image restoration tasks with rich and diverse textures. We obtain the LR bicubic, LR bilinear, and LR nearest images by downsampling HR images by the scaling factor $\times 4$ of the DIV2K and Flickr2K datasets using the Pytorch bicubic, bilinear, and nearest kernel function. The LR real images by the scaling factor $\times 4$ are provided with their corresponding HR images in the RealSR dataset. We consider the five domains as shown in Fig. 5.1. We obtain the source domain images from the three datasets with their corresponding domain labels. We randomly generate the target domain labels with their corresponding images from the three datasets. We train our network in RGB color space.

5.1.2.2 Data augmentation

We augment the training data with random vertical and horizontal flipping, and 90° rotations. Moreover, we also consider another effective data augmentation technique, called mixture of augmentation (MOA) [112] strategy. In the MOA, a data augmentation (DA) method, among *i.e.*, Blend, RGB permutation, Mixup, Cutout, Cutmix, Cutmixup, and CutBlur is randomly selected and then applied on the inputs. This MOA technique encourages the network to acquire more generalization power by partially blocking or corrupting the training sample.

5.1.2.3 Training details

At the training time, we set the input image patch size as 128×128. We train the network for 51000 training iterations with a batch size of 16 using Adam optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ without weight decay for both

Table 5.1: ×4 SR quantitative results comparison of our method with others over the DIV2K (100 images of validation-set) and RealSR (93 images of testset) that are total 393 images of the testset with the four LR degradation. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red and the second best performance is shown in blue.

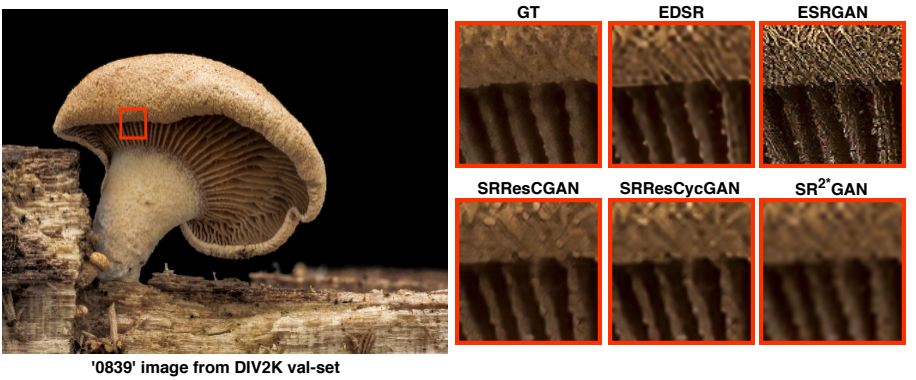
SR methods	#Params	Bicubic	Bilinear	Nearest	Real	Average
		PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓
EDSR [69]	43.0M	21.33 / 0.66 / 0.3477	23.05 / 0.72 / 0.3083	19.34 / 0.56 / 0.3653	28.06 / 0.82 / 0.4182	22.95 / 0.69 / 0.3599
ESRGAN [106]	16.7M	16.02 / 0.31 / 0.6008	17.25 / 0.37 / 0.5186	15.29 / 0.26 / 0.6254	27.98 / 0.82 / 0.3840	19.14 / 0.44 / 0.5322
SRResCGAN [81]	380K	23.30 / 0.67 / 0.2900	24.43 / 0.70 / 0.2720	21.10 / 0.60 / 0.3044	27.96 / 0.82 / 0.3676	24.20 / 0.69 / 0.3085
SRResCycGAN [84]	380K	24.56 / 0.73 / 0.3380	25.58 / 0.75 / 0.3183	21.75 / 0.63 / 0.3541	28.02 / 0.82 / 0.3827	24.98 / 0.73 / 0.3483
SR^2 *GAN (ours)	380K	25.52 / 0.75 / 0.4250	26.23 / 0.76 / 0.4083	22.75 / 0.66 / 0.4415	28.02 / 0.82 / 0.4353	25.63 / 0.75 / 0.4275

generator and discriminator to minimize the loss functions in the Eqs. (5.1) and (5.2). The learning rate is initially set to 10^{-4} and then multiplies by 0.5 after 5K, 10K, 20K, and 30K iterations. Our all experiments are performed with a scaling factor of ×4 for the LR and HR images.

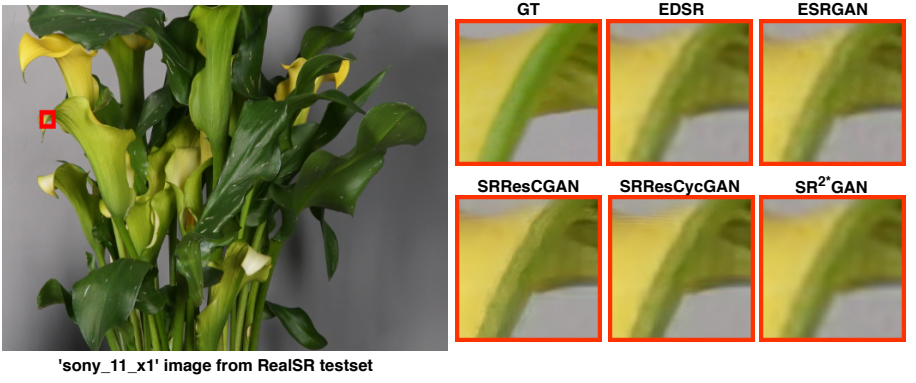
5.1.2.4 Comparison with the state-of-art methods

We compare our method with other state-of-art SISR methods including EDSR [69], ESRGAN [106], SRResCGAN [81], and SRResCycGAN [84]. We compare the SR results with one deep feed-forward residual network (*i.e.*, EDSR) and other three GAN-based approaches (*i.e.*, ESRGAN, SRResCGAN, SRResCycGAN). Although there are existing SR methods [122], [123], and [102] by handling multiple degradation settings, but they are the non-blind SR techniques with deep feed-forward networks, while our proposed method is a blind SR GAN-based approach. So, the comparison with [122], [123], and [102] methods is not fair. We run all the original source codes and trained models by the default parameters settings for the comparison. The competing methods are trained for just one degradation setting and tested on the different ones. Unlike them, the proposed approach can be trained with multiple degradations instead of single degradation.

We evaluate the trained model under the Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), and LPIPS [124] metrics, refer to the section 4.1.4.4 for more details of the evaluation metrics. Table 5.1 shows the quantitative results comparison of our method with the others over the DIV2K (100 images of validation-set) and RealSR (93 images of testset) that are total 393 images of the testset with the four LR degra-



(a) SR results on the LR bicubic



(b) SR results on the LR real

Figure 5.3: Visual comparison of our method with other state-of-art methods on $\times 4$ super-resolution.

dation. We have excellent results in terms of PSNR and SSIM compared to the other methods, while in the case of LPIPS, we lag by the others because our training objective is to get better PSNR/SSIM. The reason for none of the competing methods achieved their best performance against the bicubic down-sampling, is that the other GAN-based approaches focus more on the perceptual quality, as good perceptual image quality reduces the PSNR/SSIM scores. Our model is trained with reasonably perceptual quality, while good PSNR/SSIM, refers to section 5.1.2.5.

