

# Image Restoration Using Deep Learning



**Trinity College Dublin**  
Coláiste na Tríonóide, Baile Átha Cliath  
The University of Dublin

by

**Fatma Albluwi**

**Dissertation**

Presented to the University of Dublin, Trinity College  
in fulfillment of the requirements the Degree of  
**Doctor of Philosophy**

**University of Dublin, Trinity College**

September 2020



To my loving mother, Sabah.  
To my loving brothers and sisters.



## **Declaration**

I declare that this thesis has not previously been submitted as an exercise for a degree at this or any other University, and it is my own work.

by

**Fatma Albluwi**  
**September 2020**



## **Acknowledgements**

I want to thank my supervisor, Professor Rozenn Dahyot, for her help, support and patience with me. I appreciate all the guidance and encouragement she has given me over these years. I want to express my gratitude to Dr Vladimir A. Krylov for his useful advice and support. I would also like to thank the King Abdullah Scholarship Program from the Saudi Arabian Government for funding that made this PhD possible. I want to acknowledge the support of Trinity College Dublin. Special thanks to my mother, sisters: Samar, Amira and Amani, and my brothers Talal, Ahmed and Mohammed for their love and encouragement. I will be forever grateful for them for everything. Especially thanks to my mother; without her, I would never have had so many opportunities. I had a wonderful time with my friends: Amira, Nermin, Ohoud, Khaula and Mai; I am very grateful for their friendship. Thanks to all GV2 group and in particular everyone in our team: Hana, Matej, Juile, Reem, Jing, Iman, and Mairead. They are wonderful people, and I would like to thank all of them for all their help and the good times. Finally, my gratitude to the chance of living and studying in Ireland, because it helped me in many directions in my own life. It was a beautiful and fruitful journey in which I learned a lot of things. In general, thanks to everyone who smiled for me and gave me support when I felt weak. Thanks to everyone from the bottom of my heart.

Fatma Albluwi





## **Abstract**

In this thesis, we propose several convolutional neural network (CNN) architectures with fewer parameters compared to state-of-the-art deep structures to restore original images from degraded versions. Employing fewer parameters corresponds to a new trend in deep learning that seeks to create lighter/smaller models without affecting performance (i.e., quality of outcomes). Our models are used for image restoration tasks, such as single image super-resolution (SISR), denoising and artefacts reduction. Image restoration is a fundamental application in computer vision which is used in several practical fields such as medical imaging and security systems. Recently, several state-of-the-art algorithms have been developed to infer high-resolution (HR) images from only low-resolution (LR) images using deep learning algorithms. Tackling distortion such as blurring in addition to down-sampling in LR images is significant, although it has received much less attention. In this work, we propose multiple deep learning architectures for simultaneously deblurring and producing HR images from blurred and down-sampled images, which is considered a more challenging problem in image super-resolution. Two situations are considered: non-blind, where the nature and level of noise are known; and blind, where less information about the blurring process is available. The non-blind method uses a specific blur level while training the model and testing data, and this method is used when we have prior knowledge of the blur kernel. Also, it requires training a separate model for each blur kernel. However, when the blur kernel is not known, when the blur level is different for each image, the blind approach is used instead, where a single deep learning model is trained to restore a LR and blurry image. Furthermore, image compression formats such as MPEG and JPEG can present a combination of degradation effects in images such as blurring, ringing and blocking artefacts. We propose an approach that mitigates this undesirable compression drawback based on the use of CNN models. Our solution improves the visual quality of

degraded images by automating the correct of any such artefacts. We also experimentally show that our proposed CNN architectures can give the same or slightly better results compared to the existing deeper structures that are more costly computationally. Finally, we employ one of the proposed CNN models (DBSR) for denoising the natural noise due to low light conditions in a RENOIR dataset [12], to assess our model using real noisy images.

# Table of contents

<b>List of Figures</b>	<b>xv</b>
<b>List of Tables</b>	<b>xxiii</b>
<b>Acronyms</b>	<b>xxv</b>
<b>1 Introduction</b>	<b>3</b>
1.1 Overview and Motivation . . . . .	5
1.2 Summary of Contributions . . . . .	7
1.3 Thesis Outline . . . . .	8
1.4 List of Publications . . . . .	9
<b>2 Related Work</b>	<b>11</b>
2.1 Capturing Images . . . . .	12
2.1.1 The Image Processing Steps . . . . .	12
2.1.2 Image Compression . . . . .	13
2.2 Image Restoration (IR) . . . . .	14
2.2.1 Degradation models . . . . .	15
2.2.2 Denoising . . . . .	19
2.2.3 Enhancement: Super-Resolution (SR) . . . . .	27
2.3 Deep Learning for Image Denoising and SR Restoration . . . . .	38
2.3.1 Deep Architectures for Single Image SR (SISR) . . . . .	39
2.3.2 Deep Architectures for Denoising . . . . .	43
2.3.3 Challenges and Trends . . . . .	43
2.4 Evaluation of Restoration . . . . .	44

2.4.1	Metrics . . . . .	44
2.4.2	Dataset and benchmarks for evaluation . . . . .	46
2.5	Conclusion . . . . .	47
<b>3</b>	<b>Convolutional Neural Networks (CNNs)</b>	<b>49</b>
3.1	Super-Resolution CNN (SRCNN) . . . . .	50
3.1.1	SRCNN Architecture . . . . .	50
3.1.2	SRCNN Optimisation . . . . .	51
3.1.3	Computation Time . . . . .	52
3.2	Introducing Concatenation . . . . .	54
3.2.1	Compression Artefacts Removal (CAR) Networks . . . . .	54
3.2.2	De-Blurring Super-Resolution CNN (DBSRCNN) . . . . .	57
3.3	Introducing more layers . . . . .	60
3.3.1	De-Blurring Super-Resolution (DBSR) Architecture . . . . .	60
3.3.2	DBSR Optimisation . . . . .	61
3.4	Using Harmonic Blocks (Harm-net) . . . . .	62
3.4.1	Harm-DBSR Architecture . . . . .	62
3.4.2	Harm-DBSR Optimisation . . . . .	64
3.4.3	Compression of the Harm-DBSR Network . . . . .	64
3.5	Conclusion . . . . .	65
<b>4</b>	<b>Experimental Comparisons of DeBlurring Super-Resolution</b>	<b>67</b>
4.1	Methodology . . . . .	68
4.1.1	Training and Testing Datasets . . . . .	68
4.1.2	Degradation Model . . . . .	70
4.1.3	Non-Blind and Blind SISR Scenarios . . . . .	71
4.1.4	Experiments on Colour Images . . . . .	72
4.1.5	Quantitative Metrics for Comparisons . . . . .	73
4.2	Experimental Results . . . . .	73
4.2.1	Evaluation of SRCNN . . . . .	73
4.2.2	Evaluation of DBSRCNN . . . . .	79
4.2.3	Evaluation of DBSR . . . . .	88
4.2.4	Evaluation of Harm-DBSR . . . . .	94

---

4.3	Conclusion . . . . .	102
<b>5</b>	<b>Artefact Reduction in JPEG-Compressed Images</b>	<b>105</b>
5.1	Related Work . . . . .	106
5.2	Data and Training . . . . .	108
5.3	Quantitative Metrics for Comparisons . . . . .	109
5.4	Benchmark Comparisons . . . . .	109
5.5	Conclusion . . . . .	118
<b>6</b>	<b>Denoising in a Real Scenario</b>	<b>119</b>
6.1	Related Work . . . . .	120
6.2	RENOIR Dataset . . . . .	121
6.3	Training and Testing Data . . . . .	122
6.4	Evaluation of Denoising Methods . . . . .	122
6.5	Conclusion . . . . .	128
<b>7</b>	<b>Conclusion and Future Work</b>	<b>129</b>
7.1	Summary . . . . .	129
7.2	Limitations and Future Work . . . . .	132
<b>Appendix A</b>	<b>A Brief Review of Deep Learning</b>	<b>135</b>
A.1	Statistical Learning (SL) . . . . .	135
A.2	Machine Learning (ML) . . . . .	136
A.2.1	Types of Learning Problems . . . . .	136
A.2.2	Regression in ML . . . . .	137
A.2.3	The Gradient Descent (GD) Algorithm . . . . .	138
A.3	Artificial Neural Networks (ANN) . . . . .	140
A.3.1	A Brief Historical Review of ANN . . . . .	141
A.3.2	Forward-Propagation . . . . .	142
A.3.3	Backward Propagation . . . . .	145
A.4	Practical Considerations . . . . .	147
A.4.1	Dataset . . . . .	147
A.4.2	Activation Functions . . . . .	147

---

A.4.3	The Cost Function . . . . .	150
A.4.4	Over-Fitting and Regularisation . . . . .	150
A.4.5	The Learning Rate . . . . .	151
A.4.6	Stochastic GD (SGD) and Mini-Batch GD . . . . .	152
A.5	Convolutional Neural Networks . . . . .	153
A.6	Deep Learning . . . . .	155
<b>Appendix B Quantitative and Some Qualitative Results of SISR Models</b>		<b>157</b>
B.1	Evaluation of SRCNN . . . . .	157
B.2	Quantitative Evaluation of DBSRCNN . . . . .	160
B.3	Qualitative Examples of DBSRCNN and DBSR . . . . .	164
<b>References</b>		<b>177</b>

# List of Figures

1.1	Some types of degradation images (e.g., noisy, blurred, compressed, low-resolution (LR), and blurred LR images) on a colour image, the blur factor is used Gaussian blur with $\sigma = 2$ , and the down-sampling scale is 2. Each image is accompanied by zoom. . . . .	4
2.1	Flowchart of the BM3D. The operations of the BM3D repeats for each group, the reference block marked by "R" is used as a reference to group the similar blocks for it, image taken from [35]. . . . .	26
2.2	Formation of the model of the LR image. . . . .	28
2.3	Description of EDI methods. Figure taken from [10]. . . . .	32
2.4	Internal learning procedure benefits from a multiscale analysis of the LR input image to generate example pairs for learning. This image is taken from [59]. . . . .	38
2.5	Sketch of some deep structures for SISR . . . . .	42
3.1	The SRCNN architecture: the network contains three-layers. Given a LR image $\mathbf{x}$ , the first convolutional layer extracts $n_1$ LR feature maps. Then, the second convolutional layer non-linearly maps the LR feature maps to $n_2$ HR feature maps. Finally, the output layer produces the final SR image $F(\mathbf{x})$ , image taken from [42]. . . . .	52

- 3.2 The proposed structures of the CNN networks: direct and skip architectures. In the direct architecture (DA-CAR), the information directly transfers from the input to the output, while in the skip architecture (SA-CAR) there are some merged layers. We have illustrated the structure of each network as follows: The name of network + the number of layers (the size of filters in each layer)(the number of feature maps in each layer). For instance: DA-CAR3(9-7-1)(64-32-1). We implement the standard JPEG compression method with different JPEG quality factors ( $q = 10, 20$ ) using MATLAB JPEG encoder. We focus on the restoration of the luminance Y channel in the YCbCr space that has been JPEG compressed. . . . . 55
- 3.3 Proposed architecture DBSRCNN: This network involves five layers; four convolutional layers plus concatenation (merge) layer, each layer is responsible for a particular operation; feature extraction, feature enhancement, merge the first two layers, non-linear mapping and finally reconstruction. *Deeper DBSRCNN* contains six layers; the same layers in DBSRCNN plus another non-linear mapping layer. . . . . 59
- 3.4 Proposed architecture DBSR: This network comprises eight layers: the five convolutional layers of DBSRCNN, in addition to extra three enhanced layers inserted after the concatenated layer to further refine the merged feature maps. 61
- 3.5 Visualisation example of the harmonic block implemented on an input layer; taken from [160]. Each 2D filter of the DCT filter bank with size  $K \times K$  is applied to each input feature, to generate the spectral coefficients of the DCT basis functions. Then the weighted linear combination of these coefficients is performed by convolutional filter with size  $1 \times 1$ , to create new feature maps. 63
- 3.6 Visualisation example of the compressed harmonic block implemented on input features. For example, we employ a  $3 \times 3$  DCT filter bank, and applied  $\lambda$ , which is used as a hyper-parameter to reduce the DCT coefficients by limiting the spectral frequencies and truncating the high frequencies. If  $\lambda = 1$ , then the zero frequency (DC component) only will be used. If  $\lambda = 3$ , then six coefficients starting from the DC component will be utilised, and three coefficients will be truncated as illustrated in this example. . . . . 65



4.1	An example of the training phase: Firstly, the blurred LR images are created from HR images (the only pre-processing), using a Gaussian filter to smooth the HR images using $\sigma$ value (the blurring level). Then down-sampling the blurred images using a down-scaling factor, for instance, $s = 3$ . The blurred LR images are zoomed using bicubic interpolation by an up-sampling factor $s = 3$ . These degraded images are used as inputs to the network. The reconstructed SR images resulted from the network should be as similar as possible to HR images. . . . .	71
4.2	SR with SRCNN on a colour image after Gaussian blur with $\sigma = 2$ . The second row shows the results of the non-blind scenario and the blind scenario. Each result is accompanied by zoom and PSNR dB. . . . .	75
4.3	SR with SRCNN on a colour image after Gaussian blur with $\sigma = 3$ . The second row shows the results of the non-blind scenario and the blind scenario. Each result is accompanied by zoom and PSNR dB. . . . .	76
4.4	Example of blind and non-blind SRCNN (9-1-5)(64-32-1), to discover the effect of using the non-blind model on different input images with varying levels of blurring. The yellow boxes show the outputs when the training and testing assumptions match over the blurring value in the non-blind scenarios.	78
4.5	SR with SRCNN and DBSRCNN on grey-scale images after Gaussian blur with $\sigma = 1$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind $\sigma \in [1 - 3]$ SR scenarios). Each result is accompanied by zoom and PSNR dB. . . . .	82
4.6	SR with SRCNN and DBSRCNN on grey-scale images after Gaussian blur with $\sigma = 2$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind $\sigma \in [1 - 4]$ SR scenarios). Each result is accompanied by zoom and PSNR dB. . . . .	83
4.7	SR with different models on images after Gaussian blur with $\sigma = 3$ . The results show the non-blind and blind scenarios. Each result is accompanied by zoom and PSNR. . . . .	91

4.8	SISR performance of different models on 'Butterfly' image after Gaussian blur at $\sigma = 2$ . In the blind scenario $\sigma \in [0.5, 3]$ . . . . .	93
4.9	SR with different models on images after Gaussian blur with $\sigma = 4$ . The results show the non-blind and blind scenarios. Each result is accompanied by zoom and PSNR. . . . .	97
4.10	Average PSNR/ and SSIM results for the blind convolutional network DBSR $\sigma = [1, 3]$ , and its compression using the harmonic blocks; the results are calculated for $\sigma = 1$ , scale factor $s = 3$ on 'Set5'. The Bubble size is according to the number of parameters. The full harmonic network Harm8L-DBSR outperforms the standard DBSR, by using the same number of parameters. From the results, we can notice that the compressed networks with $\lambda = 5, 4$ and 3 except $\lambda = 2$ give good results when compared to the standard convolutional DBSR. However, the network with $\lambda = 6/4L$ used about two and a half times fewer parameters gives the second-best result after Harm8L-DBSR. Adopting the DCT filters enabled us to compress the harmonic nets and use fewer parameters by choosing the most critical low-frequency coefficients. .	100
4.11	SR with different models on images after Gaussian blur with $\sigma = 2$ . The results show the blind scenario $\sigma = [1, 3]$ . Each result is accompanied by zoom and PSNR dB. . . . .	101
5.1	Visualisation example from LIVE1 dataset of compressed corrupted quality images for sailing 2 image at JPEG quality $q = 10$ (Lower quality) & 20 (higher quality). . . . .	108
5.2	Average image reconstruction results reported by PSNR (dB) on the LIVE1 dataset for JPEG quality $q = 10$ . The Bubble size is according to the number of parameters. Our proposed DA-CAR4 network has fewer parameters (where it uses only 30% of the parameters of AR-CNN) but with better performance, and this is achieved by selecting a suitable filter size with the number of layers. DA-CAR5 network realises even better results by using 75% of the parameters of AR-CNN network. . . . .	112

---

5.3	Average image reconstruction results reported by PSNR (dB) on the LIVE1 dataset for JPEG quality $q = 20$ . The Bubble size is according to the number of parameters. Our architecture SA-CAR6 outperforms L8 model, by using around 60% of the number of parameters. . . . .	113
5.4	Qualitative evaluation of reconstruction quality for parrots image using different networks for JPEG quality $q = 10$ . Each result is accompanied by zoom and PSNR dB/ SSIM/ PSNR-B dB. . . . .	115
5.5	Qualitative evaluation of reconstruction quality using different networks for JPEG quality $q = 10$ . Each result is accompanied by zoom and PSNR dB/ SSIM/ PSNR-B dB. . . . .	116
5.6	Qualitative evaluation of reconstruction quality using different networks for JPEG quality $q = 20$ . Each result is accompanied by zoom and PSNR dB/ SSIM/ PSNR-B dB. . . . .	117
6.1	Results of DBSR and BM3D methods on the RENOIR dataset with zoomed crops. . . . .	126
6.2	Results of DBSR and BM3D methods on the RENOIR dataset with zoomed crops. . . . .	127
6.3	Results of DBSR and BM3D methods on the RENOIR dataset with zoomed crops. . . . .	128
A.1	Recurrent Neural Network (RNN). . . . .	141
A.2	Feed Forward Neural Network with three layers. . . . .	144
A.3	Backward Propagation. . . . .	147
A.4	Charts of some activation functions. . . . .	149
A.5	Two different problems of learning rate. . . . .	151
A.6	An illustrated example of a convolutional layer. . . . .	154
A.7	An example of Deep Neural Network Architecture (DNN/ Multilayer neural network) with four fully connected layers. . . . .	156
B.1	The test convergence curves of PSNR (dB) for different SRCNN models with three layer structure (9-1-5)(64-32-1) on the Set5 dataset. . . . .	158

- B.2 The boxplot of different blurring levels, which collects the boxplots of all networks together in one graph to ease comparison, demonstrates that when the blurring level increases, the resolution performance decreases. . . . . 158
- B.3 The boxplots for different trained non-blind and blind super-resolution models. The figure shows that non-blind SR models produce the highest resolution performance for testing images when the suitable model is used, when the model and the input images have the same level of blurring. Also, the wrong blur kernel assumption negatively affects the quality of the restored images. . . . . 159
- B.3 The performance of the different deep learning networks which we applied on non-blind SR input images; using different structures of SRCNN networks and different structures of the proposed network DBSRCNN. It is very obvious from all figures that the worse performance was using (9-1-5)SRCNN, and the better performance was for the deeper DBSRCNN which involves concatenate operation. . . . . 162
- B.4 SR with SRCNN and DBSRCNN on grey-scale images after Gaussian blur with  $\sigma = 1$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind  $\sigma \in [1 - 3]$  SR scenarios). Each result is accompanied by zoom and PSNR dB. . . . . 165
- B.5 SR with SRCNN and DBSRCNN on grey-scale images after Gaussian blur with  $\sigma = 3$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind  $\sigma \in [1 - 3]$  SR scenarios). Each result is accompanied by zoom and PSNR dB. . . . . 166
- B.6 SR with SRCNN and DBSRCNN on grey-scale images after Gaussian blur with  $\sigma = 3$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind  $\sigma \in [1 - 3]$  SR scenarios). Each result is accompanied by zoom and PSNR dB. . . . . 167

B.7	SR with SRCNN and DBSRCNN on grey-scale images after Gaussian blur with $\sigma = 4$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind $\sigma \in [1 - 4]$ SR scenarios). Each result is accompanied by zoom and PSNR dB. . . . .	168
B.8	SR with SRCNN and DBSRCNN on a colour image after Gaussian blur with $\sigma = 2$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind $\sigma \in [1 - 3]$ SR scenarios). Each result is accompanied by zoom and PSNR dB. . . . .	169
B.9	SR with SRCNN and DBSRCNN on a colour image after Gaussian blur with $\sigma = 3$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind $\sigma \in [1 - 3]$ SR scenarios). Each result is accompanied by zoom and PSNR dB. . . . .	170
B.10	Example of non-blind SRCNN (9-1-5)(64-32-1). . . . .	171
B.11	Example 1: example for different networks for each application. . . . .	172
B.12	Example 2: example for different networks for each application. . . . .	173
B.13	SR with different models on images after Gaussian blur with $\sigma = 1$ . The results show the non-blind and blind scenarios. Each result is accompanied by zoom and PSNR. . . . .	174
B.14	SR with different models on images after Gaussian blur with $\sigma = 2$ . The results show the non-blind and blind scenarios. Each result is accompanied by zoom and PSNR. . . . .	175
B.15	SR with different models on images after Gaussian blur with $\sigma = 4$ . The results show the non-blind and blind scenarios. Each result is accompanied by zoom and PSNR. . . . .	176



# List of Tables

2.1	Some of the popular image datasets which are used for super-resolution benchmarks. . . . .	47
3.1	Some differences aspects between the two different implementation. . . . .	53
4.1	Average of PSNR (dB)/ SSIM results with different blur levels $\sigma = 0$ (i.e., without adding any blur), 1, 2, 3, 4, scale factor $s = 3$ on test set 'Set5'. . . . .	74
4.2	Average of PSNR (dB)/ SSIM results with different blur levels $\sigma = 0$ (i.e., without adding any blur), 1, 2, 3, 4, scale factor $s = 3$ on test set 'Set14'. . . . .	74
4.3	Average of PSNR (dB) with different blur levels $\sigma = 0$ (i.e., without adding any blur), 1, 2, 3, 4, scale factor $s = 3$ , on test sets 'Set5' and 'Set14' together as one group. . . . .	77
4.4	Average of PSNR (dB)/ SSIM results for Non-blind Models with different blur levels $\sigma = 0$ (i.e., without adding any blur), 1, 2, 3, 4, scale factor $s = 3$ on test set 'Set5'. . . . .	80
4.5	Average of PSNR (dB)/ SSIM results for Blind Models with different blur levels $\sigma = 0$ (i.e., without adding any blur), 1, 2, 3, 4, scale factor $s = 3$ on test set 'Set5'. 80	
4.6	Average of PSNR (dB)/ SSIM results for Non-blind Models with different blur levels $\sigma = 0$ (i.e., without adding any blur), 1, 2, 3, 4, scale factor $s = 3$ on test set 'Set14'. . . . .	81
4.7	Average of PSNR (dB)/ SSIM results for Blind Models with different blur levels $\sigma = 0$ (i.e., without adding any blur), 1, 2, 3, 4, scale factor $s = 3$ on test set 'Set14'. . . . .	81

4.8	Average of PSNR (dB)/ SSIM results for all non-blind models with different blur levels $\sigma = 0$ (i.e., without adding any blur), 1, 2, 3, 4, scale factor $s = 3$ on test set 'Set5'.	86
4.9	Average of PSNR (dB)/ SSIM results for all non-blind models with different blur levels $\sigma = 0$ (i.e., without adding any blur), 1, 2, 3, 4, scale factor $s = 3$ on test set 'Set14'.	87
4.10	Average PSNR (dB) and SSIM results for $\sigma = 0$ (i.e., without adding any blur) on datasets 'Set5' and 'Set14', with different scale factor $s = 2, 3, 4$ .	88
4.11	Average PSNR (dB)/ and SSIM results with different blur levels $\sigma = 1, 2, 3, 4$ , scale factor $s = 3$ on 'Set5' and 'Set14'.	90
4.12	Average PSNR (dB)/ and SSIM results with different kernel width of blur kernel with scale factor $s = 3$ on 'Set5'.	92
4.13	Average PSNR (dB)/ and SSIM results for non-blind networks with different blur levels at $\sigma = 1, 2, 3, 4$ , scale factor $s = 3$ on 'Set5' and 'Set14'.	95
4.14	Average PSNR (dB)/ and SSIM results with different kernel width of blur kernel with scale factor $s = 3$ on 'Set5'. Our presented results of DBSR and Harm3L-DBSR are from blind scenarios.	95
4.15	Average PSNR (dB)/ and SSIM results for blind networks with different blur levels $\sigma = 1, 2, 3, 4$ , scale factor $s = 3$ on 'Set5' and 'Set14'.	96
4.16	Average PSNR (dB)/ and SSIM results for the blind convolutional network DBSR $\sigma = [1, 3]$ , and its compression using the harmonic blocks; the results are calculated for different blur levels $\sigma = 1, 2, 3$ , scale factor $s = 3$ on 'Set5' and 'Set14'.	99
5.1	Average image reconstruction results reported by PSNR (dB), PSNR-B (dB) and SSIM on the LIVE1 dataset for JPEG quality $q = 10$ .	111
5.2	Average image reconstruction results reported by PSNR (dB), PSNR-B (dB) and SSIM on the LIVE1 dataset for JPEG quality $q = 20$ .	111
5.3	Average image reconstruction results reported by PSNR (dB), PSNR-B (dB) and SSIM on the BSDS500 validation dataset for JPEG quality $q = 10$ .	114
5.4	Average image reconstruction results reported by PSNR (dB), PSNR-B (dB) and SSIM on the BSDS500 validation dataset for JPEG quality $q = 20$ .	114



---

6.1	PSNR (in dB) performance of DBSR and other denoising algorithms reported from [12] on the RENOIR real dataset. . . . .	125
6.2	SSIM performance of DBSR and BM3D algorithms on the RENOIR real dataset, and the average resolution of the images of the different three cameras with an average of the inference CPU time. . . . .	125
B.1	The convergence curves performance of the default and deep models of SRCNN and DBSRCNN for non-blind LR applications on Set5. . . . .	163



# Acronyms

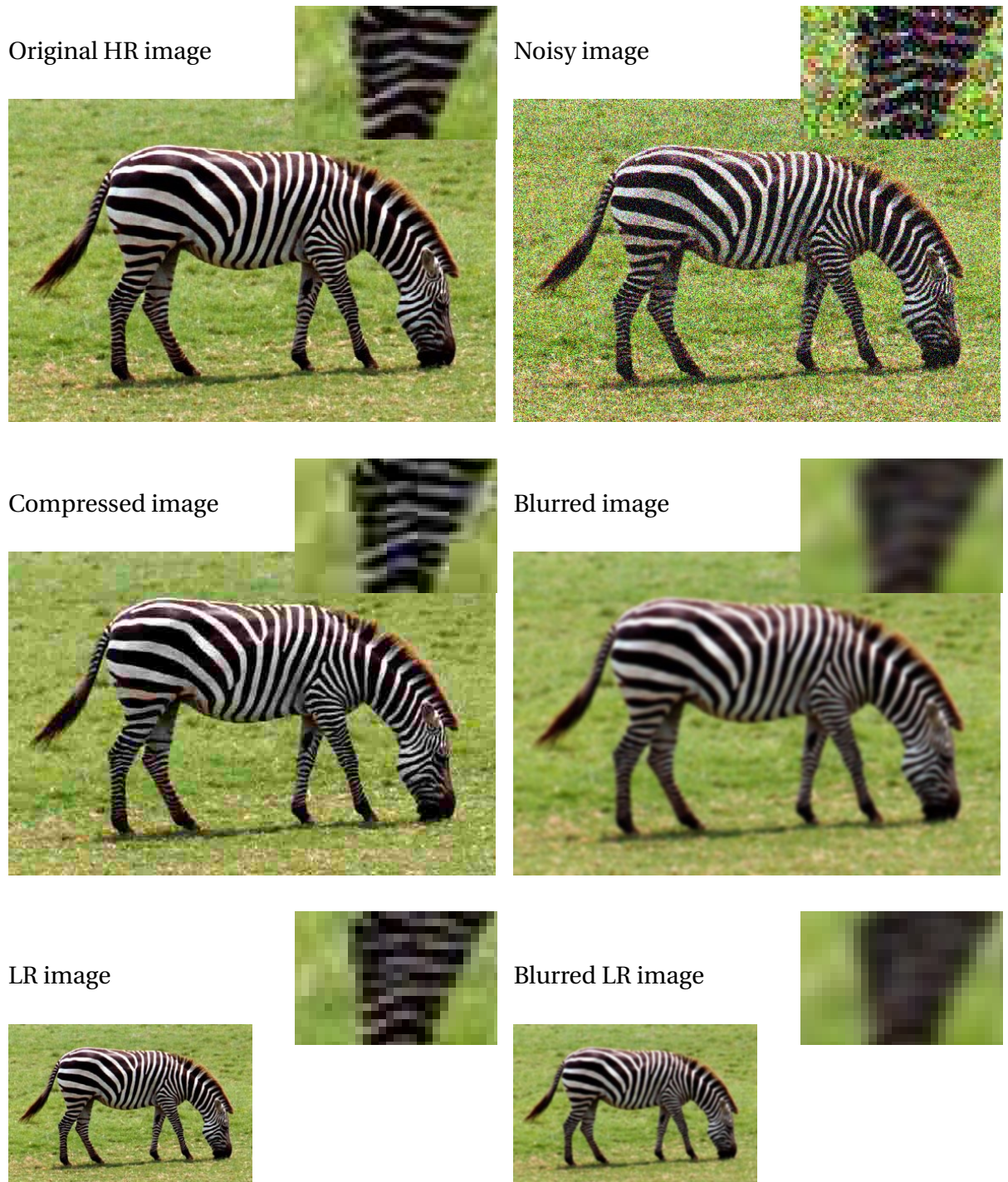
Acronym	Full name	Acronym	Full name
AI	Artificial Intelligence	K-SVD	k-Singular Value Decomposition
ANN	Artificial Neural Network	LR	Low-Resolution image
ANR	Anchored Neighbour Regression	LRR	Low-Rank Representation
AWGN	Additive White Gaussian Noise	LS	Least Square
A+	Adjust ANR	MISR	Multiple-image SR
BM3D	Block Matching and 3D Filtering	ML	Machine Learning
BN	Batch Normalisation	MRF	Markov Random Field
CNN	Convolutional Neural Network	MSE	Mean Square Error
DBSR	Deblurring SR Network	NE	Neighbour Embedding
DBSRCNN	Deblurring SR CNN	NEDI	a New EDI
DL	Deep learning	NLM	Non-Local Means
DnCNN	Denoising CNN	NN	Nearest Neighbour
DNN	Deep Neural Network	NSS	Non-local Self-Similarity
DRCN	Deeply Recursive CNN	PSNR	Peak Signal-to-Noise Ratio
EDI	Edge Directed Interpolation	ResNet	Residual Network
EDSR	Enhanced Deep SR Network	RL	Residual Learning
ESPCN	Efficient Sub-Pixel CNN	RNN	Recurrent neural network
FSRCNN	Fast SRCNN Network	SGD	Stochastic Gradient Descent
GD	Gradient Descent	SISR	Single-image Super-Resolution
GGD	Generalised Gaussian Distribution	SL	Statistical Learning
GPR	Gaussian Process Regression	SR	Super-Resolution
HR	High-Resolution image	SRCNN	Super-Resolution CNN
IFC	Information Fidelity Criterion	SRMD	SR network for Multiple Degradation
IR	Image Restoration	SRResNet	SR Residual Network
IRCNN	Image Restoration CNN	SSIM	Structural Similarity Index Measure
IQA	Image Quality Assessment	TNRD	Trainable Non-linear Reaction Diffusion
K-NN	k-nearest neighbour	VDSR	Very Deep SR network



# Chapter 1

## Introduction

Captured digital images present several types of degradation compared to the real images, which is attributed to many factors such as the capturing processor or by some phenomena of deterioration, for example, fog and wind speed. Therefore, image restoration has become an essential task in computer vision and image processing, and also in several areas such as medical imaging and security systems, to restore the original scene  $y$  from degraded images  $x$ . Many overlapping noises in the environment cause degradation in an imaging process, and this is regarded as a major challenge in image restoration. These factors include imaging conditions (e.g., being out-of-focus), atmospheric turbulence and motion of the scene or degradation as a result of the imaging systems. Distortions also can be produced during image compression or transmission. These factors can introduce many types of image distortions, for instance, noise, blurring, down-sampling or all these together. Figure 1.1 shows some degradations in corrupted images. All these factors lead to a loss of some image information. The dominant approach is the design of task-specific algorithms, such as image deblurring, super-resolution (SR), denoising and removing artefacts produced from the compression. In this thesis, we address the concept of a “real” or “ideal” digital image; thus, we provide an outline description of how a digital camera acquires an image in Chapter 2.



**Fig. 1.1** Some types of degradation images (e.g., noisy, blurred, compressed, low-resolution (LR), and blurred LR images) on a colour image, the blur factor is used Gaussian blur with  $\sigma = 2$ , and the down-sampling scale is 2. Each image is accompanied by zoom.

The linear model which defines image degradation can be formulated as:

$$x = Hy + \eta \quad (1.1)$$

where  $x$  denotes the corrupted image,  $y$  refers to the unknown ground truth image and  $\eta$  is noise.  $H$  is a degradation matrix that defines the degradation type and thus specifies the type of restoration task. If  $H$  is an identity matrix, the restoration task is denoising. However, if the original image is convolved with a kernel as blurring operator, the problem is image deblurring. Also,  $H$  can be designated as a down-sampling operator, and in this case, the restoration problem is super-resolution. Generally, image restoration can be grouped into non-blind and blind image restoration. In the case of non-blind image restoration, the blur kernel estimation is available for restoring the image. However, blind image reconstruction use the corrupted image only, without any information about the blur kernel. The various image restoration techniques seek to model the degradation and then inverse the process to restore the original image. However, image restoration is regarded as an ill-posed problem, which means that a small change in the observation data may lead to a significant change in the solution. Therefore, image restoration is considered a challenging problem in computer vision and the solutions focus on finding strategies to alleviate the ill-posed problem.

## 1.1 Overview and Motivation

We have entered the Big Data era, and this quantity of data calls for automated methods of data analysis and statistical techniques that are capable of extracting information from complex data. Artificial Neural Networks (ANNs) are a collection of powerful Machine Learning (ML) algorithms that can deal with high dimensional data such as images, and they are used to learn a mapping function that can estimate the outputs from the inputs [39]. ANNs are complex models that are able to solve main supervised learning problems, regression and classification. These models attempt to estimate the unknown relationship between the outputs and the inputs without making any assumption around data. Since 2006, Geoffrey Hinton has shown a new kind of neural network known as Deep Learning (DL) which attains great flexibility and power by learning. DL is regarded as a rebranding

of ANNs with several hidden layers. ANN algorithms can be trained in an end-to-end manner with a large amount of training data and automatically extract the necessary features. The pipeline of ANN attempts to automatically extract low-level signal features such as an object's edges, contours or corners. These features are then combined to form intermediate representations which express the object portions (e.g., an eye, a wheel). In the discrimination tasks such as classification, the underlying representations are used to classify the objects into several classes.

Recently, the topic of image restoration has been studied extensively, especially taking into account the advances produced by using the ANNs compared to the classical methods. Furthermore, this topic has become a standard subject to illustrate advances in artificial intelligence (AI) and ML. Tsai and Huang discussed the topic of SR for the first time in 1984 [159]. Since then, this topic has received extensive attention from researchers. SR methods are modelled to restore a High-Resolution (HR) image from multiple Low-Resolution (LR) images or one LR image. The SR methods produce more pleasant images than the interpolated images by designing the relation between LR and HR images. Many approaches have been adopted to accomplish improvement. The first strategies of SR are the classical Multiple Image SR (MISR) methods which suppose that multiple LR images of the same scene exist, and where each LR image includes a small difference from the original scene. Therefore, merging the scattered visual information into one image could enhance the final result. However, if only one LR image exists, then this problem is known as Single Image SR (SISR). The SISR task is a significant application in computer vision, and it is necessary for several areas such as satellite imaging, microscopy, medical imaging and security systems. However, the SISR task is an ill-posed problem, in which one LR image produces many solutions for the HR image. The state-of-the-art approaches of SISR are example-based approaches, in which they learn prior information which alleviates the multiplicity of solutions. The example-based methods are divided into two types of processes: internal techniques and external example-based methods which use external knowledge to enhance the resolution — this thesis focuses on using external knowledge by applying deep learning (DL) techniques.



## 1.2 Summary of Contributions

Recently, the advanced deep architectures have performed well, however, they do so by utilising an extremely large number of parameters. Therefore the new trend in deep learning is to design lighter models with fewer parameters with little or no degradation in performance. In this thesis, we propose deep architectures with fewer parameters compared to the state-of-the-art deep structures (cf. Chapter 3), to restore the real images from the degraded images. The main contributions reported in this thesis are:

1. The task of SISR has witnessed a dramatic improvement in recent years through the use of DL algorithms and, in particular, convolutional neural networks (CNNs), to recover HR images from LR images. Most of the DL algorithms are interested in obtaining estimated HR images from LR images without considering other degradations such as blurring, which is regarded as one of the primary keys of the SR task. Although the distortions are a critical part in restoring SR images, it has been received much less attention. Blurring is a crucial and challenging aspect of the SR procedure because it removes high-frequencies. Therefore, research should put more emphasis on the restoration of the blurring. In this work, we propose new CNN structures that simultaneously addresses deblurring and SR from blurred LR images (cf. Chapter 4). The contributions of this Chapter are summarised as follows:
  - (a) We evaluated the benchmark super-resolution convolutional neural network (SRCNN) architecture proposed in [42] for the blurred reconstruction situation. In addition, we propose a revised deeper model (DBSRCNN) that proves its superiority experimentally in both scenarios when the levels of blur are known and unknown a priori.
  - (b) To address this challenging problem, we propose another new architecture (called DBSR) to tackle blur with the down-sampling of images by extending the DBSRCNN architecture by adding three feature enhancement layers. We validated our new architecture experimentally against several state-of-the-art SR techniques.
  - (c) We have used Harmonic blocks in the proposed DBSR network (Harm-DBSR) to use the transformation methods instead of using standard convolution layers

as in CNN. We have demonstrated that the DBSR results can be enhanced if it is trained on spectral information. At the same time, we can utilise fewer parameters by applying the Harmonic network.

2. The most common image compression formats such as MPEG and JPEG can produce a combination of distorting effects in images such as blurring, ringing and blocking artefacts. We propose an approach that mitigates this undesirable compression drawback based on the use of CNN (called SA-CAR6). Our solution enhances the visual quality of corrupted images by automatically correcting any such artefacts. We also experimentally show that our proposed CNN architectures (DA-CAR3, DA-CAR4 and DA-CAR5) can give the same or slightly better results compared to the existing deeper structures that are more costly computationally (cf. Chapter 5).
3. Noise reduction algorithms have often been evaluated using images degraded by artificially synthesised noise. A RENOIR dataset [12] provides an alternative way to test noise reduction algorithms on real noisy images. We propose to assess our DBSR network to reduce the natural noise due to low light conditions in the RENOIR dataset (cf. Chapter 6).

### 1.3 Thesis Outline

The work carried out in this thesis is structured into seven chapters. Firstly, Chapter 2 reviews the state-of-the-art image restoration techniques, with these techniques being divided into classical approaches and DL methods. Additionally, we present the evaluation of restoration by presenting the metrics (PSNR and SSIM) used in this thesis and the benchmark training and testing datasets. Then, the following four chapters (Chapter 3 to Chapter 6) introduce the contributions of this thesis. In Chapter 3, we present the different CNN models that are proposed in this work. We used the proposed architectures in image restoration such as SISR (cf. Chapter 4), artefacts reduction (cf. Chapter 5) and denoising applications (cf. Chapter 6). We classified the experiments in three chapters (Chapters 4 to 6) depending on the type of task.

In Chapter 4, we address an additional degradation factor of an unknown amount of blurring applied to LR images that are received by the SR pipeline. It is thus necessary to

tackle simultaneously deblurring and SR reconstruction in a unified procedure. Also, we apply SR on blurred images with two different scenarios: with a known priori (non-blind) and unknown (blind) amount of blurring. We used some CNN architectures (DBSRCNN, DBSR, Harm-DBSR) that we have proposed in Chapter 3 to enhance the final results.

In Chapter 5, we propose some CNN architectures of reduced size that effectively suppress artefacts of JPEG-compressed images. The size reduction of CNN allows one to handle training as well as deployment more efficiently. Furthermore, smaller models are a better option since they reduce computational complexity, and will enable us to avoid problems of large networks such as overfitting, vanishing or exploding gradients. Firstly, we present a novel CNN (SA-CAR) for image restoration, which delivers slightly better performance than the state-of-the-art CNN-based models and employs fewer parameters. Besides this, we applied several CNN architectures (DA-CAR3, DA-CAR4 and DA-CAR5) which we proposed with different parameter counts to show that we can achieve the same or better results using smaller networks with a lower parameter count.

In Chapter 6, we train the DBSR model which we designed to recover the deblurred HR images from the blurred LR images, to remove noise and restore meaningful information for the real RENOIR dataset, and also to assess its performance on real noise.

In Chapter 7, we summarise the conclusion of the work carried out in this thesis. In Appendix B, we give additional qualitative results of the experiments. Finally, in Appendix A, we present a concise historical review of ANN and a brief overview of deep learning.

## 1.4 List of Publications

Part of the work performed in this thesis has been published in the subsequent articles:

1. Fatma Albluwi, Vladimir A Krylov and Rozenn Dahyot, Artefacts Reduction in JPEG Compressed Images Using CNNs, Irish Machine Vision and Image Processing Conference (IMVIP), Belfast, Northern Ireland, August 2018.
2. Fatma Albluwi, Vladimir A Krylov and Rozenn Dahyot, Image Deblurring and Super-Resolution Using Deep Convolutional Neural Networks, 28th International Workshop on Machine Learning for Signal Processing (MLSP), Aalborg, Denmark, September 2018.

3. Fatma Albluwi, Vladimir A Krylov and Rozenn Dahyot, Denoising Renoir Image Dataset with DBSR, Irish Machine Vision and Image Processing Conference (IMVIP), Dublin, Ireland, August 2019.
4. Fatma Albluwi, Vladimir A Krylov and Rozenn Dahyot, Super-Resolution on Degraded Low-Resolution Images Using Convolutional Neural Networks, 27th European Signal Processing Conference (Eusipco), A Coruña, Spain, September 2019.

# Chapter 2

## Related Work

In the digital domain, acquiring images (imaging) introduces several types of degradation to the final result. Therefore, digital image restoration techniques play a vital role in many different fields, and this has been extensively studied for the purpose of recovering an original scene from a distorted one. The literature in this field is vast; therefore, we will review the fundamental and state-of-the-art algorithms. Firstly, we discuss how a digital camera acquires an image and the image compression process in Section 2.1. Then in this chapter, we present an overview of image restoration, defining the degradation models in subsection 2.2.1, where the techniques of restoration need some knowledge of the nature of the degradation. Following this, we present a brief review of denoising and super-resolution algorithms for image restoration, in subsections 2.2.2 and 2.2.3. Deep learning plays a vital role nowadays in computer vision tasks, in particular for applications such as image restoration, detection, classification and recognition. Also, image restoration methods based on Convolutional Neural Network (CNN) attained outstanding results compared to the classical methods. Therefore, in Section 2.3, we address deep learning for image restoration. In Section 2.4, we review the evaluation of recovery. Finally, we present the Conclusion in Section 2.5.

## 2.1 Capturing Images

Capturing and focusing the light into the camera is done through the lens. The cameras have become smaller with the appearance of handheld cameras and smartphones, which led to shrinking their optical systems. Thus some physical limits that have its origins in the optics have arisen such as image blurring [32]. However, the improvement in the manufacturing and the technologies has led to enhance the optical quality. In a digital camera to acquire the image, sensors are used to convert the light into pixel data. A pixel is the primary sensor element in the digital camera, and it is considered as a light bucket [1]. The modern camera focuses the light onto complementary metal-oxid semiconductor (CMOS) sensor, which is a silicon-based sensing element. The sensor is sensitive to the light conditions where the low light conditions lead to increasing levels of noise in the captured image. Therefore, the image sensor needs post-processing and correcting the captured raw data. Many different processing steps happen in modern digital cameras of the raw image data known as the image processing pipeline (IPP). This processing is to obtain a high-quality image to display it on the camera screen, or for receiving a suitable image for further processing and compression such as in JPEG (image) or MPEG (video) data format.

### 2.1.1 The Image Processing Steps

We will present here an outline description of the IPP, which shows the sequence of manipulating of the original raw image to obtain the final image on the camera's screen. Bayer pattern is a colour filter array (CFA) of RGB filters on a grid of sensor pixels. The pattern of Bayer mimics the human eye, which is more sensitive to green wavelengths. It obtains luminance (brightness) information principally from green wavelengths and colour information from blue and red wavelengths. Therefore, the Bayer (raw) image has twice as green pixels as blue and red; this image is the unprocessed image. The final image can be automatically achieved after the post-processing of the raw image, which is provided in most consumer cameras. There are primary sources of image degradation in a consumer camera that needs correction, such as the imperfect lens system and the image sensor, which could be a source of the noise. Most modern consumer cameras use ISO sensitivity

that uses as a measure of the ability of the camera to capture light. This is important for CMOS sensors; where an integral amplifier is utilised to decrease or increase the sensitivity of active pixel sensors depending on the light conditions. Bayer scaling is used to convert the image from raw format into another suitable format for further post-processing [32]. In this step, the camera's settings of the white balance are used to scale each of R, G, and B values in the raw image. The best image format from the 1980s is JPEG. However, before performed JPEG compression of the captured raw image, colour transformations are performed. There is an initial RGB conversion which allows for gamma-corrected. Then the RGB information is encoded to another format YCbCr (YCC) which benefits of the system of the human visual perception, where our vision system is more sensitive to luminance than the compression of colour data.

### 2.1.2 Image Compression

Image compression is performed to reduce the memory footprint of an image. JPEG is an abbreviation for the Joint Photographic Experts Group, and it is probably the most extensive used lossy image compression method. It is called lossy compression because of some original information from the image are lost during the process of compression. Firstly the 24-bit RGB input image is encoded into a YCbCr image as the last step in the IPP, and then the converted YCbCr image is further manipulated into JPEG format. Practically, the YCbCr format improves the performance of the JPEG compression process for many reasons; where the YCbCr channels can be separated into the luminance Y channel which is important for the human visual perception, and the less important CbCr chrominance channels. Also, the three channels in YCbCr are decorrelated compared to the highly correlated channels in the RGB colour space. After obtaining the output YCbCr image from the IPP, each component of the three YCbCr is split into  $8 \times 8$  blocks in no downsampling case [32]. Then, 2D discrete cosine transform (DCT) is performed on each  $8 \times 8$  block to transform it into the frequency-domain representation. The DCT is used to eliminate redundant high-frequency data. The human visual system doesn't detect the high-frequency variations in brightness. Therefore, the quantisation method is used to reduce the amount of high-frequency information in the image. The final step in the process of JPEG compression is entropy encoding which helps to reduce strings

of recurring values. JPEG format utilises a quality factor to determine the amount of compression. The lower the quality factor value, the higher the compression, the smaller the memory footprint and the less visual quality of the image. Lossing information in the compression process causes the appearance of different drawbacks in the image such as blocking artefacts, blurring and ringing, especially with the high degree of compression.

## 2.2 Image Restoration (IR)

Image restoration (IR) is a fundamental problem in computer vision and image processing. Furthermore, it is widely studied in different fields such as medical imaging, remote sensing, molecular spectroscopy, satellite imaging, microscopy, digital media restoration and filmography, security systems and surveillance videotapes. In the 1950s and 1960s, the Soviet Union and the United States took photographs of the earth and the solar system. However, these images suffered from various degradations. Since that time, IR began to be an active area of research, seeking less expensive IR techniques, because at this time IR was highly costly. Digital IR aims to recover or reconstruct the original image from a degraded image [17], which has been corrupted by some degradation phenomena. The different techniques of IR try to model the degradation and then inverse the process to restore the original image. Image restoration differs from image enhancement; while image enhancement manipulates an image to obtain more pleasing results, IR is used to recover the original scene  $y$  from degraded images  $x$  [17]. There is a lot of overlapping noise in the environment, which is considered a major challenge in IR, for instance, impulse noise, multiplicative noise and Gaussian noise. Noise may be the result of the camera and imaging system, such as long exposure times, wide-angle lens, size of the camera's lens and wind speed. Other factors leading to degradation include noise during acquisition, compression and transmission, which drives the loss of some image information. Additional factors include blurring degradation which is, for instance, Gaussian blur, motion blur, uniform blur and atmospheric blur. In the domain of image processing, the IR technique used to remove or suppress unwanted features or components from the images is known as filtering. There are multiple filtering methods for restoring the image for instance, Wiener filtering, Median filter, inverse filtering, Harmonic mean filter, Max filter and Arithmetic mean filter. Nevertheless, the inverse filtering method is the simplest and most beneficial



method used in IR. In general, IR can be classified into non-blind and blind IR. In the case of non-blind IR, the blur kernel estimation is available for restoring the image. Within the blind case, recovery is from the corrupted image only without any information about the blur kernel.

Elad et al. [49] addressed the Single Image Super-Resolution (SISR) restoration problem from multiple blurred, noisy and down-sampled images by applying a hybrid algorithm from three methods. These methods are the maximum a posteriori probability (MAP) estimator, the maximum likelihood (ML) estimator, and the set-theoretic approach using projection onto convex sets (POCS). The proposed method supposes explicit knowledge of the linear blur, the additive Gaussian noise and the different measured resolutions. Brown et al. [23] introduced a technique for out-of-focus blur and projector blur to decrease the blur in an image. Yamauchi et al. [176] presented a method to restore digital photographs by using a multi-resolution approach of texture synthesis and image inpainting, to remove image defects such as scratches and annoying objects such as subtitles. Neural networks were used in IR for training the weights, which concentrates on spatial variation in terms of regularisation parameter [120]. The different filtering methods were effective, but strong assumptions about the degradation properties were required. Therefore, these strategies lack the generality to be performed on diverse image sets. Consequently, unsupervised, information-theoretic, adaptive filtering (UINTA) was presented to overcome the filtering limitation [15]. The UINTA automatically recognises the statistical signal properties to restore a broad spectrum of images.

### 2.2.1 Degradation models

Several factors cause distortions in an imaging process; some of the more common degradation mechanisms include imaging conditions such as wrong focus (out-of-focus), atmospheric turbulence and motion of the scene or from imaging systems. These factors can introduce several types of image degradation, for example, noisy, blurred, down-sampled or all these together. Traditionally, the dominant approach was the design of task-specific algorithms [156], such as image deblurring, super-resolution (SR), denoising and inpainting. In this subsection, we describe the equations and processes of the degradation model before discussing the restoration methods. There are several postulated degradation mod-

els related to the recording of a corrupted image. There are non-linear degradation models and linear degradation models. In this work, we will deal with the linear models which reasonably describe many degradations in photography. The linear degradation model is the most commonly used because it is simple and faster than non-linear, while the non-linear model is used with critical data. Mathematically, the linear model which describes image degradation can be written as:

$$x = Hy + \eta \quad (2.1)$$

where  $x$  denotes the distorted image and  $y$  refers to the unknown original image,  $\eta$  is noise, and  $H$  is a degradation matrix (known as the linear distortion operator or the transformation) which defines the restoration task. If  $H$  is an identity matrix, the restoration problem is denoising. However, if the original image is convolved with a kernel as a blurring operator (e.g., Gaussian blur kernel), the problem is image deblurring. Also,  $H$  can be represented as a down-sampling operator; in this case, the restoration problem is a SR.  $H$  is usually associated with the blur kernel or the point spread function (PSF). The noise random variable  $\eta$  is usually assumed to be an Additive White Gaussian Noise (AWGN) with zero mean and a standard deviation  $\sigma$ , where  $\eta \approx N(0, I\sigma^2)$ . The additive Gaussian noise is used in many different scenarios because it efficiently models the noise [17].