Regarding the visual quality, Fig. 5.3 shows the visual comparison of our method

Table 5.2: This table reports the quantitative SR results of our method over the DIV2K and RealSR validation-set (20 images, not used during the training phase) with the four LR domains (*i.e.*, Bicubic, Bilinear, Nearest, Real) for our ablation study. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red.

SR method	**Domain label Conditioning (ℓ)**	**PSNR↑**	**SSIM↑**	**LPIPS↓**
SR^2 *GAN (v1)	**G** Conditioned on only HR target domain & used two separate Disc. ($\mathbf{D}_{trg}$, $\mathbf{D}_{src}$)	26.95	0.75	0.3423
SR^2 *GAN (v2)	**G** Conditioned on only HR target domain & used one Disc. ($\mathbf{D}_{trg}$)	27.20	0.76	0.3176
SR^2 *GAN (v3)	**G** Conditioned on random target domains & used one Disc. ($\mathbf{D}_{trg}$)	28.09	0.78	0.3891

with other SR methods at the $\times 4$ upscaling factor on the test-set. We have comparable visual SR results with others.

5.1.2.5 Ablation Study

For our ablation study, we train three variants of the proposed SR^2 *GAN network structure with conditioning of the domain labels. By referencing to the Fig. 5.2, the first network variant v1 is trained by conditioning the first branch generator **G** on only HR target domain labels, while, the other branch generator **G** (shared weights) is conditioned on the multiple LR source domain labels. We use two separate discriminators *i.e.*, $\mathbf{D}_{trg}$ for the target domain supervision, $\mathbf{D}_{src}$ for the source domain supervision in the SR^2 *GAN (v1). Similarly, the second network variant v2 is trained as the v1, but only uses one discriminator *i.e.*, $\mathbf{D}_{trg}$ for the target domain supervision. Finally, the third network variant v3 is trained by conditioning both generators **G** on random domain labels and use only one discriminator $\mathbf{D}_{trg}$ for the real target domain supervision by minimizing the the total loss functions in the Eqs. (5.1) and (5.2). Table 5.2 shows the quantitative SR results of SR^2 *GAN (v1, v2, v3) over the DIV2K and RealSR validation-sets (20 images, not used during the training phase) with the four LR domains *i.e.*, Bicubic, Bilinear, Nearest, Real. The third network variant v3 performs better in terms of PSNR/SSIM among others. This shows the effectiveness of random target domain labeling and real target domain image supervision by the discriminator. Therefore, we opt for the third version v3 and used it for the evaluation.

5.2 Deep Iterative Convolutional Neural Networks for Raw Burst Super-Resolution

Modern digital cameras and smartphones mostly rely on image signal processing (ISP) pipelines to produce realistic colored RGB images. However, compared to DSLR cameras, low-quality images are usually obtained in many portable mobile devices with compact camera sensors due to their physical limitations. Since the low-quality images have multiple degradations *i.e.*, sub-pixel shift due to camera motion, mosaick patterns due to camera color filter array, low-resolution due to smaller camera sensors, and the rest are corrupted by the noise, the current Single Image Super-resolution (SISR) methods have limited performance to recover high-resolution (HR) image details from a single low-resolution (LR) image. Burst photography pipeline with multi-frame super-resolution (MFSR) is the most common way to deal with such a scenario that generates the HR image from a low-quality burst of raw sensor images. In this work, we propose a deep Burst Super-Resolution Iterative Convolutional Neural Network (BSRICNN) that follows the burst photography pipeline as a whole by a forward (physical) model. The proposed Burst SR scheme solves the problem with classical image regularization, convex optimization, and deep learning techniques, compared to existing black-box data-driven methods. The proposed network produces the final output by iterative refinement of the intermediate SR estimates. The proposed network exploits powerful image regularization, large-scale optimization, and deep learning techniques for multi-frame image restoration. Our model requires much less parameters and 2d convolution operation in comparison to other competing methods. Our method achieves good burst SR results in terms of the synthetic data as well as comparable visual quality of the real-world bursts with respect to the existing approaches.

There have been proposed numerous works towards the task of SISR [51, 69, 120,

122, 113, 68, 123, 102, 83] and the real SISR [61, 106, 75, 31, 81, 84, 103]. Due to the ill-posed nature of the SISR problem, the existing methods have limited performance to recover high frequency details through learned image priors. Besides the development of the SISR approaches based on image priors, the recent works have demonstrated the potential of MFSR methods that aim to fuse multiple LR views to reconstruct a HR output. Wronski *et al.* [111] developed a Super-Res Zoom method, which implements a burst processing pipeline that achieves MFSR by aligning, merging, and enhancing a sequence of raw frames to sub-pixel accuracy. Deudon *et al.* [21] proposed a MFSR network (*i.e.*, HighRes-net) by aligning each input frame to a reference frame implicitly, and then merges them using a recursive fusion method for satellite imagery. Bhat *et al.* [10] developed a DeepBurstSR network for Raw Burst super-resolution task. The DeepBurstSR aligns deep embeddings of the input LR frames using pixel-wise optical flow and then adaptively merges the information from all frames by using an attention-based fusion module. Moreover, there are other approaches [34, 56, 57] based on two stage structure which perform single-frame joint demosaicking and denoising, and after that, one can super-resolve the resulting RGB image using the existing SISR methods. However, the deep learning based Burst SR methods are black-box data-driven approaches with larger model size, while our proposed scheme has the merit of interpretability and small model size.

Mathematically, the Burst SR is described as the following forward observation model for the image degradation process:

$$\mathbf{y}_i = \mathbf{MHS}_i(\tilde{\mathbf{x}}) + \eta_i, \quad i = 1, \ldots, B \tag{5.3}$$

where, $\mathbf{y}_i$ is the observed LR burst images of total size B, $\mathbf{M}$ is a *mosaicking operator* that corresponds to the CFA (Color Filter Array) of a camera (usually Bayer), $\mathbf{H}$ is a *down-sampling operator* (*i.e.*, bilinear, bicubic, etc.) that resizes an HR image $\tilde{\mathbf{x}}$ by a scaling factor r, $\mathbf{S}_i$ is an *affine transformation* of the coordinate system of the

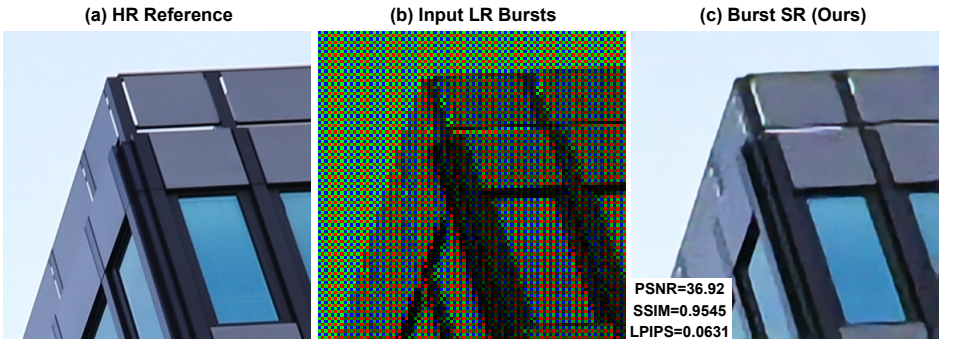


Figure 5.4: An image from the Zurich RAW to RGB Dataset [44] (testset), where we present (a) the ground truth HR reference image of size $384 \times 384 \times 3$, (b) the input LR bursts of size ($W \times H \times C \times B$) $48 \times 48 \times 4 \times 14$, and (c) the Burst SR output of size $384 \times 384 \times 3$ of our network (BSRICNN). All images are converted from raw sensor space to sRGB for visualization purpose.

image $\tilde{\mathbf{x}}$ (*i.e.*, translation and rotation), and η_i is an additive *heteroskedastic Gaussian noise* related to photon shot and read noise. Due to the ill-posed nature of an inverse problem (*i.e.*, refer to the operator $\mathbf{A}$ in section-5.2.1.1), Multi-frame Super-Resolution (MFSR) is common way to conditioning on multiple LR frames. The MFSR aims to recover the latent HR image using multiple LR frames by exploiting the additional signal information due to sub-pixel shifts, compared to the SISR methods, refer to Fig. 5.4.

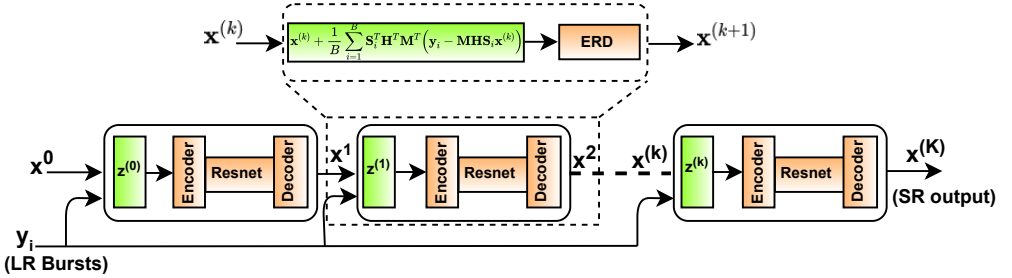


Figure 5.5: The structure of our proposed iterative Burst SR scheme. Given an LR burst images ($\mathbf{y}_i$), each network's stage produces a new estimate $\mathbf{x}^{(k+1)}$ from the previous step estimate $\mathbf{x}^{(k)}$. A single optimizer is used for all network stages with shared structures and parameters.

The proposed Burst SR scheme is shown in the Fig. 5.5. We unroll the proposed BSRICNN into K stages, where each stage computes the refined estimate of the solution.

$\mathbf{y}_i$ is an input Raw LR burst, $\mathbf{x}^0$ is an initial estimate, and $\mathbf{x}^K$ is the final estimated SR image. We learn the shared parameters across stages by jointly minimizing the loss $\mathcal{L}(\Theta)$ function with respect to all network parameters Θ.

We evaluate our proposed approach on synthetic and real burst SR datasets. We use the BurstSR dataset [10] to show the effectiveness of our method through quantitative and qualitative experiments. Finally, we also participated in the NTIRE2021 Burst SR challenges (track-1 and track-2) associated with the CVPR 2021 workshops. Table 5.4 and 5.5 show the preliminary testsets results of the track-1 and track-2 of our method (**MLP_BSR**) with other participants.

5.2.1 Proposed Method

5.2.1.1 Problem Formulation

To solve the Raw Burst Super-Resolution task (5.3), the recovery of $\mathbf{x}$ from $\mathbf{y}_i$ mostly relies on the variational approach for combining the observation and prior knowledge, and the solution is obtained by minimizing the following objective function as:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2 B} \sum_{i=1}^{B} \|\mathbf{y}_i - \mathbf{MHS}_i(\mathbf{x})\|_2^2 + \lambda \mathcal{R}(\mathbf{x}), \tag{5.4}$$

The Eq. (5.4) can be also written in the following form:

$$\mathbf{J}(\mathbf{x}) = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2 B} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \mathcal{R}(\mathbf{x}), \tag{5.5}$$

where, $\mathbf{A} = \mathbf{MHS}$, the first term is a data fidelity term that measures the proximity of the solution to the observations, the second term (*i.e.*, $\mathcal{R}(\mathbf{x})$) is the regularization term that is associated with image priors, and λ is the trade-off parameter that governs the compromise between the data fidelity and the regularizer term. In this work, we propose a deep iterative Burst SR learning method (BSRICNN, refers to Fig. 5.5) that solves

the Burst SR task in an iterative manner by using Majorization-Minimization (MM) framework [42] with K finite steps. In the next section 5.2.1.2, the proper optimization strategy is employed to find the solution that minimizes the objective function (5.5) to get the required latent HR image.

5.2.1.2 Majorization-Minimization (MM) Strategy

In an MM [42, 29, 65] approach, an iterative algorithm for solving the minimization problem

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \mathbf{J}(\mathbf{x}), \tag{5.6}$$

takes the form

$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}} \mathbf{Q}(\mathbf{x}; \mathbf{x}^{(k)}), \tag{5.7}$$

where $\mathbf{Q}(\mathbf{x}; \mathbf{x}^{(k)})$ is the majorizer of the function $\mathbf{J}(\mathbf{x})$ at a fixed point $\mathbf{x}^{(k)}$ by satisfying the following two conditions:

$$\mathbf{Q}(\mathbf{x}; \mathbf{x}^{(k)}) > \mathbf{J}(\mathbf{x}),\ \forall \mathbf{x} \neq \mathbf{x}^{(k)} \text{ and } \mathbf{Q}(\mathbf{x}^{(k)}; \mathbf{x}^{(k)}) = \mathbf{J}(\mathbf{x}^{(k)}). \tag{5.8}$$

Here, we want to upper-bound the $\mathbf{J}(\mathbf{x})$ by a suitable majorizer $\mathbf{Q}(\mathbf{x}; \mathbf{x}^{(k)})$, and instead of minimizing the actual objective function (5.5) due to its complexity, we minimize the majorizer $\mathbf{Q}(.)$ to produce the next estimate $\mathbf{x}^{(k+1)}$. By satisfying the properties of the majorizer given in (5.8), iteratively minimizing $\mathbf{Q}(.; \mathbf{x}^{(k)})$ also decreases the actual objective function $\mathbf{J}(.)$ [42]. Thus, we can write a quadratic majorizer for the complete objective function (5.5) as the following form:

$$\tilde{d}(\mathbf{x}; \mathbf{x}^{(k)}) = \frac{1}{2\sigma^2 B} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + g(\mathbf{x}, \mathbf{x}^{(k)}), \tag{5.9}$$