There are two methods used for IR. The first utilises hardware techniques [116, 124], in which the degradations are removed before recording the image on the sensor. Although their results are outstanding, the degradation process should be accurately known. Besides, they are usually valid to only one type of distortion. Therefore, these techniques are usually expensive. The second method employs software techniques, which we focus on in this work. These techniques are more flexible for handling the image distortions. Also, some research has combined the methods of hardware and software to obtain better results, such as [93].

## Characteristics of Image Reconstruction

The unknown original image  $y$  is estimated by utilising the degraded image  $x$  and knowing the type of degradation. Therefore, IR is regarded as an inverse problem. In the mid 20<sup>th</sup> century, ill-posed and inverse problems started to be studied within several branches of classical mathematics, such as computational algebra, differential and integral equations.

J. Hadamard formed the well-posedness concept in 1902 [78]. According to Hadamard, there are three conditions which must be satisfied for the linear equation (such as the linear equation 2.1) to be defined as a well-posedness problem. These conditions are as follows:

- The existence condition: A solution exists; where the operator  $H$  is defined on the entire space,  $y$  exists for every image  $x$ .
- The uniqueness condition: A unique solution; where  $H$  is a one-to-one map, it means there is at most one image  $y$  for every image  $x$ .
- The stability condition: A stable solution; it means  $H$  is continuous, where  $y$  depends continuously on  $x$ , and a small change in  $x$  leads to a slight change in  $y$  and vice versa.

The problem of ill-posedness occurs if one condition of the three Hadamard criteria is not met. The ill-posed problem is ill-conditioned if a small change in the input image  $x$  leads to a significant change in the output image  $y$ . In other words, the solution can exist, but it is difficult to find. In this case, the equation will not meet the stability condition, and the solution will be unstable. The IR is possible if the linear equation is well-posed, which implies that the matrix  $H$  is nonsingular (invertible matrix). Nonetheless, the inverse transformation  $H^{-1}$  of the degradation does not exist in most realistic distortions.

Image formation is modelled as a continuous model in infinite-dimensional space, and it is described as a Fredholm integral equation of the first kind [37]. This equation does not meet the criteria of well-posedness; therefore, IR is considered an ill-posed problem [144]. The ill-posed nature of IR indicates that a small variation in the observation data may lead to a significant variation in the solution; hence, trial and error methods cannot be employed in IR.

The pseudo-inverse is one of the simplest procedures used in IR to find a solution for ill-posed problems in linear algebra [4]. This method achieves the first two conditions of Hadamard's criteria (existence and uniqueness), but it does not meet the stability condition. There is, however, a solution that satisfies all the requirements of Hadamard's well-posed problem, which is the regularisation technique. Therefore regularisation is considered to

be one of the most widely-used methods. The regularisation concept indicates finding a solution from data, including some extra or prior information [112].

Most IR methods include a cost function that is minimised to obtain the optimal solution to equation 2.1. The cost function contains two terms; a fidelity term and a regularisation (penalty) term. The fidelity term fits the data whereas the regularisation term is used to regularise the optimisation function through the prior image model  $R(y)$ . The typical cost function used to estimate  $y$  from observation  $x$  is written as [156]:

$$\hat{y} = \arg \min_y \|x - Hy\|_2^2 + \lambda R(y) \quad (2.2)$$

where  $\hat{y}$  is an estimate of ideal image  $y$ , and  $\|\cdot\|_2$  is the Euclidean norm.  $\|x - Hy\|_2^2$  is the data fidelity term that indicates the difference between the ideal image and corrupted images.  $R(y)$  is the regularisation term and  $\lambda$  is the regularisation parameter.

### 2.2.2 Denoising

Denoising procedures are employed to remove the noise in noisy images and recover the real image by minimising the loss of original features. Noise components such as texture and edge are high-frequency features; thus, it is hard to differentiate them during the process of reducing noise. Therefore, the reconstructed images could unavoidably lose some information. Although image denoising is a classical task in image processing, it is still an open and challenging problem. The noise reduction process is considered an inverse problem from the mathematical perspective, but its solution is not unique; therefore, it has been studied for many years. The mathematical formulation of the denoising task can be modelled as:

$$x = y + \eta \quad (2.3)$$

where  $x$  is the noisy image,  $y$  is the unknown real image, and  $\eta$  represents the Additive White Gaussian Noise (AWGN) with a standard deviation  $\sigma_\eta$ . The proposed denoising techniques can be split into three main classifications: classical techniques, which can be termed spatial domain methods; transform techniques; and CNN-based methods for image denoising. The first two categories are covered in this section, but the last type (CNN denoising methods) are included in Section 2.3.

#### Spatial domain methods

Spatial domain methods are traditional methods in image denoising, and they can be separated into two divisions: spatial domain filtering and regularisation-based techniques.

**Spatial domain filtering** The noise has high spectral frequencies. Therefore, spatial filters apply low pass filtering (LPF) on neighbouring pixels. Although filtering gives plausible results of denoised images, these images suffer from image blurring and lose sharp edges. There are many different types of spatial filters which are divided into two main categories:

- **Linear filters:** Mean filtering is used for Gaussian denoising; however, if it is implemented with high-level noise, it can give smoother images [64]. Wiener filtering [19] has been applied to overcome the limitation of mean filtering; nevertheless, it also

produces blurred edges. The linear filters generally fail to maintain the textures of the image.

- **Non-linear filters:** These include median filtering and weighted median filtering [182], which have the advantages of edge-preserving and suppressing noise efficiently. Also, bilateral filtering [157] is commonly used for noise reduction. A weighted average of intensity values of the neighbouring pixels is computed, and then the result substitutes the intensity value of each pixel. However, bilateral filtering takes time for implementation.

**Regularisation-based methods** These are also called variational denoising methods. One of the most common techniques used to solve the ill-posedness problem in denoising restoration is the regularisation approach and energy minimisation. The maximum a posteriori (MAP) probability estimate was the motivation for these techniques. The MAP probability estimate of  $y$  can be written as:

$$\hat{y} = \arg \max_y P(y|x) = \arg \max_y \frac{P(x|y) P(y)}{P(x)} \quad (2.4)$$

$$\propto \arg \max_y P(x|y) P(y) \simeq \arg \max_y \log P(x|y) + \log P(y) \quad (2.5)$$

where  $P(x|y)$  is a likelihood function of  $y$ , and  $P(y)$  is the image prior. The existing denoising techniques minimise the objective function and use the image priors which are represented by a regularisation item to estimating the denoised image  $\hat{y}$ . The objective function can be modelled as:

$$\hat{y} = \arg \min_y \|x - y\|_2^2 + \lambda R(y) \quad (2.6)$$

where  $\|x - y\|_2^2$  is the data fidelity term, and  $R(y) = -\log P(y)$  is the regularisation term.  $\lambda$  is the regularisation parameter ( $\lambda > 0$ ). The primary key in the regularisation denoising techniques is obtaining a proper image prior. Different types of image priors have been used successfully, such as gradient priors, sparse priors, Non-local Self-Similarity (NSS) priors and low-rank priors.

- **Gradient priors:** The Tikhonov method is the most straightforward regularisation, where its cost function involves the  $L_2$  norm. However, it produces over-smoothed image details [100]. Rudin-Osher and Fatemi (ROF) have proposed Total variation (TV)-based regularisation [131] for preserving the edges in the restored image and solving the problem of smoothness. The TV method uses the  $L_1$  norm, instead of the  $L_2$  norm. Although this method has accomplished success in noise reduction, the reconstructed images suffer from over-smoothed textures and losses of contrast [27, 130]. The regularisation term of the general TV ( $L^2$  TV) method can be written as follows:

$$R(y) = \|\nabla y\|_1 = \left\| \sum_{C=c} \sqrt{(D_i y_c)^2 + (D_j y_c)^2} \right\|_1 \quad (2.7)$$

where  $\nabla y$  is the gradient of the image which has the horizontal and vertical partial derivatives ( $D_i, D_j$ ) along with spatial coordinates  $i$  and  $j$  in the image,  $C$  is the number of image channels, where  $C = 3$  for RGB colour images and  $C = 1$  for grayscale images.

- **Non-local priors:** The advantage of local methods is low computational complexity. However, their performance is limited, especially when dealing with high-level noise, where the correlations of neighbouring pixels are severely disturbed by noise. The image comprises similar patches at different locations. Therefore, Non-local Self-Similarity (NSS) priors, which is considered the most successful priors for image reconstruction, have been used in other extended methods such as in [24, 62, 63]. Non-Local Means (NLM) [24] is a pioneer work which utilised the weighted filtering of the NSS prior. Although it produces improvement in image denoising more than the traditional NSS based methods, it can not preserve the image structure properly, leading to degradation in the visual image quality. The idea is that each pixel  $y_i$  is obtained as a weighted average of all pixels in the image  $y$ , where  $\text{NLM}(y_i)$  denotes the filtered value. Let  $\mathbf{y}_i$  and  $\mathbf{y}_j$  are image patches (a square neighbourhood of fixed

size) centred at  $y_i$  and  $y_j$  respectively.

$$\text{NLM}(y_i) = \sum_{j \in \mathcal{Y}} w_{i,j} y_j \quad (2.8)$$

$$w_{i,j} = \frac{1}{z_i} \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|_2^2}{h}\right); \quad z_i = \exp\left(-\frac{\|\mathbf{y}_i - \mathbf{y}_j\|_2^2}{h}\right) \quad (2.9)$$

where  $z_i$  is a normalising factor, and  $h$  denotes a filter parameter.  $w_{i,j}$  is the weight of  $\mathbf{y}_i$  and  $\mathbf{y}_j$  which depends on the similarity between the patches at locations  $i$  and  $j$ . The regularisation methods which have been developed using NSS prior can be defined as [50]:

$$R(y) = \sum_{y_i \in \mathcal{Y}} \|y_i - \text{NLM}(y_i)\|_2^2 \quad (2.10)$$

- **Sparse priors:** In denoising methods, sparsity prior has received a lot of attention. Sparse representation depends on the idea of learning a dictionary where each patch is represented as a linear combination of multiple vectors (image patches, also known as atoms) from the pre-defined dictionary [48]. The reason for naming this sparse representation that the representation uses a small number of atoms of the dictionary. An image is encoded over an over-complete dictionary  $D$  and  $L_1$ -norm sparsity regularisation, as follows:

$$\hat{\alpha} = \arg \min_{\alpha} \|x - D\alpha\|_2^2 + \lambda \|\alpha\|_1 \quad (2.11)$$

where  $x$  is a noisy image, a true image  $y = D\alpha$ ;  $\alpha$  is a matrix involving vectors of sparse coefficients. Instead of estimating  $y$  in equation 2.6, the equation 2.11 estimates  $\alpha$ . A dictionary that describes the image content efficiently can be obtained by using the K-singular value decomposition (K-SVD) algorithm [3, 48]. The model of a sparse representation can be learned from the corrupted image itself or from a dataset to learn the dictionary  $D$ . The idea of K-SVD for image denoising is learning the dictionary from a noisy image  $x$  by optimising:

$$\{\hat{y}, \hat{D}, \hat{\alpha}\} = \arg \min_{y, D, \alpha} \lambda \|x - y\|_2^2 + \sum_i \|R_i y - D\alpha_i\|_2^2 + \sum_i \mu_i \|\alpha_i\|_1 \quad (2.12)$$



where  $R_i$  is the matrix which extracts the image patch  $y_i$  from image  $y$  at location  $i$ . Although sparse representation methods such as K-SVD with learned dictionaries give better results than designed dictionaries [105], they are still local methods. Therefore, these methods are not concerned with the correlation between non-local information of an image; this means that if the level of noise is high, the restored image will not be useful as a result of the local information being severely disturbed. Thus, the sparse representation has been integrated with the NSS prior to take advantage of the non-local self-similarity properties of natural images, to average out the noise among analogous patches [44, 106]. This combination of the methods of non-local and sparse coding make the similar patches share the same dictionary atoms in their sparse decomposition. The non-local centralised sparse representation (NCSR) [44] is one of the pioneering works for noise reduction, which uses this combination. Although it works well to restore the textured regions and smooth, it suffers from the computational burden, so it is not practical in many applications.

- **Low-rank priors:** Low-rank representation (LRR) also represents a data vector as a linear combination of the other data vectors as in the sparse representation. In the case of the sparse representation model, it calculates the sparsest representation of each image patch as a vector individually. On the other hand, given a set of data vectors, the LRR method tries to find the lowest rank representation, which represents a collection of vectors jointly as a linear combination of the bases in a dictionary. The LRR method corrects the corruption in data better than the sparse representation in reducing the noise in images. Low-rank matrix factorisation methods are the first category of the low-rank techniques for the restoration of noisy data. These methods approximate a provided data matrix as a product of two low-rank matrices. For example, Ji et al. [75, 76] have proposed a robust patch-based restoration algorithm for removing noise from all frames of video depending on a proposed low-rank matrix recovery, wherein the similar patches are decomposed by low-rank decomposition. However, the disadvantage of these methods is that the rank must be given. If the value is too low, the method will produce a loss of detail, but if the value is too high, it will preserve the noise.

The second category of the low-rank techniques is nuclear norm minimisation (NNM), which aims to the lowest rank approximation  $\mathbf{y}$  of an observed matrix  $\mathbf{x}$ , which contains noisy patches. Equation 2.13 describes the NNM method which estimates the low-rank matrix  $\mathbf{y}$  [26]. In the NNM model, the weights for singular values are equal, and all singular values also have the same threshold. The NNM treated all singular values equally, therefore this limits its capability and flexibility in practice, because each singular value has a different level of importance. Therefore, Gu et al. [62, 63] have proposed the weighted nuclear norm minimisation (WNNM) model as shown in equation 2.14, where different singular values have different weights of  $w$ , which leads to better denoising performance than the NNMs. Generally, the low-rank minimisation techniques give better results than the previous methods of noise reduction, particular WNNM, but their computational cost is comparatively high.

$$\hat{\mathbf{y}} = \arg \min_{\mathbf{y}} \|\mathbf{x} - \mathbf{y}\|_F^2 + \lambda \|\mathbf{y}\|_* = \mathbf{U}\mathbf{D}_\lambda(\Sigma)\mathbf{V}^T; \quad \mathbf{D}_\lambda(\Sigma) = \max(\text{diag}(\sigma_i - \lambda), 0) \quad (2.13)$$

$$\hat{\mathbf{y}} = \arg \min_{\mathbf{y}} \|\mathbf{x} - \mathbf{y}\|_F^2 + \|\mathbf{y}\|_{w,*} = \mathbf{U}\mathbf{D}_w(\Sigma)\mathbf{V}^T; \quad \mathbf{D}_w(\Sigma) = \max(\Sigma - \text{Diag}(w), 0) \quad (2.14)$$

where  $\|\cdot\|_F^2$  indicates the frobenius norm,  $\|\mathbf{y}\|_* = \sum_i \|\sigma_i\|_1$  is the nuclear norm and  $\sigma_i$  is the  $i^{\text{th}}$  singular value of  $\mathbf{y}$ . The SVD of  $\mathbf{x}$  is  $\mathbf{x} = \mathbf{U}\Sigma\mathbf{V}^T$ , and  $\mathbf{D}_\lambda(\Sigma)$  is the singular value soft-thresholding operator. However,  $\|\mathbf{y}\|_{w,*} = \sum_i \|w_i\sigma_i\|_1$  is the weighted nuclear norm of  $\mathbf{y}$ , and  $w_i$  is the weight that is related to singular value  $\sigma_i$ .

## Transform techniques

Image denoising techniques started with the classical spatial-based methods, and have gradually developed into transform-based techniques. Transform techniques suggest that the characteristics of noise and the image information are different. The methods of transform have been developed from the Fourier transform method. Later various methods have been merged, such as wavelet methods, cosine transform, and block-matching & 3D filtering (BM3D).

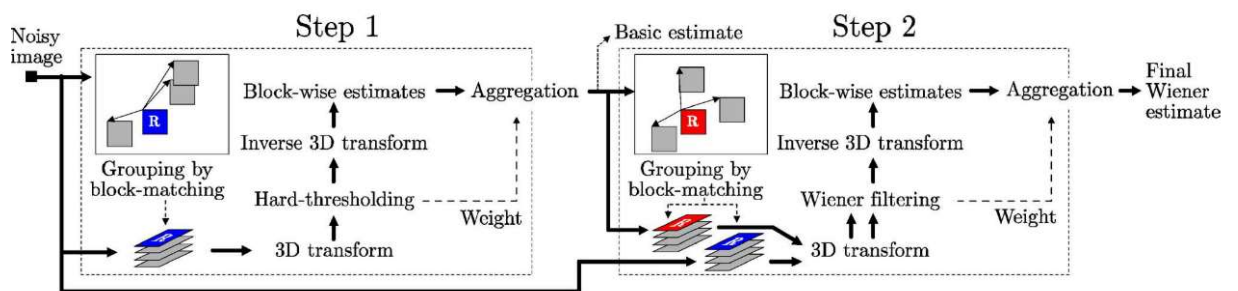
**Transform-based filtering methods** The corrupted image is transformed into a different domain. The procedure of denoising is then applied to the obtained image depending on the characteristics of the image information. The high-frequency parts, such as edges and texture, are indicated by high coefficients, while the smaller coefficients indicate the noise of the image. The most common reviewed transform in denoising is wavelet transform [108], where it has been demonstrated that it can successfully reduce noise while maintaining the image information [34, 107, 121]. The input data in the wavelet transform is decomposed into a scale-space representation. However, the drawback of the wavelet transform is that it depends on the wavelet bases selection. In the case of an inappropriate choice of wavelet bases, this leads to an inability to gain a good representation of the image given in the wavelet domain, which gives unsatisfactory denoising results.

**BM3D** This is the most common denoising technique, proposed by Dabov et al. [35]. It is considered as a powerful and efficient extension of the NLM method. BM3D comprises two main stages, as presented in Figure 2.1. Each stage consists of two steps, grouping and non-locally collaborative filtering process in the transform domain. The BM3D technique can be described as follows:

- **Grouping:** The similar 2D blocks are stacked into 3D arrays by block-matching (BM), which are known as groups. It is used to enhance the sparsity, and to enable the use of higher-dimensional filtering of each group.
- **Collaborative filtering:** This includes the following steps:

1. Employing a 3D linear transform to the group, to take advantage of the two types of correlation (between and inside the blocks). Each 3D group is transformed into the wavelet domain.
2. Shrinking the transform coefficients by using hard-thresholding or Wiener filtering to reduce the noise.
3. Inversing the 3D linear transform to output the estimations of all grouped blocks.

Finally, all estimated blocks are returned to their original positions to restore the whole image, by aggregating the overlapped estimates using weighted averaging. However, the higher the noise in the image, the less improvement is introduced from the BM3D method, and it produces artefacts.



**Fig. 2.1** Flowchart of the BM3D. The operations of the BM3D repeats for each group, the reference block marked by "R" is used as a reference to group the similar blocks for it, image taken from [35].

### 2.2.3 Enhancement: Super-Resolution (SR)

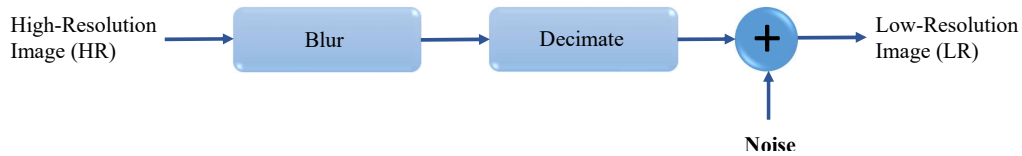
Image resolution is a critical problem in assessing the quality of different image acquisition and processing systems. The definition of image resolution is the precision of detail that can be obviously distinguished in the image. Several applications need the techniques of image processing for generating high-resolution (HR) images for detailed information, where an HR image is described as an image with a high pixel density. While image Super-Resolution (SR) methods are the methods which increase the image resolution by resampling the image at a higher sampling rate, also, these methods aim to recover a HR image without distortions. The recorded images sometimes suffer from some effects such as blur, noise and aliasing. The down-sampling factor discards some pixels of the image and leaves others unchanged. The prime purpose of SR algorithms is to produce HR images from under-sampled low-resolution (LR) images. However, the primary concern of image reconstruction is to restore degraded image such as blurred and noisy images, without changing the size of the image. Therefore SR reconstruction is not only up-sampling of under-sampled LR images to improve the quality of images but also tries to filter out distortions such as noise and blur [118]. Blur factor is caused by many reasons such as the inadequacy of the optical systems in image acquisition devices, motion blur, and lens blur which happens by the convolution of the image with a mask (matrix). The blur model commonly uses the 2D Gaussian distribution [115]. The most common practice merely uses interpolation techniques to increase the pixel density without interest in the resolution.

**Image degradation model for SR** The corrupted LR image can be obtained from the HR image according to the following equation 2.15. The pipeline of the degradation model is shown in Figure 2.2.

$$x = DB * y + \eta \quad (2.15)$$

where  $y$  is a HR image; the HR image indicates that the image does not display artefacts such as compression or blurry areas.  $x$  is a blurred LR image,  $B$  is a blurring factor,  $D$  is a down-sampling factor,  $\eta$  is noise operator and  $*$  denotes the convolution operation. Image restoration suffers from the ill-posed nature, where the solution of equation 2.15

with  $L_2$ -norm is not unique; to obtain the reconstructed SR image  $\hat{y}$ , the least square error should be solved to minimise the cost function:  $\arg \min_y \|x - DB * y\|_2^2$ .



**Fig. 2.2** Formation of the model of the LR image.

- Blurring (B):** This is a crucial aspect of the SR procedure because it removes high-frequencies and avoids resulting in an entirely aliased LR signal. In the model of SR, it may occur by defocusing or atmospheric blur, but also it corresponds to PSF (2D impulse response) of an optical device. There are different proposed strategies for blurring, a low-pass filter convolution kernel is most often used. The most applied blur kernels are the Gaussian kernel of different blurring levels  $\sigma$  and out-of-focus blur, which mimics the natural blurring kernel. Single-Image Super Resolution (SISR) can be divided into two different sub-tasks. Firstly, blind SR techniques suppose a blur kernel  $\sigma$  and HR image is unknown and attempt to restore both from the LR image. Secondly, non-blind SR techniques assume that the blur kernel is known and only recover the HR image from both the blur kernel and the LR image. Therefore, the issue is extended to obtain HR images from blurred LR images. Most of the blind SR methods concentrate on estimating the blur kernel and then performing non-blind SR methods for obtaining the SR image [126]. The success of SR reconstruction algorithms is based on estimating an accurate blur kernel, and to improve SR algorithms, research should put more emphasis on the restoration of the blurring [47].
- Down-sampling (D):** It is also known as sub-sampling or decimation. This operation leads to a spatial dimension shrinkage. Therefore, the SR process is distinguished from the related linear problems such as deblurring.
- Noise ( $\eta$ ):** This is considered to be an additional term which can be added, but a low level of noise is taken because the primary purpose is restoring the missing

information via blurring and down-sampling processes. Nevertheless, noise usually exists during saving or loading the original image.

**Application Areas** The SR process is a significant application in computer vision, and it has been shown that it is beneficial in many areas, for instance, image processing, medical imaging, security systems and forensic analysis, and many others. SR methods can be implemented as a pre-processing step, as in some applications as recognition and face verification systems, the resolution of images plays an important role. The SR process is used to enhance corrupted images and obtain HR images that help in detecting and recognising the objects better. LR images are a sensitive issue, mainly when related to the military or security applications. There are many reasons for the obtaining of these LR images, for example, atmospheric conditions, the distance between the subject and the camera, out-of-focus problems, the motion of the camera or the subject. The SR methods in these situations are beneficial to overcome such issues. Also, the SR methods can be used in image enhancement for natural images and for general-purpose to obtain HR images from the LR images to improve the quality of distorted images. These methods focus on the computation burden and the speed of implementation.

## A Review of SR Techniques

In the early 1980s, the task of image SR appeared for the first time in works by Tsai and Huang [159]. Since then, the problem has received much attention. Numerous SR techniques have been proposed to reconstruct HR images from their LR images. SR techniques aim to increase the image size and also restore the missing HR information. These techniques are divided into two main types, according to the problem modelling and the processed data. Firstly, traditional methods discussed the Multiple-Image Super-Resolution (MISR) task, where many observations of the same scene are available, with slight differences. These methods focus on reconstructing the HR image through aligning inputs and controlling noise [118]. Should only one LR image be used to obtain the HR image, the task is termed Single Image SR (SISR). The SISR techniques depend on external knowledge, where these techniques try to build and learn the relationship between the HR and LR images. The problem of SR is an inverse problem that is inherently ill-posed and poorly

conditioned because a single LR image can map to several HR images. This problem makes image restoration complicated, which means its solution is ambiguous or unstable. However, a stable solution can be computed through only the LR image and prior knowledge. This external knowledge helps to alleviate the ill-posedness problem and provides the prior information that outputs the final SR image.

In this literature review, we give an overview of SISR techniques. There are four essential categories of SISR methods: Interpolation-based methods, restoration-based methods, edge-based methods, and example-based (or patch-based/ learning-based) methods. The state-of-the-art performance has been attained by the example-based techniques, where these techniques learn prior information, which mitigates the multiple solutions (the ill-posedness) problem of SR methods [178]. The main algorithms of the SISR are divided as follows:

### **Interpolation-Based SR Methods**

Interpolation methods are the fastest and simplest ways to increase image size. Therefore, they have less computational complexity. These algorithms commonly used in SR or image processing are such as the nearest neighbour, bilinear and bicubic. Practically, the idea of the interpolation procedures depends on resampling to produce a new version of the image with different dimensions by using the adjacent neighbourhood pixels to estimate the missing pixel values. The new pixels in the interpolated (zoomed) image are merely the average of intensities of the neighbouring pixels, which cause the blur effect and lacking fine details [65]. These methods have some limitations, such as the inability to acquire the high-frequency information in the resampled image, where this information was removed during the low pass filtering. Also, the results usually have artefacts along the edges, and even seem over-smooth. The interpolated image is bigger size but presents blurred or aliased information of the LR image, hence the SR image reconstruction algorithms are employed to enhance image resolution and avoid the blurring effect. Further, in many SR techniques, the interpolation is broadly used to produce initial HR images. The nearest neighbour interpolation assigns the closest pixel value to the new treated sampling grid (or space) to resample an image. However, the bilinear interpolation method calculates the new pixel value on the new 2D grid by a weighted sum of the four closest neighbours



$(2 \times 2)$ . In image processing, the bicubic interpolation is a commonly employed procedure in image resampling, and it is used for both down-sampling and up-sampling. It takes 16 pixels into account ( $4 \times 4$ ) to compute a new pixel value on the sampling grid. The bicubic interpolated image is smoother than the resampled image by the nearest neighbour or bilinear procedures, and it also produces better visual quality and contains fewer artefacts.

### Reconstruction-Based SR Methods

Although the interpolation methods are straightforward and cost less, they yield over-smoothed images because they decrease the contrast (i.e., the blurring of edges), apparent artefacts and incapacity to generate fine details. Therefore these methods can not be used to recover the HR images from LR images in the SR techniques [118]. The reconstruction SR algorithms [11, 16, 36, 51, 113, 136, 146, 147] exploit the image properties as prior knowledge on HR images and impose a reconstruction constraint to producing the HR images from the LR one. The restoration constraint demands that the HR image through the down-sampling and the smoothing should reproduce the same LR image as far as possible. The heavy-tailed gradient distribution has been applied to image and video SR [136]. The sparse property is used in [83] to reduce the time complexity. The total variation (TV) regularisation has been employed for producing the HR images, for example, [11, 190]. However, these methods suffer from some limitations, such as the fact that they can produce artefacts in the HR result. Also, the imposed prior is not valid for arbitrary images [196].

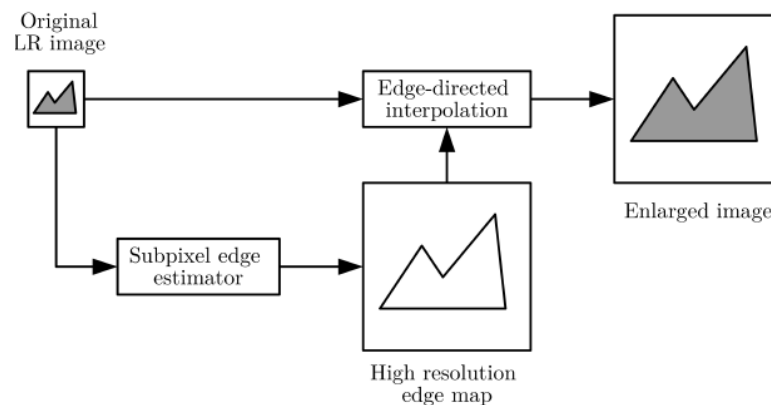
### Edge-Based SR Methods

Edges play a fundamental role in visual perception. Therefore, many works were proposed to obtain a high-quality image with sharp edges. These methods can be categorised as reconstruction-based (statistical) approaches.

**Edge-Directed Interpolation:** The EDI method [10] is used for preserving edges in the image after resampling, by adopting the orientation of the gradient maps, which depends on the underlying shapes being interpolated. EDI used an adaptive interpolation approach, which is proposed in [92]; this approach depends on nearest neighbour interpolation and

b-spline interpolation. The adaptive interpolation method is directed by edge presence, where it is calculated for each pixel as absolute differences with the adjacent pixels, and used to classify each pixel. This classification is used to prevent attributing the value of neighbouring pixels to the following edge direction. EDI has been proposed to generate HR edge maps that adapt the interpolation method and rely on the presence of edges. It uses an iterative algorithm between generating the HR image as described and LR image refinement from this generated image, as illustrated in Figure 2.3.

A similar approach has been proposed, termed new edge-directed interpolation (NEDI) [94], which models the interpolation methods utilising the relationship between the LR image and the covariance of the HR image. The NEDI method is not the same as the EDI method as it does not assume any prior knowledge except the LR image. It is just one-pass, rather than being an iterative projection as in the EDI method which suffers from computational complexity. In addition to the edge map, which the EDI procedure depends on, often suffers from a wrong decision. Although NEDI was proposed to reduce the computational burden of the covariance-based adaptive interpolation procedure, the NEDI procedure is still suffering from high computational time cost. Furthermore, inferring the hidden pixel values from the information existing in the LR images is still challenging in interpolation methods, which is a severely ill-posed problem.



**Fig. 2.3** Description of EDI methods. Figure taken from [10].

**Gradient Priors:** A smooth edge is one of the primary artefacts that result after interpolating LR images. Accordingly, instead of adapting the interpolation method as EDI procedure, various works have been proposed for focusing on refining the edge regions such as EDI but by modifying the gradient profile to give sharper images. The gradient profile is defined as a 1D profile along the gradient direction of the zero-crossing pixel in the image, while in a natural image, the gradient profile prior is considered as a parametric distribution defining the shape and the sharpness of the gradient profiles. In [51], the 1D gradient profile is modelled as a random variable  $\ell$  via a conditioned Normal distribution. At each pixel, a feature vector identifying its nearest edge is calculated. This is applied to form the SR image with adapted gradients by conditioning the distribution of  $\ell$ . This modelling leads to a Gauss-Markov Random Field model that can sample the whole SR image. But this method gives a sharp-edged image lacking texture and absence of fine detail. It is also computationally intensive.

Sun et al. [146] model the gradient profile as a 1D Generalised Gaussian Distribution (GGD), where this gradient profile prior learned from a large number of images can supply a constraint on image gradients when the HR image is estimated from the LR image. This method has fewer parameters than the method proposed in [51]. The HR gradient profile map  $\nabla y^T$  is generated by transforming the LR image. Then, the HR image is restored via a gradient descent over the energy  $E(y|x, \nabla y^T)$ , defined as:

$$E(y|x, \nabla y^T) = \|x - DB * y\|_2^2 + \|\nabla y - \nabla y^T\|_2^2 \quad (2.16)$$

where  $x$  is a LR image,  $y$  is a HR image,  $D$  is the down-sampling operation,  $B$  is a spatial filter (a Gaussian filter here),  $*$  is the convolution operator,  $\|x - DB * y\|_2^2$  is the reconstruction constraint in the image domain which measures the difference between the LR image and the smoothed and down-sampled version of HR image, and  $\|\nabla y - \nabla y^T\|_2^2$  is the gradient constraint in the gradient domain. Although the restored HR images have sharp, high-quality edges because the priors are mainly learned from edges, still edge priors are not efficient to model other high-frequency structures such as textures [178].

## Example-Based SR Methods

Example-based methods or Learning-based methods are methods which can learn mapping functions from a set of paired LR and HR training images. The example-based methods are divided into two main types. Firstly, internal example-based methods which employ a property of self-similarities of the same image [33, 59, 72]. The second type of example-based methods is external example-based methods which attempt to learn mapping functions between exemplar pairs of LR and HR patches [21, 28, 135, 154, 155, 180, 181]. The pipeline of most external example-based approaches is shared, where the focus is on learning and optimising the dictionaries or learning mapping functions. Despite some similarities, convolutional neural networks (CNNs) based techniques are different because they attempt to learn an end-to-end mapping function between LR and HR images without addressing explicitly the problem of selection and composition of dictionaries and / or manifolds. In the 2000s, these unusual methods for SR started to be used. The paper [54] had a significant influence on using such systems in the SR process. Also, the work of Freeman et al. [55] is considered a pioneer piece of work in the training-based method using patch pairs  $(x, y)$  for reconstruction, where the nearest neighbour (NN) of the input LR patch is found with its corresponding HR patch, and the target HR image is predicted via a Markov Random Field (MRF) model. On the other hand, this method depends on image patches directly. Therefore, it requires a large training set to learn patterns that can be faced in testing.

This term of example-based methods can incorporate several types of methods, where example-based SISR can be classified into four divisions; neighbour embedding and manifold learning, sparse dictionary learning, internal learning, and Artificial neural networks (ANNs). These different categories will be discussed in the following paragraphs and we will present some of the essential techniques for each kind. Notice that some procedures may be put in mixed divisions (e.g. sparse code prediction using ANNs).

- **Neighbour Embedding and Manifold Learning:** Chang et al. [28] proposed a method based on neighbour embedding (NE). It is a straightforward and effective method. This procedure is based on the assumption that LR patches can be represented by a linear combination of the  $K$ -nearest neighbours ( $K$ -NN). Further, it considers that HR and LR manifolds are formed from a set of HR and LR spaces, with similar local

geometry in the two different spaces. Euclidean distance is used to find the nearest neighbours. Also, first and second order derivatives are used as features,  $K$  assigned to equal 5, and the linear combination of the neighbours  $K$  are solved to compute the weights which minimise the reconstruction error. Although the results are less noisy than [55], the linear combination gives smoother results in some regions that make them look less realistic. Therefore, several methods have been proposed to improve the performance and reduce the computational burden. Bevilacqua et al. in [21] utilised different patch features; normalised (centred) luminance and derivative patch features. Also, they used a non-negative constraint in the LR neighbourhood estimation, which is derived from the Least Square problem (NNLS). The restored HR patches were better, as the non-negative weights calculated from the LR embeddings worked with more coherent manifolds.

Zeyde et al. [189] proposed sparse-representation modelling to restore an HR image from its blurred and down-sampled noisy version. Based on the dictionary construction from [189], two state-of-the-art methods have been proposed: Anchored Neighbour Regression (ANR) [154] and Adjust ANR (A+) [155]. However, this was done by utilising different techniques for regression and sparse search through the dictionary atoms (dictionary elements/class). The ANR method [154] has been proposed as a fast SR method. The  $K$ -nearest neighbours for each atom  $d_j$  in the dictionary are computed, to represent its neighbourhood. They depend on the correlation between the dictionary atoms to find the nearest neighbours. After defining the neighbourhoods, a separate projection matrix  $P_j$  is calculated for each dictionary atom which allows to providing the corresponding HR patch. The steps of the algorithm can be written as follows:

1. For learning, sparse dictionaries ( $D_l, D_h$ ) can be built utilising the K-SVD [3] learning approach of [189].
2. The  $K$ -nearest neighbours for each atom  $d_j$  in the dictionary are computed, then the projection matrix  $P_j$  is calculated as:

$$P_j = D_h (D_L^T D_L + \lambda I)^{-1} D_L^T \quad (2.17)$$

where  $D_h/D_l$  are the HR/LR dictionaries,  $\lambda$  is a regularisation parameter to alleviate the singularity (ill-posed) problems and stabilise the solution.

3. For testing, it is possible to compute a HR patch  $\hat{y}_i$  from a LR one  $x_i$  using:

$$\hat{y}_i = P_j x_i \quad (2.18)$$

Timofte et al. [155] proposed the Adjusted ANR (A+) method to improve the ANR performance and computing time. The A+ approach is built on the ANR method, but it utilises the full training material rather than learning the regressors from the dictionary. ANR and other sparse coding methods do not need the training sample after training the dictionaries. While in A+ procedure, the neighbourhood is computed for each atom on a pool of training samples. The A+ performs the same steps we mentioned before for the ANR method except the computing of the projection matrix, where they replaced the neighbourhood of atoms  $D$  with a matrix  $S$  containing the  $K$  training samples (i.e.,  $S_L = K \times D$ ). The projection matrix is computed offline. Therefore these operations do not consume time. The  $K$ -nearest pairs of examples for each atom  $d_j$  in the dictionary are selected from the training set  $(S_L, S_h)$ , then the projection matrix  $P_j$  is calculated as:

$$P_j = S_h (S_L^T S_L + \lambda I)^{-1} S_L^T \quad (2.19)$$

- **Sparse Dictionary Learning:** In sparse dictionary methods, the dictionaries of LR and HR samples are constructed to implement SR. Notice that the concept of 'neighbourhood' can exist in neighbour embedding methods and sparse dictionary methods, therefore some methods are not easy to classify. Dictionary learning depends on the idea of creating a set of atoms for LR and HR patches. The atom is created by taking a sample pair. Firstly, the LR image is split into LR patches, then each patch is represented utilising the LR dictionary elements, and its corresponding HR elements are utilised to obtain the restored HR image. In Yang et al.'s papers [179, 180], the dictionary, which is an external example-based SR method, was proposed to perform SR, where the sparse dictionary of LR and HR patches are utilised. The representation of the LR patch  $x_i$  is as a sparse linear combination of the LR dictionary atoms  $D_l$ .

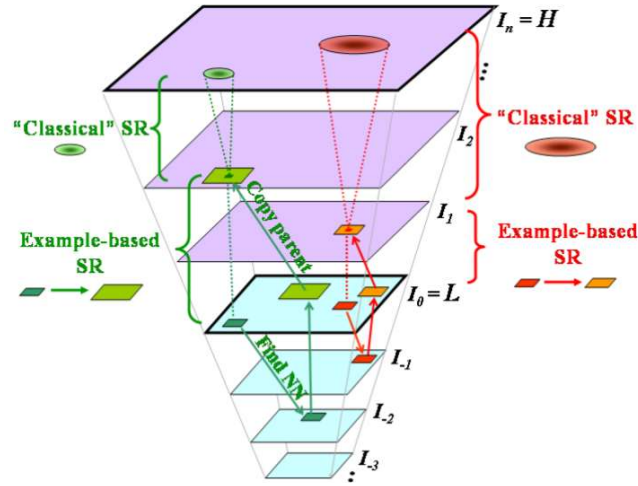
The restored HR patch  $\hat{y}_i$  is obtained by using its corresponding HR atoms  $D_h$ .

$$x_i = \alpha_i D_l \quad \& \quad \hat{y}_i = \alpha_i D_h \quad (2.20)$$

$$\arg \min_{\alpha_i} \frac{1}{2} \|x_i - \alpha_i D_l\|_2^2 + \lambda \|\alpha_i\|_1 \quad (2.21)$$

For each patch  $x_i$ , the optimisation problem in equation 2.21 should be solved to find the optimal value of sparse code  $\alpha_i$ .  $\lambda$  is added here to balance the role of the sparsity of  $\alpha$ , and  $\|\cdot\|_1$  is  $L_1$  norm.

- **Internal Learning:** Various methods exploit the internal patch similarity (redundancy) across scales and spatial dimensions for the sake of not relying on the dependency of the external database. The idea is that a group of LR and HR images is directly generated from the provided LR image. Therefore, the given images are supposed to be adequately large to include interesting and various content in order to observe it at different scales. Figure 2.4 shows the idea. The NLM method [24], non-local patch method, has been adapted to execute internal learning, as investigated in [46, 122]. The NLM procedures search non-locally (in a neighbourhood) for similar content for each position in the image using  $L_2$ -norm and implement a weighted sum based on the similarity. Moreover, various other methods have been proposed, such as [59, 177]. In these papers, the authors have exploited patch cross-scale similarities, where the LR patches in the input (given) LR image is searched for in the other LR scales. After finding a similar patch, its upper-scale counterpart is copied to a suitable location in the HR image. Gaussian Process Regression (GPR) [65] is employed to predict each pixel from its neighbours in the bicubic-interpolated image. This method also deblurs and produces HR images with sharp edges.



**Fig. 2.4** Internal learning procedure benefits from a multiscale analysis of the LR input image to generate example pairs for learning. This image is taken from [59].

### 2.3 Deep Learning for Image Denoising and SR Restoration

Deep learning (DL)[60, 90] is a branch of machine learning (ML) techniques, which automatically learns different representations of data. Recently, DL has recorded impressive results in computer vision problems using CNNs and in the speech recognition field utilising recurrent neural networks (RNN). A brief review of DL is presented in Appendix A. In many computer vision fields, the CNN based methods have demonstrated state-of-the-art results as in classification task [85], object detection task [58], face recognition [152] and image captioning [165]. Also, CNNs have been successfully applied to reconstruct images corrupted by JPEG compression [41, 149], as well as to related image enhancement problems, such as super-resolution [42, 81, 88, 95], motion deblurring [148, 150], non-blind image deconvolution [175], text deblurring [71] and image denoising [73]. The general model for CNN based reconstruction models can be formulated as:

$$\arg \min_{\Theta} C(y, \hat{y}); \quad s.t. \quad \hat{y} = F(x; \Theta) \quad (2.22)$$

Where  $C(\cdot)$  is the cost function, used for estimating the proximity between the restored image and the original image, and  $F(\cdot)$  is a CNN with a set of parameters  $\Theta$ . In this section,



we review the effective DL architectures, which have been proposed in recent years for denoising and SISR tasks.

### 2.3.1 Deep Architectures for Single Image SR (SISR)

Currently, the majority of the best performing state-of-the-art methods for SR tasks are based on deep neural networks (DNNs). Firstly, we review the Super-Resolution Convolutional Neural Network (SRCNN) proposed by Dong et al. [40, 42], which is considered to be benchmark architecture for SISR. SRCNN is a relatively simple network, which contains just three layers of CNN, as shown in Figure 2.5(a). It learns the mapping between the LR and HR images in an end-to-end manner. Each layer executes a specific function of non-linear transformation; these functions are patch extraction, non-linear mapping and reconstruction. The SRCNN model is trained on luminance components, as in many traditional methods. The filter size (and the number of feature maps) of each layer can be written as follows:  $9 \times 9$  ( $1 \times 64$ ),  $5 \times 5$  ( $64 \times 32$ ),  $5 \times 5$  ( $32 \times 1$ ). For optimising SRCNN, the mean square error (MSE) is employed as the loss function. Despite the simplicity of SRCNN, it has surpassed the other traditional methods, which may be attributed to the capability of CNN to learn efficient representations in an end-to-end manner using Large datasets. However, SRCNN does suffer from a number of issues which has led to researchers proposing many different architectures to avoid these problems and improve performance, e.g. [81, 82, 87, 97, 99, 139, 167]. One of this that SRCNN utilises the bicubic LR images as inputs to estimate the SR outputs. Therefore, SRCNN suffers from some drawbacks because it uses interpolated inputs, for example, it is very time-consuming, and the interpolated inputs provide smoother detail that may affect the final image structure estimate. Therefore, there are CNN architectures have been proposed to solve this task by directly using the LR inputs and up-sampling them within CNN.

Dong et al. have proposed the Fast SRCNN (FSRCNN) model through redesigning the SRCNN, where they used a compact hourglass-shape CNN architecture [43]. FSRCNN employs the deconvolution operation only at the final mapping layer to increase the resolution of the LR image, which reduces the computation burden. In CNN architecture, deconvolution operation (or transposed convolution) is the opposite concept of the convolution operators (e.g., filtering, pooling), where the former is used as an up-sampling operation,

in contrast, the latter is used to down-sample [187]. Instead of using the deconvolution layer, which explicitly enlarges the LR feature maps to increase the resolution from LR to HR, Shi et al. [139] proposed the ESPCN model, which implicitly learns the processing for SR by using an efficient sub-pixel convolution layer only at the end of the network, as shown in Figure 2.5(b).

**Deeper Architectures for SISR** The SRCNN model that includes just three layers has been followed by various deeper and more complex networks by increasing its width or depth to enhance the performance of the SR despite multiple training difficulties, such as [81, 97, 123, 151, 158, 195]. Kim et al. [81] proposed a deep network for SISR, called the very deep super-resolution (VDSR) network. It contains twenty convolutional layers of VGG net [141], which uses the smallest filter size ( $3 \times 3$ ), as shown in Figure 2.5(c). The VDSR model is applied to the bicubic LR inputs as the SRCNN. SRCNN is a direct mapping function between the inputs and the HR. However, VDSR learns the mapping function from the interpolated LR inputs to the residual between the inputs and the HR, which led to improving the performance.

At the same time, they noticed a similarity in the convolutional kernels of the non-linear mapping part of the VDSR model. Therefore, they proposed another architecture named a deeply-recursive convolutional network (DRCN) [82] to reduce the parameters used in VDSR. They have replaced multiple layers of VDSR with a recursive convolutional layer, as shown in Figure 2.5(d). They applied different strategies to overcome difficulties of training the depth and the recursive network. They have found that multi-supervised training in DRCN is essential, as it uses intermediate representation to reconstruct intermediate HR outputs. The final output fuses all intermediate HR outputs by using a listing of trainable positive weight scalars. However, the drawback of this strategy is that after training, the list of weight scalars will not change with different inputs.

Residual Networks (ResNet) [68], that is one of the very deep architectures based on skip-connection, has attained state-of-the-art performance in several tasks. Therefore, Ledig et al. [91] have proposed utilising ResNet for SISR called SRResNet. It is comprised of 16 residual units; each unit made up two non-linear convolutions followed by batch normalisation (BN) with residual learning; it is shown in Figure 2.5(e). Lee et al. [97] proposed an enhanced deep super-resolution network (EDSR). In the reconstruction tasks,

there is a strong relationship between the input and the output (i.e., an image-to-image relation). Therefore, the authors have removed the BN used for normalising the features from the residual unit of the SRResNet model, where BN layers rid networks of range flexibility. They showed that removing BN enhances performance. Moreover, the depth of EDSR was increased, as shown in Figure 2.5(f), which contains 32 residual blocks with 256 feature maps for each layer. This network includes 43 million parameters.

**CNN architectures combined with the SISR properties** Recently, some deep architectures have been proposed which are inspired by the representative methods for SISR. The DNNs have recorded tremendous successes depending on extracting beneficial representations using end-to-end methods. However, some recent research has demonstrated that using more information explicitly helps improve SISR performance. The blur kernel and noise are two critical factors in SISR's success. Therefore, Zhang et al.[194] took these factors into account. They used the LR images concatenated with degradation maps as inputs to the super-resolution network to treat multiple degradations (SRMD). For degradation, the anisotropic Gaussian kernel was used for blur kernel and the AWGN for noise. They have used a dimensionality stretching strategy to create the degradation maps. However, the concatenating between the degradation maps and the image used as input of the network only affects the first layer. By using deeper architectures, the deeper layers are not affected by the information of the degradation maps as in the first layer [61].

In IR tasks (e.g., deblurring, denoising and SR tasks), priors are considered to be key in the efficient reconstruction of algorithms to handle the different inverse problems flexibly. Notably, the regularisation part in equation 2.2 is also known as the prior part from the Bayesian view. There is an approach called the Plug-and-Play (P&P) method [163] suggests splitting equation 2.2 into two parts: a data part and a prior part with variable splitting procedures. Afterwards, the prior is replaced by existing denoising algorithms for solving any inverse problems. Recently researches showed that DNN could be used as an efficient denoiser prior such as in [156, 192]. In this case, the DNN acts as a pre-processing step before reconstruction. The IRCNN model in [192] has been used for denoising and SISR tasks.