To start an estimate $\mathbf{x}^{(k)}$, we have:

$$g(\mathbf{x}, \mathbf{x}^{(k)}) = \frac{1}{2\sigma^2 B} \left(\mathbf{x} - \mathbf{x}^{(k)}\right)^T \left[\alpha \mathbf{I} - \mathbf{A}^T \mathbf{A}\right] \left(\mathbf{x} - \mathbf{x}^{(k)}\right), \tag{5.10}$$

where, $\tilde{d}(\cdot, \cdot)$ is a distance function between $\mathbf{x}$ and $\mathbf{x}^{(k)}$. In order to get a valid majorizer, we need to satisfy the two conditions in Eq. (5.8) as $g(\mathbf{x}, \mathbf{y}) > 0, \forall \mathbf{x} \neq \mathbf{y}$ and $g(\mathbf{x}, \mathbf{x}) = 0$. This suggests that $\alpha \mathbf{I} - \mathbf{A}^T \mathbf{A}$ must be a positive definite matrix, which only holds if $\alpha > \|\mathbf{A}^T \mathbf{A}\|_2 \Rightarrow \alpha \geq B$. Finally, we proceed with the MM optimization scheme to iteratively minimize the quadratic majorizer function $\mathbf{Q}(.)$ by the following formulation as:

$$\begin{aligned} \hat{\mathbf{x}}^{(k)} &= \arg\min_{\mathbf{x}} \mathbf{Q}(\mathbf{x}; \mathbf{x}^{(k)}) = \tilde{d}\left(\mathbf{x}; \mathbf{x}^{(k)}\right) + \lambda \mathcal{R}(\mathbf{x}) \\ &= \frac{\alpha}{2\sigma^2 B} \|\mathbf{x} - \mathbf{z}^k\|_2^2 + \lambda \mathcal{R}(\mathbf{x}) + const. = \mathrm{Prox}_{(\lambda/\alpha\sigma^2)\mathcal{R}(.)}(\mathbf{z}^k) \end{aligned} \tag{5.11}$$

where, $\mathbf{z}^k = \mathbf{x}^k + \mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x}^k) \Rightarrow \mathbf{z}^k = \mathbf{x}^{(k)} + \frac{1}{B} \sum_{i=1}^{B} \mathbf{S}_i^T \mathbf{H}^T \mathbf{M}^T \left(\mathbf{y}_i - \mathbf{M}\mathbf{H}\mathbf{S}_i \mathbf{x}^{(k)}\right)$ (see Fig. 5.5), and the *const.* does not depend on $\mathbf{x}$ and thus it is irrelevant to the optimization task. The $\mathrm{Prox}_{(.)}$ is the proximal operator [85], which is defined as:

$$\mathbf{P}_{\mathbb{C}}(\mathbf{z}) = \arg\min_{\mathbf{x} \in \mathbb{C}} \frac{1}{2\sigma^2} \|\mathbf{x} - \mathbf{z}\|_2^2 + \frac{\lambda}{\alpha} \mathcal{R}(\mathbf{x}). \tag{5.12}$$

It can be noted that the Eq. (5.11) is treated as the objective function of a denoising problem, where $\mathbf{z}$ is the noisy observation. So, we employ a deep denoising neural network to get the required estimate $\hat{\mathbf{x}}^{(k)}$ by unrolling the network on K finite steps. Moreover, in the Eq. (5.11), we decouple the degradation operator $\mathbf{A}$ from $\mathbf{x}$ and now we need to tackle it with a less complex denoising problem. For the fast convergence and less computationally expensive, we adopt the similar strategy as done in [56, 83, 57], where the trainable extrapolation weights $\mathbf{w^{(k)}}$ are learnt directly from the training data instead of the fixed ones [67]. Our overall proposed Burst SR scheme is shown in the Fig. 5.5.

5.2.2 Proposed Burst SR Network (BSRICNN)

5.2.2.1 Network Architecture

In the Fig. 5.5, we adapt the similar Encoder-Resnet-Decoder (ERD) architecture as done in [83]. In each ERD block, the *Encoder* and *Decoder* have *Conv* and *TConv* layers respectively with 64 feature maps of 5×5 kernel size with $C \times H \times W$ tensors, where C is the number of channels of the input image. The *Resnet* consists of 5 residual blocks with two Pre-activation *Conv* layers, each of 64 feature maps with kernels support 3×3, and the pre-activation is the parametrized rectified linear unit (PReLU) [38] with 64 out feature channels. There is a trainable projection layer [64] inside the *Decoder* which computes the proximal map for the Eq. (5.12) with given noise standard deviation σ and handles the data fidelity and prior terms. The noise realization is estimated in the intermediate *Resnet* that is sandwiched between the *Encoder* and the *Decoder*. Finally, the clipping layer incorporates our prior knowledge about the valid range of image intensities and enforces the pixel values of the reconstructed image to lie in the range $[0, 255]$. Reflection padding is also used before all *Conv* layers to ensure slowly-varying changes at the boundaries of the input images.

5.2.2.2 Network Training via TBPTT

Due to the iterative nature of our Burst SR approach, the network parameters are updated using back-propagation through time (BPTT) algorithm by unrolling K steps to train the network, which is previously used in recurrent neural networks training such as LSTMs. However, it is computationally expensive with the increase of the number of iterative steps K, so both K and mini-batch (N) size are upper-bounded by the GPU memory. Therefore, to tackle this problem, we use the Truncated Backpropagation Through Time (TBPTT) algorithm as done in [56, 83] to train our network, where the sequence is unrolled into a small number of k-steps out of total K and then the back-

Algorithm 2: The proposed iterative Burst SR approach. The ERD structure and parameters are shared across all stages.

Input : $\mathbf{y}_i$: Raw LR burst input of size B, $\mathbf{M}$, $\mathbf{H}$, $\mathbf{S}$: Degradation operators, K: iterative steps, $\mathbf{w} \in \mathbb{R}^K$: extrapolation weights, σ: estimated noise, α, λ: projection parameters

Initialization: $\mathbf{x}^{(0)} = 0$, $\mathbf{z}^{(0)} = 0$, $\mathbf{x}^{(1)} = \mathbf{y}_{ref}$;

Estimate wrapping matrix: $\mathbf{S}_i, i = 1, \ldots, B$;

for $k \leftarrow 1$ **to** K **do**

  Extrapolation step: $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{w}^{(k)}(\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)})$;

  for $i \leftarrow 1$ **to** B **do**

    $\mathbf{z}^{(k)} = \mathbf{z}^{(k)} + \mathbf{S}_i^T \mathbf{H}^T \mathbf{M}^T (\mathbf{y}_i - \mathbf{MHS}_i \mathbf{x}^{(k+1)})$;

  end

  Proximal step (ERD-block): $\hat{\mathbf{x}}^{(k+1)} = \text{ERD}(\mathbf{x}^{(k)} - \mathbf{z}^{(k)}/B, \sigma, \alpha, \lambda)$;

end

Output: $\hat{\mathbf{x}}^K$: SR output

propagation is performed on the small k-steps. The Algorithm 2 describes the inputs, initial conditions, and desired updates for each network stage. The ERD structure and parameters are shared across all iterative steps. Finally, a single optimizer is used to minimize the ℓ_1-Loss with respect to ground-truth images after k iterative steps according to Eq. (5.13)

5.2.2.3 Network Loss

During the training, we use the following function to minimize the ℓ_1-Loss between the estimated latent SR image ($\mathbf{x}^{(k)}$) and ground-truth (GT) ($\mathbf{x}^{(gt)}$) after k-steps as:

$$\mathcal{L} = \arg\min_{\Theta} \mathcal{L}(\mathbf{\Theta}) = \frac{1}{2} \sum_{i=1}^{N} \|\mathbf{x}_i^k - \mathbf{x}_i^{gt}\|_1 \tag{5.13}$$

where, N is the mini-batch size and $\mathbf{\Theta}$ are the trainable parameters of our network. For computational efficiency, a single optimizer is used during the training.