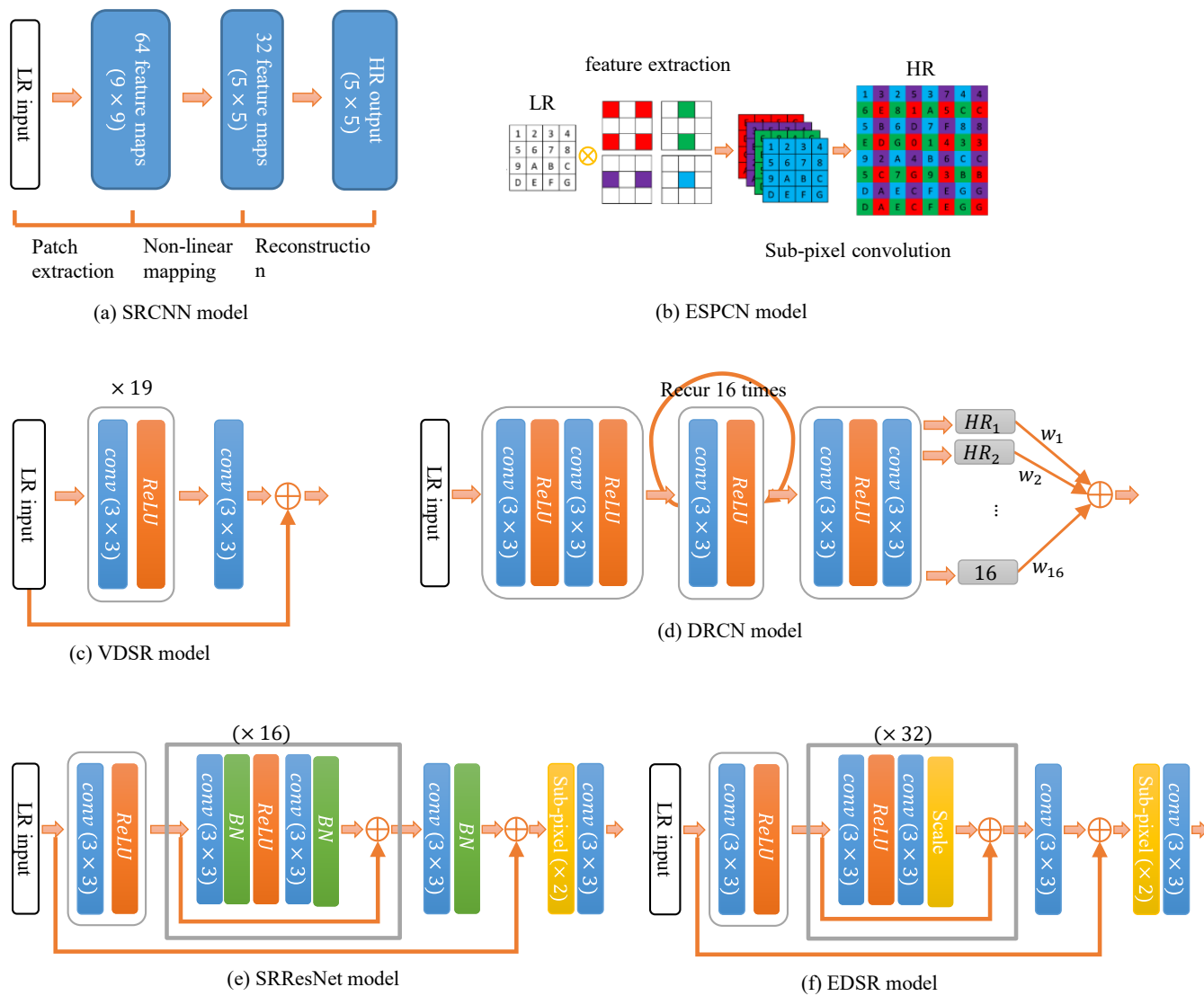


Fig. 2.5 Sketch of some deep structures for SISR

### 2.3.2 Deep Architectures for Denoising

Several attempts have been made to use DNNs for image denoising. These techniques can be separated into two classes, namely multi-layer perception (MLP) and DL techniques. The auto-encoders methods [164, 172] were utilised to remove noise from images. The MLP model [25] has been successfully applied for image denoising. Besides, the trainable non-linear reaction-diffusion (TNRD) model [29] proposed by Chen and Pock is a feed-forward deep network with a fixed number of gradient descent inference steps. The MLP and TNRD models accomplished promising performance, which can compete with BM3D. However, state-of-the-art DL denoising methods [191–193] depend on CNNs because of their outstanding denoising ability.

The feed-forward denoising convolutional neural network (DnCNN) was proposed by Zhang et al. [191]. Residual learning was utilised to learn a mapping function  $\hat{y} = F(x; \Theta_\sigma)$ , and batch normalisation was also used to accelerate the training process. The DnCNN model is trained on noisy images with a fixed noise level of  $\sigma$ . Therefore, the trained network is not proper for other noise levels. Consequently, another CNN has been introduced to be flexible in handling the different noise levels, called fast and flexible denoising CNNs (FFDNet) [193]. It is expressed as  $\hat{y} = F(x, M; \Theta)$ ; where  $M$  is a map of the noise that was designed to be input with noisy input  $x$ . The parameter set  $\Theta$  is invariant to the noise level. Therefore, this model works well when the noise level in the noisy input and the map of noise level are matched. However, the time complexity of the learning procedure is very high.

### 2.3.3 Challenges and Trends

DL algorithms have provided improved performance (i.e, improved outcomes) in image restoration compared to traditional methods. However, there remain various challenges that need new trends to be followed. Recently, the advanced SISR deep architectures have accomplished high performance, however, with a vast amount of parameters. Therefore the new trend is to design lighter models with fewer parameters with little or no degradation in the performance [183]. Besides, SISR with unknown degradation is considered a prominent challenge for the SR task, where most of the DL algorithms are focused on obtaining estimated HR images from LR images without considering other degradations, which is

regarded as the primary key of the SR task. Generally, the biggest challenge in DL is the need for a theoretical understanding of the deep architectures and how and why these architectures work. Thus, until now, we have dealt with it as a black box, and we concentrate only on the performance to assess its success.

## 2.4 Evaluation of Restoration

### 2.4.1 Metrics

Processes such as SR cause distortion, for instance, noise and artefacts like ringing. Similarity measures are used for image quality assessment (IQA) and defining the quality of a reconstructed image as to how similar it is to its actual image. Image quality assessment can be split into subjective and objective methods. While subjective methods depend on human judgement, objective methods depend on comparisons applying explicit numerical measures. There are many evaluation metrics used in the SR task. However, the two different metrics that are most often utilised to compare the performance of results are a peak signal-to-noise ratio (PSNR) and a structural similarity (SSIM) [169]. Besides, there is another group of measures that achieves a high correlation with the human perceptual scores [178], namely the information fidelity criterion (IFC) [137] and visual information fidelity (VIF). However, these measures don't respond to the structural information of the image explicitly.

- **The Peak Signal-to-Noise Ratio (PSNR):** The PSNR is a prevalent quality measure in image processing, which depends on the Mean Squared Error (MSE). It is computed by averaging the squared intensity differences between the original image and the reconstructed SR image pixels. PSNR takes into account the maximum value of the signal. There is an inverse relationship between PSNR and MSE, where the higher PSNR value corresponds to a lower error. The higher the value of PSNR, the higher the quality of the image is. Although PSNR metric provides poor performance in describing the image quality, and we are concerned with human visual perception in the image, it is considered the most widely used evaluation metric. This is attributed to the necessity to compare any work with literature works which depended on using PSNR to assess the performance. In addition to the lack of entirely accurate

perceptual metrics. Given the reconstructed image  $\hat{y}$  and the original image  $y$  whose size is  $M \times N$ , the MSE and PSNR are defined as:

$$\text{MSE} = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (\hat{y}(i, j) - y(i, j))^2 \quad (2.23)$$

$$\text{PSNR} = 10 \times \log_{10} \left( \frac{\max |\hat{y}|^2}{\text{MSE}} \right) \quad (2.24)$$

$$= 20 \times \log_{10} \left( \frac{\max |\hat{y}|}{\sqrt{\text{MSE}}} \right) \quad (2.25)$$

$$= 20 \times \log_{10} (\max |\hat{y}|) - 10 \times \log_{10} (\text{MSE}) \quad (2.26)$$

PSNR is defined by the MSE and the maximum signal value,  $\max |\hat{y}|$ . PSNR is expressed in decibels (dB). When providing an unscaled input,  $\max |\hat{y}| = 255$ , then  $20 \times \log_{10} 255 = 48.1308036087$ . However, if the input is scaled, then  $\max |\hat{y}| = 1$ . Therefore  $20 \times \log_{10}(1) = 0$ . Thus, that component is removed completely and it only computes the remaining MSE component, so the PSNR for scaled input becomes as follows:

$$\text{PSNR} = -10 \times \log_{10} (\text{MSE}) \quad (2.27)$$

- **The Structural SIMilarity Index (SSIM):** Although the PSNR is efficient and easily implemented, it does not necessarily represent what is observed by the human visual system. The SSIM index is created to improve conventional measures, which is confirmed to be incompatible with human eye perception. This index is interested in the highly structured characteristics of the natural image, wherein the visual scene of the strong neighbourhood dependencies carry essential information about the structures of the objects. SSIM is a criterion for measuring the similarity between two images [169]. For two images,  $x$  and  $y$  with the same size  $M \times N$ , the SSIM is calculated as:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (2.28)$$

Where,

$$\mu_x = \frac{1}{MN} \sum_{i=1}^{MN} x_i, \quad \mu_y = \frac{1}{MN} \sum_{i=1}^{MN} y_i, \quad \sigma_x = \sqrt{\frac{1}{MN-1} \sum_{i=1}^{MN} (x_i - \mu_x)^2}$$

$$\sigma_y = \sqrt{\frac{1}{MN-1} \sum_{i=1}^{MN} (y_i - \mu_y)^2} \quad \text{and} \quad \sigma_{xy} = \frac{1}{MN-1} \sum_{i=1}^{MN} (x_i - \mu_x)(y_i - \mu_y)$$

$\mu_x$  and  $\mu_y$  are the local sample means of  $x$  and  $y$ , respectively.  $\sigma_{xy}$  is the sample cross-correlation of  $x$  and  $y$ , and  $\sigma_x$  and  $\sigma_y$  are the local sample standard deviations of  $x$  and  $y$ , respectively. Constant numbers  $C_1$  and  $C_2$  are used to avoid instability, so that near-zero sample means, variances or correlations do not lead to numerical instability.  $\mu_x^2 + \mu_y^2$  and  $\sigma_x^2 + \sigma_y^2$  are very close to zero. The SSIM index is a value from 0 to 1, where 0 means there is no correlation between two images, and 1 means they are the same. SSIM is a more effective signal fidelity measurement than MSE [168], because MSE treats all image pixels equally, and it does not account for content-dependent variations in image fidelity.

#### 2.4.2 Dataset and benchmarks for evaluation

The training dataset has a significant influence on the performance of the DL algorithms [42, 139]. Usually, the large dataset in training leads to better results, especially on the deep structures. The different models have been trained using different datasets. For example, a relatively small training dataset that contains 91 small-sized RGB flower images from Yang et al. [180] was utilised to train some DL models. Some research used 291 training images, where they use 200 images from Berkeley Segmentation Dataset (BSD) [110] and the 91 images from Yang et al. [180]. Also, a much larger set of images was derived from ImageNet [38]. Besides these, the recently published DIVERse 2K (DIV2K) dataset [2] contains 800 RGB images for training. While, for the testing stage, the two standard benchmark image sets Set5 [21] and Set14 [189] are mostly used, they consist of five images and 14 images respectively. Also, B100 from the Berkeley segmentation dataset [110], and Urban100 [72] are utilised and both contain 100 images. Additionally, there is the newly proposed testing



dataset DIV2K [2], which includes 100 test images. Table 2.1 shows the image datasets commonly utilised in the SR task, and it indicates their average resolution and number of images.

**Table 2.1** Some of the popular image datasets which are used for super-resolution benchmarks.

Dataset	Number of images	Average resolution
Training 91 [179]	91	(264 × 204)
BSD300 [110]	300	(435 × 367)
DIV2K [2]	1000	(1972 × 1437)
Set5 [21]	5	(313 × 336)
Set14 [189]	14	(492 × 446)
Urban100 [72]	100	(984 × 797)

## 2.5 Conclusion

This chapter addresses the literature review of the various methods that have been performed in the task of restoration, especially for denoising and SISR. Several factors cause distortions in digital images during the imaging process, and these factors can introduce several types of image degradations, for example, noisy, blurred, down-sampled or all of these together. Therefore, image restoration is vital in the computer vision field. Example-based SISR methods give good results compared to other methods, because these methods automatically learn the relationship between the LR and the HR images from example pairs. The state-of-the-art results have been introduced by DNNs, using a large amount of data to train the networks. In the next chapter, we present the different CNN architectures that we propose in this thesis. These structures have been employed in image restoration applications such as SISR, denoising and artefact reduction tasks.



# Chapter 3

## Convolutional Neural Networks (CNNs)

In this chapter, we introduce the different convolutional neural networks (CNNs), which we have proposed in our work. These networks have been used in image reconstruction applications such as super-resolution application (SR), as well as denoising and artefact reduction applications. We begin by briefly reviewing the SRCNN architecture proposed by Dong et al. [42] and its optimisation procedure in Section 3.1. The SRCNN recovers the unknown high-resolution (HR) image from its corresponding low-resolution (LR) image. In Section 3.2, we introduce different architectures using concatenation layers for image restoration. In Subsection 3.2.1, we introduce two different architectures for reducing image artefacts; Direct Architecture Compression Artefacts Removal (DA-CAR) network and Skip Architecture Compression Artefacts Removal (SA-CAR). To tackle efficiently simultaneous deblurring and SR, we propose a novel DeBlurring Super-Resolution Convolutional Neural Network (DBSRCNN) architecture with concatenation layer, which is presented in Subsection 3.2.2. Then, in Section 3.3, we extend the DBSRCNN architecture, proposing a new architecture called DeBlurring Super-Resolution (DBSR) with more enhancing layers. We also provide the program implementations of the architectures online. In addition, we suggest replacing some convolutional layers in DBSR architecture with harmonic blocks to investigate the performance of the network in the frequency domain instead of the spatial domain; this network is named as Harm-DBSR. The Harm-DBSR architecture is presented in Section 3.4.

### 3.1 Super-Resolution CNN (SRCNN)

Many researchers around the world have studied the problem of obtaining a single image super-resolution (SISR) from a low-resolution (LR) image. There are many traditional methods for obtaining SR, but alternative approaches have been proposed by using DNNs which try to learn an SR image from a LR one. The model used here is named as a super-resolution convolutional neural network (SRCNN) [42]. SRCNN is considered as the first CNN model for the task of SR. SRCNN directly learns end-to-end mapping between the LR and HR images. SRCNN estimates a mapping function  $F$  that takes the LR image as the input and produces the SR image. Its structure contains only three layers. Although the SRCNN is fully feed-forward, its original implementation takes a long time (around three days) in training. We have reimplemented a code which accomplishes the training in just eight minutes.

#### 3.1.1 SRCNN Architecture

The architecture of SRCNN [42] is a relatively small network that has 8,032 parameters. It is composed of two hidden layers in addition to the input and the output layers, as is apparent in Figure 3.1. The objective is to learn the mapping function  $F$ , which performs three operations: patch extraction, non-linear mapping and reconstruction. The structure of the SRCNN network is specified below:

- **Input layer:** the input  $\mathbf{x}$  is a two-dimensional representation of the sub-image with  $c = 1$  for grey level image (channel Y) and  $c = 3$  for colour images (YCbCr).

$$F_0(\mathbf{x}) = \mathbf{x} \quad (3.1)$$

- **Patch extraction and representation:** The first convolutional hidden layer which extracts overlapping patches from the input sub-image and represents each patch as a high-dimensional vector. It uses a Rectified Linear Unit (ReLU) activation function  $F_1$ , kernel size  $f_1 = 9$  and contains  $n_1 = 64$  feature maps.

$$F_1(\mathbf{x}) = \max(0, W^{(1)} * \mathbf{x} + b^{(1)}) \quad (3.2)$$

where  $W^{(1)}$  contains  $n_1$  filters of size  $c \times f_1 \times f_1$  that output  $n_1$  feature maps.  $b^{(1)}$  is a  $n_1$ -dimensional bias vector, where each element is related to a kernel, and '\*' denotes the convolution operation.

- **Non-linear mapping:** or second-order mapping, it is the second convolutional hidden layer. It maps each high-dimensional vector of the previous hidden layer to another high-dimensional vector, which is the representation of a high-resolution patch. ReLU activation function is employed with  $n_2 = 32$  feature maps and filter size  $f_2 = 1$ .

$$F_2(\mathbf{x}) = \max(0, W^{(2)} * F_1(\mathbf{x}) + b^{(2)}) \quad (3.3)$$

where  $W^{(2)}$  involves  $n_2$  filters of size  $n_1 \times f_2 \times f_2$  to produce  $n_2$  feature maps.  $b^{(2)}$  is a  $n_2$ -dimensional bias vector.

- **Reconstruction operation:** The output layer. This is the last operation in the network. It is a convolutional layer that outputs the reconstructed SR image by aggregating the patch-wise representations.

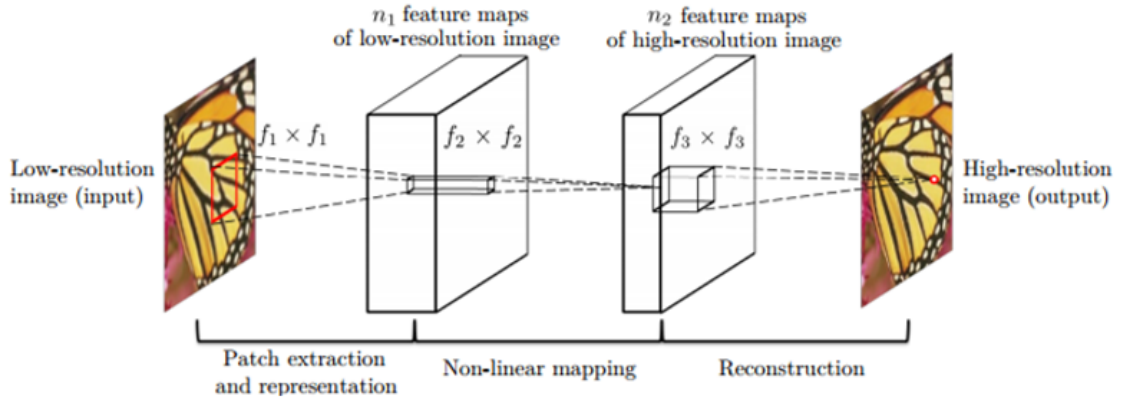
$$F(\mathbf{x}) = W^{(3)} * F_2(\mathbf{x}) + b^{(3)} \quad (3.4)$$

where  $W^{(3)}$  includes  $c$  filters of size  $n_2 \times f_3 \times f_3$  with filter size  $f_3 = 5$ .  $b^{(3)}$  is a  $c$ -dimensional bias vector; it is associated with the number of image channels, and we have used  $c = 1$ .

Finally, the structure of the standard version of SRCNN (or (9-1-5)SRCNN) can be written as network with three layers (the filters size)(the number of feature maps): (9-1-5)(64-32-1).

### 3.1.2 SRCNN Optimisation

In each layer, the filter weights are initialised by the initialisation method described in He et al. [67], considered a robust method for ReLU. ReLU is used as the activation function. The biases are set to 0, and the learning rate is 0.001. The aim is to recover an SR image  $F(\mathbf{x})$  from LR  $\mathbf{x}$  that is as similar as possible to the original HR image  $\mathbf{y}$ . An estimation of the optimal network parameters  $\Theta = \{W^{(1)}, W^{(2)}, W^{(3)}, b^{(1)}, b^{(2)}, b^{(3)}\}$  is required to learn the



**Fig. 3.1** The SRCNN architecture: the network contains three-layers. Given a LR image  $\mathbf{x}$ , the first convolutional layer extracts  $n_1$  LR feature maps. Then, the second convolutional layer non-linearly maps the LR feature maps to  $n_2$  HR feature maps. Finally, the output layer produces the final SR image  $F(\mathbf{x})$ , image taken from [42].

end-to-end mapping function  $F$ . This is accomplished by minimising the cost between the reconstructed images  $F(\mathbf{x}, \Theta)$  and its original HR images  $\mathbf{y}$ . The MSE function  $C(\Theta)$  is used as the cost function, which is used to compare the squared error between the reconstructed image and the original image:

$$C(\Theta) = \frac{1}{m} \sum_{i=1}^m \|F(\mathbf{x}_i; \Theta) - \mathbf{y}_i\|^2 \quad (3.5)$$

This cost function is minimised using Adam [84], which is used to optimise the network to find optimal weights and for faster convergence rates.

### 3.1.3 Computation Time

In DL, achieving the best performance is sometimes regarded as an issue because the training time may extend to weeks or months to train one trial. The standard (9-1-5)SRCNN takes roughly three days [40] to train the Caffe library depending on the number of epochs, where the increase in the number of epochs leads to an increase in training time. The Python implementation of SRCNN [153] uses Keras-1 with the Theano library as a backend.

However, we have reimplemented this code using Keras-2 with TensorFlow as backend [5], and the changes in the code have been undertaken accordingly [7]. Our implementation<sup>1</sup> relies on Adam optimisation to render the SRCNN training more computationally efficient. Adam is an improved version of gradient descent with adaptive learning rate and RMS propagation; therefore, it is computationally efficient. For more details, see Appendix A.4.6. The training time of the standard SRCNN in this implementation is 8.33 minutes with NVIDIA GTX 1050 GPU. The spec of the computer which we used in our experiment appear in Table 3.1. There are two versions of SRCNN. Dong et al. [42] indicate that the (9-5-5) SRCNN network achieved better performance than the (9-1-5) SRCNN network but at the cost of training time. When the filter size is increased from (9-1-5) to (9-5-5) the training time is also increased. In our Keras implementation of (9-5-5) SRCNN network, the training time was around 11 minutes. In this section, we have used the first version (9-1-5) SRCNN [40] because we worked on it before publishing version 2 [42], but in Chapter 4, we have used the first and the second versions in comparison.

**Table 3.1** Some differences aspects between the two different implementation.

	Characteristic	<b>Dong's Code</b> [40]	<b>Keras Code (Ours)</b> [5]
Computer Spec	Processor	Intel CPU 3.10 GHz	Intel i7 CPU 2.80 GHz
	Memory	16 GB	16 GB
	GPU	NVIDIA GTX 770	NVIDIA GTX 1050
Code	Implementation	Caffe Code	Keras-2 [31] with TensorFlow backend
	Optimiser	SGD	Adam
Training Time		Roughly 3 days	8.33 minutes

<sup>1</sup>DBSRCNN-Keras Code: <https://github.com/Fatma-ALbluwi/DBSRCNN>, 2018.

## 3.2 Introducing Concatenation

CNN can be shaped to different structures for deep learning problems such as direct architecture and skip architecture. In direct architecture, the image is transferred successively through layers from the input layer to the output layer. This approach is used in many tasks of low-level image processing [40, 41, 71, 150], and it often results in high performance. On the other hand, a skip architecture interrupts the successive flow of data by allowing skip connections to merge layers that have undergone different processing paths. There are several main merger operations, such as concatenating, adding, subtracting, etc. The skip architecture may alleviate some specific problems of propagating information through deeper networks, such as, specifically, vanishing and exploding gradients [68, 101, 149].

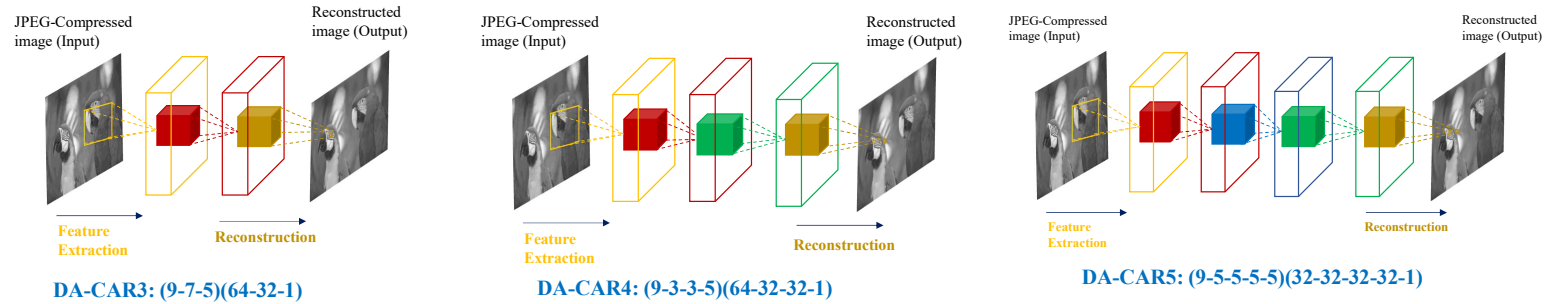
### 3.2.1 Compression Artefacts Removal (CAR) Networks

The proposed CNNs are used to learn an end-to-end mapping  $F$  which takes corrupted images  $\mathbf{x}$  as input and outputs the restored images  $\hat{\mathbf{x}} = F(\mathbf{x})$ , without pooling, deconvolution or full-connected layers. We only resort to convolutional layers and element-wise activation functions in the form of leaky rectified linear unit (leaky ReLU) [104] with hyper-parameter 0.3. In the absence of pooling and deconvolution layers, the filter size is the only factor that affects the output image size. Zero padding should be used to compensate for the size decrease if one wants the output to be the same size.

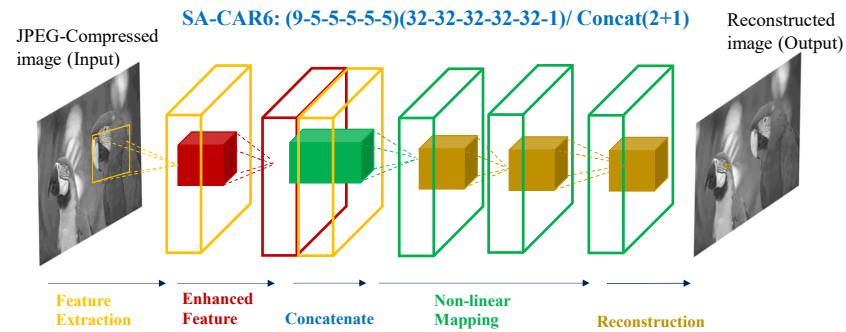
#### Direct and Skip CAR (DA-CAR & SA-CAR) Architectures

We consider both types of CNN architectures. We propose two different CNN structures [6] which we refer to as Direct Architecture Compression Artefacts Removal (DA-CAR) network (Figure 3.2(a)) and Skip Architecture Compression Artefacts Removal (SA-CAR) as shown in Figure 3.2(b). For DA-CAR version, we experiment with three-, four- and five-layer architectures (DA-CAR3, DA-CAR4 and DA-CAR5, respectively). For the skip-based architecture we consider six layers (SA-CAR6), the third layer is concatenating activation (feature maps) between the first layer and the second layer (2+1).





(a) DA-CAR Architectures.



(b) SA-CAR Architecture.

**Fig. 3.2** The proposed structures of the CNN networks: direct and skip architectures. In the direct architecture (DA-CAR), the information directly transfers from the input to the output, while in the skip architecture (SA-CAR) there are some merged layers. We have illustrated the structure of each network as follows: The name of network + the number of layers (the size of filters in each layer)(the number of feature maps in each layer). For instance: DA-CAR3(9-7-1)(64-32-1). We implement the standard JPEG compression method with different JPEG quality factors ( $q = 10, 20$ ) using MATLAB JPEG encoder. We focus on the restoration of the luminance Y channel in the YCbCr space that has been JPEG compressed.

**Formulation:** The first hidden layer (patch extraction and representation) extracts the overlapping patches from the input images and then represents each patch as a high-dimensional vector. The last layer (the reconstruction layer) assembles the restored image by aggregating the patch-wise representations. The structure of a CNN is as follows:

$$F_0(\mathbf{x}) = \mathbf{x} \quad \text{The input image.} \quad (3.6)$$

$$F_i(\mathbf{x}) = \max(0, W^{(i)} * F_{i-1}(\mathbf{x}) + b^{(i)}) \quad i \in \{1, 2, \dots, L-1\} \quad (3.7)$$

$$F(\mathbf{x}) = W^{(L)} * F_{L-1}(\mathbf{x}) + b^{(L)} \quad \text{The output image.} \quad (3.8)$$

where  $\mathbf{x}$  is the compressed image and  $W^{(i)}$  and  $b^{(i)}$  are the filters and biases.  $W^{(i)}$  is comprised of  $n_i$  filters which support  $n_{i-1} \times f_i \times f_i$ ;  $n_0$  is the number of channels in the input image and  $f_i$  is the filter size.  $F_i(\mathbf{x})$  are the feature maps in the layer  $i$ , and  $F(\mathbf{x})$  is the restored output image which has the same size as the input image.  $L$  is the number of layers in the CNN architecture. The activation function type employed is Leaky ReLU.

### DA-CAR & SA-CAR Optimisation

The filter weights in each layer are initialised by a robust initialisation method proposed in [67] for ReLU. The aim is to recover images  $\{F(\mathbf{x}_i)\}$  from the corresponding JPEG-compressed image  $\{\mathbf{x}_i\}$  that are as similar as possible to the set of ground truth images  $\{\mathbf{y}_i\}$ . The estimation of  $\Theta = \{W^{(1)}, W^{(2)}, \dots, W^{(L)}, b^{(1)}, b^{(2)}, \dots, b^{(L)}\}$  is required to specify the end-to-end mapping function  $F$ . The cost minimisation between the reconstructed images  $F(\mathbf{x}, \Theta)$  and its original images  $\mathbf{y}$  is applied. The MSE function  $C(\Theta)$  is employed as the training loss function:

$$C(\Theta) = \frac{1}{m} \sum_{i=1}^m \|F(\mathbf{x}_i; \Theta) - \mathbf{y}_i\|^2 \quad (3.9)$$

The minimisation is performed using Adam [84], which is a technique to optimise CNNs at faster convergence rates. For CNN architectures the training time is typically proportional to the size of training samples and the total parameter count  $W_{total}$ :

$$W_{total} = \sum_{i=1}^L n_{i-1} \times n_i \times f_i^2 \quad (3.10)$$

### 3.2.2 De-Blurring Super-Resolution CNN (DBSRCNN)

We have proposed a new architecture for the de-blurring SISR task: the de-blurring super-resolution convolutional neural network (DBSRCNN) [7]. This architecture is motivated by the SRCNN network. SRCNN uses one layer to extract features, and it contains 64 feature maps. However, we propose to use two layers to obtain the features; each layer involves 32 feature maps. The first layer is to extract the low-level features, and the second layer is to enhance the low-level features from the previous layer. Then, the first and second layers are merged by using a concatenation operation before mapping them.

#### DBSRCNN Architecture

The proposed network aims to learn an end-to-end mapping  $F$ , which takes the blurred LR image  $\mathbf{x}$  as input, and directly provides the deblurred HR reconstruction  $F(\mathbf{x})$ . This network includes four convolutional layers, each one is responsible for a particular task, in addition to the concatenation layer, as demonstrated in Figure 3.3.

#### The operations of the proposed network can be described as follows:

- The first layer is *feature extraction*, which extracts the low-level features; it contains 32-feature maps (or 32 filters) with filter size  $9 \times 9$ .
- The second layer is the *feature enhancement layer*, which provides enhanced features from low-ones (the output of the first layer); also it contains 32-features maps with filter size  $5 \times 5$ .
- The third layer is the *concatenation layer*, which merges the first two layers to form a new layer that contains low-level features and enhanced features together. The concatenated features create a merged vector with low-level and enhanced features together.

The merged layer contains 64-features maps or 32-features maps according to the merge procedure. There are many operations of merge which perform specific tasks on the inputs (feature maps). Such operations include summation, maximum, subtraction, averaging, multiplication and concatenation. All these operations take

the same size of inputs and return the same shape. However, the concatenation operation merges all inputs [31], allowing inputs of different sizes. We have used sum and concatenation operations. We have empirically observed that the best performance is associated with the concatenation operation.

- The fourth layer is for the *non-linear mapping operation*, which performs the second-order mapping. It contains 32-features maps with filter size  $5 \times 5$ .
- Finally, the fifth layer is the *reconstruction layer*, which reconstructs the output HR image. It includes  $c$  filter with size  $5 \times 5$ , where  $c$  is associated with the number of image channels, and we have used  $c = 1$ .

$$F_0(\mathbf{x}) = \mathbf{x} \quad \text{The input image.} \quad (3.11)$$

$$F_i(\mathbf{x}) = \max(0, W^{(i)} * F_{i-1}(\mathbf{x}) + b^{(i)}) \quad i \in \{1, 2, 4\} \quad (3.12)$$

$$F_{12}(\mathbf{x}) = \text{merge}(F_1(\mathbf{x}), F_2(\mathbf{x})) \quad (3.13)$$

$$F_3(\mathbf{x}) = \max(0, W^{(3)} * F_{12}(\mathbf{x}) + b^{(3)}) \quad (3.14)$$

$$F(\mathbf{x}) = W^{(5)} * F_4(\mathbf{x}) + b^{(5)} \quad \text{The output image.} \quad (3.15)$$

where  $W^{(i)}$  and  $b^{(i)}$  are the filters and biases of the  $i^{th}$  layer. The  $W^{(i)}$  comprises of  $n_i$  filters which support  $n_{i-1} \times f_i \times f_i$ , where  $n_i$  is the number of filters (number of feature maps), and  $n_0$  is the number of channels in the input image.  $F_i(\mathbf{x})$  is the output feature maps and  $F_{i-1}(\mathbf{x})$  is the input feature maps.  $F_{12}(\mathbf{x})$  is the merge operation.  $F(\mathbf{x})$  is the reconstructed output image which is of the same size as the input image.

**The structure of DBSRCNN:** The number of feature maps (and filter size) of each layer as follows 32(9), 32(5), 32(5), 32(5) and 1(5), although to make it simpler, we can write it as (32-32-32-32-1)(9-5-5-5-5).

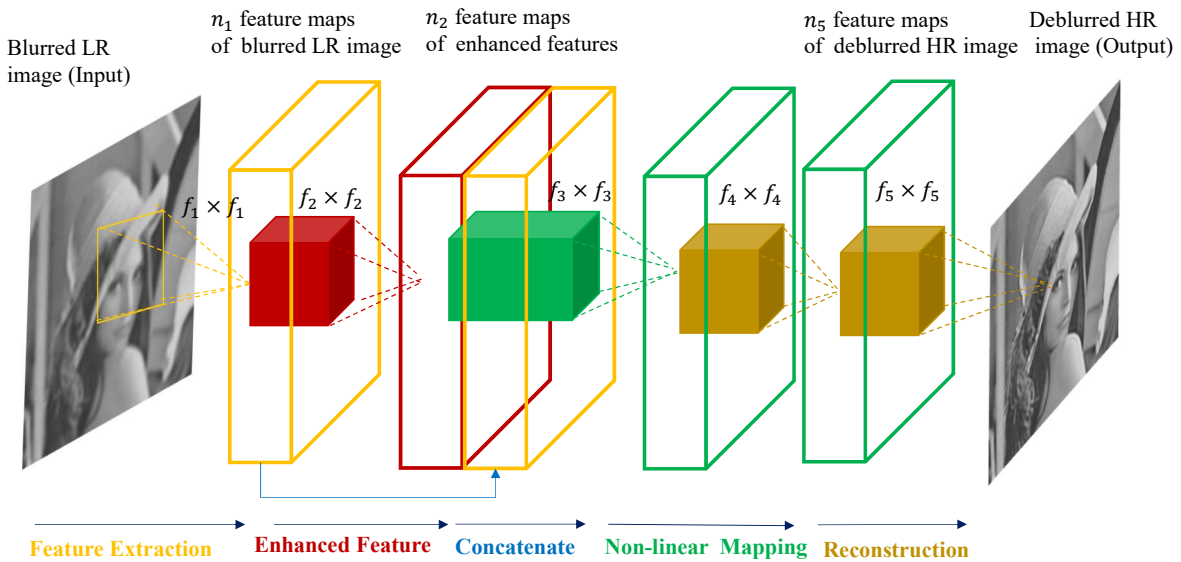
### DBSRCNN Optimisation

In each layer, the filter weights are initialised by the initialisation method described in He et al. [67]. ReLU is used as the activation function [114]. The biases are set to 0, and the learning rate is 0.001. We train all experiments for 60 epochs with a batch size of 64. The

estimation of the optimal network parameters  $\Theta = \{W^{(1)}, W^{(2)}, W^{(3)}, W^{(4)}, W^{(5)}, b^{(1)}, b^{(2)}, b^{(3)}, b^{(4)}, b^{(5)}\}$  is required to learn the end-to-end mapping function  $F$ . This is accomplished by minimising the cost between the reconstructed SR images  $F(\mathbf{x}, \Theta)$  and their original HR images  $\mathbf{y}$ . The MSE function  $C(\Theta)$  is used as the cost function, which is used to compare the squared error between the reconstructed image and the original image:

$$C(\Theta) = \frac{1}{m} \sum_{i=1}^m \|F(\mathbf{x}_i; \Theta) - \mathbf{y}_i\|^2 \tag{3.16}$$

This cost function is minimised using Adam [84], which is used to optimise the network to find optimal weights and for faster convergence rates.



**Fig. 3.3** Proposed architecture DBSRCNN: This network involves five layers; four convolutional layers plus concatenation (merge) layer, each layer is responsible for a particular operation; feature extraction, feature enhancement, merge the first two layers, non-linear mapping and finally reconstruction. *Deeper DBSRCNN* contains six layers; the same layers in DBSRCNN plus another non-linear mapping layer.

### 3.3 Introducing more layers

Inspired by the step of feature enhancement used in super-resolution [173] and JPEG compression artefacts reduction [186], we propose to introduce three feature enhancement layers after the merged layer in DBSRCNN [7] to create a more efficient and deeper network (DBSR). Indeed, a single layer has a limited capacity to enhance the noisy extracted features in complex applications like blurred SISR. Therefore, we increase this number to improve the capacity to suppress blur (noise) in the features. We have employed the DBSR network to recover the deblurred HR image from the blurred LR image [9]. Furthermore, DBSR has been trained to denoise images in the real dataset (i.e., dataset with real noise) named RENOIR to evaluate its performance on the real noise [8].

#### 3.3.1 De-Blurring Super-Resolution (DBSR) Architecture

The new model DBSR is shown in Figure 3.4. Overall it consists of eight layers. The five layers of DBSRCNN remain unchanged in the new network. The first enhanced feature layer located after the first layer is designed to extract new features from the extracted noisy features, and then merge these features together using the concatenation layer to map them together. While in DBSRCNN, we mapped these features directly; in DBSR these features are further processed by three layers before the final mapping. The operations of the proposed network can be described as follows:

$$F_0(\mathbf{x}) = \mathbf{x} \quad \text{The input image.} \quad (3.17)$$

$$F_i(\mathbf{x}) = \max(0, W^{(i)} * F_{i-1}(\mathbf{x}) + b^{(i)}) \quad i \in \{1, 2, 4, 5, 6, 7\} \quad (3.18)$$

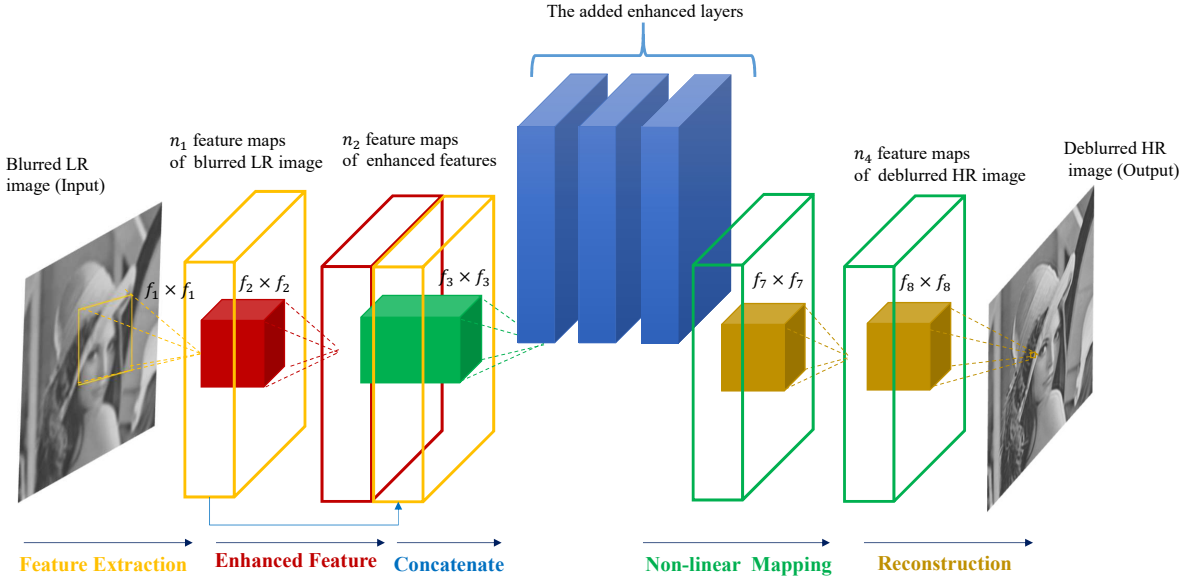
$$F_{12}(\mathbf{x}) = \text{merge}(F_1(\mathbf{x}), F_2(\mathbf{x})) \quad (3.19)$$

$$F_3(\mathbf{x}) = \max(0, W^{(3)} * F_{12}(\mathbf{x}) + b^{(3)}) \quad (3.20)$$

$$F(\mathbf{x}) = W^{(8)} * F_7(\mathbf{x}) + b^{(8)} \quad \text{The output image.} \quad (3.21)$$

where  $W^{(i)}$  and  $b^{(i)}$  are the filters and biases of the  $i^{th}$  layer. The  $W^{(i)}$  comprises of  $n_i$  filters which supports  $n_{i-1} \times f_i \times f_i$ , where  $n_i$  is the number of filter (number of feature maps), and  $n_0$  is the number of channels in the input image.  $F_i(\mathbf{x})$  is the output feature maps and  $F_{i-1}(\mathbf{x})$  is the input feature maps.  $F_{12}(\mathbf{x})$  is the merge operation.  $F(\mathbf{x})$  is the

reconstructed output image which is of the same size as the input image. The structure of the proposed network is eight layers (64-32-32-32-32-32-1)(9-5-5-5-5-5-5).



**Fig. 3.4** Proposed architecture DBSR: This network comprises eight layers: the five convolutional layers of DBSRCNN, in addition to extra three enhanced layers inserted after the concatenated layer to further refine the merged feature maps.

### 3.3.2 DBSR Optimisation

Consider a set  $\{\mathbf{y}_i, \mathbf{x}_i\}_{i=1}^m$ , where  $\mathbf{y}$  is a high-resolution image and  $\mathbf{x}$  is its corresponding interpolated blurred low-resolution image. Mean Squared Error (MSE) is used as the cost function to find the optimal parameters  $\Theta$  of the model. This is achieved by minimising the difference between the reconstructed images  $F(\mathbf{x}, \Theta)$  and their ground truth high-resolution images  $\mathbf{y}$ :

$$C(\Theta) = \frac{1}{m} \sum_{i=1}^m \|F(\mathbf{x}_i; \Theta) - \mathbf{y}_i\|^2, \quad (3.22)$$

where the network parameters  $\Theta = \{W^{(1)}, W^{(2)}, \dots, W^{(8)}, b^{(1)}, b^{(2)}, \dots, b^{(8)}\}$  and  $m$  is the number of training samples. The cost function is minimised using Adam optimisation [84]. Similar to DBSRCNN, DBSR adopts Rectified Linear Unit (ReLU) as the activation function.

We train all experiments for 60 epochs with a batch size of 64. In each layer, the filter weights are initialised by the initialisation method described in He et al. [67], considered a robust method for ReLU. The learning rate is 0.001.

### 3.4 Using Harmonic Blocks (Harm-net)

CNNs depend on learning the convolutional filters to extract the local correlation of input patterns in feature space since the spatial convolution using learned filters performed on natural images is based on the idea of a strong correlation between pixels in local neighbourhoods. In contrast, Harmonic Network uses the transformation methods which decorrelate the signals composing an image [160, 161]. The harmonic network is a feature learning that is implemented by weighted combinations of responses to predefined spectral filters. Instead of using standard convolution operation as in CNN, the harmonic net utilises harmonic blocks, which is implemented in two stages to process the data. In the first stage, the input features are decomposed by window-based Discrete Cosine Transform (2D DCT). Secondly, the responses (transformed signals) of the DCT filters are weighted by the learned weights. The harmonic block for each feature map can be computed by the following equation [160]:

$$F^l = \sum_{i=0}^{n-1} \sum_{u=0}^{K-1} \sum_{v=0}^{K-1} W_{i,u,v}^l \psi_{u,v} * F_i^{l-1} \quad (3.23)$$

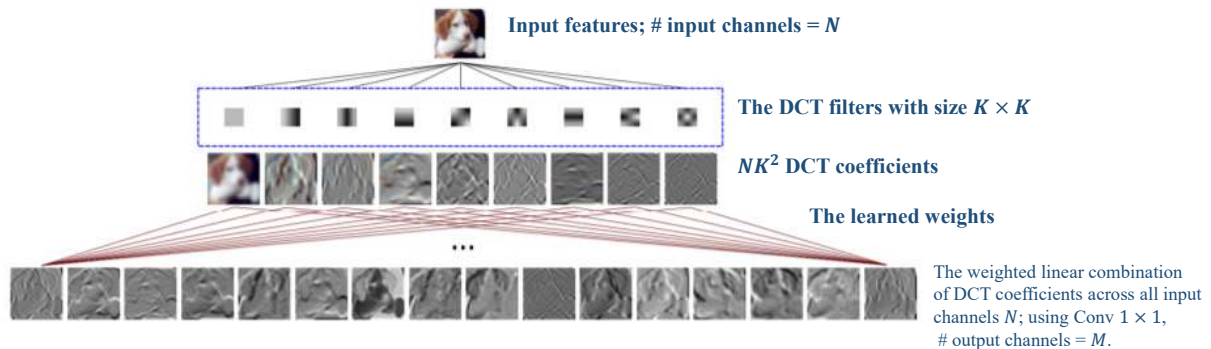
where  $F^l$  is the output feature map at layer  $l$ , which is computed as a weighted linear combination of DCT filters over the input channels  $n$ .  $F_i^{l-1}$  is an input feature map.  $\psi_{u,v}$  is a  $u, v$  spectral frequency of DCT filter with size  $K \times K$ , and  $'*$  is a 2D convolution operator.  $W_{i,u,v}^l$  is the learned weight of the  $i^{th}$  feature for  $u, v$  frequency. The linear combination of DCT coefficients is applied by a convolutional filter with size  $1 \times 1$ .

#### 3.4.1 Harm-DBSR Architecture

Some studies have conducted learning on the frequency domain, where the networks have been trained on DCT features. For example, Chen et al. [29] have used the linear filters to train the diffusion network using clean/degraded image pairs for image restoration



tasks. The filters are defined as a weighted linear combination of DCT basis filters. The diffusion model is trained in stages (iterations) involving convolution operations with a set of linear filters, and hence it can be treated as a CNN. Therefore, we were motivated to integrate the DCT transformation with our DBSR network, to investigate the influence of using spectral information in this network on the performance of the image restoration. We have followed the same strategy proposed in [160] to build the harmonic block. The convolutional layer in CNN can be replaced by the harmonic block to construct a fully or a partially harmonic network. We have used the harmonic blocks in the DBSR architecture to evaluate the influence of the predefined filters experimentally. The DBSR consists of eight convolutional layers as shown in Figure 3.4. The DBSR network has been implemented with different numbers of harmonic blocks, starting by replacing one convolutional layer with one harmonic block, and ending with a full harmonic net with eight harmonic blocks. Figure 3.5 demonstrates a visual example of applying the harmonic block.



**Fig. 3.5** Visualisation example of the harmonic block implemented on an input layer; taken from [160]. Each 2D filter of the DCT filter bank with size  $K \times K$  is applied to each input feature, to generate the spectral coefficients of the DCT basis functions. Then the weighted linear combination of these coefficients is performed by convolutional filter with size  $1 \times 1$ , to create new feature maps.

### 3.4.2 Harm-DBSR Optimisation

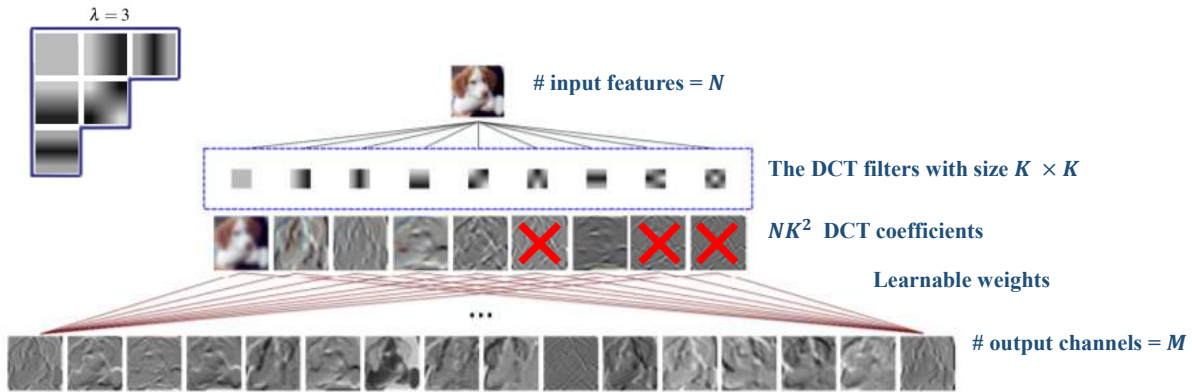
We have performed here the same optimisation strategy we implemented in DBSR with the same hyper-parameters and the same datasets, to compare the performance of the same network with and without the harmonic blocks. The backward pass through the transform layer is implemented as in the convolutional layer since the DCT is a linear transform. Consider a set  $\{\mathbf{y}_i, \mathbf{x}_i\}_{i=1}^m$ , where  $\mathbf{y}$  is a ground truth image and  $\mathbf{x}$  is its corresponding corrupted image. Mean Squared Error (MSE) is used as the cost function to find the optimal parameters  $\Theta$  of the model. This is achieved by minimising the difference between the reconstructed images  $F(\mathbf{x}, \Theta)$  and its ground truth images  $\mathbf{y}$ :

$$C(\Theta) = \frac{1}{m} \sum_{i=1}^m \|F(\mathbf{x}_i; \Theta) - \mathbf{y}_i\|^2, \quad (3.24)$$

where  $\Theta$  is the network parameters and  $m$  is the number of training samples. The cost function is minimised using Adam optimisation [84]. Similar to DBSR, Harm-DBSR uses ReLU as the activation function. We train all experiments for 60 epochs with a batch size of 64. In each layer, the filter weights are initialised by the initialisation method in He et al. [67]. The learning rate is 0.001.

### 3.4.3 Compression of the Harm-DBSR Network

The objective of our research is to achieve competitive results with state-of-the-art methods with a small number of parameters. Thus, we will follow the same strategy proposed in [160], to limit the DCT coefficients used in the harmonic block by using the most informative low frequencies and truncate the high-frequencies; see Figure 3.6. Consequently, this strategy enables us to decrease the number of parameters and their following operations in the network. A hyper-parameter  $\lambda$  represents the number of the coefficients used in the harmonic block beginning from the DC component (zero frequency). The following equation can calculate the number of levels of coefficients  $\lambda(\lambda + 1)/2$ , for instance, if we want to use 9x9 DCT filters with  $\lambda = 9$ ; then the number of coefficients =  $9(10)/2 = 45$  coefficients from 81 coefficients.



**Fig. 3.6** Visualisation example of the compressed harmonic block implemented on input features. For example, we employ a 3x3 DCT filter bank, and applied  $\lambda$ , which is used as a hyper-parameter to reduce the DCT coefficients by limiting the spectral frequencies and truncating the high frequencies. If  $\lambda = 1$ , then the zero frequency (DC component) only will be used. If  $\lambda = 3$ , then six coefficients starting from the DC component will be utilised, and three coefficients will be truncated as illustrated in this example.