5.2.3 Experiments

5.2.3.1 Training data

For Synthetic data, we used 46,839 sRGB images from the Zurich RAW to RGB dataset [44], provided in the Burst SR challenge track-1 for the training. We generate the synthetic RAW LR bursts with their corresponding HR using the data generation code as done in [10]. For the validation and testing phase, 1204 RAW LR synthetic bursts have been generated by the data generation code using the sRGB images from the test split of the Zurich RAW to RGB dataset [44]. For the evaluation of our method on the real dataset, we use the BurstSR testset containing 639 real-world LR bursts, provided in the Burst SR challenge track-2.

We estimate the warp matrix (*i.e.*, $\mathbf{S}_i$, refers to Algorithm-2 and section-5.2.1) to align the bursts to the base/reference frame using the Enhanced Correlation Coefficient (ECC) [26] method as done in [57] for all experiments. For those bursts whose are not aligned by the ECC method, we keep them in the training and testing data by making the assumption of the identity matrix.

5.2.3.2 Training description

For the training phase, we set the input Raw burst LR patch sizes as $48 \times 48 \times 4$ with their corresponding Raw HR patch sizes as $384 \times 384 \times 3$ by the scaling factor $\times 4$. We use the LR burst size of 14. We train the network for 368k iterations with a batch size of 2 using Adam optimizer [52] with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ without weight decay to minimize the loss (5.13). The learning rate is set to 10^{-3} for all iterations. We unroll the proposed network into K stages, where we set K as 10.

We implemented our method with Pytorch 1.7.1. The experiments are performed under Windows 10 with i7-8700 CPU with 32GB RAM and on NVIDIA GeForce RTX-3090 GPU with 24GB memory. The average running times (image per second on GPU)

Table 5.3: We compare our method with the common evaluation metrics (PSNR / SSIM / LPIPS). The quantitative SR results (×4 upscale) are shown over the synthetic and real Burst SR test sets. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red and the second best performance is shown in blue.

Burst SR Method	#Params [M]	#Conv2d	Synthetic data			Real data			Fine-tuned on Real data
			PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	
DeepJoint [34]+RRDB [106]	17.26	371	33.25	0.881	0.195	42.13	0.957	0.088	✓
DeepBurstSR [10]	5.25	48	34.48	0.905	0.118	45.17	0.978	0.037	✓
HighRes-net [21]	1.11	25	34.30	0.891	0.170	43.99	0.972	0.051	✓
BSRICNN (ours)	0.38	12	37.62	0.895	0.166	41.40	0.952	0.101	✗

are 0.3350 and 0.8838 over the synthetic and real testsets, respectively.

5.2.3.3 Comparison with the Burst SR methods

We compare our method with existing Burst SR methods including DeepJoint [34] + RRDB [106], DeepBurstSR [10], and HighRes-net [21]. In order to do a fair comparison, the other methods are trained to perform joint denoising, demosiacking, and super-resolution using the same training data used by our method, expect that we are not fine-tuned the pretrained synthetic data model on the real data, while others are trained on that.

We evaluate the trained model under the Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), and LPIPS [124] metrics, refer to the section 4.1.4.4 for more details of the evaluation metrics. The quantitative Burst SR results are evaluated on the raw *linear sensor* space. Table 5.3 shows the quantitative results of our method over the synthetic and real Burst SR testset. We have achieved excellent PSNR on the synthetic data, compared to DeepBurstSR method, while, lags the SSIM/LPIPS score, even though the parameters and depth of the proposed network is much less than the DeepBurstSR.

On the real-world burst data comparison, the DeepBurstSR outperforms the others methods in terms of PSNR/SSIM/LPIPS, while we have a comparable SSIM score with others, even though our method is not fine-tuned on the real data. Since the fine-tuning [10] increases the performance, but it also further requires significant additional labelled training data that is difficult to collect in practice, and it is also more

computational expensive in terms of training time and hardware resources.

Table 5.4: The participating methods results on the synthetic test set from Track-1 in the Burst SR Challenge, in terms of PSNR, SSIM, and LPIPS.

Team Name	PSNR↑	SSIM↑	LPIPS↓
Noah_TerminalVision_SR	46.85	0.983	0.018
MegSR	46.72	0.983	0.020
Inria	44.76	0.969	0.034
TTI	44.40	0.973	0.038
BREIL	39.22	0.918	0.104
MLP_BSR	37.62	0.895	0.166

Table 5.5: The participating methods results on BurstSR test set from Track-2 in the Burst SR Challenge. The PSNR, SSIM, and LPIPS scores are computed after spatial and color alignment of the network prediction to the ground truth.

Team Name	PSNR↑	SSIM↑	LPIPS↓
MegSR	45.45	0.979	0.032
Noah_TerminalVision_SR_B	45.26	0.978	0.026
Noah_TerminalVision_SR_A	45.36	0.979	0.035
TTI	44.16	0.974	0.040
MLP_BSR	41.40	0.952	0.101
BREIL	29.93	0.797	0.141

5.2.3.4 The NTIRE2021 Burst SR Challenge

We participated in the NTIRE2021 Burst SR Challenges, namely Synthetic (Track-1) and Real-world (Track-2). The goal of the challenge is to super-resolve ($\times 4$) images from real-world burst LR input. We train firstly the proposed network on the synthetic Burst SR dataset provided in the Track-1 to jointly learn the denosining, demosaicking, and super-resolution, and after that apply the pretrained synthetic data model on the real-world BurstSR data provided in the Track-2. Table 5.4 and 5.5 provide the preliminary $\times 4$ SR results for track-1 and track-2 (testset) of our method (**MLP_BSR**) with other participants.

5.2.3.5 Visual comparison on the Real-World LR Bursts

Regarding the visual quality, Fig. 5.6 shows the qualitative comparison of our method with other Burst SR methods at the $\times 4$ upscaling factor on BurstSR test-set. Our method has still produced satisfying results on the real LR bursts without fine-tuning

Figure 5.6: SR visual comparison of the proposed BSRICNN method with the existing Burst SR methods on the real-world BurstSR testset at the $\times 4$ upscaling factor. All images are converted from the raw sensor space to sRGB for visualization purpose.

cost, while the DeepBurstSR [10] fine-tune the model on the real LR burst dataset to the get the SR results which require extra training cost with more GPU hours consumption.

5.2.3.6 Ablation Study

For our ablation study, we compare the impact of different numbers of input burst frames and iterative steps for the proposed Burst SR method. Table 5.6 shows the average PSNR/SSIM/LPIPS score after iterative steps (K) and LR bursts size on the synthetic burst testset at $\times 4$ upscaling factor. Our model is trained using a fixed number of iterative steps of 10 and burst size of 14. Our model achieves better performance (PSNR/SSIM/LPIPS) with the larger burst size and the number of iterative steps. It

Table 5.6: Impact of different number of input burst frames (B) and number of iterative steps (K). The quantitative results are reported on the synthetic burst testset. The arrows indicate if high ↑ or low ↓ values are desired. The best performance is shown in red.

Burst Size (B)	iterative steps ($K = 5$)			iterative steps ($K = 10$)		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
2	34.19	0.8790	0.2498	34.12	0.8777	0.2480
4	34.69	0.8852	0.2359	34.66	0.8842	0.2317
8	35.09	0.8887	0.2277	34.99	0.8876	0.2217
14	35.12	0.8896	0.2255	35.30	0.8903	0.2165
16	35.21	0.8907	0.2232	35.30	0.8909	0.2168
32	35.23	0.8902	0.2236	35.41	0.8909	0.2159

shows the generalization ability of our method to other input bursts and the effective utilization of the information from multiple LR frames in order to improve Burst SR performance.