### 3.5 Conclusion

This chapter introduces the different CNN architectures we designed with their optimisation strategies. In the following chapters, we will present the performance of these networks in various image restoration applications. In Chapter 4, we perform the Single Image Super-Resolution (SISR) task using the proposed CNNs that are capable of handling low-resolution and input image blurring simultaneously. We present in this chapter the experimental evaluation of the new architectures DBSRCNN, DBSR and Harm-DBSR that are applied to both non-blind and blind SR scenarios. Chapter 5 addresses the artefacts reduction in JPEG-compressed images using CNNs. We employ in this application, the different versions of DA-CAR (DA-CAR3, DA-CAR4 and DA-CAR5) and SA-CAR6, we also compare the performance of these architectures with the state-of-the-art methods proposed for artefact reduction. The DBSR model is designed to recover the deblurred high-resolution image from the blurred low-resolution image, by learning an end-to-end mapping. Furthermore, in Chapter 6, DBSR is trained to denoise images in the real RENOIR dataset [12] to assess its performance on real noise.



## Chapter 4

# Experimental Comparisons of Deblurring Super-Resolution

Many researchers around the world have studied the problem of obtaining SISR from a LR image. There are many traditional methods for obtaining SR; however, the majority of the best performing state-of-the-art methods for SR are based on deep neural networks, such as [42, 81, 82, 87, 97, 99, 139, 167]. All these algorithms assume that a small amount of noising or blurring is applied to all training and testing images, where they focus only on getting HR images from the LR images with low blurring levels produced from upscaling the images using bicubic interpolation. Indeed, the images are sometimes not solely suffering from LR but issues also extend to other problems such as blurring or noise. Therefore, in this study, we address an additional factor of an unknown amount of blurring applied to images that are received by the SR pipeline. It is thus necessary to tackle simultaneously deblurring and SR reconstruction in a unified procedure.

This chapter is organised as follows: Firstly, we present the methodology of the study in Section 4.1. Secondly, Section 4.2 displays the experimental results for different CNNs we proposed in Chapter 3 for SR application. In Subsection 4.2.1, we begin by implementing the SRCNN model proposed by Dong et al. [40], which is considered as a pioneering CNN model for the SR task. The SRCNN network introduced a simple yet efficient architecture for image restoration. It has been used to obtain HR images from LR images, although in this research, we have applied SRCNN to obtain deblurred SR images from blurred LR

images with different levels of blurring. To tackle efficiently simultaneous deblurring and SR, we experimentally validate the proposed DBSRCNN model in Subsection 4.2.2. The evaluation of the deeper network DBSR, which allows for enhanced non-linearity mapping is exhibited in Subsection 4.2.3. Then, the performance of the harmonic blocks used in DBSR model (Harm-DBSR), is shown in Subsection 4.2.4. We apply SR on blurred images with two different scenarios: with a priori known (non-blind) and unknown (blind) amount of blurring. Finally, we present the conclusions in Section 4.3.

## 4.1 Methodology

### 4.1.1 Training and Testing Datasets

Various training datasets have been used for training different networks. For example, Dong et al. [40] used 91 images from Yang et al. [180] to train SRCNN, which is considered as a relatively small dataset. Depending on the idea that the performance of deep learning is boosted from training a larger training dataset, Dong et al. [42] have trained the SRNN using a large dataset that comprises 395,909 images from the ILSVRC 2013 ImageNet [38]. Nevertheless, this large data yielded a slight improvement. SRCNN is considered to be a small model, and the 91 images have captured sufficient variability of natural images; therefore, this dataset was enough to train the SRCNN model. Indeed, to learn from training with a large dataset which combines more diverse data such as ImageNet, it needs a model with a large learning capacity [85]. In the VDSR model [81]; 291 images have been used: 91 images from Yang et al. [180] and 200 images from Berkeley Segmentation Dataset (BSD) [110] to train a large network with 20 layers.

Training the DNNs on a significant amount of data helps to avoid the overfitting problem, and the image data augmentation techniques are usually used in computer vision tasks to create more data from available training data [119, 140]. There are traditional transformations for augmenting training data such as cropping patches from the available images, flipping, rotating and colour transfer. In our work, we have used the most common practical augmentation techniques: cropping, flipping and rotating methods to increase the size of original images. In this chapter, we train different networks (SRCNN, DBSRCNN,

DBSR, Harm-DBSR and compressed Harm-DBSR); hence, we have used various training datasets with data augmentation methods.

**For training the reimplemented SRCNN and DBSRCNN models:** The 91 images from Yang et al.[179] are used. We make this choice of training data to allow a fair comparison with [42] where they were employed. To fully exploit the available data, we rely on the augmentation strategy: the training HR set  $\mathbf{y}$  are randomly cropped to obtain  $f_{sub} \times f_{sub} \times c$  pixel sub-images, with a stride of 14.  $f_{sub} \times f_{sub}$  is the number of pixels and the number of channels is  $c$ . The size of training sub-images we employ is  $f_{sub} = 33$ , accordingly the 91-images can be divided into 21,824 training sub-images.

**For training the DBSR and Harm-DBSR models:** These models are larger than SRCNN and DBSRCNN; consequently, they need more data for training. We employ 291 images as in [81], with data augmentation (flipping and rotation techniques) resulting in a total of 2,328 images. These training images are then cropped into sub-images with size  $f_{sub} = 31$  resulting in 573,632 sub-images by employing a stride of 21. The augmentation data with flipping and rotating techniques were used for training the non-blind networks and blind DBSR with  $\sigma = [0.5, 3]$ , but, it took 18 hours to train 60 epochs. Therefore, we trained blind DBSR with  $\sigma = [1, 3]$  and  $\sigma = [1, 4]$  using the 291 images with only cropping augmentation, which yields 204,288 sub-images and seven hours for training 60 epochs. We have employed the same training strategy used in DBSR for Harm-DBSR, to investigate the impact of using harmonic blocks in the existing CNN (DBSR) without any other changes.

**Test Datasets:** Set5 (5 images) [21] and Set14 (14 images) [189] have been used in the testing stage. The model is trained on sub-images, whereas the inference is carried on the whole image to avoid the step of averaging as post-processing for estimated HR patches. Each result of our experiments written in the following tables is the median of five runs.

### 4.1.2 Degradation Model

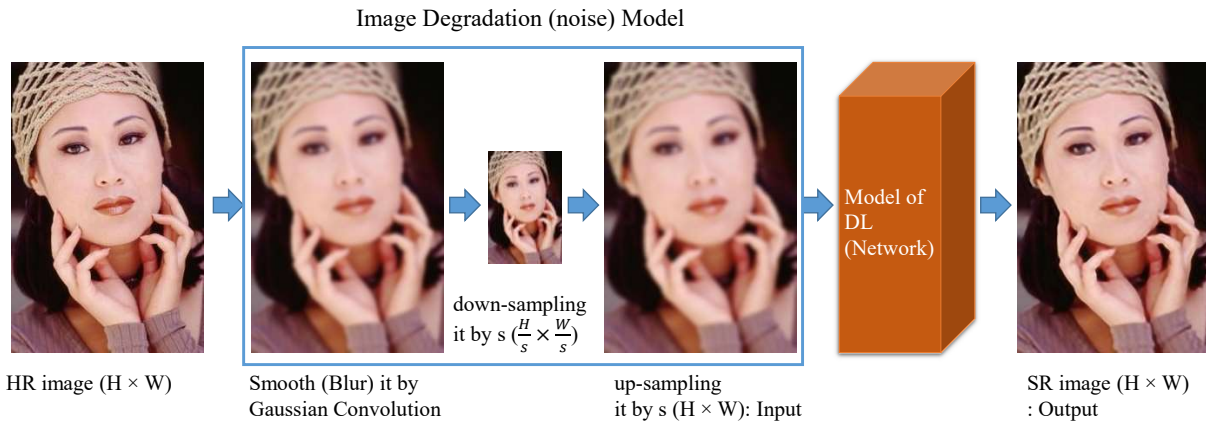
The standard model for degradation is formulated as a linear combination, as follows:

$$\mathbf{x} = D_s B_\sigma \mathbf{y} + \eta \quad (4.1)$$

where the HR image  $\mathbf{y}$  is first blurred by operator  $B_\sigma$ ; at the blurring level  $\sigma$  [45, 126, 178, 194]. Then, it is down-sampled by  $D_s$  with fixed down-sampling scale  $s$ . We have used the bicubic down-sampling method in our study as in [33, 47, 56, 59, 155, 180]. The noise  $\eta$  is additive noise. In practice, the pipeline for degradation can be non-linear due to compression artefacts or noise [127]. Before generating LR images according to equation (4.1), the blur kernels should be defined. The degradation model assumes that an HR image can be degraded into many LR images depending on the blur kernel and the noise. To produce blur, we applied the Gaussian kernel model  $N(0, \sigma)$  with a fixed kernel width (the value of  $\sigma$ ), which has been established to be practically feasible in SISR applications.

To generate a single blurred LR image  $\mathbf{x}_i$  (input) for training and testing according to equation (4.1), and as shown in Figure 4.1, the HR image  $\mathbf{y}_i$  is first blurred using a Gaussian kernel  $N(0, \sigma)$  with different values of standard deviation  $\sigma$  (i.e.  $\sigma = 1, 2, 3$  or  $4$ ). Secondly, the blurred image is down-sampled using the down-scaling factor  $s$ . Then up-sampling (zooming) the down-sampled image  $\mathbf{x}_i$  to the HR input resolution size is undertaken using bicubic interpolation. The down/up-scaling factors employed in this study are  $s = 2, 3, 4$ . The only pre-processing method that has been performed is obtaining corrupted data for training the networks. Note that padding is required to ensure that both the input and the output images in the network are of the same spatial dimensions.





**Fig. 4.1** An example of the training phase: Firstly, the blurred LR images are created from HR images (the only pre-processing), using a Gaussian filter to smooth the HR images using  $\sigma$  value (the blurring level). Then down-sampling the blurred images using a down-scaling factor, for instance,  $s = 3$ . The blurred LR images are zoomed using bicubic interpolation by an up-sampling factor  $s = 3$ . These degraded images are used as inputs to the network. The reconstructed SR images resulted from the network should be as similar as possible to HR images.

### 4.1.3 Non-Blind and Blind SISR Scenarios

There are two main scenarios which have been conducted to obtain the deblurred super-resolution images from blurred low-resolution images:

**1- Non-Blind SR scenario:** This is concerned with obtaining deblurred SR images from blurred LR images when the levels of blurring in images are known. The 2-D Gaussian filtering  $N(0, \sigma)$  is used to blur the images with different values of  $\sigma$ . The original assumption of the SRCNN algorithm is that the blurring in the LR images is produced from bicubic interpolation operation without adding any blurring (i.e.  $\sigma = 0$ ).  $\sigma = i$  is the blur kernel width which we added to the LR images using Gaussian kernel with  $i = 1, 2, 3$  or  $4$ , and the network is trained and tested with it. A Python script was created to build five different networks according to the level of blurring.

**2- Blind SR scenario:** This focuses on obtaining deblurred SR images from blurred LR images when the levels of blurring in images are unknown. The blind SR model is trained on a collection of images with different blurring levels. The benefit of the blind model

is that one solution solves many problems simultaneously instead of each task having a specific solution. As in the non-blind case; in the first place, the blurring level should be determined first, followed by finding the solution according to that. A Python script was created to build two different blind SR networks according to the range of blurring levels in images. While the first network is trained on images with kernel width  $N(0, \sigma)$  with  $\sigma$  ranging between [1-3], the second network is trained on ranging between [1-4]. The blind models are tested on images at any kernel width value. The aim of training two different blind models is to find out the different behaviours of these networks and then compare their results to discover if the network trained on a more significant number of blurring levels will give worse resolution or the same. Moreover, we have trained the DBSR and Harm-DBSR networks on images with  $\sigma$  varying between [0.5, 3] to investigate the influence of the standard deviation values on the performance.

#### 4.1.4 Experiments on Colour Images

- The majority of existing SR methods concentrate on single-band or grey-scale input images. In this case, colour images are transformed to another colour space as YCbCr, and the luminance channel Y is used in the SR algorithms ( $c = 1$ ), where Y channel is considered as a grey-scale copy of the main image.
- However, for working on colour images, there are three main strategies to perform the SR approaches:
  - The straightforward approach is to perform SR separately on the input colour channels and then merge them into a colour image.
  - Another method consists in dealing with all three channels in a unified manner by expanding the sizes of layers in the deep architecture ( $c = 3$ ).
  - Finally, human vision has more accurate spatial sensitivity to the differences provided in the image brightness (luminance) more than variations in colour. Therefore, the colour images can be super-resolved by casting the colour images to YCbCr colour space where the SR is performed solely on the luminance channel Y ( $c = 1$ ). Chroma components Cb, and Cr are up-scaled by bicubic interpolation. Then all channels (Y, Cb, Cr) are combined again to produce the output.

In this work, we follow the third strategy of color images and perform SR on the Y channel (i.e.,  $c = 1$ ) in the first/last layer for training networks. Then, the bicubic interpolation is used on chroma components (Cb, Cr) to produce the final output.

### 4.1.5 Quantitative Metrics for Comparisons

The peak signal-to-noise ratio (PSNR) metric in dB is widely used to evaluate the quality of image restoration quantitatively. The PSNR metric is related to the optimiser of the network (MSE; which is used as a cost function), where there is a negative relationship between the MSE and PSNR. Therefore PSNR is used as an indicator of the performance of the networks. Also, the structural similarity index measure (SSIM) is used as an alternative evaluation metric to evaluate model performance.

## 4.2 Experimental Results

### 4.2.1 Evaluation of SRCNN

To evaluate the extent of improvement, we computed the average of PSNR and SSIM between the blurred LR input images (corrupted images) and the original HR images for all pipelines. The default baseline comparison is with bicubic interpolation. After training the networks, the average of PSNR (dB) and SSIM results on the Set5 and Set14 testing datasets were reported in Tables 4.1 and 4.2 respectively<sup>1</sup>. In the non-blind scenario, five pipelines were trained on images with blur  $N(0, \sigma)$  with  $\sigma = 0, 1, 2, 3$ , and 4, and tested on images having the same level of blur; all these AI pipelines improved the PSNR and SSIM. The performance improvement becomes less pronounced in the blind scenarios. From the results of blind SR models of the blurred LR images with different levels of  $\sigma$ , it appears that the best results were for  $\sigma = 2$  and  $\sigma = 3$ , and performance decreases on the two sides ( $\sigma = 1$  and  $\sigma = 4$ ). From these results, it is clear that the weights of the blind models are treated as an average of the weights of non-blind models. Although the blind method is better than the non-blind method when we do not know information about

---

<sup>1</sup>Part of these results was published at the Machine Learning for Signal Processing (MLSP) conference 2018, in Denmark [7].

blurring, blind SRCNN did not introduce any enhancement in LR images with  $\sigma = 0$ , where the reconstructed images were worse than the input images. The reason for this is that the blind networks were not trained on LR images; they just were trained on blurred LR images with different levels  $\sigma = [1-3]$  or  $[1-4]$ . Figures 4.2 and 4.3 demonstrate examples of the output of processing a colour input along with relevant comparisons with SRCNN.

**Table 4.1** Average of PSNR (dB)/ SSIM results with different blur levels  $\sigma = 0$  (i.e., without adding any blur), 1, 2, 3, 4, scale factor  $s = 3$  on test set 'Set5'.

Kernel Width	LR Input <sup>1</sup>	Non-Blind Model		Blind Model		
		SRCNN	SRCNN Improvement	SRCNN $\sigma = [1-3]$	SRCNN $\sigma = [1-4]$	SRCNN Improvement
$\sigma = 0$	30.40/ 0.881	31.95/ 0.905	+1.55/ +0.025	<b>28.49/ 0.848</b>	27.64/ 0.831	-1.91/ -0.032
$\sigma = 1$	29.47/ 0.860	31.55/ 0.899	+2.08/ +0.039	<b>30.05/ 0.879</b>	29.05/ 0.862	+0.58/ +0.019
$\sigma = 2$	27.45/ 0.802	30.29/ 0.868	+2.84/ +0.066	<b>30.02/ 0.871</b>	29.55/ 0.866	+2.57/ +0.069
$\sigma = 3$	25.65/ 0.735	29.01/ 0.833	+3.36/ +0.097	27.54/ 0.801	<b>27.80/ 0.809</b>	+2.15/ +0.074
$\sigma = 4$	24.33/ 0.680	27.35/ 0.778	+3.02/ +0.098	25.43/ 0.723	<b>25.91/ 0.738</b>	+1.58/ +0.059

<sup>1</sup> Baseline comparison with up-scale bicubic (No AI). The improvement is calculated as the difference between the average PSNR of the degraded images (LR input) and the reconstructed images using the non-blind and blind SRCNN networks.

**Table 4.2** Average of PSNR (dB)/ SSIM results with different blur levels  $\sigma = 0$  (i.e., without adding any blur), 1, 2, 3, 4, scale factor  $s = 3$  on test set 'Set14'.

Kernel Width	LR Input	Non-Blind Model		Blind Model		
		SRCNN	SRCNN Improvement	SRCNN $\sigma = [1-3]$	SRCNN $\sigma = [1-4]$	SRCNN Improvement
$\sigma = 0$	27.54/ 0.859	28.67/ 0.885	+1.13/ +0.026	<b>26.46/ 0.849</b>	25.85/ 0.832	-1.08/ -0.010
$\sigma = 1$	26.86/ 0.833	28.40/ 0.882	+1.54/ +0.049	<b>27.57/ 0.871</b>	26.92/ 0.856	+0.71/ +0.038
$\sigma = 2$	25.37/ 0.766	27.41/ 0.855	+2.04/ +0.089	<b>27.16/ 0.844</b>	26.93/ 0.841	+1.79/ +0.078
$\sigma = 3$	24.04/ 0.694	26.33/ 0.805	+2.29/ +0.111	25.34/ 0.759	<b>25.52/ 0.772</b>	+1.48/ +0.078
$\sigma = 4$	23.05/ 0.634	25.19/ 0.749	+2.14/ +0.115	23.85/ 0.677	<b>24.20/ 0.697</b>	+1.15/ +0.063

The improvement is calculated as the difference between the average PSNR of the degraded images (LR input) and the reconstructed images using the non-blind and blind SRCNN networks.

Original HR

blurred LR with  $\sigma = 2$ SRCNN  $\sigma = 2$ blind SRCNN  $\sigma = [1 - 3]$ 

**Fig. 4.2** SR with SRCNN on a colour image after Gaussian blur with  $\sigma = 2$ . The second row shows the results of the non-blind scenario and the blind scenario. Each result is accompanied by zoom and PSNR dB.



**Fig. 4.3** SR with SRCNN on a colour image after Gaussian blur with  $\sigma = 3$ . The second row shows the results of the non-blind scenario and the blind scenario. Each result is accompanied by zoom and PSNR dB.

## General Comparison

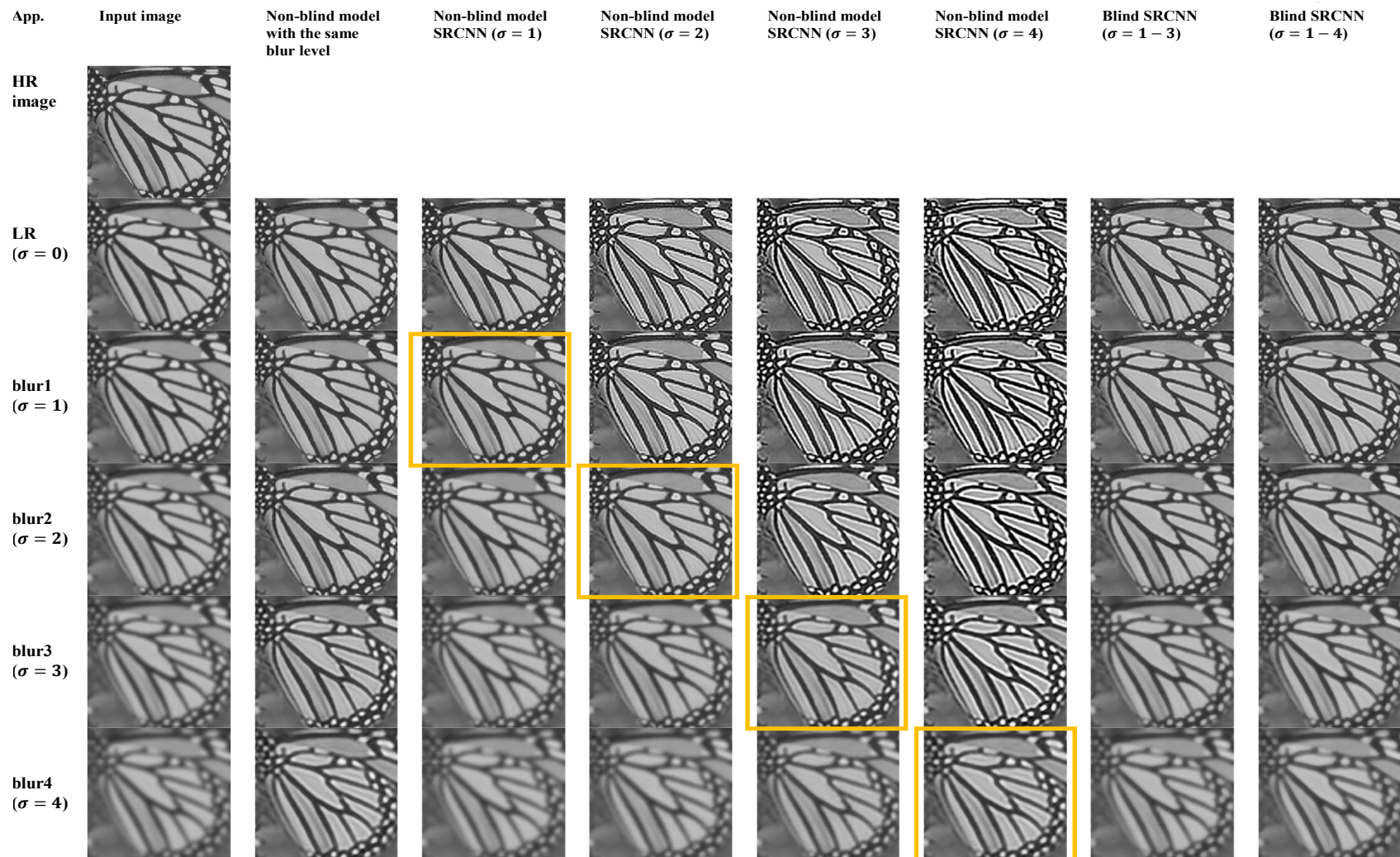
We want at this point to illustrate the influence of mismatch of the degradation method in training and testing in the non-blind scenario. As can be seen from Table 4.3 and Figure 4.4, when the training and testing assumptions match the blurring value, the result of the model will be best, see diagonal PSNR in blue in Table 4.3. However, if the assumption of the degradation at training and testing are mismatched, the results will deteriorate substantially. However, if the blind network is used, there is only a slight quality decrease, and the results are still good. Therefore, if we do not have information about the blurring value, we can use the blind pipelines.

By tacking a quantitative example, we can notice the difference if we compare between the results of different input images for any non-blind SRCNN and SRCNN  $\sigma = [1 - 3]$ . For example, input images blurred with  $\sigma = 2$  using SRCNN ( $\sigma = [1 - 3]$ ) = 27.92 and with SRCNN ( $\sigma = 2$ ) = 28.16. However, for the remainder of the results of SRCNN  $\sigma = [1 - 3]$  for input images with different blurring levels, results are are better than the results provided by SRCNN  $\sigma = 2$ . If we see the LR input images  $\sigma = 0$  with SRCNN ( $\sigma = 2$ ) = 20.66 and with SRCNN ( $\sigma = 1 - 3$ ) = 26.99, this has a huge effect on the result. Therefore, if we estimate blurring levels in the corrupted image wrongly or use the mismatched model, the obtained results will be worse than if we used the blind network. A qualitative example is evident in Figure 4.4. Note that Table 4.3 gives a brief of the average PSNR over the merged test sets (we have combined Set5 and Set14 in one group). For more details see Appendix B.

**Table 4.3** Average of PSNR (dB) with different blur levels  $\sigma = 0$  (i.e., without adding any blur), 1, 2, 3, 4, scale factor  $s = 3$ , on test sets 'Set5' and 'Set14' together as one group.

Kernel Width	LR Input	Non-Blind Model <sup>1</sup>					Blind Model <sup>2</sup>	
		SRCNN <sup>3</sup> $\sigma = 0$	SRCNN $\sigma = 1$	SRCNN $\sigma = 2$	SRCNN $\sigma = 3$	SRCNN $\sigma = 4$	SRCNN $\sigma = [1 - 3]$	SRCNN $\sigma = [1 - 4]$
$\sigma = 0$	28.29	29.53	28.18	20.66	16.86	14.75	26.99	26.32
$\sigma = 1$	27.55	28.90	29.23	23.64	18.80	16.04	28.22	27.48
$\sigma = 2$	25.92	26.53	27.11	28.16	23.98	19.59	27.92	27.62
$\sigma = 3$	24.46	24.63	24.87	25.78	27.04	24.35	25.92	26.12
$\sigma = 4$	23.39	23.42	23.54	23.92	24.88	25.76	24.27	24.65

The diagonal blue results appear the outputs when the training and testing assumptions match over the blurring value in the non-blind scenarios. <sup>1</sup> Re-trained SRCNN Architecture with different level of noise on input. <sup>2</sup> Re-trained SRCNN Architecture with mix of noise level. <sup>3</sup> Dong et al. PAMI 2016.



**Fig. 4.4** Example of blind and non-blind SRCNN (9-1-5)(64-32-1), to discover the effect of using the non-blind model on different input images with varying levels of blurring. The yellow boxes show the outputs when the training and testing assumptions match over the blurring value in the non-blind scenarios.

Resolution



### 4.2.2 Evaluation of DBSRCNN

Tables 4.4 and 4.6 report the average of PSNR and SSIM for Set5 and Set14 respectively using non-blind pipelines. Tables 4.5 and 4.7 describe the average of PSNR and SSIM for Set5 and Set14 respectively using blind pipelines<sup>2</sup>. One observes a clear improvement of DBSRCNN's performance over SRCNN on blurred images. DBSRCNN architecture allows for improvement in the quality of the images as measured with PSNR and SSIM. A possible explanation of this performance is that the concatenation of features extracted at an early stage acts similarly to traditional image processing techniques such as unsharp masking that boosts relevant (high) frequencies partially lost in the blurring stage, to enhance the reconstructed image. Furthermore, in the provided tables, we computed the difference between the improvement of (9-1-5) SRCNN applied in Section 4.2.1 and the improvement of deep DBSRCNN for non-blind and blind SR applications.

Reconstruction examples with various levels of blur are shown in Figure 4.5 and Figure 4.6 for qualitative comparison: DBSRCNN allows for a visually better reconstruction than SRCNN confirming the quantitative assessment results reported by PSNR and SSIM. SRCNN has been compared with the earlier state-of-the-art methods (SC, NE+LLE, KK, ANR, A+) in [42], and hence DBSRCNN compares favourably with these as well. More examples are found in Appendix B. In general, we conclude that performance depends on the extracted features and representations, and the results of our proposed network emphasise that when the network contains enhanced features, this leads to improving the outcomes of the network. Also, when the network involves enhanced features and adopts more non-linear mapping layers, the network gives better results.

---

<sup>2</sup>Part of these results was published at the Machine Learning for Signal Processing (MLSP) conference 2018, Denmark [7].

**Table 4.4** Average of PSNR (dB)/ SSIM results for Non-blind Models with different blur levels  $\sigma = 0$  (i.e., without adding any blur), 1, 2, 3, 4, scale factor  $s = 3$  on test set 'Set5'.

Kernel Width	LR Input	SRCNN	SRCNN Improvement	DBSRCNN	DBSRCNN Improvement
$\sigma = 0$	30.40/ 0.881	31.95/ 0.905	+1.55/ +0.025	32.60/ 0.918	+2.20/ +0.038
$\sigma = 1$	29.47/ 0.860	31.55/ 0.899	+2.08/ +0.039	32.65/ 0.918	+3.18/ +0.058
$\sigma = 2$	27.45/ 0.802	30.29/ 0.868	+2.84/ +0.066	32.09/ 0.908	+4.64/ +0.106
$\sigma = 3$	25.65/ 0.735	29.01/ 0.833	+3.36/ +0.097	30.48/ 0.872	+4.83/ +0.136
$\sigma = 4$	24.33/ 0.680	27.35/ 0.778	+3.02/ +0.098	28.69/ 0.817	+4.36/ +0.138

The improvement is calculated as the difference between the average PSNR of the degraded images and the reconstructed images using the SRCNN and DBSRCNN networks.

The results in green colour introduce how much the SRCNN improves the results.

The results in blue colour introduce how much the DBSRCNN improves the results.

**Table 4.5** Average of PSNR (dB)/ SSIM results for Blind Models with different blur levels  $\sigma = 0$  (i.e., without adding any blur), 1, 2, 3, 4, scale factor  $s = 3$  on test set 'Set5'.

Kernel Width	LR Input	SRCNN $\sigma = [1 - 3]$	SRCNN $\sigma = [1 - 4]$	SRCNN Improvement	DBSRCNN $\sigma = [1 - 3]$	DBSRCNN $\sigma = [1 - 4]$	DBSRCNN Improvement
$\sigma = 0$	30.40/ 0.881	<b>28.49/ 0.848</b>	27.64/ 0.831	-1.91/ -0.032	<b>29.89/ 0.877</b>	29.63/ 0.869	-0.51/ -0.004
$\sigma = 1$	29.47/ 0.860	<b>30.05/ 0.879</b>	29.05/ 0.862	+0.58/ +0.019	<b>31.24/ 0.900</b>	30.85/ 0.892	+1.77/ +0.040
$\sigma = 2$	27.45/ 0.802	<b>30.02/ 0.871</b>	29.55/ 0.866	+2.57/ +0.069	<b>30.14/ 0.881</b>	30.08/ 0.875	+2.69/ +0.079
$\sigma = 3$	25.65/ 0.735	27.54/ 0.801	<b>27.80/ 0.809</b>	+2.15/ +0.074	<b>29.51/ 0.854</b>	28.40/ 0.834	+3.86/ +0.119
$\sigma = 4$	24.33/ 0.680	25.43/ 0.723	<b>25.91/ 0.738</b>	+1.58/ +0.059	26.28/ 0.758	<b>27.72/ 0.796</b>	+3.39/ +0.116

The improvement is calculated as the difference between the average PSNR of the degraded images and the reconstructed images using the SRCNN and DBSRCNN networks.

The results in green colour introduce how much the SRCNN improves the results.

The results in blue colour introduce how much the DBSRCNN improves the results.

**Table 4.6** Average of PSNR (dB)/ SSIM results for Non-blind Models with different blur levels  $\sigma = 0$  (i.e., without adding any blur), 1, 2, 3, 4, scale factor  $s = 3$  on test set 'Set14'.

Kernel Width	LR Input	SRCNN	SRCNN Improvement	DBSRCNN	DBSRCNN Improvement
$\sigma = 0$	27.54/ 0.859	28.67/ 0.885	+1.13/ +0.026	29.11/ 0.893	+1.57/ +0.034
$\sigma = 1$	26.86/ 0.833	28.40/ 0.882	+1.54/ +0.049	29.11/ 0.893	+2.25/ +0.061
$\sigma = 2$	25.37/ 0.766	27.41/ 0.855	+2.04/ +0.089	28.80/ 0.886	+3.43/ +0.120
$\sigma = 3$	24.04/ 0.694	26.33/ 0.805	+2.29/ +0.111	27.50/ 0.842	+3.46/ +0.148
$\sigma = 4$	23.05/ 0.634	25.19/ 0.749	+2.14/ +0.115	26.28/ 0.788	+3.23/ +0.155

The improvement is calculated as the difference between the average PSNR of the degraded images and the reconstructed images using the SRCNN and DBSRCNN networks.

The results in green colour introduce how much the SRCNN improves the results.

The results in blue colour introduce how much the DBSRCNN improves the results.

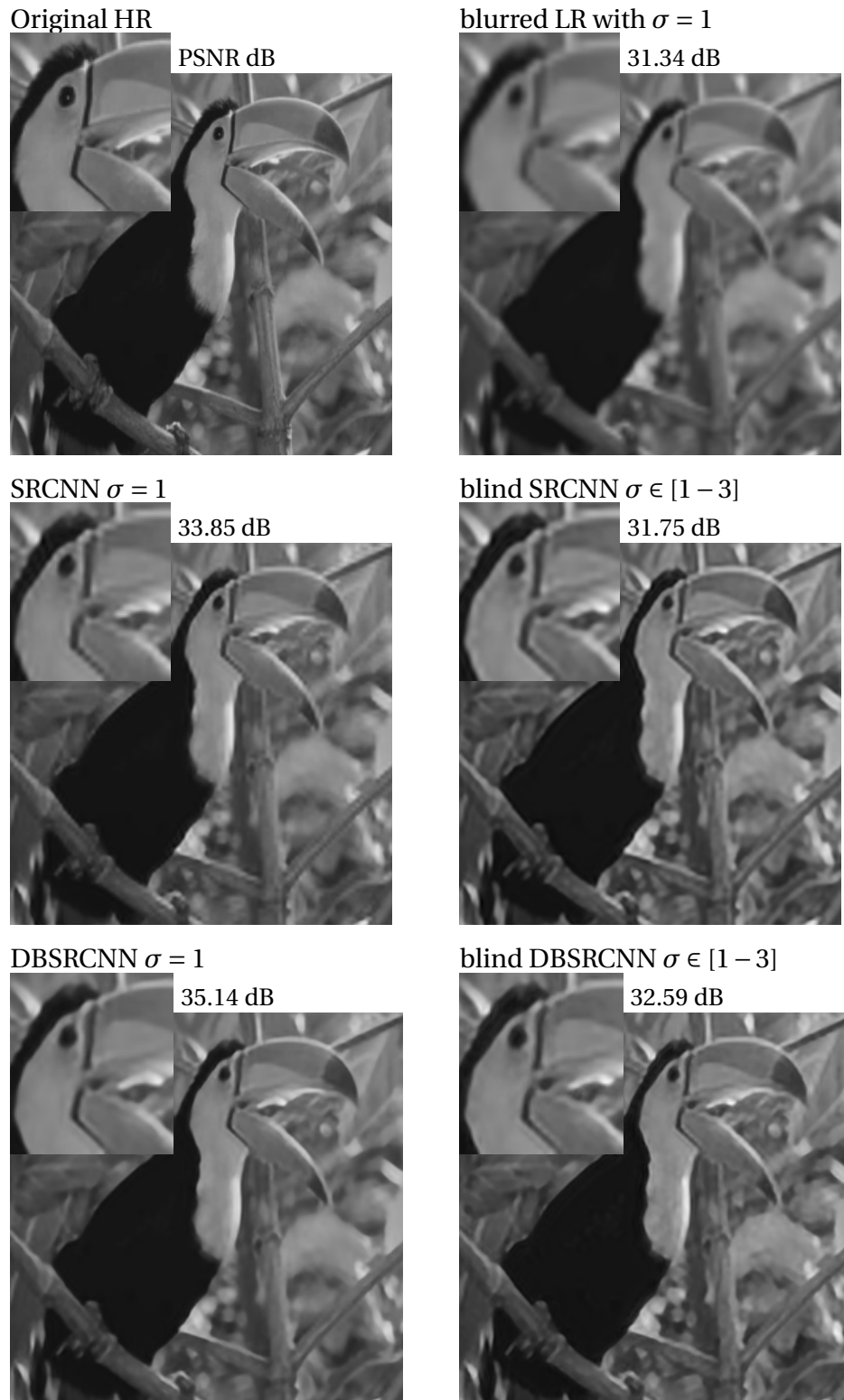
**Table 4.7** Average of PSNR (dB)/ SSIM results for Blind Models with different blur levels  $\sigma = 0$  (i.e., without adding any blur), 1, 2, 3, 4, scale factor  $s = 3$  on test set 'Set14'.

Kernel Width	LR Input	SRCNN	SRCNN	SRCNN	DBSRCNN	DBSRCNN	DBSRCNN
		$\sigma = [1 - 3]$	$\sigma = [1 - 4]$	Improvement	$\sigma = [1 - 3]$	$\sigma = [1 - 4]$	Improvement
$\sigma = 0$	27.54/ 0.859	<b>26.46/ 0.849</b>	25.85/ 0.832	-1.08/ -0.010	<b>27.20/ 0.869</b>	27.10/ 0.863	-0.34/ +0.010
$\sigma = 1$	26.86/ 0.833	<b>27.57/ 0.871</b>	26.92/ 0.856	+0.71/ +0.038	<b>28.38/ 0.883</b>	28.09/ 0.878	+1.52/ +0.051
$\sigma = 2$	25.37/ 0.766	<b>27.16/ 0.844</b>	26.93/ 0.841	+1.79/ +0.078	<b>27.43/ 0.849</b>	27.28/ 0.842	+2.06/ +0.083
$\sigma = 3$	24.04/ 0.694	25.34/ 0.759	<b>25.52/ 0.772</b>	+1.48/ +0.078	<b>26.68/ 0.812</b>	25.99/ 0.798	+2.64/ +0.118
$\sigma = 4$	23.05/ 0.634	23.85/ 0.677	<b>24.20/ 0.697</b>	+1.15/ +0.063	24.55/ 0.713	<b>25.46/ 0.756</b>	+2.41/ +0.122

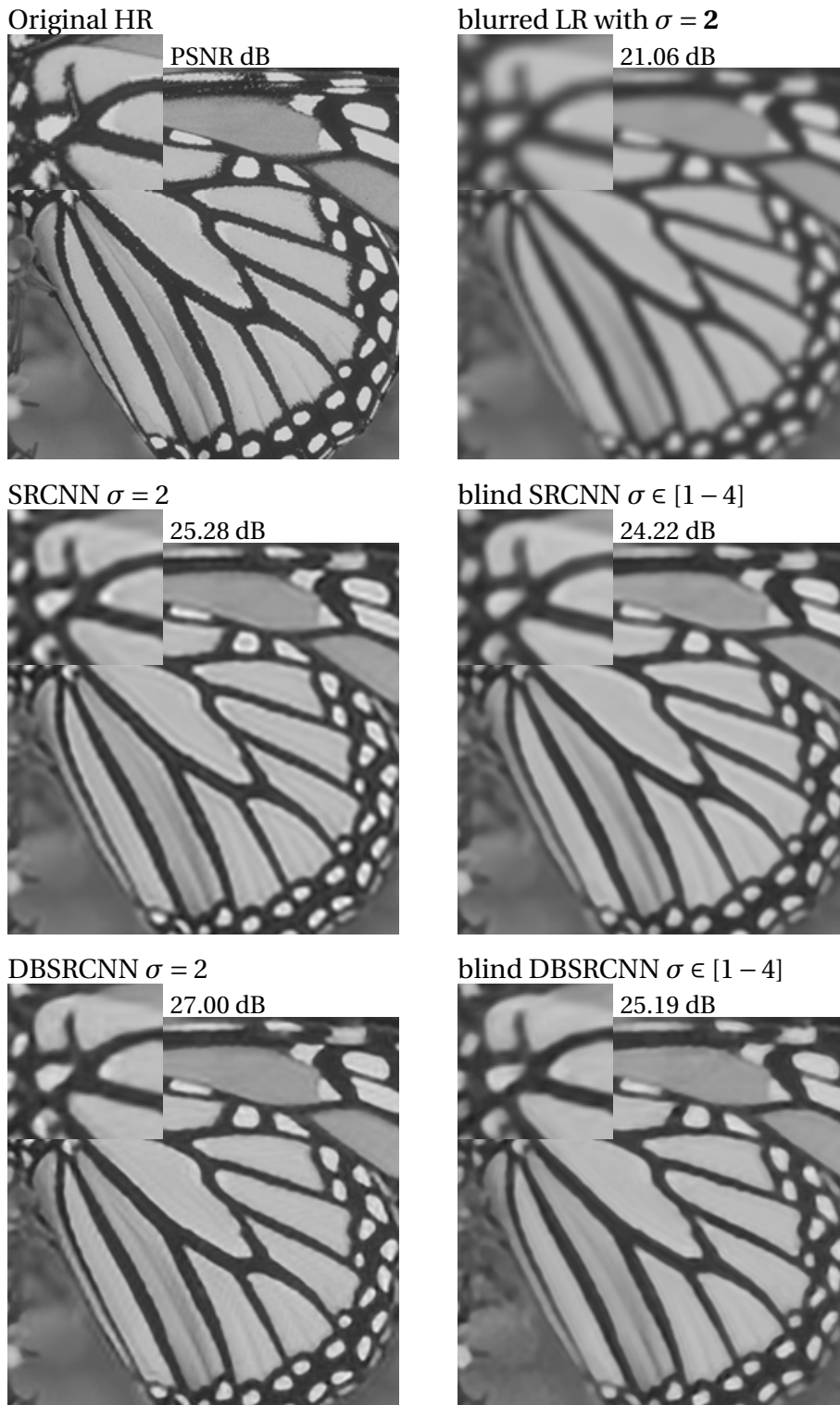
The improvement is calculated as the difference between the average PSNR of the degraded images and the reconstructed images using the SRCNN and DBSRCNN networks.

The results in green colour introduce how much the SRCNN improves the results.

The results in blue colour introduce how much the DBSRCNN improves the results.



**Fig. 4.5** SR with SRCNN and DBSRCNN on grey-scale images after Gaussian blur with  $\sigma = 1$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind  $\sigma \in [1 - 3]$  SR scenarios). Each result is accompanied by zoom and PSNR dB.



**Fig. 4.6** SR with SRCNN and DBSRCNN on grey-scale images after Gaussian blur with  $\sigma = 2$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind  $\sigma \in [1 - 4]$  SR scenarios). Each result is accompanied by zoom and PSNR dB.

## Comparison of the performance of different networks

In this part of the study, we want to investigate the behaviours of different CNN architectures (the direct and skip nets) in dealing with the deblurring SR application. He and Sun [66] have suggested that the CNN networks could benefit from increasing network depth by adding more layers in the network. From this point of view, Dong et al. [42] wanted to improve the result of the SRCNN network by adding extra non-linear mapping layers in the network (known as deep SRCNN). However, when deep SRCNN with four layers was applied to LR images, the result was not improved. Additionally, when they went deeper by using five layers, the result was worse. Hence, they adopted SRCNN with only three layers. In this section, we want to see the influence of the deep SRCNN on deblurring the SISR task. Therefore, deep SRCNN, which contains four layers, was applied to our applications to check the effect of adding the extra layer to enhance the features in the SRCNN network. Tables 4.8 and 4.9 summarise the results using PSNR and SSIM for all pipelines we performed for blurred input images with different blurring levels.

**For the deep direct nets:** The results show that the deep SRCNN with four layers that we used to enhance the LR images at  $\sigma = 0$  has provided minor enhancement in the performance, as indicated in Dong et al.'s paper [42]. However, the results confirm that the deep SRCNN has improved the reconstructed images with different levels of blurring, which means that more layers are required for enhancing the blurred LR images. We conclude that only one layer in the LR application for extracting the features was enough, while blurred LR application requires deeper networks to give better results because of the increasing blurring levels in the input images.

**For the skip nets:** The only difference between SRCNN and proposed DBSRCNN pipelines is that the first layer in SRCNN (64-feature maps) has been separated into two layers (32-feature maps for each layer). These layers have then been merged in the concatenation layer before mapping them non-linearly. However, we can observe a definite improvement of DBSRCNN's performance over SRCNN on blurred images, where this combination of low-level features and enhanced features has led to improving the performance. Furthermore, we have added another non-linear mapping layer to the proposed network DBSRCNN to gain a deeper network and adopt a more robust regressor between the ex-

tracted features and the output. The deep DBSRCNN network improves the performance more than the default DBSRCNN model, as shown in the tables.

We conclude that the proposed DBSRCNN network (which combines the low-level features and enhanced features in one layer before mapping operation) still presents a significant enhancement compared to the three-layer SRCNN model ((9-1-5)SRCNN or (9-5-5)SRCNN which does not have an enhanced feature layer and only maps the low-level features) — also compared to deep SRCNN (which only maps the improved features and ignores the low-level features). Furthermore, the results of DBSRCNN show that the concatenation operation is better than using the summation operation in the merge layer. Therefore, we adopt the concatenation operation in the proposed network. For more information around these comparisons of the different pipelines, please review Appendix B.

**Table 4.8** Average of PSNR (dB)/ SSIM results for all non-blind models with different blur levels  $\sigma = 0$  (i.e., without adding any blur), 1, 2, 3, 4, scale factor  $s = 3$  on test set 'Set5'.

Net.	Network	Type of Network	# layers	# filters	Filter size	# par. <sup>1</sup>	$\sigma = 0$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$	$\sigma = 4$	Time/ep <sup>2</sup>	Total time <sup>3</sup>
Bicubic	No AI	LR-HR	-	-	-	-	30.40/ 0.8806	29.47/ 0.8600	27.45/ 0.8022	25.65/ 0.7354	24.33/ 0.6796	-	-
Dong et al. PAMI 2016	Re-trained SRCNN	Default	3 layers	(64-32-1)	(9-1-5)	8,032	31.95/ 0.9054	31.55/ 0.9027	30.29/ 0.8853	29.01/ 0.8330	27.35/ 0.7776	10s	8.33m
				(64-32-1)	(9-5-5)	57,184	32.37/ 0.9139	32.14/ 0.9092	31.42/ 0.8930	29.62/ 0.8494	27.52/ 0.7820	13s	10.83m
New Networks	Deep SRCNN	Deep	4 layers	(64-32-16-1)	(9-5-5-5)	69,584	32.39/ 0.9136	32.22/ 0.9108	31.87/ 0.9035	29.85/ 0.8570	28.05/ 0.7999	15s	12.5m
				(64-32-32-1)	(9-5-5-5)	82,784	32.44/ 0.9148	32.28/ 0.9118	31.90/ 0.9042	29.99/ 0.8595	28.18/ 0.8024	18s	15m
	DBSRCNN: Merge operation is concatenation	Default	4 layers + concat	(32-32-64*	(9-5-5-5)	80,192	32.51/ 0.9167	32.37/ 0.9143	31.90/ 0.9043	30.14/ 0.8628	28.32/ 0.8074	18s	15m
				(32-32-64*	(9-5-5-5-5)	105,792	<b>32.60/ 0.9181</b>	<b>32.65/ 0.9179</b>	<b>32.09/ 0.9078</b>	<b>30.48/ 0.8716</b>	<b>28.69/ 0.8172</b>	23s	19.17m

<sup>1</sup> Number of parameters.    <sup>2</sup> Training time in seconds for one epoch.    <sup>3</sup> Total training time in minutes/ 50 epochs.

\* The concatenation layer.    The results in blue colour indicate the best results.



**Table 4.9** Average of PSNR (dB)/ SSIM results for all non-blind models with different blur levels  $\sigma = 0$  (i.e., without adding any blur), 1, 2, 3, 4, scale factor  $s = 3$  on test set 'Set14'.

Net.	Network	Type of Network	# layers	# filters	Filter size	# par. <sup>1</sup>	$\sigma = 0$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$	$\sigma = 4$	Time/ep <sup>2</sup>	Total time <sup>3</sup>
Bicubic	No AI	LR-HR	-	-	-	-	27.54/ 0.8589	26.86/ 0.8325	25.37/ 0.7660	24.04/ 0.6937	23.05/ 0.6337	-	-
Dong et al. PAMI 2016	Re-trained SRCNN	Default	3 layers	(64-32-1)	(9-1-5)	8,032	28.67/ 0.8852	28.40/ 0.8841	27.41/ 0.8661	26.33/ 0.8051	25.19/ 0.7494	10s	8.33m
				(64-32-1)	(9-5-5)	57,184	28.96/ 0.8905	28.81/ 0.8878	28.21/ 0.8724	26.83/ 0.8206	25.40/ 0.7536	13s	10.83m
New Networks	Deep SRCNN	Deep	4 layers	(64-32-16-1)	(9-5-5-5)	69,584	28.98/ 0.8905	28.89/ 0.8884	28.60/ 0.8815	27.05/ 0.8264	25.78/ 0.7701	15s	12.5m
				(64-32-32-1)	(9-5-5-5)	82,784	29.02/ 0.8909	28.90/ 0.8892	28.64/ 0.8815	27.12/ 0.8296	25.90/ 0.7757	18s	15m
	DBSRCNN: Merge operation is concatenation	Default	4 layers + concate	(32-32-64*	(9-5-5-5)	80,192	29.06/ 0.8921	28.98/ 0.8903	28.65/ 0.8826	27.18/ 0.8314	26.02/ 0.7787	18s	15m
				(32-32-64* -32-32-1)	(9-5-5-5-5)	105,792	<b>29.11/ 0.8927</b>	<b>29.11/ 0.8930</b>	<b>28.80/ 0.8860</b>	<b>27.50/ 0.8416</b>	<b>26.28/ 0.7882</b>	23s	19.17m

<sup>1</sup> Number of parameters.    <sup>2</sup> Training time in seconds for one epoch.    <sup>3</sup> Total training time in minutes/ 50 epochs.

\* The concatenation layer.    The results in blue colour indicate the best results.

### 4.2.3 Evaluation of DBSR

#### Comparison with the state-of-the-art models

The proposed DBSR model is compared with several CNN models designed to handle down-sampling without blurring<sup>3</sup>. Table 4.10 shows the PSNR and SSIM [169] results of state-of-the-art CNN models. While our proposed method may not always perform best, our DBSR pipeline still achieves competitive results with a significantly smaller number of parameters in comparison with the state-of-the-art models. We can notice that our model provides competitive performance with LapSRN [87] by utilising around three times fewer parameters, and with SRMD [194] by using about six times fewer parameters. The results of the DBSR model are better than those obtained by SRCNN and DBSRCNN, which is achieved by adding more layers in the DBSRCNN model to enhance the low-level features before mapping.

**Table 4.10** Average PSNR (dB) and SSIM results for  $\sigma = 0$  (i.e., without adding any blur) on datasets 'Set5' and 'Set14', with different scale factor  $s=2, 3, 4$ .

Dataset	Scale Factor	LR Input	SRCNN	DBSRCNN (Ours)	LapSRN [87]	SRMD [194]	DBSR (Ours)
		PSNR / SSIM					
Set5	$s = 2$	33.66 / 0.930	35.68 / 0.948	36.27 / 0.951	37.52 / 0.959	37.53 / 0.959	37.23 / 0.957
	$s = 3$	30.40 / 0.868	31.95 / 0.845	32.60 / 0.908	33.82 / 0.922	33.86 / 0.923	33.24 / 0.917
	$s = 4$	28.42 / 0.810	29.79 / 0.844	30.25 / 0.858	31.54 / 0.885	31.59 / 0.887	30.84 / 0.873
Set14	$s = 2$	30.24 / 0.869	31.74 / 0.899	32.10 / 0.903	33.08 / 0.913	33.12 / 0.914	32.83 / 0.911
	$s = 3$	27.54 / 0.774	28.67 / 0.806	29.11 / 0.817	29.89 / 0.834	29.84 / 0.833	29.56 / 0.827
	$s = 4$	25.99 / 0.703	27.00 / 0.735	27.29 / 0.746	28.19 / 0.772	28.15 / 0.772	27.69 / 0.759
No of parameters		-	8k	105k	813K	1,478k	236k

The results of LapSRN and SRMD models are taken from Zhang et al. [194].

<sup>3</sup>Part of these results was published at the European Signal Processing (EUSIPCO) conference 2019, Spain [9].

### Non-Blind and Blind Scenarios

Table 4.11 shows the evaluation of the performance of DBSR on images with different degrees of blur. We have considered two different scenarios: the non-blind and blind scenarios. The non-blind scenario corresponds to the case when the network is trained and tested on images with the same  $\sigma$  in  $N(0, \sigma)$ ;  $\sigma = 1, 2, 3$  or  $4$ . In the blind scenario, the network is trained on images with kernel width  $N(0, \sigma)$  with  $\sigma$  ranging between  $[0.5-3]$  /  $[1-3]$  / or  $[1-4]$ . The blind models are tested on images at any kernel width value. The quantitative results (PSNR/ SSIM) for Set5 and Set14 point out that DBSR enhances the quality of images over SRCNN and DBSRCNN models, for both non-blind and blind scenarios. A possible explanation is that the added enhanced layers led to improved results relying on cleaner features with less noise.

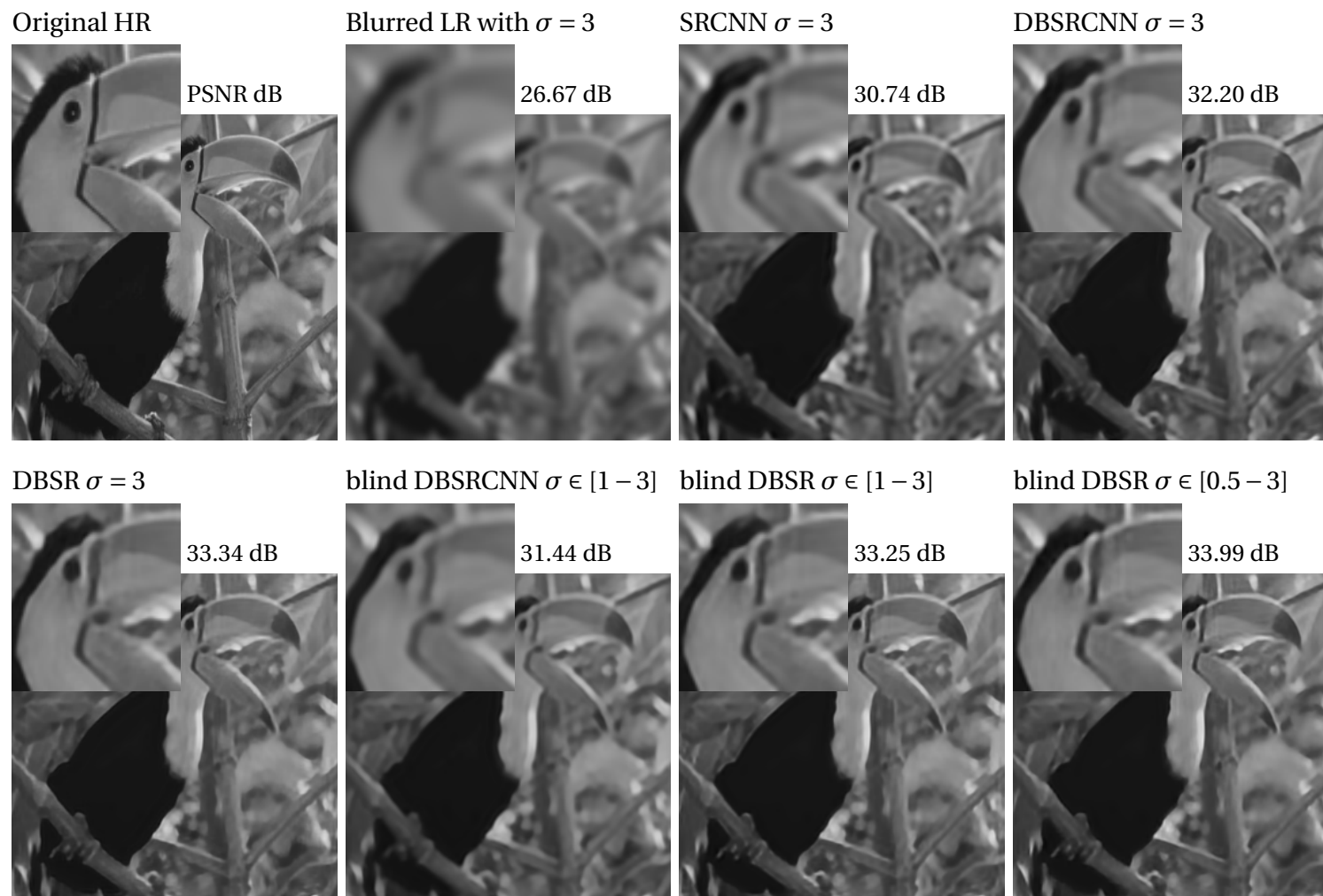
Although the performance of blind networks (SRCNN and DBSRCNN) decreases when training with  $\sigma = [1-4]$  comparing with when training with  $\sigma = [1-3]$ , the DBSR for  $\sigma = [1-4]$  gives better performance than these models. This means that the capacity of the model is critical when dealing with a problematic task such as a high level of blurring. Also, the results of DBSR with  $\sigma = [0.5, 3]$  are better than DBSR with  $\sigma = [1, 3]$ . The reason for that improvement could be the use of data augmentation for training the network. Qualitative comparison of reconstruction are shown in Figure 4.7. More examples are presented in Appendix B.

**Table 4.11** Average PSNR (dB)/ and SSIM results with different blur levels  $\sigma = 1, 2, 3, 4$ , scale factor  $s = 3$  on 'Set5' and 'Set14'.

Dataset	Kernel Width	LR Input	Non-blind Networks			Blind Networks				
			SRCNN	DBSRCNN	DBSR	DBSRCNN $\sigma \in [1, 3]$	DBSR $\sigma \in [1, 3]$	DBSR $\sigma \in [0.5, 3]$	DBSRCNN $\sigma \in [1, 4]$	DBSR $\sigma \in [1, 4]$
			PSNR/ SSIM							
Set5	$\sigma = 1$	29.47/ 0.847	31.55/ 0.892	32.65/ 0.907	33.24/ 0.917	31.24/ 0.888	31.50/ 0.898	32.60/ 0.909	30.85/ 0.881	31.47/ 0.899
	$\sigma = 2$	27.45/ 0.789	30.29/ 0.873	32.09/ 0.897	33.05/ 0.914	30.14/ 0.884	31.51/ 0.896	31.78/ 0.900	30.08/ 0.862	31.27/ 0.893
	$\sigma = 3$	25.65/ 0.724	29.03/ 0.819	30.48/ 0.858	31.36/ 0.879	29.51/ 0.840	31.01/ 0.878	31.67/ 0.886	28.40/ 0.820	30.15/ 0.868
	$\sigma = 4$	24.33/ 0.672	27.36/ 0.763	28.69/ 0.803	29.83/ 0.840	26.28/ 0.746	26.29/ 0.747	26.34/ 0.746	27.72/ 0.782	29.02/ 0.822
Set14	$\sigma = 1$	26.86/ 0.745	28.40/ 0.805	29.11/ 0.818	29.56/ 0.827	28.38/ 0.805	28.70/ 0.816	29.10/ 0.820	28.09/ 0.798	28.65/ 0.815
	$\sigma = 2$	25.37/ 0.679	27.41/ 0.780	28.80/ 0.808	29.47/ 0.825	27.43/ 0.765	28.53/ 0.801	28.80/ 0.813	27.28/ 0.758	28.41/ 0.798
	$\sigma = 3$	24.04/ 0.617	26.33/ 0.712	27.50/ 0.753	28.00/ 0.782	26.68/ 0.722	27.80/ 0.770	28.29/ 0.788	25.99/ 0.709	27.39/ 0.765
	$\sigma = 4$	23.05/ 0.574	25.19/ 0.659	26.28/ 0.698	27.06/ 0.742	24.55/ 0.635	24.60/ 0.644	24.67/ 0.648	25.46/ 0.668	26.65/ 0.723

The results in gray colour indicate the mismatched degradation assumptions in training and testing.

The results in blue colour indicate the best results; while the results in green colour are the second-best results.



**Fig. 4.7** SR with different models on images after Gaussian blur with  $\sigma = 3$ . The results show the non-blind and blind scenarios. Each result is accompanied by zoom and PSNR.

In Table 4.12, we follow the comparison presented in Zhang et al. [194], where Gaussian blur with  $\sigma = 1.3$  and  $\sigma = 2.6$  and scale factor  $s = 3$  was considered on the Set5 dataset. We compare our model with VDSR [81], SRMD [194] and model-based methods such as IRCNN [192]. The VDSR performance declines severely when the assumed degradation differs from the true one because the VDSR model is designed for bicubic degradation which results from LR only. This occurs when the true and assumed degradations are mismatched. Our model provides good performance compared to other models. An example of qualitative comparison of reconstruction is shown in Figure 4.8. In particular, it can be observed that SRMD, the best performing model in terms of PSNR, reports SR results of comparable visual quality with the proposed DBSR, whereas the more realistic DBSR blind pipeline achieves slightly worse sharpness (see zoom in Figure 4.8).

**Table 4.12** Average PSNR (dB)/ and SSIM results with different kernel width of blur kernel with scale factor  $s = 3$  on 'Set5'.

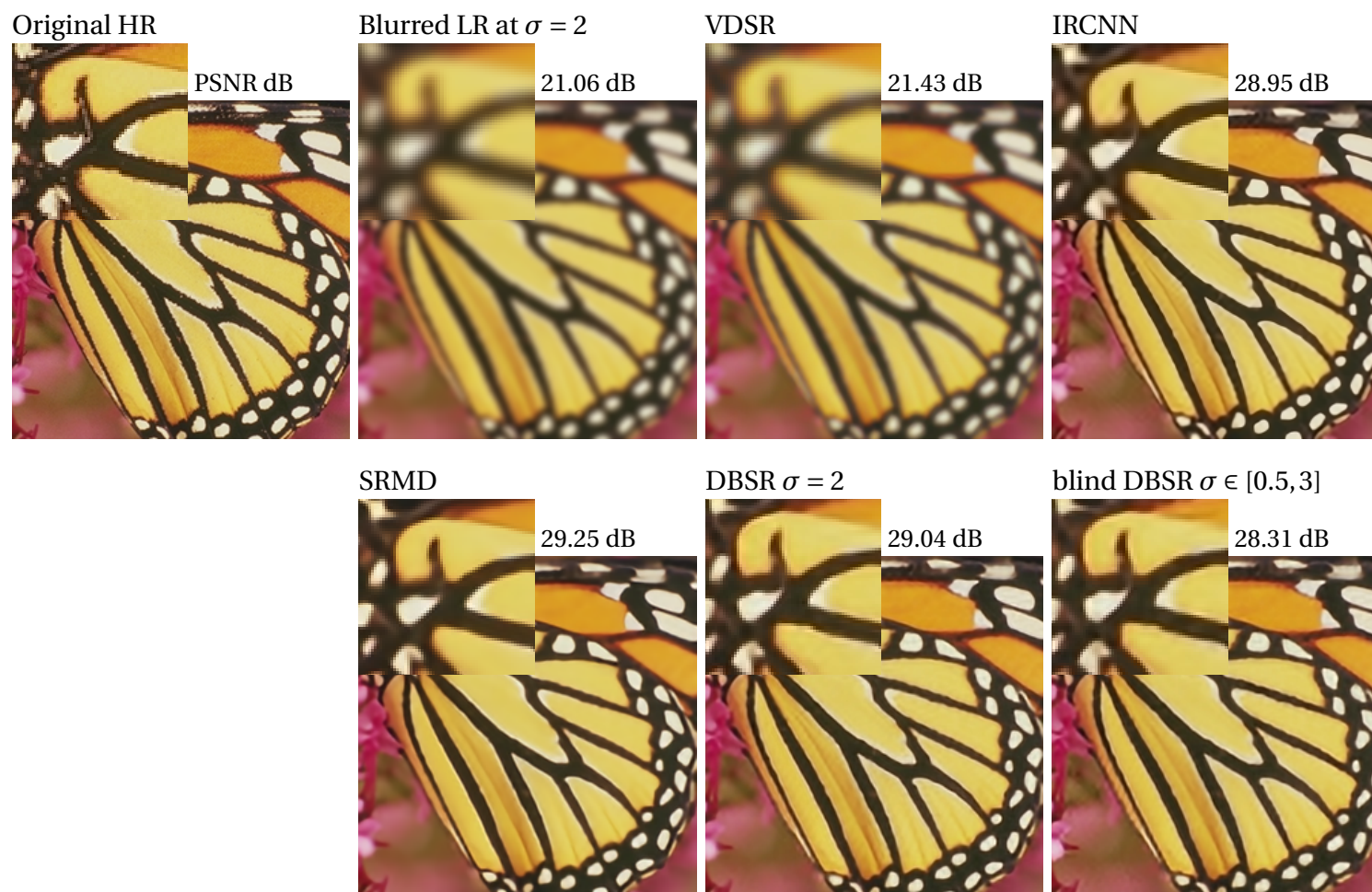
Kernel Width	LR Input	VDSR [81]	IRCNN [192]	SRMD [194]	DBSR (Ours) $\sigma \in [0.5, 3]$
$\sigma = 0.2$	30.39/0.868	33.67/0.921	33.39/ 0.939	33.86/ 0.923	31.74/ 0.901
$\sigma = 1.3$	28.84/ 0.831	30.24/ -	33.31/ 0.919	33.77/ 0.922	32.70/ 0.909
$\sigma = 2.6$	26.17/ 0.744	26.31/ -	31.48/ 0.862	32.59/ 0.900	31.85/ 0.895

The results of VDSR model is taken from Zhang et al. [194].

We have gathered the results of IRCNN and SRMD from rerunning their codes which are available online, available at <https://github.com/cszn/SRMD> and <https://github.com/cszn/IRCNN>, respectively.

The results in gray color indicate mismatched degradation assumption in training and testing. The results in blue colour indicate the best results; while the results in green colour are the second-best results.

The VDSR model has been trained on 291 images; 91 images from Yang et al. [180], and 200 images from the Berkeley Segmentation Dataset (BSD) [110]. While the IRCNN net has been trained using a collected large dataset that comprises 400 selected images from the validation set of ImageNet database [38], 4,744 images of the Waterloo Exploration Database [103], and 400 BSD images [110]. Also, the SRMD network has been trained on large-scale color images, including 800 training images from the DIV2K dataset [2], 4,744 images from the WED dataset [103] and 400 BSD images [110] are also used.



**Fig. 4.8** SISR performance of different models on 'Butterfly' image after Gaussian blur at  $\sigma = 2$ . In the blind scenario  $\sigma \in [0.5, 3]$ .

## 4.2.4 Evaluation of Harm-DBSR

### Non-Blind and Blind Scenarios

In this part of the study, we first investigate whether the DBSR results can be enhanced if the DBSR is trained on spectral information, by replacing the standard convolutional layers with harmonic blocks; a network which we have named the Harm-DBSR network. We have validated the performance of the Harm-DBSR with different modifications; from a starting point of replacing the first convolution layer with a harmonic block, we reached a point where we chose to replace all convolutional layers. Generally, utilising harmonic blocks in standard DBSR, for all modifications we have done, gives better results than using convolutional operations alone. However, the best results are produced when replacing the first three layers of DBSR with harmonic blocks (Harm3L-DBSR) as reported in Tables 4.13, 4.15 and 4.14 for the testing datasets Set5 and Set14, for non-blind and blind SR scenarios respectively. It is apparent from these tables that the Harm-DBSR version for each convolutional network produces better results thanks to its adoption of the spectral information. For example, Harm-DBSR compared to DBSR for the non-blind SR scenario, and Harm-DBSR  $\sigma \in [1 - 3]$  compared to DBSR  $\sigma \in [1 - 3]$  for blind SR. In Table 4.15, Harm-DBSR  $\sigma \in [0.5 - 3]$  produced the best results for corrupted inputs with  $\sigma = 1, 2$  or  $3$ , while the second-best results were for DBSR  $\sigma \in [0.5 - 3]$ . However, the best results were provided for corrupted images with  $\sigma = 4$  by Harm-DBSR with  $\sigma \in [1 - 4]$  then by DBSR with  $\sigma \in [1 - 4]$  since these networks trained on  $\sigma = 4$ ; this is as expected because these networks are trained on the blur level  $\sigma = 4$ . An example of qualitative comparison of reconstruction using the DBSR model and the Harm-DBSR version is shown in Figure 4.9.



**Table 4.13** Average PSNR (dB)/ and SSIM results for non-blind networks with different blur levels at  $\sigma = 1, 2, 3, 4$ , scale factor  $s = 3$  on 'Set5' and 'Set14'.

Dataset	Kernel Width	LR Input	SRCNN	DBSRCNN	DBSR	Harm3L-DBSR
Set5	$\sigma = 1$	29.47/ 0.847	31.55/ 0.892	32.65/ 0.907	33.24/ 0.917	33.39/ 0.918
	$\sigma = 2$	27.45/ 0.789	30.29/ 0.873	32.09/ 0.897	33.05/ 0.914	33.25/ 0.917
	$\sigma = 3$	25.65/ 0.724	29.03/ 0.819	30.48/ 0.858	31.36/ 0.879	31.82/ 0.892
	$\sigma = 4$	24.33/ 0.672	27.36/ 0.763	28.69/ 0.803	29.83/ 0.840	30.03/ 0.848
Set14	$\sigma = 1$	26.86/ 0.745	28.40/ 0.805	29.11/ 0.818	29.56/ 0.827	29.66/ 0.829
	$\sigma = 2$	25.37/ 0.679	27.41/ 0.780	28.80/ 0.808	29.47/ 0.825	29.55/ 0.828
	$\sigma = 3$	24.04/ 0.617	26.33/ 0.712	27.50/ 0.753	28.00/ 0.782	28.32/ 0.795
	$\sigma = 4$	23.05/ 0.574	25.19/ 0.659	26.28/ 0.698	27.06/ 0.742	27.35/ 0.749

The results in blue colour indicate the best results; while the results in green colour are the second-best results.

**Table 4.14** Average PSNR (dB)/ and SSIM results with different kernel width of blur kernel with scale factor  $s = 3$  on 'Set5'. Our presented results of DBSR and Harm3L-DBSR are from blind scenarios.

Kernel Width	LR Input	VDSR[81]	IRCNN[192]	SRMD[194]	DBSR (Ours) $\sigma \in [0.5, 3]$	Harm3L-DBSR (Ours) $\sigma \in [0.5, 3]$
$\sigma = 0.2$	30.39/ 0.868	33.67/ 0.921	33.39/ 0.939	33.86/ 0.923	31.74/ 0.901	32.10/ 0.905
$\sigma = 1.3$	28.84/ 0.831	30.24/ -	33.31/ 0.919	33.77/ 0.922	32.70/ 0.909	32.91/ 0.912
$\sigma = 2.6$	26.17/ 0.744	26.31/ -	31.48/ 0.862	32.59/ 0.900	31.85/ 0.895	31.79/ 0.894

The results of VDSR model are taken from Zhang et al. [194].

We obtained the results of IRCNN and SRMD from re-running their codes which are available online, available at <https://github.com/cszn/SRMD> and <https://github.com/cszn/IRCNN>, respectively.

The results in gray colour indicate mismatched degradation assumption in training and testing.

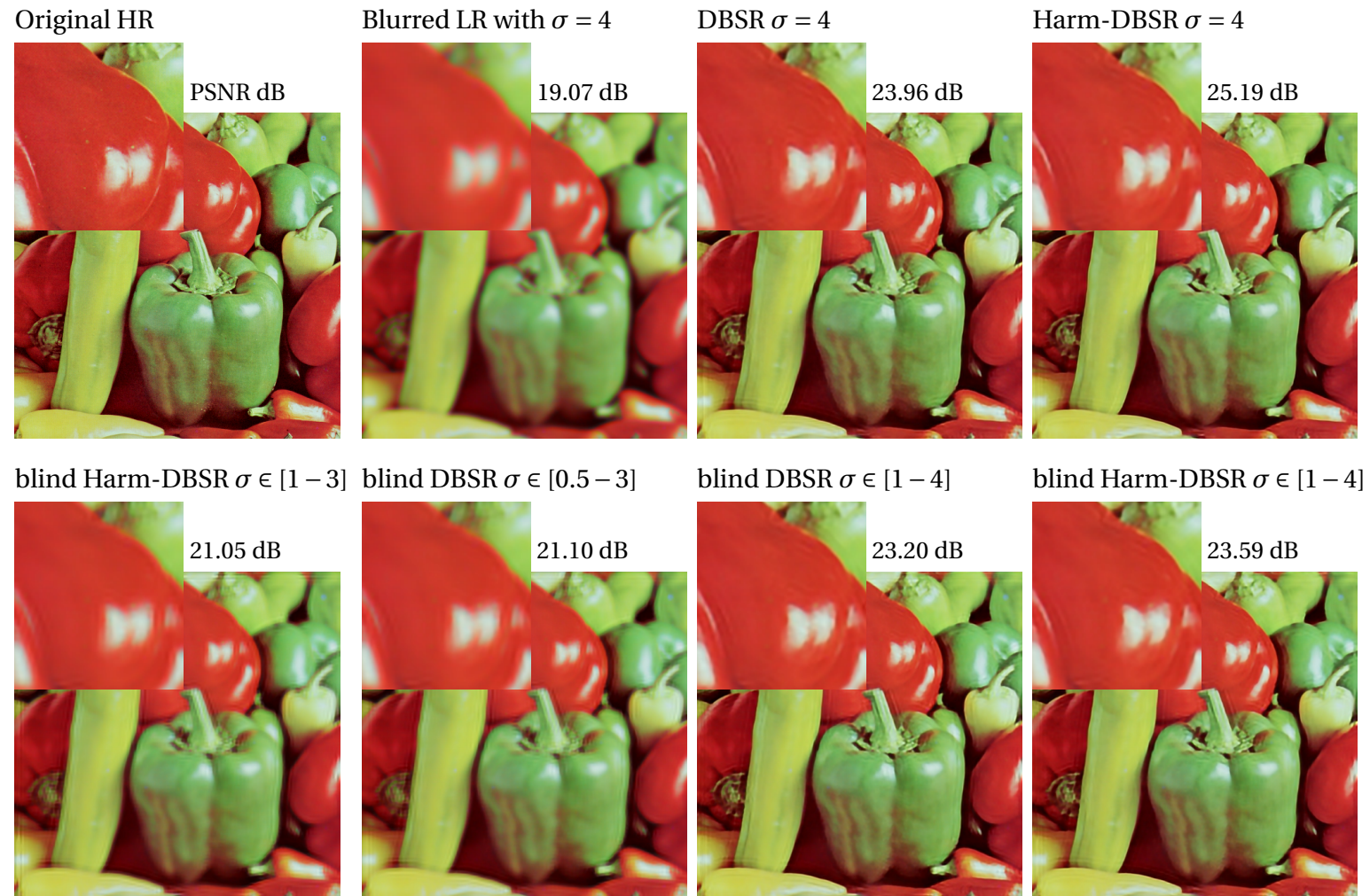
The results in blue colour indicate the best results; while the results in green colour are the second-best results.

**Table 4.15** Average PSNR (dB)/ and SSIM results for blind networks with different blur levels  $\sigma = 1, 2, 3, 4$ , scale factor  $s = 3$  on 'Set5' and 'Set14'.

Dataset	Kernel Width	LR Input	DBSRCNN $\sigma \in [1, 3]$	DBSR $\sigma \in [1, 3]$	Harm3L-DBSR $\sigma \in [1, 3]$	DBSR $\sigma \in [0.5, 3]$	Harm3L-DBSR $\sigma \in [0.5, 3]$	DBSRCNN $\sigma \in [1, 4]$	DBSR $\sigma \in [1, 4]$	Harm3L-DBSR $\sigma \in [1, 4]$
Set5	$\sigma = 1$	29.47/ 0.847	31.24/ 0.888	31.50/ 0.898	31.75/ 0.904	32.60/ 0.909	32.85/ 0.912	30.85/ 0.881	31.47/ 0.899	31.62/ 0.902
	$\sigma = 2$	27.45/ 0.789	30.14/ 0.884	31.51/ 0.896	31.82/ 0.901	31.78/ 0.900	32.33/ 0.906	30.08/ 0.862	31.27/ 0.893	31.52/ 0.896
	$\sigma = 3$	25.65/ 0.724	29.51/ 0.840	31.01/ 0.878	31.23/ 0.881	31.67/ 0.886	32.03/ 0.895	28.40/ 0.820	30.15/ 0.868	30.55/ 0.875
	$\sigma = 4$	24.33/ 0.672	26.28/ 0.746	26.29/ 0.747	26.28/ 0.746	26.34/ 0.746	26.07/ 0.736	27.72/ 0.782	29.02/ 0.822	29.25/ 0.826
Set14	$\sigma = 1$	26.86/ 0.745	28.38/ 0.805	28.70/ 0.816	28.92/ 0.821	29.10/ 0.820	29.23/ 0.823	28.09/ 0.798	28.65/ 0.815	28.84/ 0.819
	$\sigma = 2$	25.37/ 0.679	27.43/ 0.765	28.53/ 0.801	28.82/ 0.807	28.80/ 0.813	29.08/ 0.818	27.28/ 0.758	28.41/ 0.798	28.60/ 0.803
	$\sigma = 3$	24.04/ 0.617	26.68/ 0.722	27.80/ 0.770	27.94/ 0.776	28.29/ 0.788	28.55/ 0.800	25.99/ 0.709	27.39/ 0.765	27.70/ 0.774
	$\sigma = 4$	23.05/ 0.574	24.55/ 0.635	24.60/ 0.644	24.59/ 0.644	24.67/ 0.648	24.42/ 0.636	25.46/ 0.668	26.65/ 0.723	26.80/ 0.729

The results in gray color indicate the mismatched degradation assumptions in training and testing.

The results in blue colour indicate the best results; while the results in green colour are the second-best results.



**Fig. 4.9** SR with different models on images after Gaussian blur with  $\sigma = 4$ . The results show the non-blind and blind scenarios. Each result is accompanied by zoom and PSNR.

## Compression of the Harm-DBSR Network

For the purpose of this research, we want to know the performance of the networks after compression by using part of the DCT coefficients inside the harmonic blocks. Consequently, we retrained the DBSR with  $\sigma = [1, 3]$  with a different number of levels of coefficients  $\lambda$ , as shown in Table 4.16. Although the DBSR  $\sigma = [0.5, 3]$  gives the best performance, DBSR  $\sigma = [1, 3]$  is faster in training; therefore we have adopted the DBSR  $\sigma = [1, 3]$  as a baseline to perform the comparison between the performance of standard networks and the compressed nets. We have kept the model structure and the training data to train the different networks, we changed only all convolutional layers with harmonic blocks (denoted as Harm8L-DBSR), and then used different  $\lambda$  to get different nets with various levels of compression. Firstly, the harmonic network Harm8L-DBSR outperforms the standard DBSR, see Table 4.16. We trained various compressed harmonic networks on limited spectral information of hidden features with  $\lambda = 5, 4, 3, 2$ ; by applying limited DCT coefficients for each feature = 15, 10, 6, 3 respectively. Also, we have trained a network with  $\lambda = 6$  for the first layer and  $\lambda = 4$  for the remaining seven layers (denoted as  $\lambda = 6/4L$ ). The second-best result after Harm8L-DBSR, which uses the full spectrum, is Harm8L-DBSR with  $\lambda = 6/4L$ , by using about two and a half times fewer parameters. Also, from the results, we can notice that the compressed networks with  $\lambda = 5, 4$  and 3 except  $\lambda = 2$  give good results when compared to the standard convolutional DBSR. The performance of Harm8L-DBSR with  $\lambda = 5$  is slightly lower than the full harmonic network Harm8L-DBSR, however, it still outperforms the DBSR. Harm8L-DBSR with  $\lambda = 4$  and 3 gives a competitive performance with DBSR, notably with  $\lambda = 3$  by employing around four times fewer parameters. Besides, the performance degradation when implementing  $\lambda = 2$  could return to the fact that some truncated high-frequencies coefficients carry important features for the task of restoration. Figure 4.10 shows an example of the DBSR network sizes versus their performance; results for the blind convolutional network DBSR  $\sigma = [1, 3]$ , and its compression using the harmonic blocks. An example of a qualitative comparison of the reconstruction using the Harm8L-DBSR net with various levels of compression is shown in Figure 4.11.

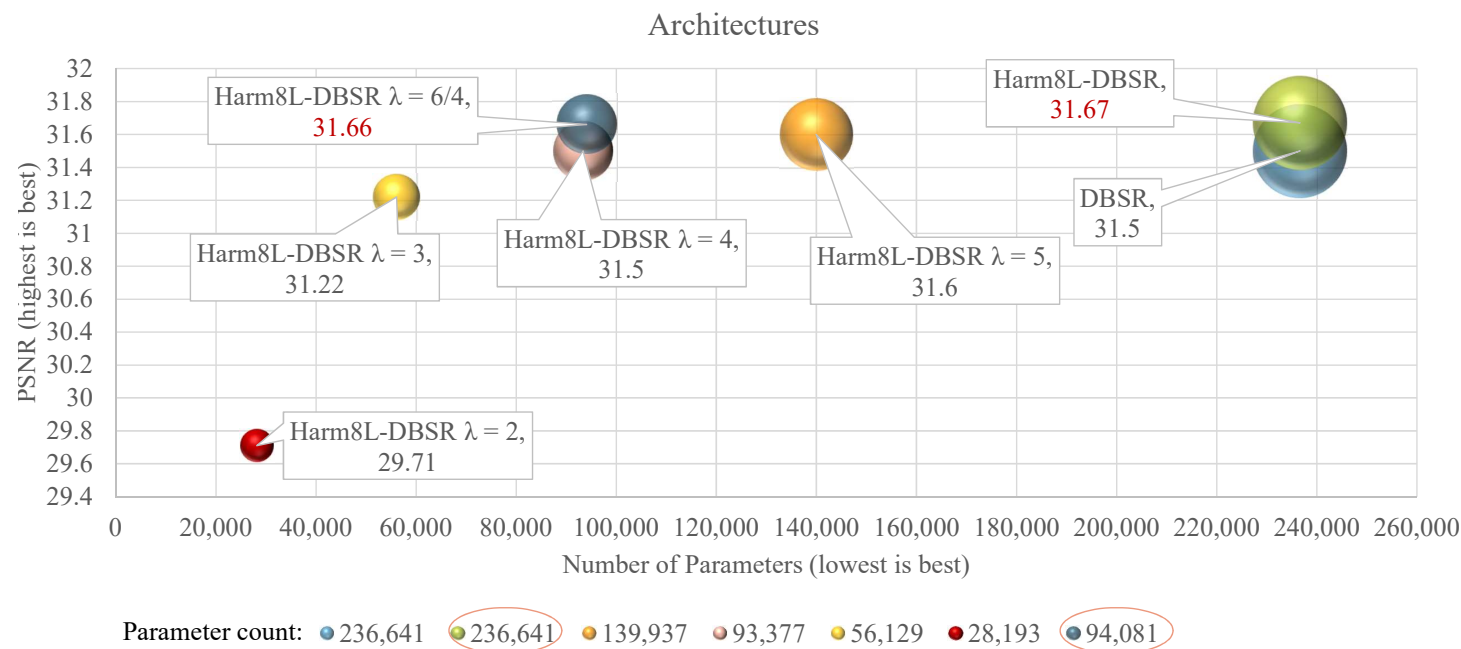
**Table 4.16** Average PSNR (dB)/ and SSIM results for the blind convolutional network DBSR  $\sigma = [1, 3]$ , and its compression using the harmonic blocks; the results are calculated for different blur levels  $\sigma = 1, 2, 3$ , scale factor  $s = 3$  on 'Set5' and 'Set14'.

Dataset	Kernel Width	LR Input	DBSR	Harm8L-DBSR	Harm8L-DBSR $\lambda = 5^*$	Harm8L-DBSR $\lambda = 4^*$	Harm8L-DBSR $\lambda = 3^*$	Harm8L-DBSR $\lambda = 2^*$	Harm8L-DBSR $\lambda = 6, 4/L^{**}$
Set5	$\sigma = 1$	29.47/ 0.847	31.50/ 0.898	<b>31.67/ 0.903</b>	31.60/ 0.900	31.50/ 0.899	31.22/ 0.893	29.71/ 0.853	<b>31.66/ 0.901</b>
	$\sigma = 2$	27.45/ 0.789	31.51/ 0.896	<b>31.74/ 0.899</b>	31.61/ 0.897	31.56/0.896	31.23/ 0.890	30.07/ 0.861	<b>31.62/ 0.898</b>
	$\sigma = 3$	25.65/ 0.724	31.01/ 0.878	<b>31.27/ 0.883</b>	<b>31.16/ 0.880</b>	31.10/ 0.879	30.82/ 0.872	29.47/ 0.838	<b>31.10 / 0.880</b>
Set14	$\sigma = 1$	26.86/ 0.745	28.70/ 0.816	<b>28.82/ 0.819</b>	28.78/ <b>0.818</b>	28.70/ 0.816	28.55/0.811	27.33/ 0.769	<b>28.81/ 0.818</b>
	$\sigma = 2$	25.37/ 0.679	28.53/ 0.801	<b>28.75/ 0.806</b>	28.67/ 0.803	28.58/ 0.803	28.28/ 0.794	27.36/ 0.759	<b>28.68/ 0.804</b>
	$\sigma = 3$	24.04/ 0.617	27.80/ 0.770	<b>28.00/ 0.777</b>	<b>27.90/ 0.774</b>	27.85/ 0.773	27.55/0.761	26.60/ 0.720	<b>27.87/ 0.774</b>
Number of parameters			236,641	236,641	139,937	93,377	56,129	28,193	94,081

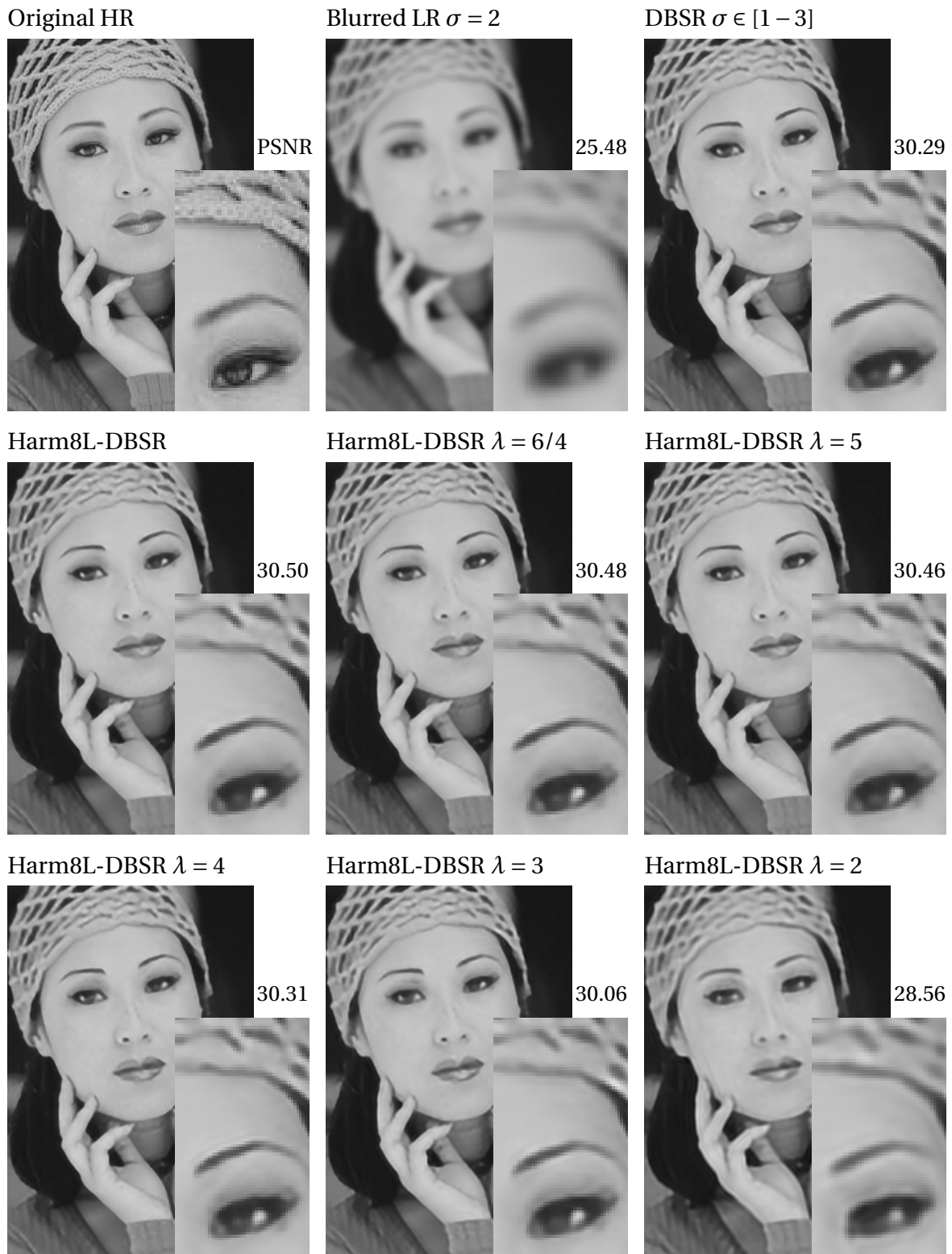
\* means the number of levels of coefficients used in all harmonic layers, where for  $\lambda = 5, 4, 3, 2 \Rightarrow$  the number of coefficients used in each layer = 15, 10, 6, 3 respectively. For \*\*, we have used  $\lambda = 6$  for the first layer and  $\lambda = 4$  for the remaining seven layers.

- The results in **blue** colour indicate the best results; while the results in **green** colour are the second-best results.

- All the networks introduced in this table were trained on the same training data, where the only difference is the compression degree.



**Fig. 4.10** Average PSNR/ and SSIM results for the blind convolutional network DBSR  $\sigma = [1,3]$ , and its compression using the harmonic blocks; the results are calculated for  $\sigma = 1$ , scale factor  $s = 3$  on 'Set5'. The Bubble size is according to the number of parameters. The full harmonic network Harm8L-DBSR outperforms the standard DBSR, by using the same number of parameters. From the results, we can notice that the compressed networks with  $\lambda = 5, 4$  and  $3$  except  $\lambda = 2$  give good results when compared to the standard convolutional DBSR. However, the network with  $\lambda = 6/4L$  used about two and a half times fewer parameters gives the second-best result after Harm8L-DBSR. Adopting the DCT filters enabled us to compress the harmonic nets and use fewer parameters by choosing the most critical low-frequency coefficients.



**Fig. 4.11** SR with different models on images after Gaussian blur with  $\sigma = 2$ . The results show the blind scenario  $\sigma = [1, 3]$ . Each result is accompanied by zoom and PSNR dB.

### 4.3 Conclusion

In this chapter, we have extensively evaluated the performance of the SRCNN architecture for recovering high-resolution images from low-resolution images corrupted by blur or noise. The SRCNN network has worked well to recover SR images from LR ones. However, the higher the blurring levels in images in addition to the decrease in resolution, the worse the results. Although the improvement by using SRCNN vs baseline increases with increased blurring levels, the results are still unsatisfactory. We note that low-level features extracted using a single layer (the first layer) in the SRCNN network are still blurred, making the model less than optimal for tackling noise. Therefore, we have proposed a new architecture DBSRCNN that enhances the reconstruction by boosting the relevant features that were originally lost in the SRCNN pipeline using the concatenation operation. Our experimental study with different levels of Gaussian blur demonstrates that our revised deeper architecture performs better in both non-blind and blind testing scenarios.

To make the architecture more efficient, we have presented the DBSR model. This model is an extension model for the DBSRCNN model, to which we have proposed adding convolutional layers to enhance the extracted features. We have reported a panel of comparisons with the state-of-the-art deep learning (e.g., VDSR[81], LapSRN [87] and SRMD[194]) and with model-based methods (e.g., IRCNN[192]), and we have highlighted the competitive performance of the proposed model DBSR for super-resolution on blurred images. Importantly, these results are obtained by a model with three to six times fewer parameters, where the SRMD model includes 1,478k of parameters, LapSRN has 813K parameters, but DBSR is designed with 236k of parameters.

Lastly, we have applied the harmonic blocks in the standard convolutional DBSR, to evaluate the usage of the spectrum information empirically. We have demonstrated that using spectral information has helped to enhance the results to a greater extent than using CNN alone, where the Harm-DBSR produced outputs that were better than the DBSR. Furthermore, adopting the DCT filters enabled us to compress the harmonic nets and use fewer parameters by choosing the most critical low-frequency coefficients. We have shown that the compressed networks gave a competitive performance with DBSR.

In the next chapters, we employ the proposed CNN architectures in different image reconstruction applications. In Chapter 5, we reduce the artefacts resulted from JPEG-



compression. In Chapter 6, we then use our proposed networks for a denoising application with a real dataset.



## Chapter 5

# Artefact Reduction in JPEG-Compressed Images

A variety of methods have been proposed to reduce the artefacts in JPEG compressed images. These can generally be divided into two main types; namely deblocking methods and restoration methods. Deblocking methods are focused on suppressing ringing and blocking artefacts, and various types of filters have been utilised in the spatial domain to remove the blocking artefacts [98, 117, 125, 166]. Wavelet transform has also been employed to perform denoising in the frequency domain [96]. One of the most modern state-of-the-art deblocking methods is the Pointwise Shape-Adaptive DCT (SA-DCT) [52]. This approach, however, is not capable of reproducing sharp edges and it oversmooths the image textures. On the other hand, restoration methods deal with the compression as a distortion operation. These methods include projection on convex sets based method (POCS) [184], solving a MAP problem (FoE) [145] and sparse-coding-based methods [77]. More recently, CNNs have been used to mitigate compression artefacts and restore distorted images [41, 149] delivering state-of-the-art results.

In this chapter, we aim to propose CNNs of reduced size that effectively suppress artefacts of JPEG-compressed images<sup>1</sup>. The size reduction allows one to handle training as well as deployment of such CNNs more efficiently. Firstly, we present a novel CNN (referred to as SA-CAR6) for image restoration, which delivers better performance than the state-

---

<sup>1</sup>Part of the results was published at the Irish Machine Vision and Image Processing (IMVIP) conference 2018, Belfast [6].

of-the-art CNN-based models [41, 149] and employs fewer parameters. Smaller models are a better option since they reduce the computational complexity, and allow us to avoid problems of large networks such as overfitting, vanishing or exploding gradients. Secondly, we consider several CNNs with different parameter counts (DA-CAR3, DA-CAR4, DA-CAR5) to show that we can achieve the same or better results using smaller networks with a lower parameter count. Section 5.1 discusses the related work. In Section 5.2, we present the dataset employed and in Section 5.3 the metrics used for evaluation and comparisons. Section 5.4 reports evaluation results and comparisons with state-of-the-art models. The conclusion is presented in Section 5.5.

## 5.1 Related Work

It has become commonplace to apply lossy compression, e.g., JPEG and HEVC-MSP, to images and videos to save storage space and bandwidth. Much attention has been attracted to enhancing the quality and compression power of algorithms, in particular, in large companies operating huge images and video volumes such as Facebook and Twitter. Compression is a family of data encoding techniques that often rely on inexact approximations to represent encoded content to achieve the most competitive performance. The compression approach leads to undesirable artefacts such as blurring, blocking and ringing, which are addressed by artefact reduction methods.

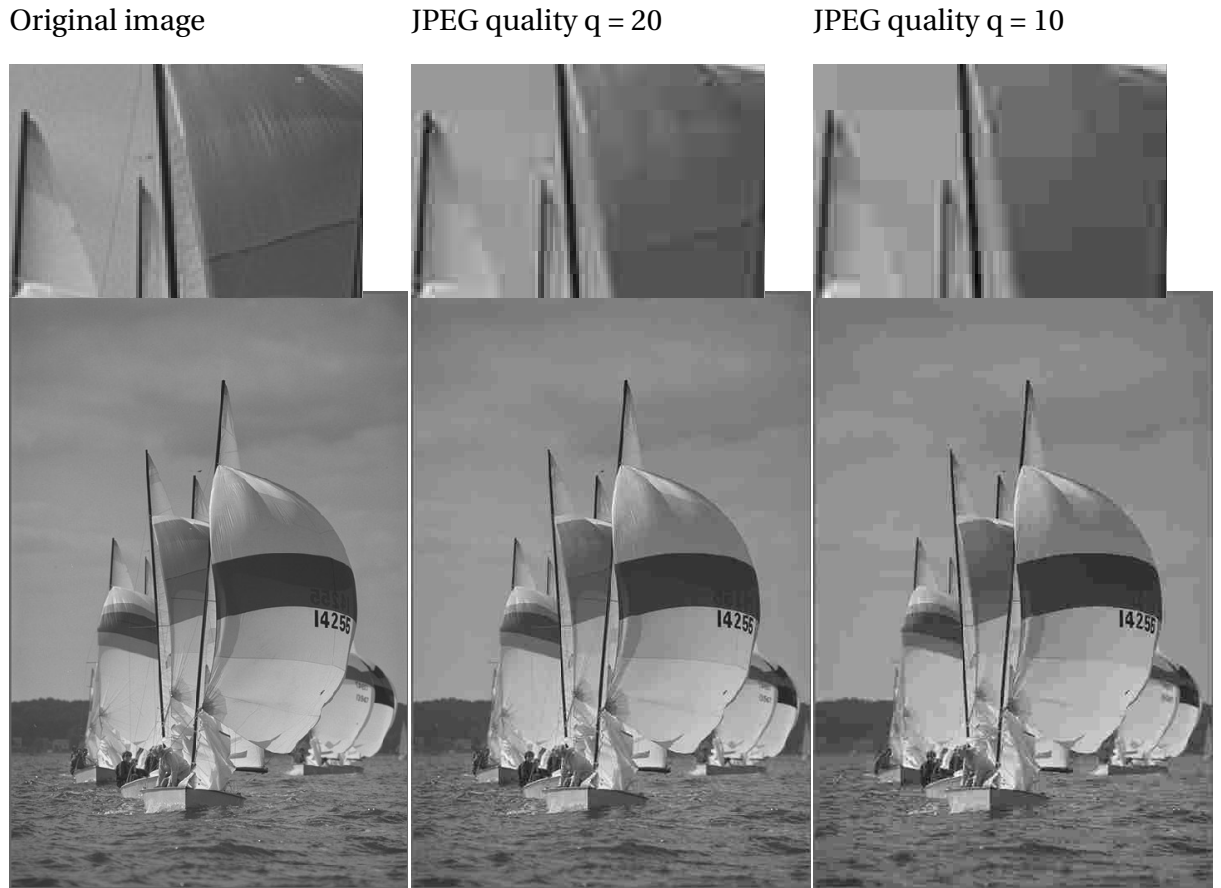
JPEG compression is probably the widest used lossy image compression approach. The underlying technique separates an image into  $8 \times 8$ -pixel blocks followed by the Discrete Cosine Transformation (DCT) being applied to each block separately. In order to achieve compression, quantisation is then performed on the DCT coefficients, which inevitably results in the appearance of visual artefacts. *Blurring* appears because the high-frequency components are lost. *Ringing*, also referred to as the *Gibbs phenomenon*, takes place because of the coarse quantisation of the high-frequency components. Finally, the *blocking* artefacts are due to the separate processing and encoding of blocks without consideration and, hence, loss of adjacency information between the blocks.

There are many different methods that have been proposed in the literature to deal with the numerous types of artefacts, some of which target specific types of compression side-effects. For example, *deblocking* methods are utilised to decrease blocking artefacts by

applying filters on or across the borders [98, 117, 125, 166], or using thresholding by wavelet transform [96] or Shape-Adaptive DCT transform (SA-DCT) [52]. Although these methods deal reasonably well with blocking artefacts and ringing effects, the results suffer from blurring. Other methods consider compression as distortion which requires reconstruction techniques [74, 77, 145, 184]. These methods output sharpened images featuring smooth regions and noisy edges.

Dong et al. utilised the CNN to remove the compression artefacts (ARCNN: Artefacts Removing using Convolutional Neural Network) [41]. The structure of ARCNN depends on Super Resolution CNN (SRCNN) which was proposed to infer high-resolution images from lower-resolution input [40]. The SRCNN architecture consists of three layers: the feature extraction layer, non-linear mapping layer and reconstruction layer, and this structure has been set according to the sparse coding method. The three-layer SRCNN was not sufficient to restore compressed images, so extra feature enhancement layers were added to SRCNN after the feature extraction layer to enhance the extracted features, which resulted in a four-layer ARCNN.

Svoboda et al. proposed two different convolutional networks for JPEG restoration called L4 and L8 Residual architectures [149]. Similar to Kim et al. for super-resolution [81], residual learning was adopted in L4 and L8, where the network learns the residual image  $\mathbf{r} = \mathbf{x} - \hat{\mathbf{x}}$ . The estimated residual image  $\mathbf{r}$  is then subtracted from the corrupted input  $\mathbf{x}$  to obtain the restored output image  $\hat{\mathbf{x}}$ , instead of learning the reconstructed whole image directly. Their approach is based on the idea of 'deeper is better', but deeper networks take a long time to train and may suffer from overfitting and vanishing or exploding gradients. To alleviate these problems, skip architecture was employed [101] as well as residual learning [68]. The skip architecture allows the network to 'shortcut' the input to the intermediate layers. This allows the more complex image content to be used in the middle layers as local context information, since this is critical for reducing the impact of artefacts.



**Fig. 5.1** Visualisation example from LIVE1 dataset of compressed corrupted quality images for sailing 2 image at JPEG quality  $q = 10$  (Lower quality) & 20 (higher quality).

## 5.2 Data and Training

The BSDS500 dataset [14] contains 400 training images that were used for training all networks. The validation set from BSDS500, which comprises 100 high-quality images, and the LIVE1 dataset [138] are used as test sets for the different networks. The LIVE1 dataset has 29 images in uncompressed BMP format; this dataset is often used for super-resolution [178] and image quality assessment [169]. The training images are split into  $33 \times 33$  sub-images  $\{y_i\}$ . These sub-images are generated from the ground truth images with a stride of 10, to create 521,984 sub-images. We have transformed the RGB images into a YCbCr colour model to deal with grey-scale by using the Y channel (luminance component) since our

focus is on removing the different artefacts, not on chrominance distortions. The network is trained on a batch size of 64 with learning rate  $\alpha = 0.002$ . The JPEG-compressed images  $\{\mathbf{x}_i\}$  are generated from the grey-scale sub-images using MATLAB JPEG encoder. We have used two different quality settings:  $q = 10$  (low quality) and  $q = 20$  (mild quality). Figure 5.1 displays a qualitative example of an image with different qualities. The number of training epochs is set to 60. All implementations are in Keras (Python, TensorFlow backend), with training and experiments run on the following system: Intel(R) Core i7 CPU (2.80 GHz), 16GB RAM, GTX Geforce 1050 GPU.