5.2.4 Discussion and Limitations

Our proposed Algorithm 2 has a close connection with other proximal algorithms such as ISTA [19] and FISTA [5] that require the exact form of the employed regularizer such as Total Variation / Hessian Schatten-norm [65]. However, in our case, the regularizer is learned implicitly from the training data (*i.e.*, non-convex form) and therefore we do not have any assumptions regarding the explicit form of the regularizer. Our proposed algorithm acts as an inexact form of the proximal gradient descent step.

Figure 5.7: Imprecise warp matrix. All images are converted from raw sensor space to sRGB for visualization purpose.

Since we are estimating the warping matrix by using the ECC method to align the observations to the reference frame, sometimes its estimation is imprecise that will introduce undesirable artifacts to the final SR result. In Fig. 5.7, we show the failure case of our method, when we use identity matrix instead of the correct matrix estimate by the ECC.

6

Conclusion

6.1 Broader Impact

Image super-resolution has a broad potential impact through a wide range of applications. These include satellite imaging, medical imaging, medicine, telescope imaging in astronomy, portable device imaging, graphics, forensics, security and surveillance imaging. In the last decade, most of the photos have been taken using built-in smartphone cameras, where the resulting low-quality images is inevitable and undesirable due to their physical limitations. Since the mobile cameras are small and versatile due to their compact camera sensors, there are several key limitations [20] of the mobile phone camera as compared to a DSLR *i.e.*, small sensor size, limited aperture, noise (*i.e.*, photon shot and read noise) and limited dynamic range, limited depth of field due to fixed aperture, limited zoom and color sub-sampling. As a result, the image quality is not comparable with that of DSLR cameras. Therefore, the focus is shifted towards software solutions of the cameras to mitigate their limitations.

6.2 Future Works

In the recent works [102, 83, 81, 84, 77, 109, 118], we proposed deep learning based SR methods for effective and efficient SISR. For the burst SR, our proposed SR method has participated to the recent CVPR-2021 challenge [9]. Recently, numerous works have been addressed towards the task of effective and efficient SISR methods, while the explorable and burst SR tasks have received little attention due to the challenging nature of the inverse problem. The existing SISR methods are usually not optimized for common smartphone AI hardware [43] due to the limited memory and storage constraints. The existing SISR methods do not allow to generate the infinitely many plausible HR images [3] vary in their textures and fine details that are consistent with the given observed LR image.

In the future work, we are interested to explore the off-the-shelf deep learning components such as Normalizing Flows with invertible neural network (INN) [53], energy-based models (EBM) [59], vision transformers [50], GAN specially score-based [94], transformer-based [41], and StyleGAN [48], and self-supervised learning approaches to develop SR methods for the existing SR problems, in particular the explorable SR task.

6.3 Conclusion

We have witnessed many advances in the field of image super-resolution over the past few years. The first thing we have explored the effective and efficient SR approaches. We proposed the deep CNN model that follows the realistic degradation process. After that, we designed the iterative SISR scheme by following the powerful image regularization and large-scale optimization techniques with the residual learning approach. Then, we proposed the efficient SISR approach by cascading the residual denoiser networks.

We then discussed the real-world SR problems. We proposed the deep GAN-based

SR approach to first generate realistic LR/HR pairs and then design the SR network that follows the image observation (physical) model by training on the generated LR/HR data. Next, we proposed the deep end-to-end GAN-based SR method by translating the LR to HR domain and vice-versa. Then, we incorporated the learnable adaptive sinusoidal non-linearities into the LR and SR learning process to further enhance the real-world SR performance.

Finally, we explored the multi-image SR tasks. In this regard, we proposed the deep StarGAN approach to super-resolve the LR images from the multiple different LR degradation domains. Next, we proposed the deep iterative burst SR network that follows the forward (physical) observation model of burst photography pipeline.

Despite the recent rapid progress attained in the image SR, there are still many open challenges due to the hard nature of the inverse problems. The important stream of research could be to generate diverse SR samples consistent with the input LR image by learning the SR space. Another important research dimension is the multi-frame SR with the increasing popularity of burst photography. Moreover, the other important aspect of the image SR is to make the existing deep networks to optimize real-time performance on mobile or edge NPUs.

Bibliography

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPRW*, pages 126–135, 2017.

[2] Pablo Andres Arbelaez, Michael Maire, Charless C. Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:898–916, 2011.

[3] Yuval Bahat and Tomer Michaeli. Explorable super resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2716–2725, 2020.

[4] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, pages 183–202, 2009.

[5] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, pages 183–202, 2009.

[6] Aleksandr Belov, Joel Stadelmann, Sergey Kastryulin, and Dmitry V Dylov. Towards ultrafast mri via extreme k-space undersampling and superresolution. *arXiv preprint arXiv:2103.02940*, 2021.

[7] Mario Bertero and Patrizia Boccacci. *Introduction to inverse problems in imaging*. CRC press, 1998.

[8] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. *BMVC*, 2012.

[9] Goutam Bhat, Martin Danelljan, Radu Timofte, et al. NTIRE 2021 challenge on burst super-resolution: Methods and results. In *CVPRW*, 2021.

[10] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Deep burst super-resolution. *arXiv preprint arXiv:2101.10997*, 2021.

[11] Giacomo Boracchi and Alessandro Foi. Modeling the performance of image restoration from motion blur. *IEEE Transactions on Image Processing*, 21:3502–3517, 2012.

[12] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, Jan. 2011.

[13] Jose Caballero, Christian Ledig, Andrew Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, and Wenzhe Shi. Real-time video super-resolution with spatio-temporal networks and motion compensation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4778–4787, 2017.

[14] Jiezhang Cao, Yawei Li, Kai Zhang, and Luc Van Gool. Video super-resolution transformer. *arXiv preprint arXiv:2106.06847*, 2021.

[15] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, May 2011.

[16] Kelvin CK Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Basicvsr: The search for essential components in video super-resolution and beyond.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4947–4956, 2021.

[17] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1256–1272, 2017.

[18] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, pages 8789–8797, 2018.

[19] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, pages 1413–1457, 2004.

[20] Mauricio Delbracio, Damien Kelly, Michael S Brown, and Peyman Milanfar. Mobile computational photography: A tour. *arXiv preprint arXiv:2102.09000*, 2021.

[21] Michel Deudon, Alfredo Kalaitzis, Israel Goytom, Md Rifat Arefin, Zhichao Lin, Kris Sankaran, Vincent Michalski, Samira E Kahou, Julien Cornebise, and Yoshua Bengio. Highres-net: Recursive fusion for multi-frame super-resolution of satellite imagery. *arXiv preprint arXiv:2002.06460*, 2020.

[22] Bernhard Dieber, Christian Micheloni, and Bernhard Rinner. Resource-aware coverage and task assignment in visual sensor networks. *IEEE Trans. Circuits Syst. Video Techn.*, 21:1424–1437, 10 2011.

[23] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *ECCV*, 2014.

[24] Weisheng Dong, Lei Zhang, Guangming Shi, and Xin Li. Nonlocally centralized sparse representation for image restoration. *IEEE Transactions on Image Processing*, 22:1620–1630, 2013.