### 5.3 Quantitative Metrics for Comparisons

Three different metrics are employed to evaluate the quantitative performance and assess the perceptual quality of the restored images. The first metric is the peak signal-to-noise-ratio (PSNR) that is related to the MSE used as a cost function for optimising the networks. PSNR does not provide an assessment of perceptual quality. The second metric is the structural similarity (SSIM) [169] which quantifies the perceptual quality. PSNR-B [185] is the third metric considered. This metric constitutes a modification of PSNR which takes into account the blocking effect factor (BEF). PSNR-B is more sensitive to blocking than SSIM.

### 5.4 Benchmark Comparisons

There is a rule in DL says "The deeper is better", but here we want to show that smaller CNNs with good design and carefully chosen filter size can give outcomes which outperform the results of deeper structures. Therefore, we have proposed different CNNs of reduced size that effectively suppress artefacts of JPEG-compressed images compared to the state-of-the-art deeper structures.

AR-CNN (9-7-1-5) [41] is based on feature enhancement and an extra layer was originally added after the feature extraction layer in SRCNN (9-1-5) to further refine and enhance the features, as we discussed in section 5.1. However, our DA-CAR3 (9-7-5) architecture allows us to achieve the same performance by using only three layers when using suitable filter sizes; see Table 5.1 and Table 5.3 for quality = 10. Therefore, we have concluded that

the ARCNN (9-7-1-5) enhancement is not due to the extra features enhancement layer added to the SRCNN (9-1-5) model as the authors argued in [41], where we have achieved almost the same result using only three layers in the DA-CAR3 (9-7-5) model. We conclude that this improvement is attributed to utilising a larger filter size in the mapping stage, which indicates that using the neighbouring information during mapping is useful.

Our proposed network DA-CAR3 with three layers using direct architecture with filter sizes (9-7-5) performs better than both the three layers SRCNN (9-1-5) and the four layers AR-CNN (9-7-1-5) while roughly using the same number of parameters. Our proposed DA-CAR4 network has 30% of the parameters of AR-CNN, see Table 5.1 and Table 5.3, with equivalent performance to DA-CAR3 and AR-CNN models. This is achieved by selecting a suitable filter size with the number of layers. The DA-CAR5 network achieves even better results using more parameters than DA-CAR4 but still fewer by 75% of the parameters of the AR-CNN network. Furthermore, DA-CAR5 is slightly better than L4 for quality  $q = 10$ , while the results of L4 on LIVE1 and BSDS500 datasets are better than AR-CNN for JPEG quality  $q=10$  (see Tables 5.1 and 5.3) and for JPEG quality  $q=20$  (see Tables 5.2 and 5.4). From these comparisons, we can conclude that deeper networks are not usually better and sometimes give worse results than smaller networks. L8 (which merges layers 4+1 and 6+1), and which has been trained for the quality  $q = 20$  only [149], is better than L4 (see Table 5.2 and 5.4). Svoboda et al. used residual learning and skip architecture to avoid the problems inherent in deeper networks. Our new architecture SA-CAR6 (which merges layers 2+1) outperforms L4 and L8, by using around 60% of the number of parameters of L8; see Table 5.2 and Table 5.4. Figures 5.2 and 5.3 illustrate examples of proposed CNN structures of different sizes versus their performance for quality =10 and 20, respectively. Figures 5.4, 5.5 and 5.6 present image processing results obtained by employing the proposed and benchmark architectures.



**Table 5.1** Average image reconstruction results reported by PSNR (dB), PSNR-B (dB) and SSIM on the LIVE1 dataset for JPEG quality  $q = 10$ .

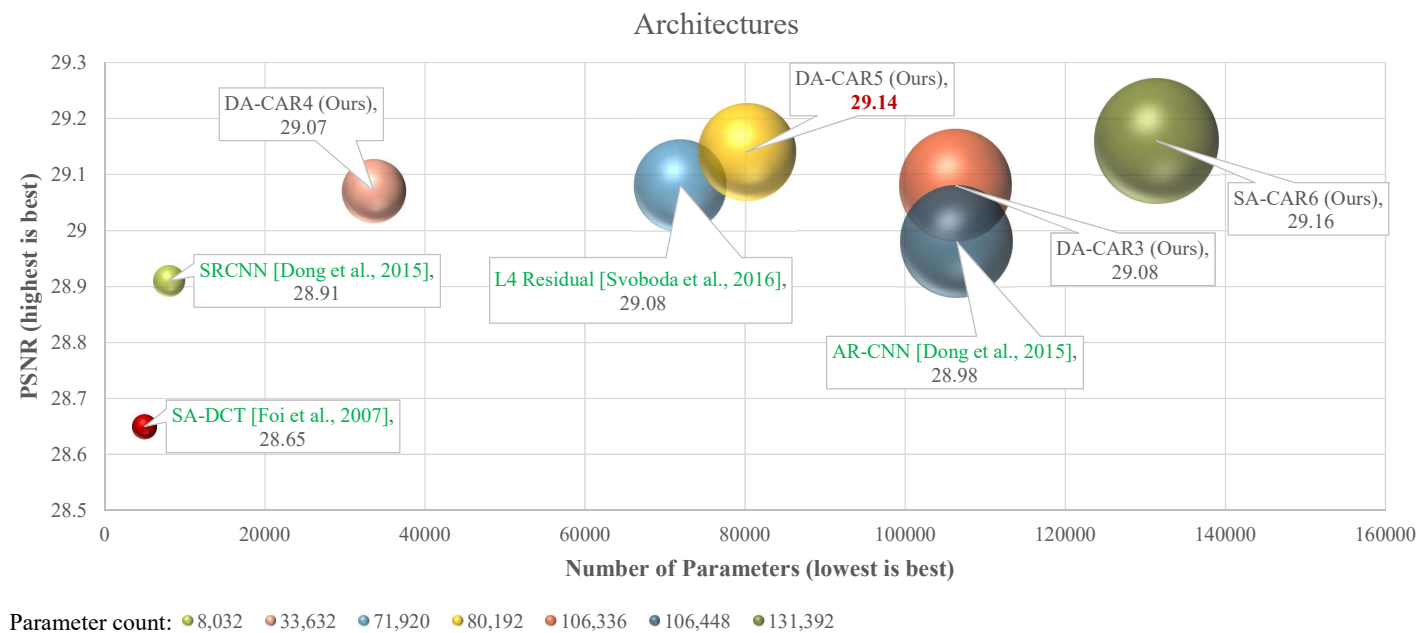
Architecture	Number of Layers	Filter size	Feature maps	PSNR	PSNR-B	SSIM	Parameter count
Input JPEG	-	-	-	27.77	25.33	0.791	-
SA-DCT [52]	-	-	-	28.65	28.01	0.809	-
SRCNN [41]	3	(9-1-5)	(64-32-1)	28.91	28.52	0.818	8,032
AR-CNN [41]	4	(9-7-1-5)	(64-32-16-1)	28.98	28.70	0.822	106,448
DA-CAR3 (Ours)	3	(9-7-5)	(64-32-1)	29.08	28.71	0.823	106,336
DA-CAR4 (Ours)	4	(9-3-3-5)	(64-32-32-1)	29.07	28.71	0.823	33,632
DA-CAR5 (Ours)	5	(9-5-5-5-5)	(32-32-32-32-1)	29.14	28.77	0.825	80,192
L4 Residual [149]	4	(11-3-3-5)	(48-64-64-1)	29.08	28.71	0.824	71,920
SA-CAR6 (Ours)	6 (2+1)	(9-5-5-5- 5-5)	(32-32-32-32- 32-1)	29.16	28.80	0.826	131,392

The results in blue colour indicate the best results, and the results in green colour are the second-best results. Lower quality, therefore, it gives lower PSNR.

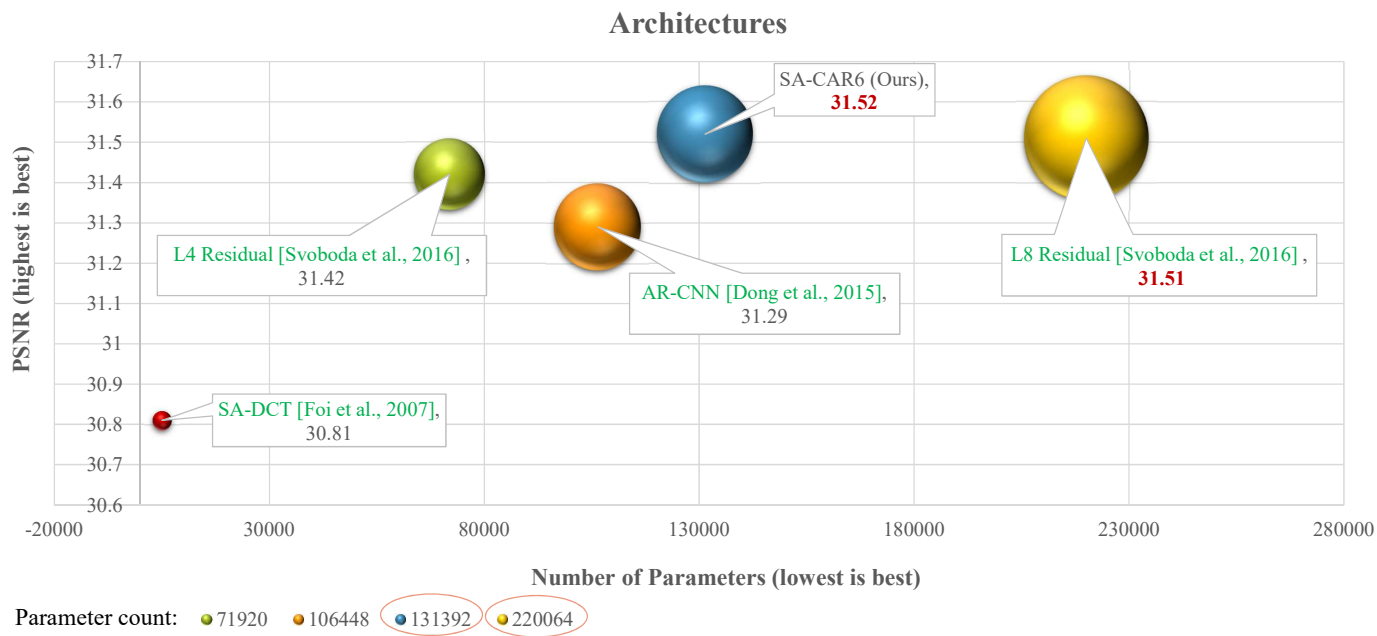
**Table 5.2** Average image reconstruction results reported by PSNR (dB), PSNR-B (dB) and SSIM on the LIVE1 dataset for JPEG quality  $q = 20$ .

Architecture	Number of Layers	Filter size	Feature maps	PSNR	PSNR-B	SSIM	Parameter count
Input JPEG	-	-	-	30.07	27.57	0.868	-
SA-DCT [52]	-	-	-	30.81	29.82	0.878	-
AR-CNN [41]	4	(9-7-1-5)	(64-32-16-1)	31.29	30.76	0.887	106,448
L4 Residual [149]	4	(11-3-3-5)	(48-64-64-1)	31.42	30.83	0.890	71,920
L8 Residual [149]	8 (4+1),(6+1)	(11-3-3-3- 1-5-1-5)	(32-64-64-64- 64-64-128-1)	31.51	30.92	0.891	220,064
SA-CAR6 (Ours)	6 (2+1)	(9-5-5-5- 5-5)	(32-32-32-32- 32-1)	31.52	30.97	0.892	131,392

The results in blue colour indicate the best results, and the results in green colour are the second-best results. Higher quality, therefore, it gives higher PSNR.



**Fig. 5.2** Average image reconstruction results reported by PSNR (dB) on the LIVE1 dataset for JPEG quality  $q = 10$ . The Bubble size is according to the number of parameters. Our proposed DA-CAR4 network has fewer parameters (where it uses only 30% of the parameters of AR-CNN) but with better performance, and this is achieved by selecting a suitable filter size with the number of layers. DA-CAR5 network realises even better results by using 75% of the parameters of AR-CNN network.



**Fig. 5.3** Average image reconstruction results reported by PSNR (dB) on the LIVE1 dataset for JPEG quality  $q = 20$ . The Bubble size is according to the number of parameters. Our architecture SA-CAR6 outperforms L8 model, by using around 60% of the number of parameters.

**Table 5.3** Average image reconstruction results reported by PSNR (dB), PSNR-B (dB) and SSIM on the BSDS500 validation dataset for JPEG quality  $q = 10$ .

Architecture	Number of Layers	Filter size	Feature maps	PSNR	PSNR-B	SSIM	Parameter count
Input JPEG	-	-	-	27.58	24.97	0.769	-
SRCNN [41]	3	(9-1-5)	(64-32-1)	28.64	28.18	0.793	8,032
AR-CNN [41]	4	(9-7-1-5)	(64-32-16-1)	28.74	28.31	0.796	106,448
DA-CAR3 (Ours)	3	(9-7-5)	(64-32-1)	28.76	28.30	0.797	106,336
DA-CAR4 (Ours)	4	(9-3-3-5)	(64-32-32-1)	28.75	28.30	0.797	33,632
DA-CAR5 (Ours)	5	(9-5-5-5-5)	(32-32-32-32-1)	28.81	28.35	0.799	80,192
L4 Residual [149]	4	(11-3-3-5)	(48-64-64-1)	28.75	28.29	0.800	71,920
SA-CAR6 (Ours)	6 (2+1)	(9-5-5-5- 5-5)	(32-32-32-32- 32-1)	28.82	28.37	0.801	131,392

The results in **blue** colour indicate the best results, and the results in **green** colour are the second-best results. Lower quality, therefore, it gives lower PSNR.

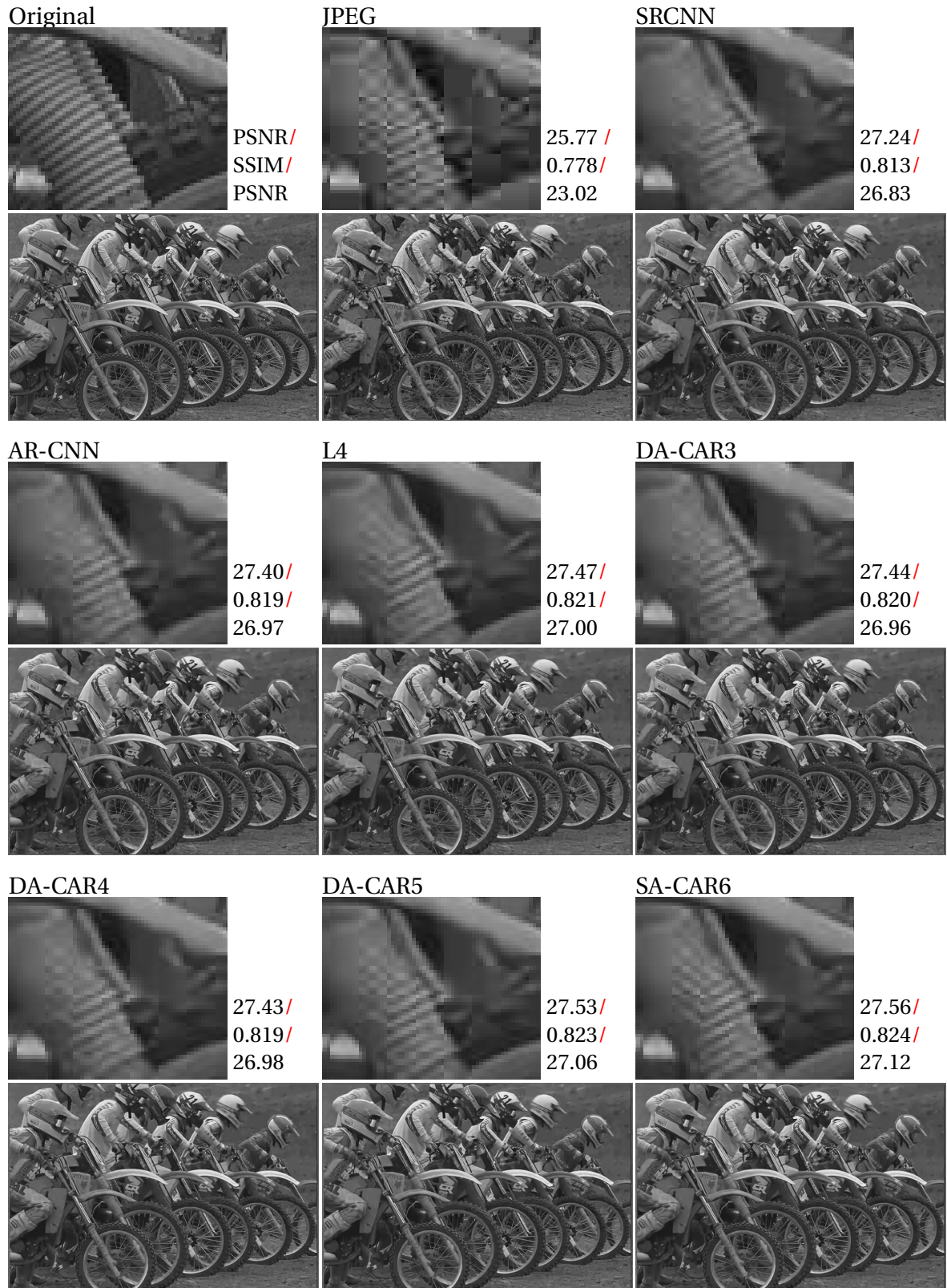
**Table 5.4** Average image reconstruction results reported by PSNR (dB), PSNR-B (dB) and SSIM on the BSDS500 validation dataset for JPEG quality  $q = 20$ .

Architecture	Number of Layers	Filter size	Feature maps	PSNR	PSNR-B	SSIM	Parameter count
Input JPEG	-	-	-	29.72	26.97	0.852	-
AR-CNN [41]	4	(9-7-1-5)	(64-32-16-1)	30.80	30.08	0.868	106,448
L4 Residual [149]	4	(11-3-3-5)	(48-64-64-1)	30.90	30.13	0.871	71,920
L8 Residual [149]	8 (4+1),(6+1)	(11-3-3-3- 1-5-1-5)	(32-64-64-64- 64-64-128-1)	30.99	30.19	0.872	220,064
SA-CAR6 (Ours)	6 (2+1)	(9-5-5-5- 5-5)	(32-32-32-32- 32-1)	31.00	30.25	0.873	131,392

The results in **blue** colour indicate the best results, and the results in **green** colour are the second-best results. Higher quality, therefore, it gives higher PSNR.



**Fig. 5.4** Qualitative evaluation of reconstruction quality for parrots image using different networks for JPEG quality  $q = 10$ . Each result is accompanied by zoom and PSNR dB/ SSIM/ PSNR-B dB.



**Fig. 5.5** Qualitative evaluation of reconstruction quality using different networks for JPEG quality  $q = 10$ . Each result is accompanied by zoom and PSNR dB/ SSIM/ PSNR-B dB.

Original



PSNR / SSIM / PSNR-B



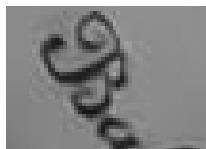
JPEG



33.97 / 0.892 / 31.44



L4



35.51 / 0.916 / 35.26



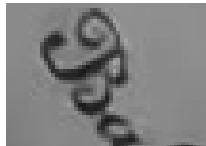
L8



35.56 / 0.917 / 35.33



SA-CAR6



35.66 / 0.919 / 35.44



**Fig. 5.6** Qualitative evaluation of reconstruction quality using different networks for JPEG quality  $q = 20$ . Each result is accompanied by zoom and PSNR dB/ SSIM/ PSNR-B dB.

## 5.5 Conclusion

In this work, we have proposed different CNNs with reduced size compared to the state-of-the-art networks with keeping the performance. Network performance significantly depends on the size of filters across the architecture. If the suitable filter size is used, the same or higher performance is achieved in comparison with the state-of-the-art structures, as we have demonstrated with our model DA-CAR3(9-7-5) that provides better performance more than the performance of SRCN(9-1-5) and AR-CNN(9-7-1-5). High numbers of parameters used in the network lead to an increase in the computation burden and the time required for both training and inference. In practice, the choice between models performing the same task may often be made based on the limitations of hardware and energy consumption requirements. We have proposed and evaluated the performance of smaller CNN architectures with a carefully selected structural layout. Our evaluation and benchmark comparisons demonstrate that many of the state-of-the-art image enhancement and JPEG artefact suppression models can be shrunk without compromising their qualitative performance. Experimentally, we have shown that DA-CAR4 gives almost the same results of AR-CNN with only 30% of parameters. DA-CAR5 outperformance the AR-CNN by using 75% of parameters. Also, SA-CAR6 achieves better performance than L8 by utilising fewer parameters with around 60%. We conclude that designing smaller nets with chosen suitable size of filters can provide better performance than large models with avoiding the problems of these nets as training time and overfitting gradients. In the next chapter, we will address the denoising of a real dataset using our proposed networks.



# Chapter 6

## Denoising in a Real Scenario

Nowadays, the process of denoising, which removes the noise and recovers meaningful information from corrupted images to obtain high-quality images, is a fundamental problem. Images during acquisition are inevitably degraded by noise, causing loss of image detail. Therefore, different methods for noise reduction (or denoising) have been proposed to deal with this issue. In particular, in recent years, deep learning-based techniques have shown excellent performance, where DNN methods depend only on data without any assumption around the distribution of the noise. This factor characterises DNN methods. Dong et al. [42] and Mao et al. [109] showed that Deep Neural Networks (DNN) surpass traditional (Non-Neural Network-based) methods for image restoration such as image denoising and image super-resolution. However, it has been recently shown that when considering real noisy image datasets, traditional techniques such as BM3D [35] outperform state-of-the-art methods for denoising [12]. We proposed a DNN pipeline (DBSR) [9], and in this chapter, we compare its performance on the real dataset (RENOIR) [12] and show how it competes against the leading techniques. The De-Blurring Super-Resolution (DBSR) model described in Section 3.3.1 is designed to recover deblurred high-resolution images from blurred low-resolution images. In this chapter, the DBSR model is trained to denoise images in the real RENOIR dataset to assess its performance on real noise<sup>1</sup>. This chapter is structured as follows: In Section 6.1 we give a brief discussion for related work. Section 6.2 discusses the real RENOIR Dataset introduced by Anaya et al. [12]. In Section

---

<sup>1</sup>Part of the results was published at the Irish Machine Vision and Image Processing (IMVIP) conference 2019, Dublin [8].

6.3, we present the training and testing of the data. Then, we discuss the evaluation of denoising methods in Section 6.4. Finally, the conclusion is introduced in Section 6.5.

## 6.1 Related Work

The applications of noise reduction focus on removing the granular discrepancies found in images to improve the quality of these images. The noise issue is essential to the performance of digital cameras and it is used as a metric for their sensor. The small sensor size and insufficient exposure time cause low-light issues for imaging devices such as mobile phones and cameras, and it leads to the capture of severely corrupted images. There are many different methods which have been applied to deal with the low-light image noise problem, such as: scale Mixtures of Gaussians (GSM) [121], the Non-local Means (NL-means) [24], Fields of Experts (FoE) [129], K-SVD [48], Block Matching and 3D Filtering (BM3D) [35], the Active Random Field (ARF) [18], Learned Simultaneous Sparse Coding (LSSC) [106], learned generative models with sampling based on the Bayesian Minimum Mean Squared Error estimate (MMSE) estimation [134], Multi-Layer Perceptron (MLP) [25] and Bilevel optimisation (opt-MRF) [30].

Most of these algorithms, however, have been evaluated solely on noisy images where the noise is artificially added to ground truth (clean) images to construct noisy images. Gaussian noise, Poisson noise and salt-and-pepper noise are the standard choices for artificial noise. Evaluation of denoising algorithms using these methods might, however, not be as accurate as evaluating their performance on real noisy digital camera images in low-light conditions. This is because the noise caused by low-light conditions is not independent identical distribution (i.i.d.) and is more complicated with its variance based on the intensity of the image [53]. Besides, different digital cameras produce various types of noise due to many factors in the imaging systems, such as sensor type and sensor size. Also, noise exists in every imaging process, particularly the low-light conditions which severely corrupt the image. Although obtaining real noisy images is easy, it is difficult to know what their noise-free counterparts should be. Therefore, the evaluation of noise reduction algorithms mostly depends on adding synthetic noise such as i.i.d. Gaussian noise to the clean images. The RENOIR image dataset [12] is a dataset with different noisy-clean image pairs from various digital cameras with diverse types of realistic noise.

## 6.2 RENOIR Dataset

Anaya and Barbu [12] have proposed a method to capture noisy-clean image pairs; naturally degraded low-light images and their low-noise counterparts, called the RENOIR dataset. This dataset has been used to test some of the popular algorithms which have been proposed to perform noise reduction. The dataset consists of 120 static scenes taken from three different cameras with different sensor sizes, and these cameras produce various kinds of noise. An equal number of scenes were taken: 40 scenes for Xiaomi Mi3 (small sensor), 40 for Canon T3i (mid-size sensor) and 40 for Canon S90 (slightly larger sensor). Also, each camera captured many images with different noise levels. Three or four images were captured per scene; one or two images involve noise and are two noise-free (low-noise) images. Anaya has followed a specific procedure (called the 'sandwich' procedure) to capture the images in this dataset under low-light conditions. The process is as follows:

- For each scene, a low-noise image was acquired with long exposure time and low-light sensitivity (ISO 100). This image is considered as the reference image.
- Then, one or two noisy images were acquired with reduced exposure time and raised light sensitivity. The value of ISO for each camera is ISO 1600 or 3200 for Mi3, ISO 640 or 1000 for S90 and ISO 3200 or 6400 for T3i.
- Lastly, another low noise image was obtained with the same parameters as the first low-noise image (the reference image). This image is considered as a clean image.

Undertaking the procedure in this way allowed the researchers to obtain two low-noise (reference and clean) images to assess the whole acquisition process quality for each scene. Also, this procedure guarantees that the only difference between the images for each scene is due to noise, not due to motion or lighting change while taking images. In addition, the low-noise images (reference ( $I^r$ ) and clean ( $I^c$ )) are noisy versions of the unknown ground truth image ( $I^{GT}$ ). These images were used to give the best estimate to the unknown ground truth image, which is:

$$I^{GT} = \frac{(I^r + I^c)}{2} \quad (6.1)$$

The PSNR measurement is calculated according to the equation 6.1, where the PSNR is estimated as the difference between the average of the two low-noise images (reference and clean images) and a noisy image, but not from applying the standard noise estimate (the difference between a noisy-clean image pair).

### 6.3 Training and Testing Data

The difference between the real dataset and generated images by adding artificial noise is that the real dataset contains different levels of noise, and this poses many challenges in training and testing. Using real datasets could be more efficient in practice. However, the denoising algorithms need a significant amount of data to train, and this amount of real data is unavailable. Therefore, the denoising algorithms are usually training on generated images by adding artificial noise with known noise levels.

The DBSR model is trained on 291 images; 91 images from Yang et al. [180] in addition to 200 images from the Berkeley Segmentation Dataset [110], with the augmentation strategy (flipping and rotation). The training dataset is divided into sub-images of size  $f_i = 50$  by employing a stride of 15. The model is trained on corrupted images with Gaussian noise at  $\sigma = 25, 30, 35$  and 50. MatConvNet is used to train the network for image denoising. We train our network for 100 epochs with a batch size 64 and learning rate 0.0001. The activation function used is Rectified Linear Unit (ReLU =  $\max(0, x)$ ) [114]. In each layer, the filter weights are initialised by the initialisation method described in He et al. [67], which is considered a robust method for the ReLU function. The cost function is minimised using Adam optimisation [84]. The number of feature maps (and filter size) of each layer is as follows 64(9), 32(5), 32(5), 32(5), 32(5), 32(5), 32(5) and 1(5). The structure of the proposed network can be written as eight layers (64-32-32-32-32-32-32-1)(9-5-5-5-5-5-5-5).

### 6.4 Evaluation of Denoising Methods

Anaya et al. [12] have selected six popular algorithms for image denoising to evaluate their performance using this real dataset: the Active Random Field (ARF)[18], Block Matching and 3D Filtering (BM3D) [35], Bilevel optimization (opt-MRF) [30], Multi Layer Perceptron (MLP) [25], Non-local means using a James-Stein estimator (NLM-JS)[171], and Non-local

means with a soft threshold (NLM-ST)[102]. These algorithms were selected because they are efficient enough to deal with the large images, and their implementations are available online. These methods depend on the level of noise  $\sigma$ . For the two versions of Non-local Means: the NLM-JS and NLM-ST methods, the noise parameter value was estimated for each image using the noise estimation method, which is discussed in [12]. For denoising, a patch size used for the NLM-JS method is  $3 \times 3$ , with a search window  $15 \times 15$ , and block size  $15 \times 15$ . However, a patch size used for the NLM-ST method is of  $5 \times 5$ , with a search window of  $13 \times 13$ , and block size of  $21 \times 21$ . The algorithms ARF[18], opt-MRF[30] and MLP [25] train their models using the noisy and clean images, to generalise on unseen noisy images. The algorithms were trained on Gaussian noise using different values of  $\sigma$ . The ARF model was trained on Gaussian noise using different values of noise at  $\sigma = 10, 15, 20, 25$  and  $50$ , with four iterations. Also, to evaluate the performance of the opt-MRF model, it was trained on Gaussian noise using  $\sigma = 15$  and  $25$  with 30 iterations. For the MLP algorithm, the model was trained on Gaussian noise at  $\sigma = 10, 25, 35, 50$ , and  $75$ . The performance of BM3D for colour image denoising was evaluated at  $\sigma = 5, 10, 15, 20, 25$ , and  $50$ . The trained filters, for the different noise level values  $\sigma$ , were used to denoise the images in this dataset. Then the parameter  $\sigma$ , which gave the best results, was selected for each method. For better performance, the NLM-JS, and NLM-ST, ARF, opt-MRF and MLP methods were applied to the YUV colour space. But BM3D was denoised in the RGB colour space where there is a particular version of the BM3D which directly handles colour images. Table 6.1 shows the PSNR results for the three cameras and different methods. The PSNR values were computed between the restored images and the best ground truth estimate (which is the average of the reference-clean images as shown in equation 6.1). The best result for BM3D is obtained with  $\sigma = 50$ ; however, the best results for the ARF, opt-MRF, and MLP algorithms occurred with  $\sigma = 25$ .

We used our DBSR model to denoise the RENOIR dataset; the best result is for  $\sigma = 35$ . Therefore, we recorded the results for this value in Table 6.1. The PSNR results of all three cameras display that the BM3D outperformed the other algorithms. Our DBSR model gives the second-best result on average after the BM3D method, and it achieves better performance than the different techniques. However, for the S90 camera, the second-best result was for the opt-MRF technique, in which case our method gives the third-best result.

Note that the SSIM metric was computed to the three cameras in [12], but the published SSIM results were higher for noisy images than denoised images. These results indicate that the noisy images are looking better than the reconstructed images. Therefore, the authors did not use the SSIM results for performance evaluation, where the SSIM values did not reflect the improvement obtained from the different methods. At the same time, the authors have not been unable to interpret the occurrence of these results. Hence the PSNR metric was used for performance evaluation. However, when we recomputed the SSIM for the corrupted images and BM3D model, we received different values compared to the published SSIM results as shown in Table 6.2, where the SSIM values for noisy images are lower than the SSIM values for reconstructed images. Our SSIM results reflect the improvement which happened after using the denoising algorithms, and this result is consistent with the PSNR results. Although the BM3D is better than our method in the performance, our DL model is about three times faster than the BM3D algorithm in the inference stage. The dimensionality of the images of the three cameras are very large, as illustrated in Table 6.2. Therefore they take a long time at the inference stage for executing the denoising models. Some qualitative examples are shown in Figures 6.1, 6.2 and 6.3.

**Table 6.1** PSNR (in dB) performance of DBSR and other denoising algorithms reported from [12] on the RENOIR real dataset.

Camera	Before Denoising	NLM-ST [102]	ARF [18]	MLP [25]	NLM-JS [171]	opt-MRF [30]	DBSR (ours)	BM3D [35]
Mi3	23.49	30.87	30.92	31.23	31.35	31.64	32.12	32.14
S90	26.19	33.08	33.80	34.07	34.14	34.98	34.83	36.31
T3i	27.44	36.82	36.55	37.58	37.40	38.65	38.76	39.91
<b>Average</b>	25.71	33.59	33.76	34.29	34.30	35.09	35.24	36.12

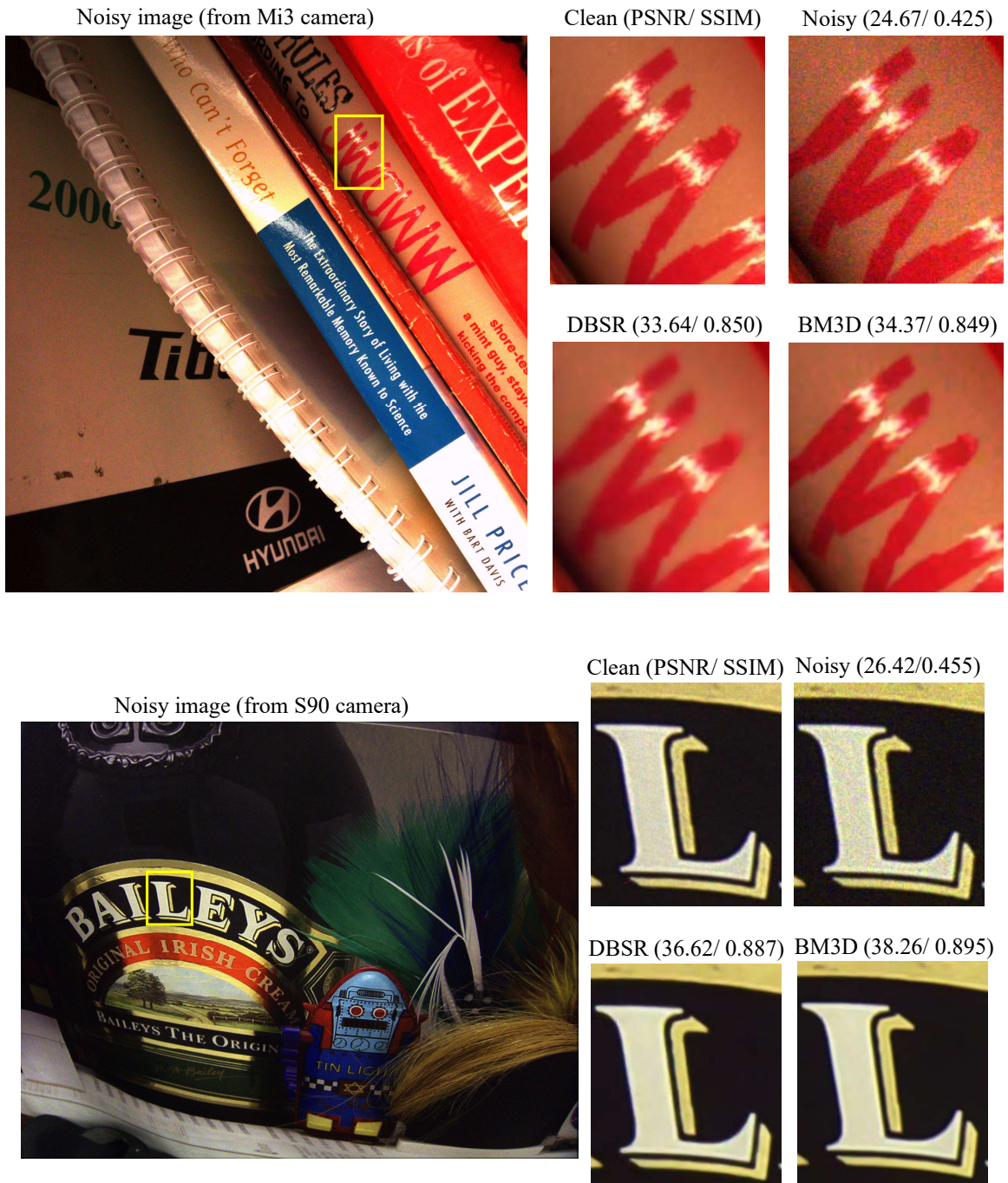
The results of the ARF, opt-MRF, MLP, NLM-JS and NLM-ST methods are reported as in [12].

The results in blue colour indicate the best results; and the results in green colour are the second-best results.

**Table 6.2** SSIM performance of DBSR and BM3D algorithms on the RENOIR real dataset, and the average resolution of the images of the different three cameras with an average of the inference CPU time.

Camera	Average Resolution	CPU Time / sec		SSIM				
		BM3D	DBSR (Ours)	Published in [12]		Our Result		DBSR (Ours)
				noisy	BM3D	noisy	BM3D	
Mi3	(2632 × 2645)	182	65	0.989	0.982	0.396	0.798	0.812
S90	(3684 × 2760)	250	92	0.988	0.979	0.450	0.882	0.861
T3i	(5187 × 3453)	450	162	0.991	0.994	0.506	0.933	0.915
<b>Average</b>	(3834 × 2953)	294	106.3	0.989	0.985	0.451	0.871	0.862

The CPU inference time is reported per image in seconds, and the average CPU time is displayed in the table for the 40 images taken from each camera, run on Intel(R) Core i7 CPU @ 2.80 GHz, 16GB RAM, and NVIDIA GTX Geforce 1050 GPU.



**Fig. 6.1** Results of DBSR and BM3D methods on the RENOIR dataset with zoomed crops .





Fig. 6.2 Results of DBSR and BM3D methods on the RENOIR dataset with zoomed crops.



**Fig. 6.3** Results of DBSR and BM3D methods on the RENOIR dataset with zoomed crops.

## 6.5 Conclusion

Dong et al. [42] and Mao et al. [109] have concluded that DNN has superior performance when compared to the traditional methods using synthesis images. However, when the algorithms have been tested on the real dataset, it has been shown that some conventional techniques such as BM3D could surpass the DNN methods [12]. We have compared our DBSR model for denoising the noisy images in the RENOIR dataset. Based on the PSNR metric, DBSR gives the second-best results on average after BM3D is compared to the other algorithms for denoising, confirming the results of Anaya and Barbu [12] that BM3D has the best performance on this dataset, even in comparison with our network. Although the traditional BM3D method exceeds our model in the performance, our model outperforms the BM3D in execution time, where the computational cost of our model is almost three times lower than the BM3D method at the inference stage.

# Chapter 7

## Conclusion and Future Work

Image restoration is a crucial task in computer vision and for many other fields. For this reason, many different techniques have been proposed in the literature. However, the most recent presented advances have been based on deep learning (DL) algorithms. In this last chapter, the main results and conclusion on the work carried out throughout this thesis is presented in Section 7.1: the three contributions on single image super-resolution (SISR), artefacts reduction and denoising tasks are summarised. Finally, the limitations of the proposed methods and the possible extension of this work in the future are discussed in Section 7.2.

### 7.1 Summary

This work has focused on image restoration applications intended to restore the original images from degraded images and improve perceptual quality by using deep learning. The example-based techniques have attained state-of-the-art performance, where it allows an improvement of the image reconstruction quality and mitigates the multiple solutions (the ill-posedness) problem of image restoration methods. Recently, example-based DL methods using convolutional neural networks (CNNs) have achieved state-of-the-art results for reconstructing images corrupted by JPEG compression, blurring, noise or down-sampling. We have proposed in this thesis several CNN architectures with fewer parameters

compared to the state-of-the-art deep structures to recover the ideal images from the corrupted images. The main contributions of this thesis are reviewed as follows:

- The first core contribution of this work is for a single image super-resolution (SISR) task. Recently, this task has witnessed a significant improvement, notably by using the CNNs to restore high-resolution (HR) images from low-resolution (LR) images. However, most of these algorithms have focused only on reconstructing the HR images from LR images without taking into account possible additional degradations such as the blurring distortion. Tackling the other degradations beside the down-sampling issue is considered as one of the fundamental keys of the SR task to obtain pleasing results. In this thesis, we have proposed light CNN architectures to address the blurring problem, which is considered a challenging aspect of the SR task since it removes high-frequencies, and the down-sampling problem to estimate deblurred HR images from the blurred LR images. The contributions of this work are summarised as follows:
  - **De-Blurring Super-Resolution CNN (DBSRCNN) model:** We have proposed a new CNN architecture for the de-blurring SISR task; called DBSRCNN. We have motivated this model by the benchmark super-resolution convolutional neural network (SRCNN) model [42], which contains three layers. We have shown that the performance of DBSRCNN architecture has outperformed SRCNN in improving the quality of the images. A potential explanation of this performance is that the concatenation of features extracted at an early stage acts similarly to traditional image processing techniques such as unsharp masking that boosts relevant (high) frequencies partially lost in the blurring stage, to enhance the reconstructed image.
  - **De-Blurring Super-Resolution (DBSR) model:** We have proposed another new architecture (called DBSR) to tackle this challenging problem; the blurring and down-sampling of images. Experimentally, we have shown that the single layer used in DBSRCNN has a limited capacity to enhance the noisy extracted features in complex applications like blurred SISR. Therefore, we have extended the DBSRCNN architecture by adding extra feature enhancement layers. We have reported a panel of comparisons with the state-of-the-art deep learning

and model-based methods to demonstrate the competitive performance of the proposed model for SR on blurred LR images. Importantly, these results are obtained by a model with three to six times fewer parameters.

- **Using Harmonic blocks in the DBSR model (Harm-DBSR):** We have proposed the use of Harmonic blocks in the DBSR network to evaluate the influence of using spectral information instead of utilising standard convolution layers. Besides, adopting the DCT filters enables the compression of the harmonic nets and uses fewer parameters. We have implemented different structures of Harm-DBSR through replacing the convolutional layers in CNN with harmonic blocks to construct a fully or partially Harmonic network. All these structures have produced better results than using convolutional operations alone.
- **Different scenarios:** We have applied the proposed CNN models for SISR by adopting two scenarios: non-blind and blind scenarios. The performance improvement was less pronounced in the blind scenarios compared to the non-blind scenarios. However, the mismatch of the degradation method in training and testing in the non-blind scenario affected the final result. Therefore, if there is no information about blurring, using the blind model is a better choice than the non-blind model.
- We have proposed an approach that alleviates undesirable image degradation resulting from image compression such as blurring, ringing and blocking artefacts based on the use of CNN. We have proposed small CNN architectures with a carefully selected structural layout, which we refer to as Direct Architecture Compression Artefacts Removal (DA-CAR) structures with a different number of layers and Skip Architecture Compression Artefacts Removal (SA-CAR6) which utilise the skip connection. Our evaluation and comparisons demonstrate that many of the state-of-the-art image enhancement and JPEG artefact suppression models can be shrunk without compromising their qualitative performance. Additionally, we have shown that if a suitable filter size is used, the same or higher performance can be achieved in comparison with deeper structures.

- **Applying the DBSR model on a real noisy dataset:** We have employed the DBSR model for denoising the noisy images in a real dataset named RENOIR [12] to evaluate its performance on real noise. The DBSR model provides the second-best results on average after the classical BM3D method compared to the other algorithms. However, our DBSR model outperforms the BM3D in terms of computational cost, where the execution time of our model is almost three times faster than the BM3D method at the inference stage.

## 7.2 Limitations and Future Work

- We have proposed CNN models for the SISR task. However, some inherent limitations have been identified. Therefore, these limitations are suggested for future lines of research:
  - The inputs of our models used in the SISR task are bicubic LR. Therefore, our architectures suffer from the same drawbacks as other proposed structures that utilise interpolated inputs. Additionally, the down-sampling factor used in training and testing should be matched, as performance can be affected by the use of a different factor should the factor be unknown or mismatched [47]. Therefore, applying the architectures directly on the LR images without interpolation helps to avoid such problems.
  - We have adopted the blurring distortion that represents the most challenging problem in the SISR. However, noise also is regarded as another degradation key in the SR task. Therefore it should be taken into consideration. Besides this, we have considered the most common blur kernel in image restoration, which is the Gaussian blur kernels. Therefore, in future work, other types of blur kernel should be adopted.
- We have applied the proposed model DBSR for the SISR task on RENOIR real noisy dataset to assess its performance, which has achieved the second-best performance compared to other techniques. Modifying the DBSR architecture by, for instance,

increasing the depth or the width of the structure could enhance the features and improve the results.

- We have adopted the Harmonic blocks which use spectral information instead of standard CNN layers in the DBSR architecture, and this has led to enhancing the results for the SISR task. Using Harmonic blocks in CNN structures for diverse image restoration tasks such as denoising would therefore be better placed to explore the effect of spectral information in restoring the original image.
- Integrating the image restoration methods using DL with the recognition and classification systems could be an interesting direction for future work, where the corrupted images which lose specific features is one of the principal challenging problems that deteriorate the performance of these systems. Furthermore, the recognition engine can be used as a metric which reflects the quality of the restoration models.





# Appendix A

## A Brief Review of Deep Learning

### A.1 Statistical Learning (SL)

Statistical Learning (SL) involves tools and models that extract information from complex data. Statistical learning and traditional statistical modelling are different because SL depends on algorithms which do not make any assumptions about the technique of generating data [22]. Methods of SL attempt to learn functions that predict the response of the generated data from a black-box of known observations. Before the 1980s, most of the SL techniques were linear models, because of the computational inability in non-linear models. However, in the 1980s, advancement in computing technology led to researchers exploring non-linear learning models. Therefore, SL methods became an interesting new domain in statistics. Moreover, the late 1980s saw the beginning of Artificial Neural Networks (ANNs) which use heavy SL mechanisms [133]. By 1990, Machine Learning (ML), was introduced as a shared domain between two fields of computer science and statistics. In the early 2000s, ML and SL became outstanding fields in statistics and computer science departments worldwide. In the late 2000s and early 2010s, with significant improvements in the computer technology and increases in data size (large datasets), both ML and SL have been widely utilised in several areas such as scientific spheres and industries.

## A.2 Machine Learning (ML)

The aim of ML is to construct learning algorithms that do the learning and generalisation automatically from data without human intervention or explicit programming. ML and deep machine learning have many software libraries used in different tasks, such as: Caffe, Keras, Torch, Theano and TensorFlow. Keras and PyTorch provide high-level abstractions built upon frameworks such as TensorFlow and Lua Torch, also they are easy to use in building deep learning models. In addition, ML tasks have been applied in a wide variety of applications in different fields. The use of ML is becoming omnipresent, for instance: in engineering, finance and medicine. For example, convolutional neural networks (CNN) have been successfully used to recognise generic objects (e.g, trucks, animals, ...) in cluttered scenes with various lighting conditions and poses by LeCun and Battou [89]. Also, Karpathy et al. used the CNN for classification tasks to classify YouTube videos into a large number of categories [79]. Furthermore, in computer vision, the CNNs have been employed to retrieve high-resolution (HR) images from low-resolution (LR) images by Donget al. [42]. In the medical field, Kuruvilla and Gunavathi have utilised neural networks in the diagnosis of cancer using the results of CT scans of the lungs [86]. ML tasks have been implemented in several other fields such as: in computer vision (as image denoising and object recognition), in robotics, automatic speech recognition, etc. The algorithms of ML are complicated models that are able to perform two main kinds of supervised learning: classification and regression.

### A.2.1 Types of Learning Problems

Machine learning is separated into two major tasks depending on how the architectures and techniques are used: generation (or synthesis) and recognition (or classification).

- **Supervised Learning:** is a predictive approach which learns from labelled data, and aims to learn a function that maps from input  $x$  to output  $y$ . In other words, the goal of supervised learning is to create a function  $f(x; \mathbf{w})$ ; for each pair of vectors  $(x, y)$  for vector of parameter  $\mathbf{w}$  where  $f(x; \mathbf{w}) = y$  [162]. Input could be a complicated structured object such as an image, a graph or an email message. There are two major kinds of supervised learning problems according to the type of output. If the

output is a categorical or nominal variable, the problem is known as *classification* or *pattern recognition*, but if the output is a continuous variable, then the problem is known as *regression*. It is also called supervised or *discriminative deep architectures*, where the target label data for this type are always available.

- **Unsupervised Learning:** is a descriptive approach, which learns from unlabelled data. It aims to seek interesting patterns within the dataset. This approach takes only inputs and the target class labels are not available. It is sometimes known as *knowledge discovery* or *generative learning*, which refers to unsupervised feature or representation learning. While supervised learning aims to estimate  $y$  from  $x$  by estimating  $p(y|x)$ , unsupervised learning attempts to learn the probability distribution  $p(x)$ , or some useful properties of that distribution. Examples of unsupervised learning problems include clustering.

## A.2.2 Regression in ML

**Linear Regression:** is a simple ML algorithm. Assuming a linear relationship between an input vector  $\mathbf{x} \in \mathbb{R}^d$  and an output variable  $y \in \mathbb{R}$ , as the following equation:

$$y = \mathbf{w}^T \mathbf{x} + b + \epsilon = \sum_{i=1}^d w_i x_i + b + \epsilon \quad (\text{A.1})$$

where  $\epsilon$  is an *error* term (an unknown term);  $\epsilon \approx N(0, \sigma^2)$ . An intercept term  $b$  is known as a bias parameter in ML.  $\mathbf{w} \in \mathbb{R}^d$  is a vector of weights that define how each feature affects the prediction. The parameters of the model are unknown. Therefore, they have to be estimated from observed data. Suppose that an input matrix of  $n$  examples and an output vector (which provide the true values of  $y$ ) will be used to estimate the parameters using the following equation:

$$\hat{y} = \mathbf{w}^T \mathbf{x} + b = \sum_{i=1}^d w_i x_i + b = \sum_{i=0}^d w_i x_i \quad ; w_0 = b$$

The model can continue with only weights by augmenting  $\mathbf{x}$  with an extra entry that is always set to 1;  $x_0 = 1$ . This extra weight will play the role of the bias parameter. Good

estimators produce the predicted value  $\hat{y}_i$  as near as possible to the true value of  $y_i$ . Therefore the measure of the nearness between  $y_i$  and  $\hat{y}_i$  should be defined. The popular choice is the square of residuals between the true value  $y_i$  and the estimated value  $\hat{y}_i$ , (i.e.  $e_i^2 = (y_i - \hat{y}_i)^2$ ) which is known as a square of the error (distances). Summing the errors for all observations will give the quantity of the total deviation of all data ( $n$  examples) and estimated model (the line). This expression is named the *Sum Square of Error* (SSE) :

$$SSE = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (\text{A.2})$$

To obtain the optimal values of parameters that minimise the SSE in equation A.2, a minimisation problem should be solved analytically as follows :

$$\arg \min_{\mathbf{w}} SSE = \arg \min_{\mathbf{w}} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (\text{A.3})$$

Also, the square distance is considered as a cost function  $c(\mathbf{w})$ , where SSE in equation A.2 expresses the overall cost of the deviation from  $\hat{y}$  for all  $n$  examples.

$$C(\mathbf{w}) = \sum_{i=1}^n (y_i - \sum_{j=1}^d w_j x_{ij})^2 \quad (\text{A.4})$$

To find the optimal values of parameters that solve the minimisation problem analytically, is very impractical and difficult especially if the amount of features is large  $\{x_1, x_2, \dots, x_d\}$ . There are other methods to find the solution to  $\arg \min_{\mathbf{w}} C(\mathbf{w})$  by applying an algorithm called *Gradient Descent*. Although gradient descent produces optimal solutions, it is computationally intensive.

### A.2.3 The Gradient Descent (GD) Algorithm

The gradient descent (GD) algorithm overcomes the difficulty of solving  $d$  equations analytically to find the minimum of function  $f$ . As the number of variables  $d$  increases, analytic solving becomes more difficult. Finding the minimum numerically using an iterative algorithm to minimise  $f$  is a more efficient way of using the gradient descent algorithm. The technique of gradient descent is to use the derivatives of the function to

follow the downhill of this function to a minimum, where  $x$  moves small steps  $\alpha$  with opposite sign of the derivative to reduce  $f(x)$ ;  $f(x) - \alpha (\text{sign} f'(x)) < f(x)$  for small enough  $\alpha$ . The concept of GD is to start from an initial point and the algorithm at each iteration moves a short distance in the direction of decrease of the function  $f$ . Given a vector of  $d$  feature variables  $\mathbf{x} = (x_1, x_2, \dots, x_d)$ ,  $f(\mathbf{x}) = f(x_1, x_2, \dots, x_d)$ , the gradient of function  $f$  is  $\nabla f(\mathbf{x}) = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_d} \right)$ , the iteration number  $\tau$ , starting from 0 in the initial value  $\tau = \{0, 1, 2, \dots\}$ . Hence,  $x^\tau$  is the value of  $x$  at iteration  $\tau$ , and  $\alpha$  is a learning rate (step size) parameter. The GD algorithm follows the following steps:

1. Initialising  $\mathbf{x}^0$ .
2.  $\mathbf{x}^{(\tau)} = \mathbf{x}^{(\tau-1)} - \alpha \nabla f(\mathbf{x}^{(\tau-1)})$   
in every iteration, the algorithm moves a short distance  $\alpha$  in the opposite direction to the gradient which is the direction of the decline, to update  $\mathbf{x}$ .
3. Calculating  $f(\mathbf{x}^{(\tau)})$ .
4. Stopping the algorithm when a particular criteria is met.

### Convergence and Stopping Criteria

There are two necessary conditions of the GD algorithm for the convergence to the minimum: the smoothness and convexity (i.e. continuously differentiable) of the cost (loss) function, the function is monotonically decreasing :

$$f(\mathbf{x}^{(\tau)}) \leq f(\mathbf{x}^{(\tau-1)}) \tag{A.5}$$

According to the problematical nature and the accuracy of the required outputs, the stopping criteria of the algorithm can vary. One of the options of the stopping criteria is that the algorithm stops when  $\| f(\mathbf{x}^{(\tau)}) - f(\mathbf{x}^{(\tau-1)}) \|_2 < \epsilon$ , where  $\epsilon$  is typically a very small number, and  $\| \cdot \|_2$  is L2-norm. This means that the algorithm will stop when the change between consequent iterations is less than  $\epsilon$ . Another option for the stopping criteria is  $\| \nabla f(\mathbf{x}^{(\tau)}) \|_2 < \epsilon$ , which indicates when the rate of decline becomes very small, and leads to a small change in the value of  $f(\mathbf{x}^{(\tau)})$  in subsequent iterations.

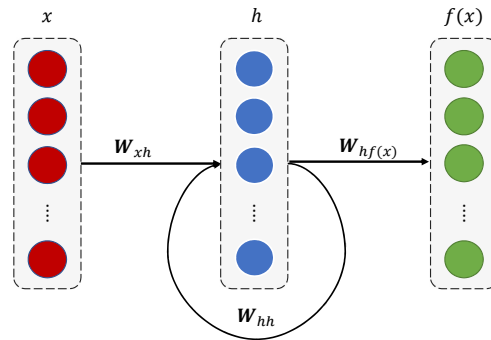
### A.3 Artificial Neural Networks (ANN)

In the early days of artificial intelligence (AI), researchers were interested in solving problems that are difficult for human beings but which can be described by mathematical rules. More recently, a challenge AI has been solving intuitive tasks for people which they can do, but which are difficult to describe such as the classification of items, recognising spoken words or faces in images. In ML algorithms, artificial neural networks (ANNs) are a set of models which are inspired by biological neural networks which attempt to mimic the behaviour of the brain neurons. These are used to learn and estimate functions that depend on high dimensional data. The computation of models in a neural network is very complex and assumes a non-linear relationship between the inputs and the outputs. Therefore, ANNs are able to learn from different data formats such as structured data, sounds, images, etc.

ANN consists of a group of layers: an input layer, an output layer and a number of hidden layers in the middle between the input layer and the output layer. Each layer is made up of a number of processing units (neurons or nodes) which applies a function known as an activation function, and neural network supports a wide range of activation functions. If there are no feedback connections (cycles or loops) among the layers of the network, then this network is known as *Feed-Forward Network*, where the information moves in only a forward direction, from the input neurons to the output neurons. While the Single-Layer Feed-forward Network contains one input layer, one output layer of processing units (known as a *Simple-Layer Perceptron*), Multi-Layer Feed-forward Network has one input layer, one output layer and one or more hidden layers of neurons between them (known as a *Multi-Layer Perceptron* (MLP)). However, a *Recurrent Neural Network* (RNN) contains at least one feed-back loop where some layers in the network take inputs from consequent layers, so the activation functions can flow in a circle, that supports the networks to perform learn sequences and temporal processing, and it may or may not contain any hidden layers. A Simple RNN is shown in Figure A.1. If all the processing units in every layer are connected to all the processing units in the subsequent layer, this type of network is *fully-connected*.

Artificial Neural Networks are extremely powerful parallel computational systems consisting of many simple processing elements connected together to perform a particular

task. This parallelism makes them efficient and robust. One of the most powerful characteristics of neural networks is their capability for learning and generalising from a set of training data. The weights (or strengths) of the connections between processing units are adapted to optimise the final response.



**Fig. A.1** Recurrent Neural Network (RNN).

### A.3.1 A Brief Historical Review of ANN

There are three historical stages relating to ANNs research and the development of deep learning. In the 1940s–1960s, the first stage started with the development of biological learning theories, McCulloch and Pitts in 1943 [111] and Hebb in 1949 [69]. The first models which had been implemented such as the perceptron allowed for the training of a single neuron by Rosenblatt in 1958 [128]. During the period from 1980 to 1995 was the second stage, to train the ANN with one or two hidden layers with back-propagation; Rumelhart et al. in 1986 [132]. The current stage is the stage of Deep Learning (DL), started in 2006, when Hinton et al. came up with the idea of deep layers networks [70]. The first stage was called cybernetics. The goal was learning a set of weights  $\{w_1, w_2, \dots, w_d\}$  and computing the output  $y$  from simple linear models  $f(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^d w_i x_i$ ; for a set of  $d$  input values  $\mathbf{x} = \{x_1, x_2, \dots, x_d\}$ . In 1943, the McCulloch-Pitts Neuron [111] was an early model of brain function, being the first work known as AI, in which a single neuron takes a weighted sum of inputs  $f(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^d w_i x_i$  (simple linear model) from other neurons, and this model can recognise two different categories of these inputs by testing whether  $f(\mathbf{x}, \mathbf{w})$  is positive or negative. In other words, the neuron is either activated by these inputs or it remains

dormant. Donald Hebb, in 1949, suggested a mechanism of learning for neural networks, which is known as the Hebbian principle "neurons that fire together, wire together" [69], and it is related to a simple updating rule for adapting the connection weights between neurons for learning.

In 1951, the first ANN was built by Marvin Minsky and Dean Edmonds. They were two graduate students in the Princeton mathematics department. Their network was named *The Stochastic Neural Analog Reinforcement Calculator* (SNARC) which built on Hebb's theory [80]. In 1958, the perceptron (a single layer perceptron) was the first attempt to create an ANN for pattern classification in that period which was made by Rosenbalt at Cornell University. A great many experiments with perceptrons had been implemented by the mid 1960s, and the perceptrons were able to learn to recognise some kinds of patterns but not others. Also, in 1969, Minsky and Papert proved that single layer perceptron was limited with pattern classification. For example, it could not implement the *XOR* logical function and several others [60]. Therefore, there has been a need to perform this task using more complex networks. The complex network which was found to be able to fix the limits of pattern classification as XOR did not exist until 1986, when it became possible to perform multi-layer networks after the popularisation of the back-propagation algorithm which is used until today, by Rumelhart, Hinton and Williams [133]. The method of the back-propagation algorithm was discovered by Werbos in 1974 [170]. In 2006, Geoffrey Hinton showed that a type of neural network called *Deep Learning* (DL) (also called *deep structured learning*, *hierarchical learning* or *deep machine learning*), has emerged as a particular kind and a sub-field of ML research implemented with deep layered networks and achieves great flexibility and power by learning [70]. Also, DL has been characterised as a rebranding of neural networks with multiple hidden layers, which has been motivated to mimic the way the brain works.

### A.3.2 Forward-Propagation

Before describing the forward propagation process of the neural network, some notation should be introduced for the neural network. The neural network consists of  $L+1$  layers (i.e.  $L-1$  hidden layers). Each layer contains  $N_l$  processing units where  $l \in \{0, 1, 2, \dots, L\}$ . The network takes  $d$  features  $(x_1, x_2, \dots, x_d)$  as inputs in the input layer; hence,  $N_0 = d$  inputs



or units. Generally, layer  $l$  involves  $N_l$  processing units, each unit takes  $N_{l-1} + 1$  inputs which is associated to the processing units in the previous layer  $l - 1$  in addition to a bias unit. The first hidden layer includes  $N_1$  processing units, where takes  $d$  inputs from the input layer plus extra bias term. The output layer produces  $N_L$  outputs. The equations that express a neural network are as follows :

- $W_{i,j}^{(l)}$  : the connection weights from  $i^{th}$  processing unit in the  $l - 1$  layer to the  $j^{th}$  processing unit in  $l$  layer;  $i \in \{1, 2, \dots, N_{l-1}\}$ ,  $j \in \{1, 2, \dots, N_l\}$  and  $l \in \{1, 2, \dots, L\}$ .
- $b_j^{(l)}$  : bias unit which is correlated to  $j^{th}$  processing unit in the  $l$  layer, where  $j \in \{1, 2, \dots, N_l\}$  and  $l \in \{1, 2, \dots, L\}$ .
- $a_j^{(l)}(\mathbf{x})$  : the hidden layer pre-activation (or input activation) is the weighted sum of the input of neurons in layer  $l - 1$  plus the bias unit, which is correlated to the  $j^{th}$  processing unit in the layer  $l$ .

$$a_j^{(l)}(\mathbf{x}) = b_j^{(l)} + \sum_{i=1}^{N_{l-1}} W_{i,j}^{(l)} h_i^{(l-1)}(\mathbf{x})$$

$$\text{For the first hidden layer } (l = 1) : a_j^{(1)}(\mathbf{x}) = b_j^{(1)} + \sum_{i=1}^d W_{i,j}^{(1)} x_i;$$

where  $\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x}$  and  $\mathbf{x} = \{x_1, x_2, \dots, x_d\}$  are  $d$  input features.

The equations have been reformulated utilising linear algebra to make the equations easier for manipulation :

$\mathbf{a}^{(l)}(\mathbf{x}) = \mathbf{b}^{(l)} + \mathbf{W}^{(l)} \mathbf{h}^{(l-1)}(\mathbf{x})$ , where  $\mathbf{W}^{(l)}$  is a  $N_l \times N_{l-1}$  matrix which represents the weights related to all connections between layer  $l - 1$  and layer  $l$ , and  $\mathbf{b}^{(l)}$  is a  $N_l \times 1$  column vector which represents all the bias units connecting to layer  $l$ .

- $h_j^{(l)}(\mathbf{x})$  : the hidden layer activation (or the output activation) is the activation function in the  $j^{th}$  neurons of the layer  $l$ ,  $h_j^{(l)}(\mathbf{x}) = g(a_j^{(l)}(\mathbf{x}))$ , in another formulation will be  $\mathbf{h}^{(l)}(\mathbf{x}) = g(\mathbf{a}^{(l)}(\mathbf{x}))$ , where  $g(\cdot)$  is the activation function.
- $f_j(\mathbf{x})$  : the output layer activation is the output of the  $j^{th}$  neuron in the layer ( $L$ );  
 $f_j(\mathbf{x}) = h_j^{(L)}(\mathbf{x}) = O(a_j^{(L)}(\mathbf{x}))$ . In another formulation using linear algebra will be  
 $\mathbf{f}(\mathbf{x}) = \mathbf{h}^{(L)}(\mathbf{x}) = O(\mathbf{a}^{(L)}(\mathbf{x}))$ , where  $O(\cdot)$  is the output activation function.

The process of forward propagation in the neural network is described by the equations above. These equations describe the relationship between the features of input  $\{x_1, x_2, \dots, x_d\}$  and the responses of the neural network  $\{h_1, h_2, \dots, h_{N_L}\}$ . Figure A.2 is an example of the forward propagation structure of the feed-forward network. Consequently, if we have  $L + 1$  layers for the neural network, the forward propagation equations using

linear algebra are written as follows :

$$\mathbf{a}^{(1)}(\mathbf{x}) = \mathbf{b}^{(1)} + \mathbf{W}^{(1)}(\mathbf{x}) \quad \left( \text{where, } a_j^{(1)}(\mathbf{x}) = b_j^{(1)} + \sum_{i=1}^d W_{i,j}^{(1)} x_i \right)$$

$$\mathbf{h}^{(1)}(\mathbf{x}) = g(\mathbf{a}^{(1)}(\mathbf{x})) \quad \left( \text{where, } h_j^{(1)}(\mathbf{x}) = g(a_j^{(1)}(\mathbf{x})) \right) \quad \text{for } j \in \{1, 2, \dots, N_1\}$$

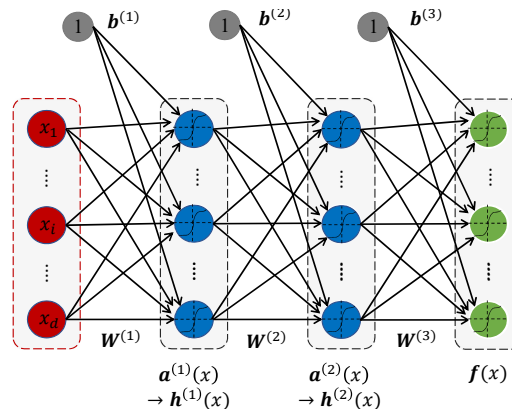
$$\mathbf{a}^{(2)}(\mathbf{x}) = \mathbf{b}^{(2)} + \mathbf{W}^{(2)}\mathbf{h}^{(1)}(\mathbf{x}) \quad \left( \text{where, } a_j^{(2)}(\mathbf{x}) = b_j^{(2)} + \sum_{i=1}^{N_1} W_{i,j}^{(2)} h_i^{(1)}(\mathbf{x}) \right)$$

$$\mathbf{h}^{(2)}(\mathbf{x}) = g(\mathbf{a}^{(2)}(\mathbf{x})) \quad \left( \text{where, } h_j^{(2)}(\mathbf{x}) = g(a_j^{(2)}(\mathbf{x})) \right) \quad \text{for } j \in \{1, 2, \dots, N_2\}$$

⋮

$$\mathbf{a}^{(L)}(\mathbf{x}) = \mathbf{b}^{(L)} + \mathbf{W}^{(L)}\mathbf{h}^{(L-1)}(\mathbf{x}) \quad \left( \text{where, } a_j^{(L)}(\mathbf{x}) = b_j^{(L)} + \sum_{i=1}^{N_{L-1}} W_{i,j}^{(L)} h_i^{(L-1)}(\mathbf{x}) \right)$$

$$\mathbf{h}^{(L)}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) = O(\mathbf{a}^{(L)}(\mathbf{x})) \quad \left( \text{where, } h_j^{(L)}(\mathbf{x}) = f_j(\mathbf{x}) = O(a_j^{(L)}(\mathbf{x})) \right) \quad \text{for } j \in \{1, 2, \dots, N_L\}$$



**Fig. A.2** Feed Forward Neural Network with three layers.

The target function for a neural network  $\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}; \mathbf{W}, \mathbf{b})$ , where  $\mathbf{W}$  and  $\mathbf{b}$  are the parameters :  $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L)}, \mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \dots, \mathbf{b}^{(L)}$ . A total cost function for the neural network using the target function :

$$C(\mathbf{W}, \mathbf{b}) = \sum_{i=1}^n c(f(\mathbf{x}_i; \mathbf{W}, \mathbf{b}), \mathbf{y}_i) \quad (\text{A.6})$$

Where  $c(\cdot)$  is the cost function (a loss function for one example), and  $y_i$  is the output vector for the  $i^{th}$  training example;  $i \in \{1, 2, \dots, n\}$ , and  $n$  is the number of training dataset.

To obtain the best values for the parameters  $\mathbf{W}$  and  $\mathbf{b}$ , the minimisation issue should be resolved as follows:

$$\arg \min_{\mathbf{W}, \mathbf{b}} C(\mathbf{W}, \mathbf{b}) \quad (\text{A.7})$$

This problem can be solved by applying a stochastic gradient descent algorithm (SGD), where SGD is an extension of the gradient descent algorithm, which will be introduced in section A.4.6. Due to the complexity of obtaining the partial derivatives for the cost function, there is an algorithm known as *the Backward Propagation of Errors* which uses the chain rule, to simplify the problem of obtaining the derivatives.

### A.3.3 Backward Propagation

The feed-forward neural network is used to accept an input  $x$  and produce an output  $f(x)$ , where data information flows forward through the network (this is called forward propagation). After obtaining the cost function, the back-propagation algorithm allows the information to flow backwards through the network from the cost function in order to compute the gradient. Back-propagation refers only to the procedure for computing the gradient by applying the chain rule, while another algorithm such as SGD is used to implement learning using this gradient. Figure A.3 shows how back-propagation works.

- **Cost gradient at hidden layers**

$$\begin{aligned} \text{Partial derivative : } \frac{\partial C}{\partial h_j^{(l)}(\mathbf{x})} &= \sum_i \frac{\partial C}{\partial a_i^{(l+1)}(\mathbf{x})} \frac{\partial a_i^{(l+1)}(\mathbf{x})}{\partial h_j^{(l)}(\mathbf{x})} \\ &= \sum_i \frac{\partial C}{\partial a_i^{(l+1)}(\mathbf{x})} \frac{\partial (b_i^{(l+1)} + \sum_j W_{i,j}^{(l+1)} h_j^{(l)}(\mathbf{x}))}{\partial h_j^{(l)}(\mathbf{x})} = \sum_i \frac{\partial C}{\partial a_i^{(l+1)}(\mathbf{x})} W_{i,j}^{(l+1)} \end{aligned}$$

The gradient is :  $(\mathbf{W}^{(l+1)})^T (\nabla_{\mathbf{a}^{(l+1)}(\mathbf{x})} C)$  ; where  $C = C(\mathbf{W}, \mathbf{b})$  in equation A.6.

- **Cost gradient at hidden layers pre-activation**

$$\begin{aligned} \text{Partial derivative : } \frac{\partial C}{\partial a_j^{(l)}(\mathbf{x})} &= \sum_i \frac{\partial C}{\partial h_j^{(l)}(\mathbf{x})} \frac{\partial h_j^{(l)}(\mathbf{x})}{\partial a_j^{(l)}(\mathbf{x})} \\ &= \sum_i \frac{\partial C}{\partial h_j^{(l)}(\mathbf{x})} \frac{\partial g(a_j^{(l)}(\mathbf{x}))}{\partial a_j^{(l)}(\mathbf{x})} = \sum_i \frac{\partial C}{\partial h_j^{(l)}(\mathbf{x})} \dot{g}(a_j^{(l)}(\mathbf{x})) \end{aligned}$$

The gradient is :  $(\nabla_{\mathbf{h}^{(l)}(\mathbf{x})} C) \odot [\dots, \dot{g}(a_j^{(l)}(\mathbf{x})), \dots]$

- **Cost gradient of parameters:**

- **For weights**

$$\begin{aligned} \text{Partial derivative : } \frac{\partial C}{\partial W_{i,j}^{(l)}} &= \frac{\partial C}{\partial a_i^{(l)}(\mathbf{x})} \frac{\partial a_i^{(l)}(\mathbf{x})}{\partial W_{i,j}^{(l)}} \\ &= \frac{\partial C}{\partial a_i^{(l)}(\mathbf{x})} \frac{\partial (b_i^{(l)} + \sum_j W_{i,j}^{(l)} h_j^{(l-1)}(\mathbf{x}))}{\partial W_{i,j}^{(l)}} = \frac{\partial C}{\partial a_i^{(l)}(\mathbf{x})} h_j^{(l-1)}(\mathbf{x}) \end{aligned}$$

The gradient is :  $(\nabla_{\mathbf{a}^{(l)}(\mathbf{x})} C) (\mathbf{h}^{(l-1)}(\mathbf{x}))^T$

- **For biases**

$$\begin{aligned} \text{Partial derivative : } \frac{\partial C}{\partial b_i^{(l)}} &= \frac{\partial C}{\partial a_i^{(l)}(\mathbf{x})} \frac{\partial a_i^{(l)}(\mathbf{x})}{\partial b_i^{(l)}} \\ &= \frac{\partial C}{\partial a_i^{(l)}(\mathbf{x})} \frac{\partial (b_i^{(l)} + \sum_j W_{i,j}^{(l)} h_j^{(l-1)}(\mathbf{x}))}{\partial b_i^{(l)}} = \frac{\partial C}{\partial a_i^{(l)}(\mathbf{x})} \end{aligned}$$

The gradient is :  $\nabla_{\mathbf{a}^{(l)}(\mathbf{x})} C$

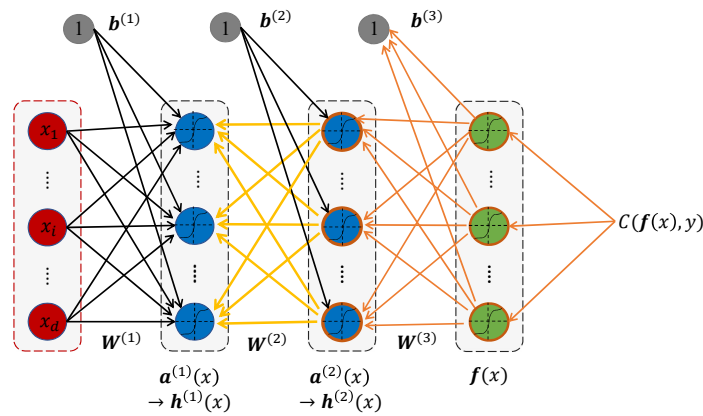


Fig. A.3 Backward Propagation.

## A.4 Practical Considerations

### A.4.1 Dataset

In ML, a very common practice is data splitting into two parts: training data and testing data. The training data is utilised to fit the model and the testing data is utilised to check the performance and provide generalisations on different sets of data. The dividing of the dataset is often performed after randomising the raw dataset. The common dividing ratios are about 70:30 to 80:20 training examples to testing examples. In ML, an important problem is that large training datasets are necessary for effective generalisation.

### A.4.2 Activation Functions

There are diverse options of activation functions that can be exploited in the neural nets. It is usually impossible to predict in advance which function will work best because the design of hidden units does not yet have any definitive guiding theoretical principles [60], therefore the design process depends on trial and error. A collection of the most popular options for activation functions in neural nets is addressed below :

**The Identity Function:** This is a linear mapping that maps its inputs to the interval  $(-\infty, \infty)$ . This function is usually utilised in the last layer to obtain the output because it

does not add any complication to the network.

$$g(x) = x \quad \& \quad \text{Its derivative: } g'(x) = 1$$

**The Sigmoid (sigm) Function:** This is a non-linear function that maps the inputs to the interval  $[0, 1]$ . The sigmoid function is used in the hidden layers and the output layer as well, because it increases the non-linear features of the neural network. Also, it is usually used for two class classification problems, where the output indicates the likelihood of belonging to one of the two classes.

$$g(x) = \text{sigm}(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \quad \& \quad \text{Its derivative: } g'(x) = g(x)(1 - g(x))$$

**The Hyperbolic Tangent (tanh) Function:** This maps each input to an interval  $[-1, 1]$ . It is a non-linear mapping that is utilised in the ANN especially in regression tasks. Vanishing gradients may occur in DNNs because of the units of the higher layer are almost saturated at  $-1$  or  $1$ , when the gradients are nearly  $0$  in lower layers of the network. Thus this problem leads to slow convergence, and also may lead to the trained network converging to a poor local minimum.

$$g(x) = \text{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad \& \quad \text{Its derivative: } g'(x) = 1 - g^2(x)$$

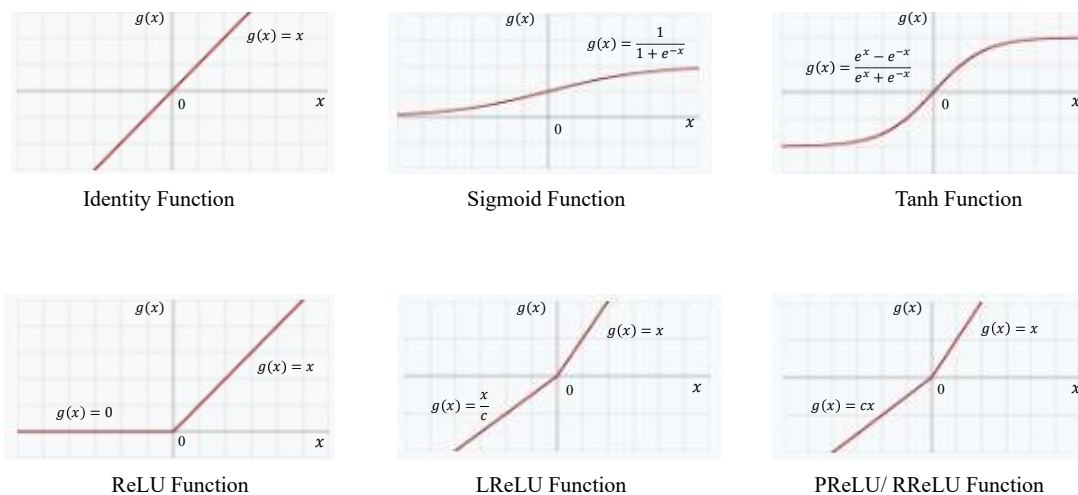
**Rectified Linear Units (ReLU) Function:** ReLU is an excellent default choice of activation function. This function has more usefulness than sigmoid and the hyperbolic tangent activation functions because ReLU function does not suffer from the gradient vanishing problem as tanh function and sigmoid function [104]. Also, ReLU function speeds up the training of the ANN because the network converges faster [188]. ReLU is easy to optimise because it is comparable to linear units, and the difference between the linear function and a ReLU is that the outputs are zero across half the domain of a ReLU. Therefore, the derivatives through a rectified linear unit remain large and are also consistent when the unit is active.

$$g(x) = \text{ReLU}(x) = \max(0, x) = \begin{cases} x & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases} \quad \& \quad g'(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}$$

**Leaky Rectified Linear Units (LReLU) Function:** The gradient of the ReLU function is 1 when it is activated higher than 0. Therefore vanishing gradients do not happen for active hidden units in DNN [20]. However, during optimisation, the gradients of ReLU are 0 for the units which are not active, and the weights of units that never activate will not be adjusted. The learning might be slow when training ReLU networks with constant 0 gradients, and this is a potential problem similar to the vanishing gradients problem. To alleviate this problem, leaky rectified linear units (LReLU) is introduced by Maas et al. [104]. The LReLU allows for a small non-zero gradient when the unit is not active. There are many modified types of ReLU: in addition to LReLU, the parametric rectified linear unit (PReLU) which is proposed by He et al.. The difference between LReLU and PReLU is that in the training, the parameter  $c$  is learned via back-propagation.

$$g(x) = \text{LReLU}(x) = \begin{cases} x & \text{for } x \geq 0 \\ \frac{x}{c} & \text{for } x < 0 \end{cases} \quad \& \quad g'(x) = \begin{cases} 1 & \text{for } x \geq 0 \\ \frac{1}{c} & \text{for } x < 0 \end{cases}$$

where  $c$  is a fixed parameter in range  $(1, \infty)$ , and it is suggested to be a large number such as 100. However, the theoretical explanation of their superior performance is still needed [174]. Figure A.4 displays some activation function charts.



**Fig. A.4** Charts of some activation functions.

### A.4.3 The Cost Function

The selection of the cost function is an important aspect of neural networks. In regression tasks, the aim of regression tasks is to estimate the outputs of continuous variables using input variables  $d$ ;  $\hat{y} = f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ . One of the ways to measure the performance of the model is to compute the mean squared error of the model (MSE). MSE (the mean of L2-norm squared) is the variation between a true value and the predicted value. This cost function ensures that the difference between predicted value and the true value is penalised. The mean squared error is given by :

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (\text{A.8})$$

This error measure equals to 0 when  $\hat{\mathbf{y}} = \mathbf{y}$ , and the error measure increases when the difference between the predictions and the outputs increases.

### A.4.4 Over-Fitting and Regularisation

The model in ML tries to learn to fit the training data well, but when it fails in generalisation on another dataset, as in this case, over-fitting occurs. When the model is too complex, over-fitting usually happens, where the number of parameters is large compared with the number of training points. It occurs when the model explains the noise in the training dataset and can not describe the relation among the inputs and their outputs. Some suggestions have been used to avoid over-fitting. Firstly, decreasing the complexity of the model by removing some of the data features that reduce the number of parameters, or by using more training examples. Also, penalising the size of the parameters using regularisation. Regularisation can be added to the total cost function used to train a neural network by adding a parameter norm penalty  $\Omega(\Theta)$  to the machine learning model, as follows:

$$C_\lambda(\Theta) = \sum_{i=1}^n c(f(\mathbf{x}_i; \Theta), \mathbf{y}_i) + \lambda \Omega(\Theta)$$

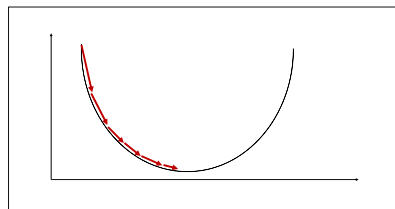
Where  $\lambda \in [0, \infty)$  is a hyper-parameter (regularisation parameter) which indicates the penalty size on the parameters  $\Theta$ , where  $\Theta \equiv \{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \dots, \mathbf{W}^{(L)}, \mathbf{b}^{(L)}\}$ . Setting  $\lambda$  to 0 results in no regularisation, and larger values of  $\lambda$  correspond to more regularisation. While the



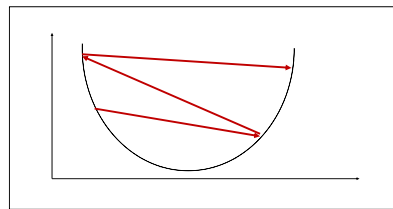
model fitting demands the minimisation of  $C_\lambda(\Theta)$ , the penalty terms make the parameters as small as possible. For neural networks, a parameter norm penalty  $\Omega$  is usually used to penalise only the weights and leaves the biases unregularised.

### A.4.5 The Learning Rate

The hyper-parameters control many aspects of the behaviour of machine learning algorithms. There are two basic approaches for choosing the hyper-parameters; manually or automatically, but choosing the hyper-parameters manually needs an understanding of how machine learning models achieve good generalisation. The learning rate  $\alpha$  perhaps is the most important hyper-parameter in machine learning, and it is usually determined manually in the gradient descent algorithm. The learning rate controls the effective capacity of the machine learning model, which is at its highest when the learning rate is correct. Therefore, if we have time to tune only one hyper-parameter, we will tune the learning rate [60]. For training error, the learning rate has a U-shaped curve. If the value of learning rate is too small, the algorithm may demand a long time to converge to the minimum, and may become permanently stuck with a high training error. However, if the value of learning rate is large, the training error may increase rather than decrease, and it may cause the algorithm to diverge. Figure A.5 shows these problems. Choosing the learning rate correctly should seek to avoid these two problems. When the algorithm approaches a value close to the minimum, a common practice is to decrease the learning rate to enhance the result, which is referred to as *fine tuning*.



**(a)** Small learning rate causes slow convergence.



**(b)** Large learning rate causes divergence.

**Fig. A.5** Two different problems of learning rate.

### A.4.6 Stochastic GD (SGD) and Mini-Batch GD

Most deep learning algorithms involve optimisation, where the optimisation indicates the process of minimising or maximising some function  $f(\mathbf{W})$  by changing  $\mathbf{W}$ . In machine learning models, optimisation problems usually minimise the cost function, and they work very well when trained with gradient descent. The number of updates needed to converge often increases with the size of a training set. Also, for good generalisation, machine learning needs large training sets, but the large training sets are more computationally expensive. To avoid this, the *Stochastic Gradient Descent algorithm* (SGD) is used for large training sets. SGD is an extension of the gradient descent algorithm, but it runs much faster than ordinary gradient descent. SGD calculates the gradient by computing the expected value of the partial derivatives from a single randomly picked training sample, rather than computing the derivatives of the total cost function in each iteration. The SGD algorithm is as follows:

1. Initialise  $\mathbf{W}^{(0)}$ .
2. For  $\tau = 1, 2, \dots$  do :

$$\begin{aligned}\mathbf{W}^{(\tau)} &= \mathbf{W}^{(\tau-1)} - \alpha \mathbb{E}(\nabla C(\mathbf{W})) = \mathbf{W}^{(\tau-1)} - \alpha \mathbb{E}\left(\nabla \sum_{i=1}^n c(\mathbf{y}_i, f(\mathbf{x}_i; \mathbf{W}))\right) \\ &= \mathbf{W}^{(\tau-1)} - \alpha \nabla c(\mathbf{y}_i, f(\mathbf{x}_i; \mathbf{W})) \quad \text{for a random } i \in \{1, 2, \dots, n\}\end{aligned}$$

3. Calculate  $C(\mathbf{W})$ .
4. Stop when a certain criteria is met.

Although SGD performs much faster than regular gradient descent, it is very slow in networks with many hidden layers and it takes a very long time to converge. Also, it can get stuck in poor local optima, usually in deep nets that are far from optimal [57]. *Mini-Batch Gradient Descent* is an optimisation method which aims to overcome the slow convergence of SGD and the computational inefficiency of normal gradient descent. The idea of mini-batch gradient descent is that the expectation could be roughly estimated by using a small set ( $p$  points) from the training dataset to estimate the gradient of the cost function, instead of using a single data point as SGD or using the whole training set as in ordinary gradient descent. A mini-batch of examples can be sampled, in each step of the

algorithm  $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p\}$  drawn uniformly from the training set. The update step becomes :

$$\mathbf{W}^{(\tau)} = \mathbf{W}^{(\tau-1)} - \alpha \nabla \left( \sum_{i=1}^p c(\mathbf{y}_i, f_i(\mathbf{x}; \mathbf{W})) \right)$$

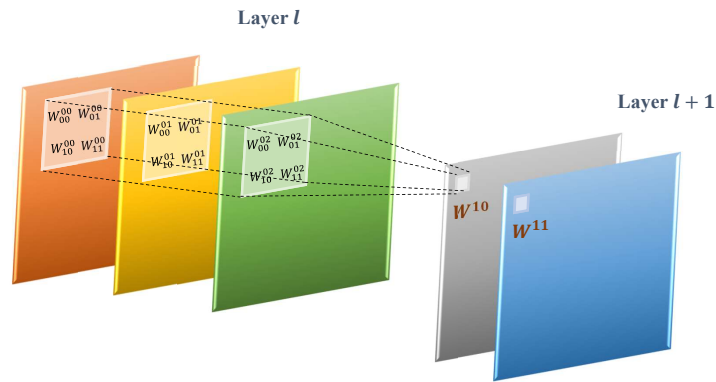
Recently, a number of mini-batch-based methods have been introduced that adapt the learning rates of model parameters, such as *Adam*. Adam is an efficient stochastic optimisation method that only needs first-order gradients with little memory requirements, and it is computationally efficient. Adam's name is derived from adaptive moment estimation [84]. A reasonable choice of optimisation algorithm is SGD, and another very reasonable alternative method for optimisation is Adam.

## A.5 Convolutional Neural Networks

Convolutional neural network (CNN) is a special type of ANN to process data which is considered as a 2D grid (image). It is a type of feed-forward ANN designed to learn directly from multi-dimensional raw data, such as images. Also, CNN references that the neural network performs a mathematical operation named convolution. A convolution on two discrete 2D functions can be regarded as multiplication by a matrix. A CNN involves of one or more convolutional layers. These layers are usually followed by fully-connected layers, but a CNN can be entirely designed by only convolutional layers. A convolutional layer is created by a 3D tensor of processing units. The size of convolutional layer is  $k \times m \times n$ , where  $k$  is the number of the feature maps. Also, each 2D plane of neurons forms one feature map; the size of each feature map is  $m \times n$ . In the convolutional layer, each neuron receives inputs from a set of neurons in the previous layer located in a small neighbourhood; this small neighbourhood is defined by a kernel.

The kernel (or filter) is a 2D matrix of a fixed size  $r \times s$  which determines patches of processing units of the same size  $r \times s$  in an input feature map. The filter convolved with the input feature maps in the convolutional layer  $l$  produces a single neuron as output for the output feature map of the convolutional layer  $l + 1$ . Consider an illustrated example of a convolutional layer in Figure A.6. The figure shows that each neuron in the output feature map in the convolutional layer  $l + 1$  shares the same group of weights associated

with the patches of the input feature maps in layer  $l$ . Also, the neurons in each output feature map in the convolutional layer  $l + 1$  can be different weights. Sharing of weights means the weights needing to be stored is fewer, hence the memory requirements of the model will decrease.



**Fig. A.6** An illustrated example of a convolutional layer.

**Forward Propagation of CNN:** The output feature maps in the convolutional layer  $l + 1$  are determined by the convolution of kernels with the input feature maps of the preceding convolutional layer  $l$ . Some notations should be defined to define the equations of forward propagation for a convolutional layer, as follows:

- $K_l$  : the number of the input feature maps in the convolutional layer  $l$ .
- $m_l \times n_l$  : the feature maps size in the convolutional layer  $l$ .
- $r_l \times s_l$  : the filters size associated with the convolutional layer  $l$ .
- $\mathbf{W}_{kk'}^{(l)}(u, v)$  : the filter weights related to the convolution with the  $k^{th}$  input feature map in the convolutional layer  $l$ , going to the  $k'^{th}$  output feature map in the convolutional layer  $l + 1$ , and for the weight in position  $(u, v)$  in the kernel.
- $b_{k'}$  : column vector represents all bias units connecting to a convolutional layer  $l + 1$ .
- $a_{k'}^{(l+1)}(i, j)$  : a pre-activation function which represents the weighted sum of the activated patches (a group of processing units in a small neighbourhood) from a convolutional layer  $l$  plus the bias.
- $h_{k'}^{(l+1)}(i, j)$  : is an activation function which is related to the  $k'^{th}$  feature map in layer  $l + 1$ , for the processing unit in position  $(i, j)$  in the output feature map.

The forward propagation equations for a convolutional layer is specified by :

$$\begin{aligned} a_{k'}^{(l+1)}(i, j) &= b_{k'} + \sum_{k=1}^{K_l} (h * \mathbf{W}_{k'})_k^{(l)} \\ &= b_{k'} + \sum_{k=1}^{K_l} \sum_{u=1}^r \sum_{v=1}^s h_k^{(l)}(i+u-1, j+v-1) \mathbf{W}_{kk'}^{(l)}(u, v) \end{aligned}$$

$$\begin{aligned} h_{k'}^{(l+1)}(i, j) &= g(a_{k'}^{(l+1)}(i, j)) \\ \text{for } k' \in \{1, 2, \dots, K_{l+1}\}, \quad i \in \{1, 2, \dots, m_{l+1}\} &; m_{l+1} = m_l + 1 - r_l \\ j \in \{1, 2, \dots, n_{l+1}\} &; n_{l+1} = n_l + 1 - s_l \end{aligned}$$

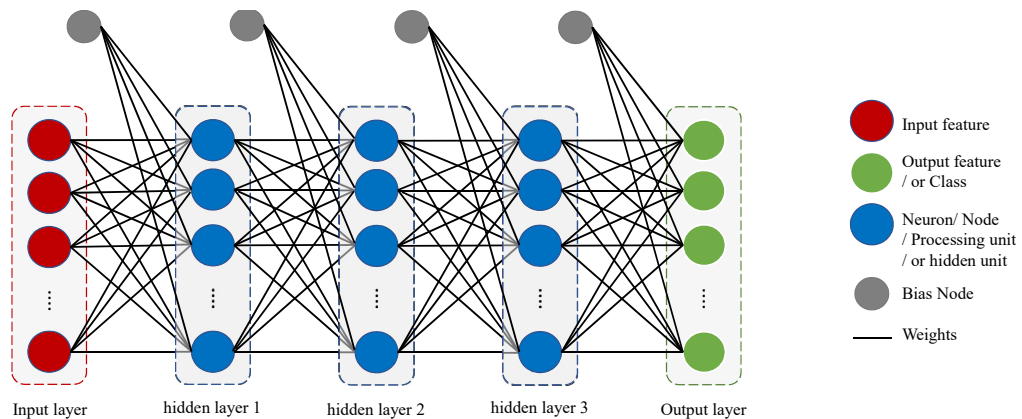
Using the backward propagation algorithm, the partial derivatives of the total cost can be easily computed with respect to the weights of the convolutional layers.

## A.6 Deep Learning

Deep learning (DL) [90] is a set of algorithms that learns in multiple non-linear transformation levels corresponding to different levels of abstraction with complex structures. It models high-level abstractions from low-level abstractions in data. The levels in these learned models correspond to different levels of features where the lower-level features can assist to represent many higher-level features, and such a hierarchy architecture of features is known as a deep architecture. Architectures of deep neural network (DNN) as shown in Figure A.7, differ from normal neural networks where DNNs have more hidden layers.

There are a very large number of different deep structures, most of them have been branched from some essential parent structures. The design of non-linear processing units in a layer relies on the task needing to be solved. DL is a fast-growing area such that new algorithms issue every few weeks. It is not always possible to compare the performance of several structures together, because they are not all estimated on the same datasets. The essential example of a DL model is the feed-forward deep network. Also, research has very successfully performed recurrent neural networks (RNNs) and convolutional deep NNs (CNNs). DL aims to learn several levels of features and abstraction which

make sense of data such as images, sound and text. DL methods depend on learning representations of data as models and inference. The data examples such as an image can be represented in several manners. For example, a vector of intensity values of pixels or in a more abstract form as a set of edges. Research in this field attempts to obtain better representations of data by designing models to learn these representations. Moreover, some feature representations of input are better than others because the performance of most learning systems depends definitively on it [142].



**Fig. A.7** An example of Deep Neural Network Architecture (DNN/ Multilayer neural network) with four fully connected layers.

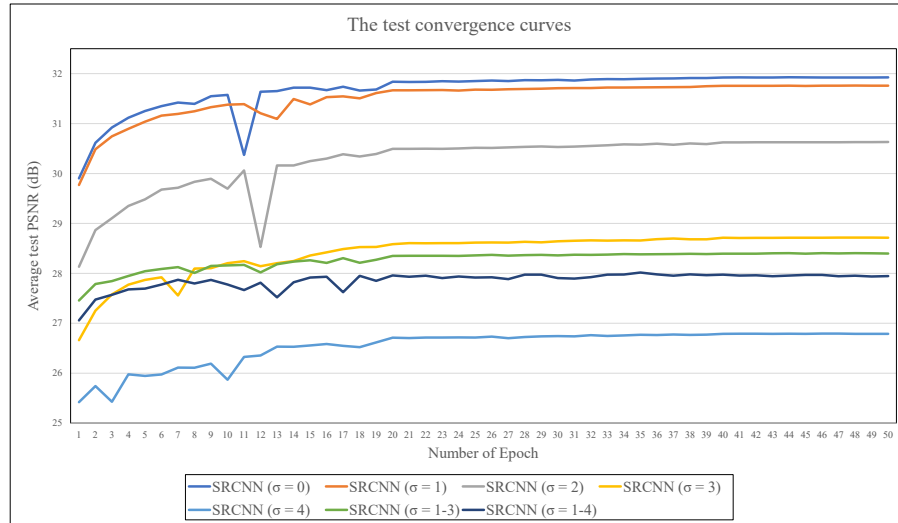
However, Deep Learning has two common issues; over-fitting and training time. Over-fitting happens when a statistical model represents a noise rather than the underlying relationship. This occurs when the model is very complicated and has too many weights relative to the number of data points. The model which has over-fitting usually produces a bad performance. Regularisation methods such as dropout regularisation [143] can be used during training to resist over-fitting, where some number of neurons are randomly deleted from the hidden layers during training. In addition, deep networks can fall into local extrema, and once trained the network is not flexible enough to adapt to another set of new data [13]. Moreover, a better theoretical understanding of deep learning and convolutional nets is a true challenge. For example, the choice of structural features and how to efficiently tune hyper-parameters of models, are still far from being a reality [13]. Also, RNNs and deep nets are suffering from vanishing or exploding gradients [13, 20].

# Appendix B

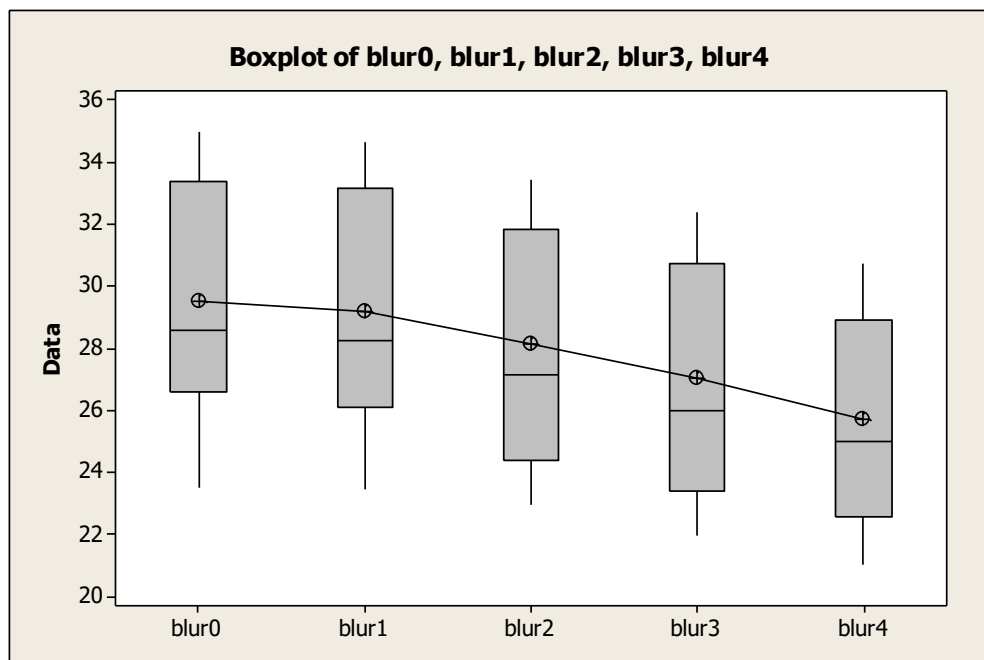
## Quantitative and Some Qualitative Results of SISR Models

### B.1 Evaluation of SRCNN

In this section, we give more details of single image super-resolution applications discussed in Chapter 4. Figure B.1 shows the test convergence curves of PSNR for all models of non-blind and blind scenarios. The curves show the difference between the performance of the different models in the non-blind scenario, where the higher the blurring level, the lower the performance. Also, in the blind situation for the models  $\sigma = 1,3$  and  $\sigma = 1,4$ , it is evident that when the model is trained on blurred images with  $\sigma = 4$ , it has led to decreasing the model performance while increasing the blurring level in the images. From boxplot graph B.2, it is apparent that there are differences in the performance mean of different networks, as when the level of blurring increases, the resolution performance decreases, and this clarifies the impact of varying blurring levels on the performance of SR using the SRCNN model.



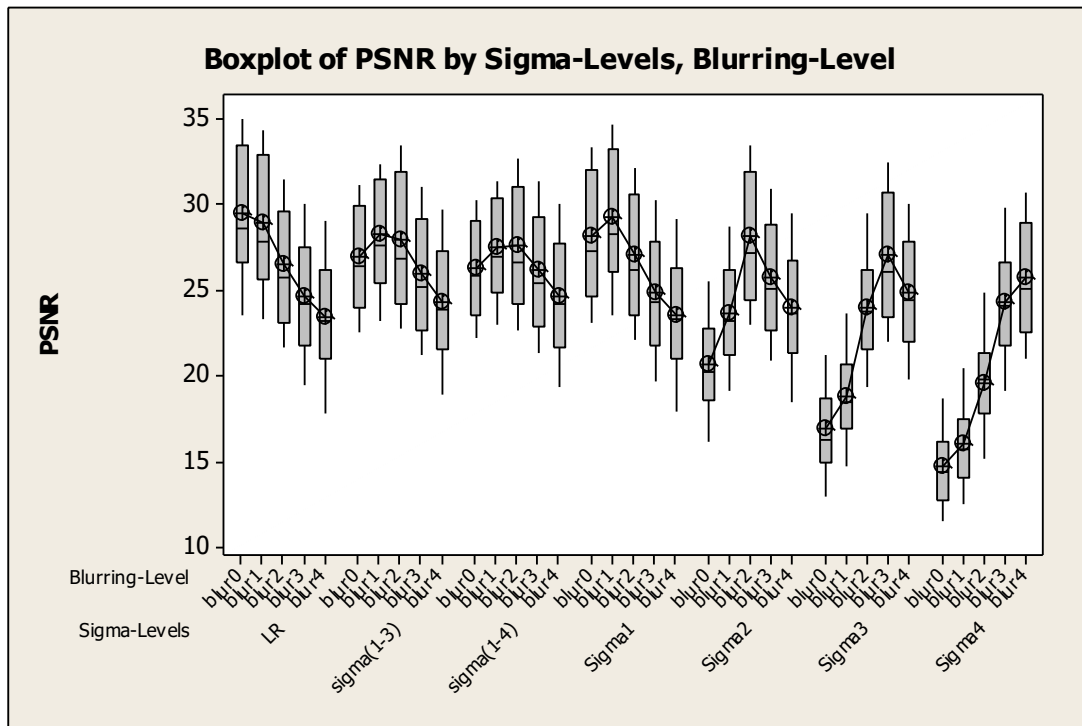
**Fig. B.1** The test convergence curves of PSNR (dB) for different SRCNN models with three layer structure (9-1-5)(64-32-1) on the Set5 dataset.



**Fig. B.2** The boxplot of different blurring levels, which collects the boxplots of all networks together in one graph to ease comparison, demonstrates that when the blurring level increases, the resolution performance decreases.