[25] Netalee Efrat, Daniel Glasner, Alexander Apartsin, Boaz Nadler, and Anat Levin. Accurate blur models vs. image priors in single image super-resolution. *2013 IEEE International Conference on Computer Vision*, pages 2832–2839, 2013.

[26] Georgios D Evangelidis and Emmanouil Z Psarakis. Parametric image alignment using enhanced correlation coefficient maximization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, pages 1858–1865, 2008.

[27] Ruicheng Feng, Jinjin Gu, Yu Qiao, and Chao Dong. Suppressing model overfitting for image super-resolution networks. *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 0–0, 2019.

[28] Mário Figueiredo, José M Bioucas-Dias, and Robert D Nowak. Majorization–minimization algorithms for wavelet-based image restoration. *IEEE Transactions on Image processing*, 16(12):2980–2991, 2007.

[29] Mário AT Figueiredo, José M Bioucas-Dias, and Robert D Nowak. Majorization–Minimization algorithms for wavelet-based image restoration. *IEEE Transactions on Image processing*, pages 2980–2991, 2007.

[30] G. L. Foresti, C. Micheloni, L. Snidaro, and C. Marchiol. Face detection for visual surveillance. In *12th International Conference on Image Analysis and Processing, 2003.Proceedings.*, pages 115–120, Sep. 2003.

[31] Manuel Fritsche, Shuhang Gu, and Radu Timofte. Frequency separation for real-world super-resolution. *ICCV workshops*, 2019.

[32] Jorge García, Niki Martinel, Alfredo Gardel, Ignacio Bravo, Gian Luca Foresti, and Christian Micheloni. Modeling feature distances by orientation driven classifiers for person re-identification. *J. Vis. Comun. Image Represent.*, 38(C):115–129, July 2016.

[33] D. Geman and Chengda Yang. Nonlinear image recovery with half-quadratic regularization. *IEEE Transactions on Image Processing*, 4(7):932–946, July 1995.

[34] Michaël Gharbi, Gaurav Chaurasia, Sylvain Paris, and Frédo Durand. Deep joint demosaicking and denoising. *ACM Transactions on Graphics (TOG)*, pages 1–12, 2016.

[35] T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.

[36] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems (NIPS)*, pages 2672–2680, 2014.

[37] Ke Han, Yan Huang, Zerui Chen, Liang Wang, and Tieniu Tan. Prediction and recovery for adaptive low-resolution person re-identification. In *European Conference on Computer Vision (ECCV)*, pages 193–209, 2020.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[39] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5197–5206, 2015.

[40] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *CVPR*, pages 5197–5206, 2015.

[41] Drew A Hudson and C Lawrence Zitnick. Generative adversarial transformers. *arXiv preprint arXiv:2103.01209*, 2021.

[42] David R Hunter and Kenneth Lange. A tutorial on MM algorithms. *The American Statistician*, pages 30–37, 2004.

[43] Andrey Ignatov, Radu Timofte, Maurizio Denna, Abdel Younes, Andrew Lek, Mustafa Ayazoglu, Jie Liu, Zongcai Du, Jiaming Guo, Xueyi Zhou, et al. Real-time quantized image super-resolution on mobile npus, mobile ai 2021 challenge: Report. *CVPRW*, 2021.

[44] Andrey Ignatov, Luc Van Gool, and Radu Timofte. Replacing mobile camera isp with a single deep learning model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 536–537, 2020.

[45] Takashi Isobe, Xu Jia, Shuhang Gu, Songjiang Li, Shengjin Wang, and Qi Tian. Video super-resolution with recurrent structure-detail network. In *European Conference on Computer Vision (ECCV)*, pages 645–660, 2020.

[46] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, pages 1125–1134, 2017.

[47] Tao Jiang, Xiaojun Wu, Zhang Yu, Wuyang Shui, Gang Lu, Shiqi Guo, Hao Fei, and Qieshi Zhang. Recursive inception network for super-resolution. *IEEE International Conference on Pattern Recognition (ICPR)*, pages 2759–2764, 2018.

[48] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8110–8119, 2020.

[49] Michael Kellman, Kevin Zhang, Jon Tamir, Emrah Bostan, Michael Lustig, and Laura Waller. Memory-efficient learning for large-scale computational imaging. *arXiv preprint arXiv:2003.05551*, 2020.

[50] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *arXiv preprint arXiv:2101.01169*, 2021.

[51] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1646–1654, 2016.

[52] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[53] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[54] Filippos Kokkinos and Stamatios Lefkimmiatis. Deep image demosaicking using a cascade of convolutional residual denoising networks. *IEEE European Conference on Computer Vision (ECCV)*, pages 303–319, 2018.

[55] Filippos Kokkinos and Stamatios Lefkimmiatis. Iterative joint image demosaicking and denoising using a residual denoising network. *IEEE Transactions on Image Processing*, pages 4177–4188, 2019.

[56] Filippos Kokkinos and Stamatios Lefkimmiatis. Iterative joint image demosaicking and denoising using a residual denoising network. *IEEE Transactions on Image Processing*, pages 4177–4188, 2019.

[57] Filippos Kokkinos and Stamatis Lefkimmiatis. Iterative residual cnns for burst photography applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5929–5938, 2019.

[58] Dilip Krishnan and Rob Fergus. Fast image deconvolution using hyper-laplacian priors. *NIPS*, 2009.

[59] Yann LeCun, Sumit Chopra, Raia Hadsell, Aurelio Ranzato, and Fu Jie Huang. A tutorial on energy-based learning. 2006.

[60] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan

Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. *CVPR*, pages 4681–4690, 2017.

[61] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 4681–4690, 2017.

[62] Stamatios Lefkimmiatis. Non-local color image denoising with convolutional neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5882–5891, 2017.

[63] Stamatios Lefkimmiatis. Universal denoising networks: A novel cnn architecture for image denoising. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3204–3213, 2018.

[64] Stamatios Lefkimmiatis. Universal denoising networks: A novel cnn architecture for image denoising. *CVPR*, pages 3204–3213, 2018.

[65] Stamatios Lefkimmiatis, Aurélien Bourquard, and Michael Unser. Hessian-based norm regularization for image restoration with biomedical applications. *IEEE Transactions on Image Processing*, pages 983–995, 2011.

[66] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. *ECCV*, pages 702–716, 2016.

[67] Huan Li and Zhouchen Lin. Accelerated proximal gradient methods for nonconvex programming. *Advances in neural information processing systems (NIPS)*, pages 379–387, 2015.

[68] Yaoman Li, Jinglei Yang, Zheng Liu, Xiaomin Yang, Gwanggil Jeon, and Wei Wu. Feedback network for image super-resolution. *CVPR*, 2019.

[69] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. *CVPRW*, pages 1132–1140, 2017.

[70] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1132–1140, 2017.

[71] Qihang Lin and Lin Xiao. An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization. *International Conference on Machine Learning (ICML)*, pages 73–81, 2014.

[72] Xinhao Liu, Masayuki Tanaka, and Masatoshi Okutomi. Single-image noise level estimation for blind denoising. *IEEE transactions on image processing*, pages 5226–5237, 2013.

[73] Y. Liu, Y. Wang, N. Li, X. Cheng, Y. Zhang, Y. Huang, and G. Lu. An attention-based approach for single image super resolution. *24th IEEE International Conference on Pattern Recognition (ICPR)*, pages 2777–2784, 2018.

[74] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021.

[75] Andreas Lugmayr, Martin Danelljan, and Radu Timofte. Unsupervised learning for real-world super-resolution. *ICCV workshops*, 2019.

[76] Andreas Lugmayr, Martin Danelljan, Radu Timofte, et al. Aim 2019 challenge on real-world image super-resolution: Methods and results. In *ICCV Workshops*, 2019.

[77] Andreas Lugmayr, Martin Danelljan, Radu Timofte, et al. Ntire 2020 challenge on real-world image super-resolution: Methods and results. *CVPR Workshops*, 2020.

[78] David Martin, Charless Fowlkes, Doron Tal, Jitendra Malik, et al. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001.

[79] Emmanuel Moebel, Antonio Martinez-Sanchez, Lorenz Lamm, Ricardo Righetto, Wojciech Wietrzynski, Sahradha Albert, Damien Lariviere, Eric Fourmentin, Stefan Pfeffer, Julio Ortiz, et al. Deep learning improves macromolecule identification in 3d cellular cryo-electron tomograms. *bioRxiv*, pages 2020–04, 2021.

[80] Rao Muhammad Umer, Gian Luca Foresti, and Christian Micheloni. Deep generative adversarial residual convolutional networks for real-world super-resolution. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[81] Rao Muhammad Umer, Gian Luca Foresti, and Christian Micheloni. Deep generative adversarial residual convolutional networks for real-world super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 438–439, 2020.

[82] Rao Muhammad Umer, Gian Luca Foresti, and Christian Micheloni. Deep generative adversarial residual convolutional networks for real-world super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 438–439, 2020.

[83] Rao Muhammad Umer, Gian Luca Foresti, and Christian Micheloni. Deep iterative residual convolutional network for single image super-resolution. In *ICPR*, January 2021.

[84] Rao Muhammad Umer and Christian Micheloni. Deep cyclic generative adversarial residual convolutional networks for real image super-resolution. In *ECCVW*, August 2020.

[85] Neal Parikh and Stephen Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, Jan. 2014.

[86] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary Devito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *Advances in Neural Information Processing Systems (NIPS)*, pages 1–4, 2017.

[87] Valeriya Pronina, Filippos Kokkinos, Dmitry V Dylov, and Stamatios Lefkimmiatis. Microscopy image restoration with deep wiener-kolmogorov filters. In *European Conference on Computer Vision (ECCV)*, pages 185–201, 2020.

[88] Asha Rani, Gian Luca Foresti, and Christian Micheloni. A neural tree for classification using convex objective function. *Pattern Recogn. Lett.*, 68(P1):41–47, Dec. 2015.

[89] Mehdi SM Sajjadi, Raviteja Vemulapalli, and Matthew Brown. Frame-recurrent video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6626–6634, 2018.

[90] J. C. SanMiguel, C. Micheloni, K. Shoop, G. Foresti, and A. Cavallaro. Self-reconfigurable smart camera networks. *Computer*, 47(05):67–73, May 2014.

[91] Uwe Schmidt and Stefan Roth. Shrinkage fields for effective image restoration. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2774–2781, 2014.

[92] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016.

[93] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

[94] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[95] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. Detail-revealing deep video super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, pages 4472–4480, 2017.

[96] Yapeng Tian, Yulun Zhang, Yun Fu, and Chenliang Xu. Tdan: Temporally-deformable alignment network for video super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3360–3369, 2020.

[97] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. In *CVPRW*, pages 114–125, 2017.

[98] Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 114–125, 2017.

[99] Radu Timofte, Rasmus Rothe, and Luc Van Gool. Seven ways to improve example-based single image super resolution. In *CVPR*, pages 1865–1873, 2016.

[100] Radu Timofte, Vincent De Smet, and Luc Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *ACCV*, 2014.

[101] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*

(CVPR), pages 9446–9454, 2018.

[102] Rao Muhammad Umer, Gian Luca Foresti, and Christian Micheloni. Deep super-resolution network for single image super-resolution with realistic degradations. In *ICDSC*, pages 21:1–21:7, September 2019.

[103] Rao Muhammad Umer, Asad Munir, and Christian Micheloni. A deep residual star generative adversarial network for multi-domain image super-resolution. In *6th International Conference on Smart and Sustainable Technologies (SpliTech)*, 2021.

[104] Michael Unser. A representer theorem for deep neural networks. *Journal of Machine Learning Research*, pages 1–30, 2019.

[105] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. Edvr: Video restoration with enhanced deformable convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[106] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. ESRGAN: Enhanced super-resolution generative adversarial networks. *ECCV*, 2018.

[107] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600–612, 2004.

[108] Pengxu Wei, Hannan Lu, Radu Timofte, et al. Aim 2020 challenge on real image super-resolution: Methods and results. 2020.

[109] Pengxu Wei, Hannan Lu, Radu Timofte, Liang Lin, Wangmeng Zuo, et al. AIM 2020 challenge on real image super-resolution: Methods and results. In *ECCVW*, August 2020.

[110] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964.

[111] Bartlomiej Wronski, Ignacio Garcia-Dorado, Manfred Ernst, Damien Kelly, Michael Krainin, Chia-Kai Liang, Marc Levoy, and Peyman Milanfar. Handheld multi-frame super-resolution. *ACM Transactions on Graphics (TOG)*, pages 1–18, 2019.

[112] Jaejun Yoo, Namhyuk Ahn, and Kyung-Ah Sohn. Rethinking data augmentation for image super-resolution: A comprehensive analysis and a new strategy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8375–8384, 2020.

[113] Yuan Yuan, Siyuan Liu, Jiawei Zhang, Yongbing Zhang, Chao Dong, and Liang Lin. Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 701–710, 2018.

[114] Kun Zeng, Hong Zheng, Yanyun Qu, Xiaobo Qu, Lijun Bao, and Zhong Chen. Single image super-resolution with learning iteratively non-linear mapping between low-and high-resolution sparse representations. *IEEE International Conference on Pattern Recognition (ICPR)*, pages 507–512, 2018.

[115] Roman Zeyde, Michael Elad, and Matan Protter. On single image scale-up using sparse-representations. In *International conference on curves and surfaces*, pages 711–730, 2010.

[116] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. *International Conference on Learning Representations (ICLR)*, 2018.

[117] Kai Zhang, Martin Danelljan, Yawei Li, Radu Timofte, et al. AIM 2020 challenge on efficient super-resolution: Methods and results. In *European Conference on*

Computer Vision Workshops, 2020.

[118] Kai Zhang, Martin Danelljan, Yawei Li, Radu Timofte, Jie Liu, Jie Tang, Gangshan Wu, Yu Zhu, Xiangyu He, Wenjie Xu, et al. AIM 2020 challenge on efficient super-resolution: Methods and results. In *ECCVW*, pages 5–40, 2020.

[119] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26:3142–3155, 2017.

[120] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2808–2817, 2017.

[121] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep CNN denoiser prior for image restoration. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2808–2817, 2017.

[122] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Learning a single convolutional super-resolution network for multiple degradations. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3262–3271, 2018.

[123] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Deep plug-and-play super-resolution for arbitrary blur kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1671–1681, 2019.

[124] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018.

[125] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, pages 47–57, 2016.

[126] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.