We use the boxplots to show the performance of different blind and non-blind models to be more evident in comparison. The boxplots This figure illustrates that the highest performance of any model occurs when the model is applied to the input images with the same level of blurring that the model is trained on and decreases gradually for other input images with different blurring levels. For example, the performance of the non-blind model with  $\sigma = 2$  is the highest when tested on blurred images with  $\sigma = 2$ . For blind models, on the other hand, the best performance occurs in the middle and decreases on both sides gradually.



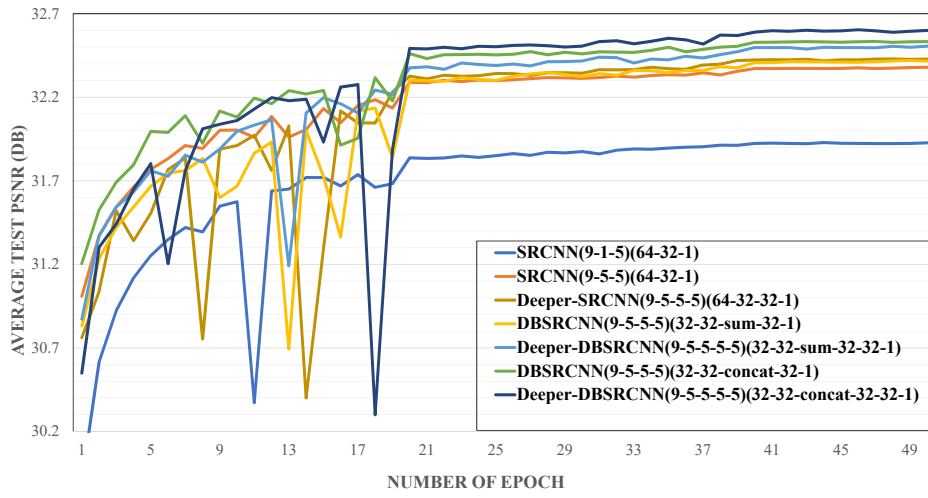
**Fig. B.3** The boxplots for different trained non-blind and blind super-resolution models. The figure shows that non-blind SR models produce the highest resolution performance for testing images when the suitable model is used, when the model and the input images have the same level of blurring. Also, the wrong blur kernel assumption negatively affects the quality of the restored images.

## B.2 Quantitative Evaluation of DBSRCNN

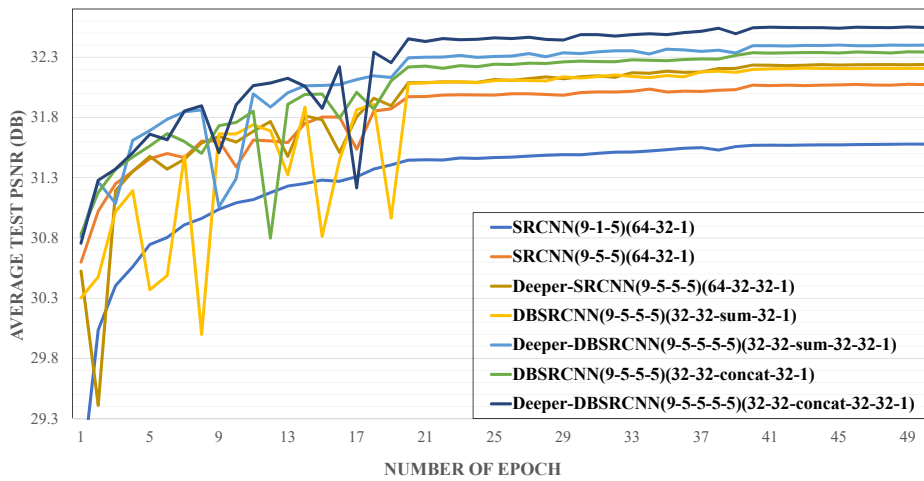
Table B.1 shows the average of evaluation PSNR for non-blind SR pipelines on Set5 for convergence curves numerically, which is revealed in Figure B.3. Also, this table gives the total training time for each model. The results show that deep SRCNN does not work well with the LR application, as indicated in Dong's paper [42]. However, it works with blurred LR applications. The improvement of the deep SRCNN is just 0.03 for a LR application. For blurred LR images with  $\sigma = 1$ , the result was increased using deep SRCNN by **0.17** more than the (9-5-5) SRCNN structure. For the blurred LR images with  $\sigma = 2$  pipeline, the difference in the result when deep SRCNN is used was **0.63**. Also, regarding the non-blind pipeline of LR with  $\sigma = 3$ , the enhancement in the result was **0.57**. Finally, the increase of performance for  $\sigma = 4$  was **0.55**. The results confirm that the higher the levels of blurring, the more layers are required for enhancing the features. Therefore, one layer in a LR application for extracting the features was enough, and the enhanced feature layer did not provide any enhancement in the performance. When we have a problematic issue such as a blurred LR application, the deeper network will be better because of the nature of its features. We can summarise this as follows: the deep SRCNN improved the results by **0.03, 0.17, 0.63, 0.57, 0.55** for  $\sigma = 0, 1, 2, 3$  and  $4$  respectively more than (9-5-5) SRCNN structure.

The proposed network (deep DBSRCNN with concatenation layer) has increased the results of different applications by **0.2, 0.48, 0.88, 0.91, 0.96** for  $\sigma = 0, 1, 2, 3$  and  $4$  respectively more than the (9-5-5) SRCNN. This means that the deep DBSRCNN is a more effective method than deep SRCNN. Besides, the higher the level of blurring in an image, the higher the improvement provided by our DBSRCNN network. For instance, the enhancement of the result using deep DBSRCNN for  $\sigma = 4$  was 0.96 but for LR was 0.20. Also, it is very clear from the results that DBSRCNN with concatenation operation provides better results than DBSRCNN with summation operation. Therefore we have chosen the concatenation operation in our experiments.

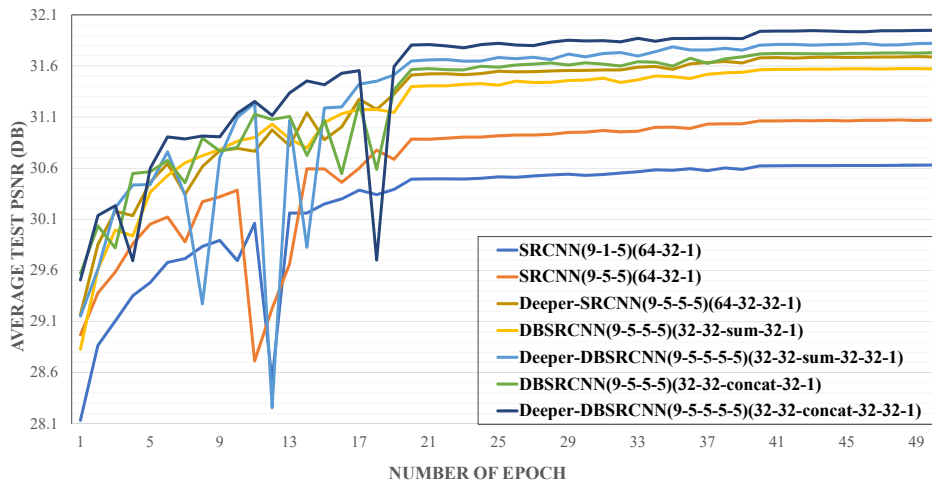
The default DBSRCNN with four layers + concatenation operation and the deep DBSRCNN with five layers + summation operation give better results than the deep SRCNN, although the number of parameters of these networks is decreased by around 2,500 parameters. Therefore, it is evident that the skip pipeline is more effective than the direct pipeline.



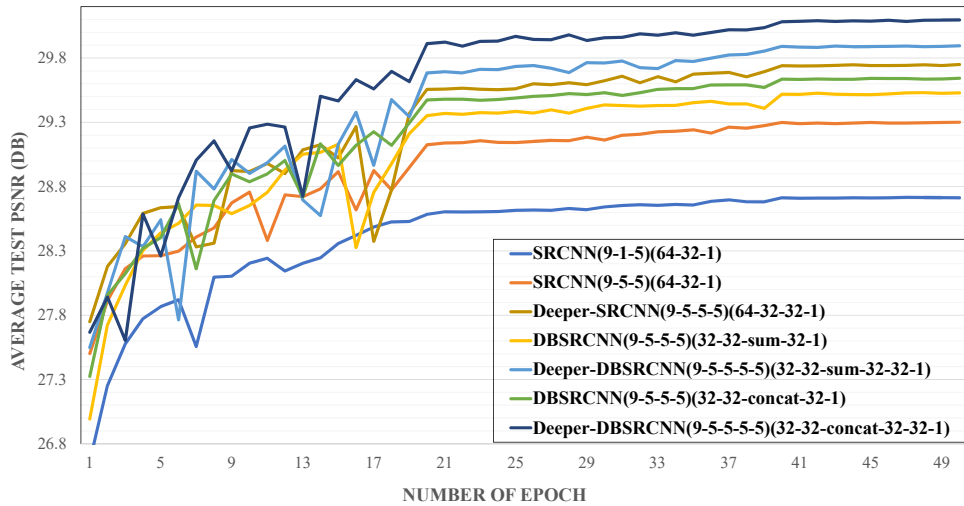
(a) SISR images from LR images.



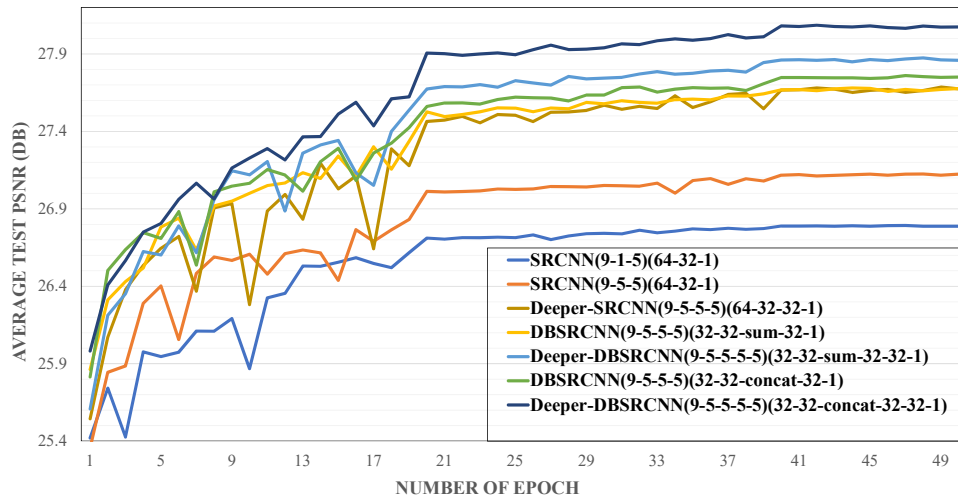
(b) SISR images from corrupted images with  $\sigma = 1$ .



(c) SISR images from corrupted images with  $\sigma = 2$ .



((a)) SISR images from corrupted images with  $\sigma = 3$ .



((b)) SISR images from corrupted images with  $\sigma = 4$ .

**Fig. B.3** The performance of the different deep learning networks which we applied on non-blind SR input images; using different structures of SRCNN networks and different structures of the proposed network DBSRCNN. It is very obvious from all figures that the worse performance was using (9-1-5)SRCNN, and the better performance was for the deeper DBSRCNN which involves concatenate operation.

**Table B.1** The convergence curves performance of the default and deep models of SRCNN and DBSRCNN for non-blind LR applications on Set5.

Net.	Network	Type of Network	# layers	# filters	Filter size	# param. <sup>1</sup>	$\sigma = 0$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$	$\sigma = 4$	Time/epoch <sup>2</sup>	Total time <sup>3</sup>
Bicubic	No AI	LR-HR	-	-	-	-	30.40	29.47	27.45	25.65	24.33	-	-
Dong et al. PAMI 2016	Re-trained SRCNN	Default	3 layers	(64-32-1)	(9-1-5)	8,032	31.91	31.58	30.67	28.63	26.74	10s	8.33m
				(64-32-1)	(9-5-5)	57,184	32.40	32.07	31.07	29.19	27.12	13s	10.83m
New Networks	Deep SRCNN	Deep	4 layers	(64-32-16-1)	(9-5-5-5)	69,584	32.35	32.15	31.66	29.52	27.54	15s	12.5m
				(64-32-32-1)	(9-5-5-5)	82,784	32.43	32.24	31.70	29.75	27.67	18s	15m
	DBSRCNN: Merge operation is summation	Default	4 layers + concate layer	(32-32-64- 32-1)	(9-5-5-5)	54,592	32.43	32.21	31.57	29.53	27.51	15s	12.5m
		Deep	5 layers + concate layer	(32-32-64-32 -32-1)	(9-5-5-5-5)	80,192	32.51	32.35	31.82	29.89	28.02	18s	15m
	DBSRCNN: Merge operation is concatenation	Default	4 layers + concate layer	(32-32-64- 32-1)	(9-5-5-5)	80,192	32.53	32.34	31.73	29.64	27.75	18s	15m
		Deep	5 layers + concate layer	(32-32-64-32 -32-1)	(9-5-5-5-5)	105,792	<b>32.60</b>	<b>32.55</b>	<b>31.95</b>	<b>30.10</b>	<b>28.08</b>	23s	19.17m

<sup>1</sup> Number of parameters; <sup>2</sup> Training time in seconds for an epoch; <sup>3</sup> Total training time in minutes/ 50 epochs

### **B.3 Qualitative Examples of DBSRCNN and DBSR**

In this section, we present more additional qualitative examples for single image super-resolution application for blind and non-blind scenarios using the SRCNN, DBSRCNN and DBSR networks.

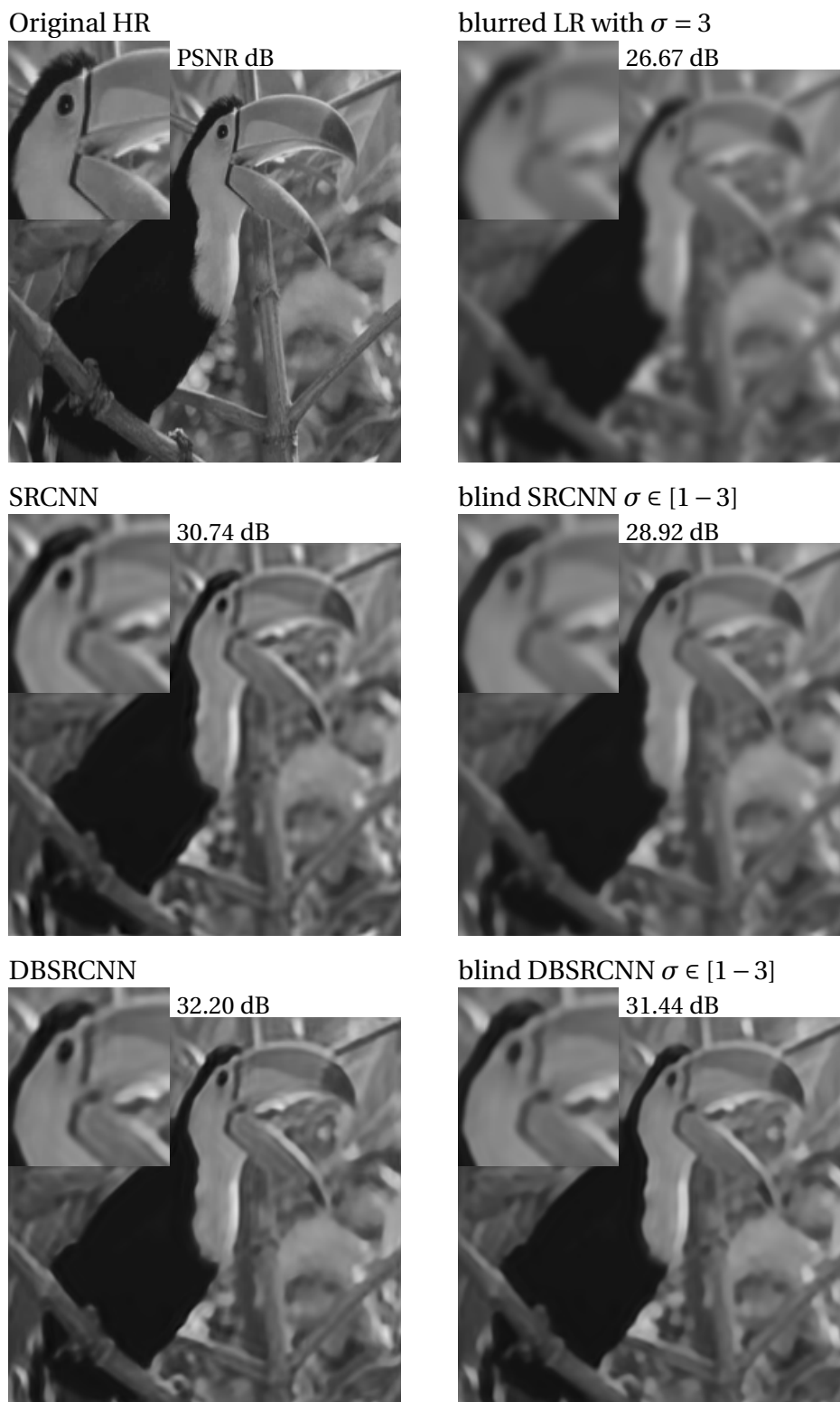


**Fig. B.4** SR with SRCNN and DBSRCNN on grey-scale images after Gaussian blur with  $\sigma = 1$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind  $\sigma \in [1 - 3]$  SR scenarios). Each result is accompanied by zoom and PSNR dB.

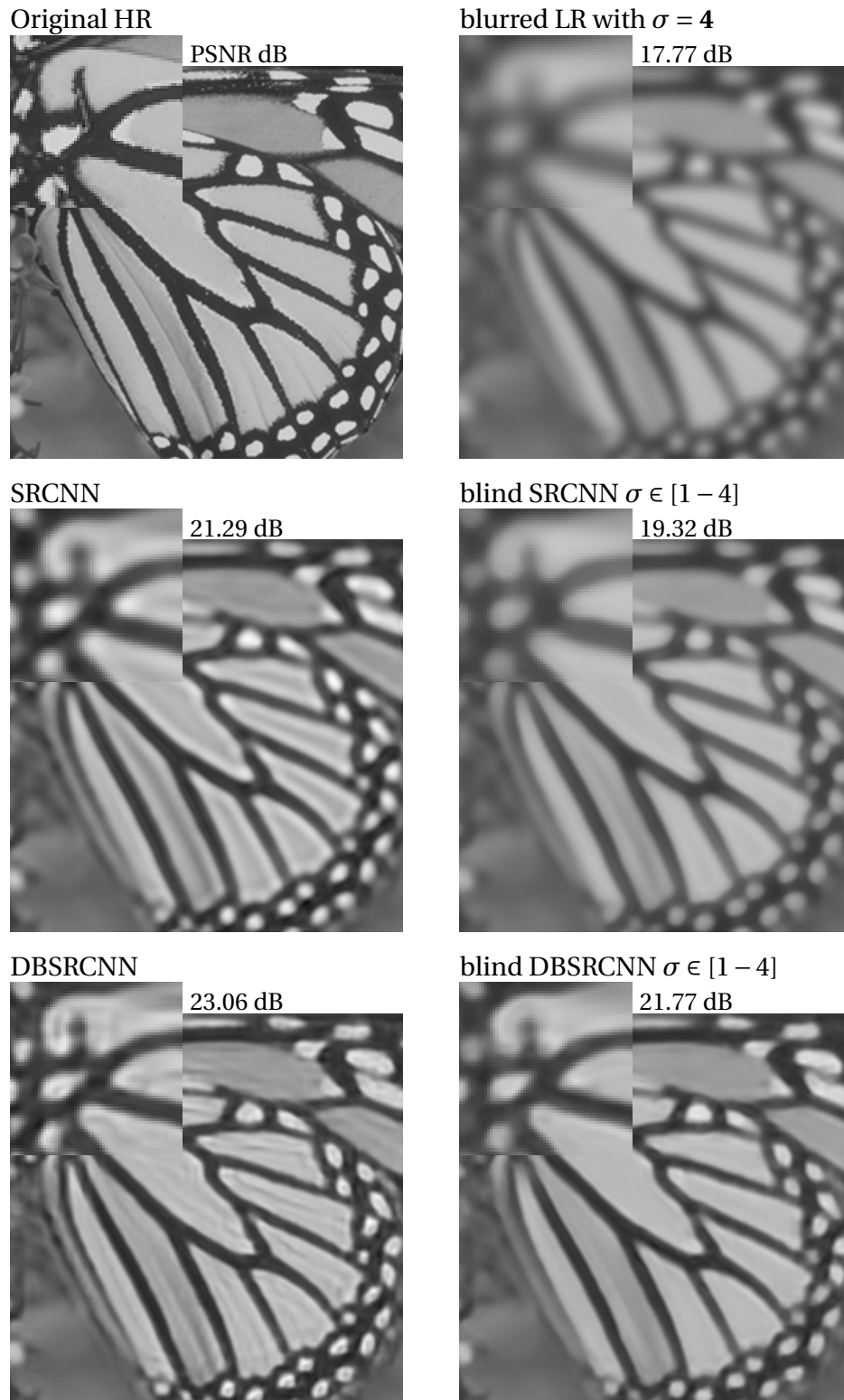


**Fig. B.5** SR with SRCNN and DBSRCNN on grey-scale images after Gaussian blur with  $\sigma = 3$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind  $\sigma \in [1 - 3]$  SR scenarios). Each result is accompanied by zoom and PSNR dB.





**Fig. B.6** SR with SRCNN and DBSRCNN on grey-scale images after Gaussian blur with  $\sigma = 3$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind  $\sigma \in [1 - 3]$  SR scenarios). Each result is accompanied by zoom and PSNR dB.



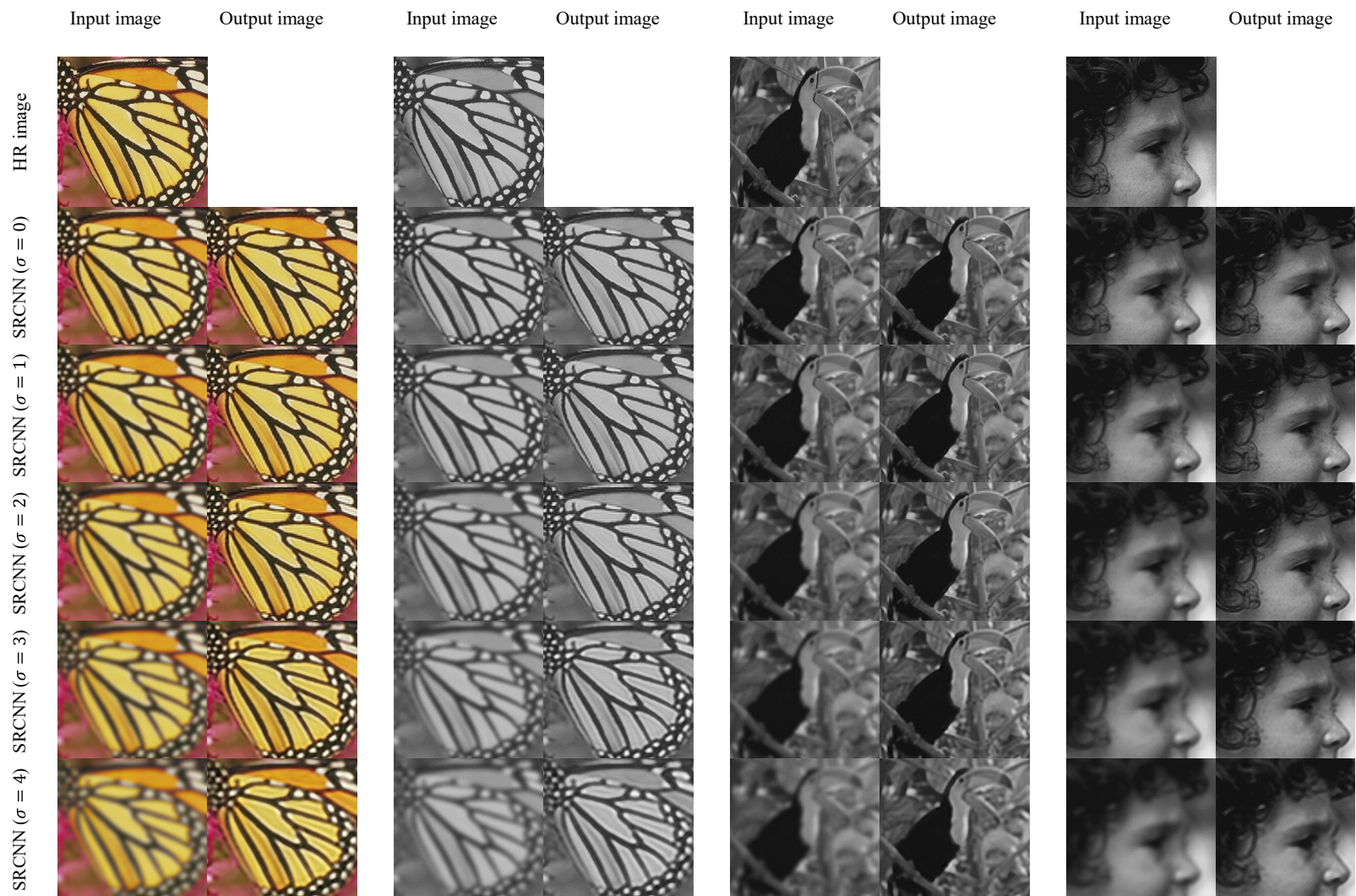
**Fig. B.7** SR with SRCNN and DBSRCNN on grey-scale images after Gaussian blur with  $\sigma = 4$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind  $\sigma \in [1 - 4]$  SR scenarios). Each result is accompanied by zoom and PSNR dB.



**Fig. B.8** SR with SRCNN and DBSRCNN on a colour image after Gaussian blur with  $\sigma = 2$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind  $\sigma \in [1 - 3]$  SR scenarios). Each result is accompanied by zoom and PSNR dB.



**Fig. B.9** SR with SRCNN and DBSRCNN on a colour image after Gaussian blur with  $\sigma = 3$ . The second and third rows show the results of SRCNN and DBSRCNN, respectively; according to the two different scenarios (the non-blind and blind  $\sigma \in [1 - 3]$  SR scenarios). Each result is accompanied by zoom and PSNR dB.



**Fig. B.10** Example of non-blind SRCNN (9-1-5)(64-32-1).








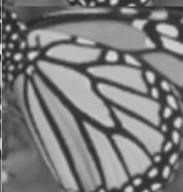











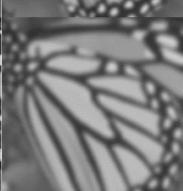





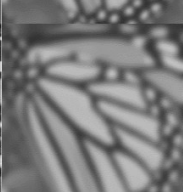

















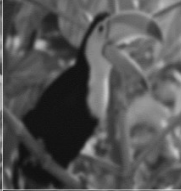


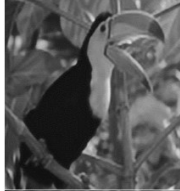


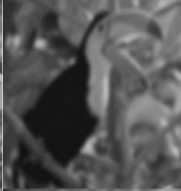





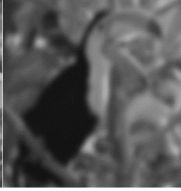
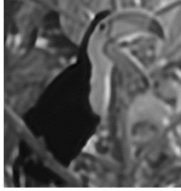
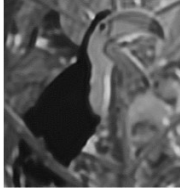
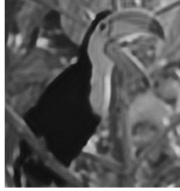
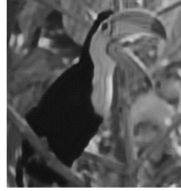
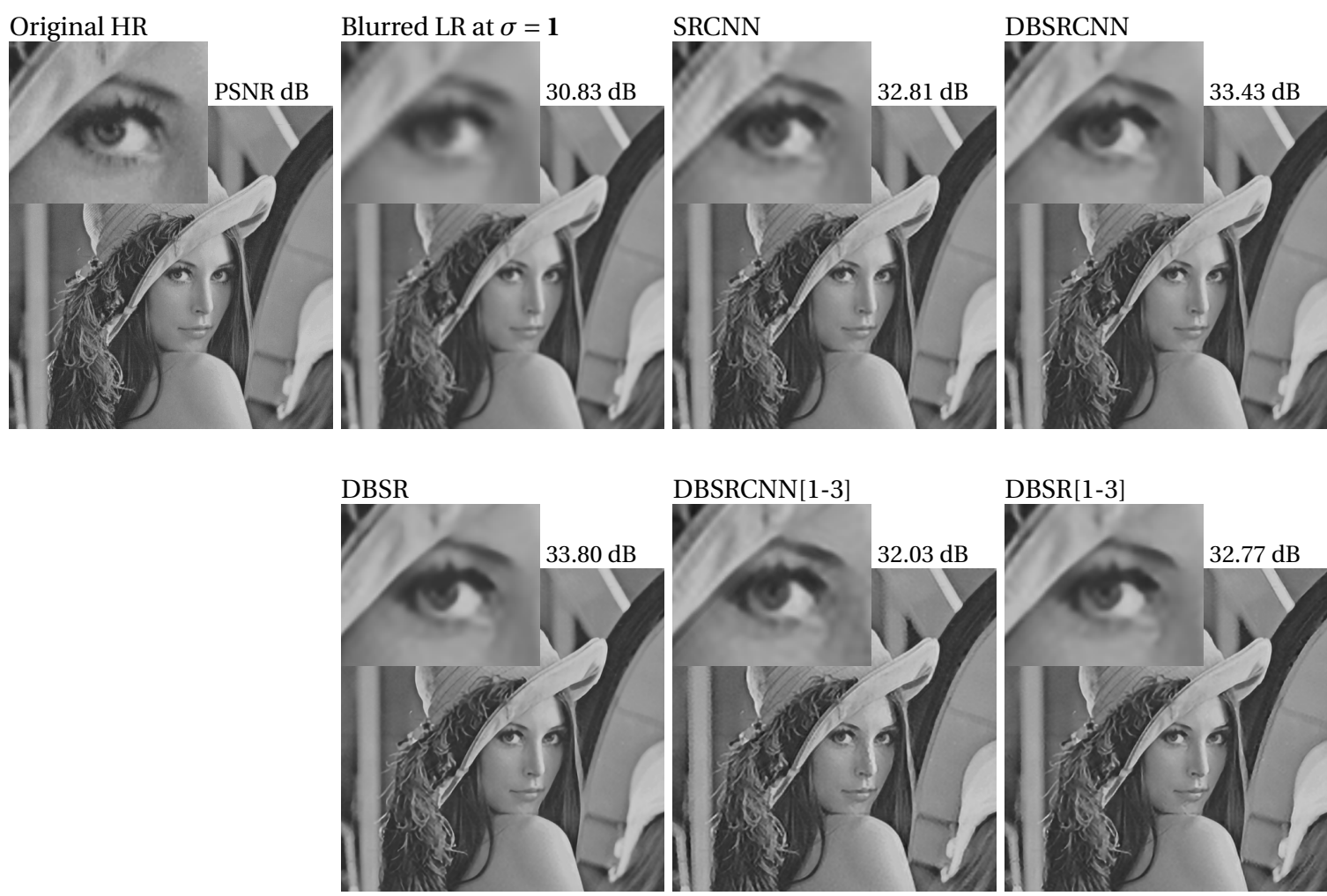
App.	HR image	Blurred LR image	PSNR/ SSIM	SRCNN (9-5-5)	PSNR/ SSIM	Deep SRCNN	PSNR/ SSIM	DB-SRCNN	PSNR/ SSIM	Deep DB- SRCNN	PSNR/ SSIM
LR ( $\sigma = 0$ )			24.04/ 0.8820		27.47/ 0.9021		27.55/ 0.9040		27.71/ 0.9090		27.79/ 0.9104
blur1 ( $\sigma = 1$ )			23.12/ 0.7925		27.00/ 0.8893		27.24/ 0.8966		27.50/ 0.9031		27.90/ 0.9106
blur2 ( $\sigma = 2$ )			21.06/ 0.7061		25.79/ 0.8519		26.73/ 0.8828		26.67/ 0.8812		26.95/ 0.8871
blur3 ( $\sigma = 3$ )			19.17/ 0.6054		24.05/ 0.7927		24.79/ 0.8236		24.59/ 0.8143		25.17/ 0.8331
blur4 ( $\sigma = 4$ )			17.77/ 0.5204		21.57/ 0.6744		22.45/ 0.7150		22.57/ 0.7229		23.06/ 0.7423

Fig. B.11 Example 1: example for different networks for each application.

models

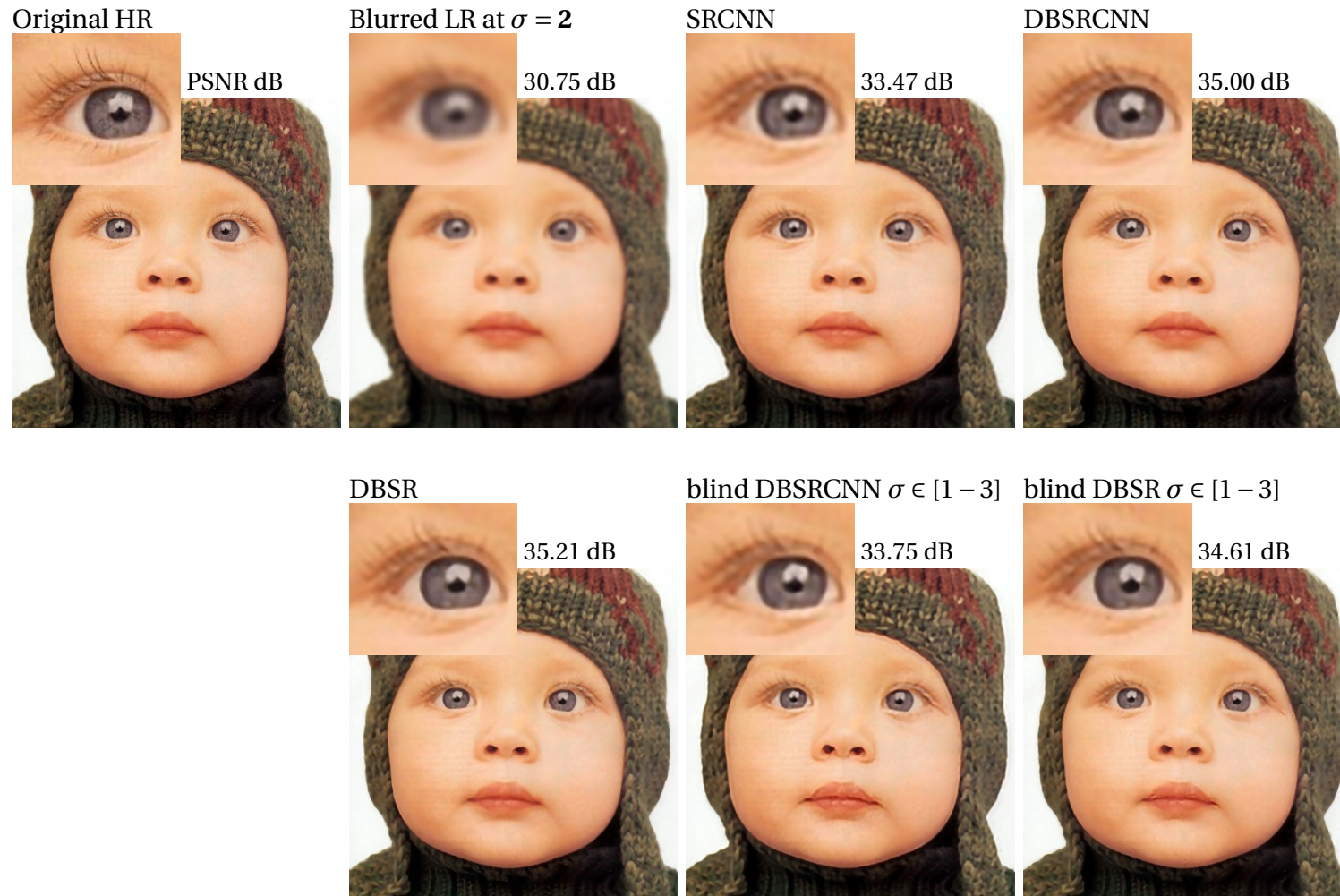
App.	HR image	Blurred LR image	PSNR/ SSIM	SRCNN (9-5-5)	PSNR/ SSIM	Deep SRCNN	PSNR/ SSIM	DB-SRCNN	PSNR/ SSIM	Deep DB- SRCNN	PSNR/ SSIM
LR ( $\sigma = 0$ )			32.58/ 0.9261		34.85/ 0.9488		34.93/ 0.9493		35.05/ 0.9511		35.22/ 0.9532
blur1 ( $\sigma = 1$ )			31.38/ 0.9067		34.57/ 0.9456		34.78/ 0.9485		34.83/ 0.9489		35.33/ 0.9531
blur2 ( $\sigma = 2$ )			28.85/ 0.8487		33.66/ 0.9342		34.19/ 0.9410		34.18/ 0.9416		34.58/ 0.9450
blur3 ( $\sigma = 3$ )			26.67/ 0.7786		31.38/ 0.8906		31.94/ 0.9007		31.79/ 0.8975		32.49/ 0.9097
blur4 ( $\sigma = 4$ )			25.10/ 0.7181		29.02/ 0.8308		29.69/ 0.8462		29.87/ 0.8503		30.36/ 0.8581

**Fig. B.12** Example 2: example for different networks for each application.

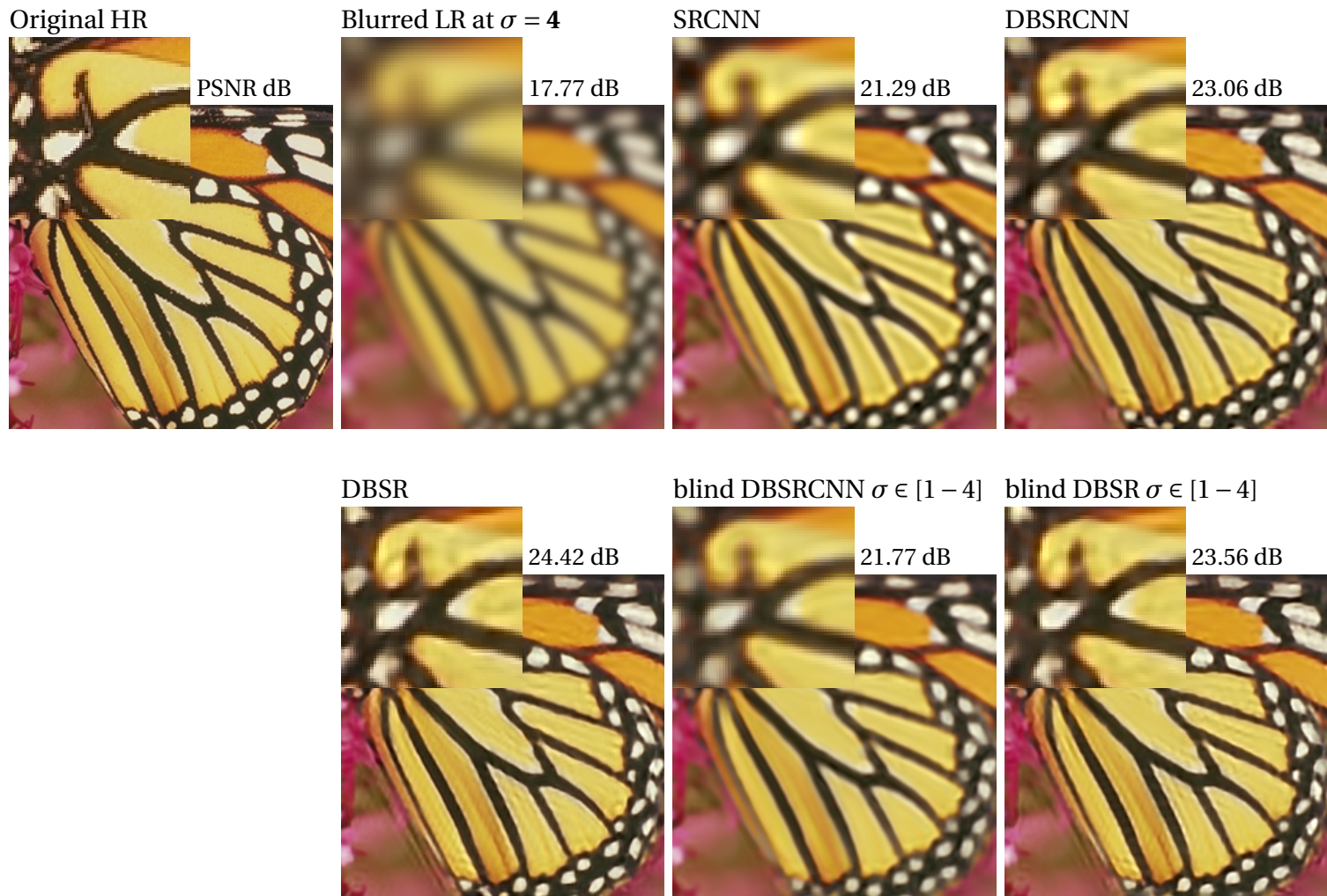


**Fig. B.13** SR with different models on images after Gaussian blur with  $\sigma = 1$ . The results show the non-blind and blind scenarios. Each result is accompanied by zoom and PSNR.





**Fig. B.14** SR with different models on images after Gaussian blur with  $\sigma = 2$ . The results show the non-blind and blind scenarios. Each result is accompanied by zoom and PSNR.



**Fig. B.15** SR with different models on images after Gaussian blur with  $\sigma = 4$ . The results show the non-blind and blind scenarios. Each result is accompanied by zoom and PSNR.

# References

- [1] James E Adams and Bruce Pillman. Digital camera image formation: Introduction and hardware. In *Digital Image Forensics*, pages 3–44. Springer, 2013.
- [2] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 126–135, 2017.
- [3] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.
- [4] Arthur Albert. *Regression and the Moore-Penrose pseudoinverse*. Elsevier, 1972.
- [5] Fatma Albluwi. DBSRCNN-Keras Code. <https://github.com/Fatma-ALbluwi/DBSRCNN>, 2018.
- [6] Fatma Albluwi, Vladimir A Krylov, and Rozenn Dahyot. Artifacts reduction in jpeg-compressed images using cnns. In *Irish Machine Vision and Image Processing conference (IMVIP)*, Belfast, United Kingdom, 2018.
- [7] Fatma Albluwi, Vladimir A Krylov, and Rozenn Dahyot. Image deblurring and super-resolution using deep convolutional neural networks. In *28th IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, Aalborg, Denmark, 2018.
- [8] Fatma Albluwi, Vladimir A Krylov, and Rozenn Dahyot. Denoising renoir image dataset with dbsr. In *Proc. Irish Machine Vision and Image Processing conference (IMVIP)*, Dublin, Ireland, 2019.

- [9] Fatma Albluwi, Vladimir A Krylov, and Rozenn Dahyot. Super-resolution on degraded low-resolution images using convolutional neural networks. In *27th European Signal Processing Conference (EUSIPCO)*, pages 1–5, Spain, 2019.
- [10] Jan Allebach and Ping Wah Wong. Edge-directed interpolation. In *Proceedings of 3rd IEEE International Conference on Image Processing (ICIP)*, volume 3, pages 707–710, 1996.
- [11] Hussein A Aly and Eric Dubois. Image up-sampling using total-variation regularization with a new observation model. *IEEE Transactions on Image Processing*, 14(10): 1647–1659, 2005.
- [12] Josue Anaya and Adrian Barbu. Renoir—a dataset for real low-light image noise reduction. *Journal of Visual Communication and Image Representation*, 51:144–154, 2018.
- [13] Plamen Angelov and Alessandro Sperduti. Challenges in deep learning. In *Proceedings of the 24th European symposium on artificial neural networks (ESANN)*, pages 489–495, 2016.
- [14] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 33(5):898–916, 2011.
- [15] Suyash P Awate and Ross T Whitaker. Unsupervised, information-theoretic, adaptive image filtering for image restoration. *IEEE Transactions on pattern analysis and machine intelligence (PAMI)*, 28(3):364–376, 2006.
- [16] Simon Baker and Takeo Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(9):1167–1183, 2002.
- [17] Mark R Banham and Aggelos K Katsaggelos. Digital image restoration. *IEEE signal processing magazine*, 14(2):24–41, 1997.
- [18] Adrian Barbu. Training an active random field for real-time image denoising. *IEEE Transactions on Image Processing*, 18(11):2451–2462, 2009.

- 
- [19] Jacob Benesty, Jingdong Chen, and Yiteng Huang. Study of the widely linear wiener filter for noise reduction. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 205–208, 2010.
- [20] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2): 157–166, 1994.
- [21] Marco Bevilacqua, Aline Roumy, Christine Guillemot, and Marie Line Alberi-Morel. Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In *Proceedings of the 23rd British Machine Vision Conference (BMVC)*, 2012.
- [22] Leo Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231, 2001.
- [23] Michael S Brown, Peng Song, and Tat-Jen Cham. Image pre-conditioning for out-of-focus projector blur. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1956–1963, 2006.
- [24] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 60–65, 2005.
- [25] Harold C Burger, Christian J Schuler, and Stefan Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2392–2399, 2012.
- [26] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimisation*, 20(4):1956–1982, 2010.
- [27] 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, 2011.

- [28] Hong Chang, Dit-Yan Yeung, and Yimin Xiong. Super-resolution through neighbor embedding. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2004.
- [29] Yunjin Chen and Thomas Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *Transactions on pattern analysis and machine intelligence (PAMI)*, 39(6):1256–1272, 2016.
- [30] Yunjin Chen, Thomas Pock, René Ranftl, and Horst Bischof. Revisiting loss-specific training of filter-based mrfs for image restoration. In *German Conference on Pattern Recognition*, pages 271–281, 2013.
- [31] François Chollet et al. *Keras*. <https://github.com/fchollet/keras>, 2015.
- [32] Peter Corcoran and Petronel Bigioi. Consumer imaging I – processing pipeline focus and exposure. In *Handbook of Visual Display Technology*, pages 1–25. Springer, 2016.
- [33] Zhen Cui, Hong Chang, Shiguang Shan, Bineng Zhong, and Xilin Chen. Deep network cascade for image super-resolution. In *European Conference on Computer Vision (ECCV)*, pages 49–64, 2014.
- [34] Ricardo Dutra Da Silva, Rodrigo Minetto, William Robson Schwartz, and Helio Pedrini. Adaptive edge-preserving image denoising using wavelet transforms. *Pattern analysis and applications*, 16(4):567–580, 2013.
- [35] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.
- [36] Shengyang Dai, Mei Han, Wei Xu, Ying Wu, and Yihong Gong. Soft edge smoothness prior for alpha channel super resolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 7, pages 1–8, 2007.
- [37] Guy Demoment. Image reconstruction and restoration: Overview of common estimation structures and problems. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):2024–2036, 1989.

- 
- [38] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 248–255, 2009.
- [39] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.
- [40] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision (ECCV)*, pages 184–199, 2014.
- [41] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. Compression artifacts reduction by a deep convolutional network. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 576–584, 2015.
- [42] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 38(2):295–307, 2015.
- [43] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *European Conference on Computer Vision (ECCV)*, pages 391–407. Springer, 2016.
- [44] Weisheng Dong, Lei Zhang, Guangming Shi, and Xin Li. Nonlocally centralized sparse representation for image restoration. *IEEE transactions on Image Processing*, 22(4):1620–1630, 2012.
- [45] Weisheng Dong, Lei Zhang, Guangming Shi, and Xin Li. Nonlocally centralized sparse representation for image restoration. *IEEE Transactions on Image Processing*, 22(4):1620–1630, 2013.
- [46] Mehran Ebrahimi and Edward R Vrscay. Solving the inverse problem of image zooming using “self-examples”. In *International Conference Image Analysis and Recognition*, pages 117–130. Springer, 2007.
- [47] Netalee Efrat, Daniel Glasner, Alexander Apartsin, Boaz Nadler, and Anat Levin. Accurate blur models vs. image priors in single image super-resolution. In *Proceedings*

- of the IEEE International Conference on Computer Vision (ICCV)*, pages 2832–2839, 2013.
- [48] Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 15(12):3736–3745, 2006.
- [49] Michael Elad and Arie Feuer. Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images. *IEEE transactions on image processing*, 6(12):1646–1658, 1997.
- [50] Linwei Fan, Xuemei Li, Hui Fan, Yanli Feng, and Caiming Zhang. Adaptive texture-preserving denoising method using gradient histogram and nonlocal self-similarity priors. *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.
- [51] Raanan Fattal. Image upsampling via imposed edge statistics. *ACM transactions on graphics (TOG)*, 26(3):95, 2007.
- [52] Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images. *IEEE Transactions on Image Processing (TIP)*, 16(5):1395–1411, 2007.
- [53] Alessandro Foi, Mejdji Trimeche, Vladimir Katkovnik, and Karen Egiazarian. Practical poissonian-gaussian noise modeling and fitting for single-image raw-data. *IEEE Transactions on Image Processing*, 17(10):1737–1754, 2008.
- [54] William T Freeman, Egon C Pasztor, and Owen T Carmichael. Learning low-level vision. *International journal of computer vision*, 40(1):25–47, 2000.
- [55] William T Freeman, Thouis R Jones, and Egon C Pasztor. Example-based super-resolution. *IEEE Computer graphics and Applications*, 22(2):56–65, 2002.
- [56] WT Freeman and Ce Liu. Markov random fields for super-resolution and texture synthesis. *Advances in Markov Random Fields for Vision and Image Processing*, 1 (155-165):3, 2011.
- [57] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.



- [58] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 580–587, 2014.
- [59] Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *IEEE 12th international conference on computer vision*, pages 349–356, 2009.
- [60] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press Cambridge, 2016.
- [61] Jinjin Gu, Hannan Lu, Wangmeng Zuo, and Chao Dong. Blind super-resolution with iterative kernel correction. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1604–1613, 2019.
- [62] Shuhang Gu, Lei Zhang, Wangmeng Zuo, and Xiangchu Feng. Weighted nuclear norm minimization with application to image denoising. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2862–2869, 2014.
- [63] Shuhang Gu, Qi Xie, Deyu Meng, Wangmeng Zuo, Xiangchu Feng, and Lei Zhang. Weighted nuclear norm minimization and its applications to low level vision. *International journal of computer vision*, 121(2):183–208, 2017.
- [64] Gajanand Gupta. Algorithm for image processing using improved median filter and comparison of mean, median and improved median filter. *International Journal of Soft Computing and Engineering (IJSCE)*, 1(5):304–311, 2011.
- [65] He He and Wan-Chi Siu. Single image super-resolution using gaussian process regression. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 449–456. IEEE, 2011.
- [66] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5353–5360, 2015.

- [67] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 1026–1034, 2015.
- [68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.
- [69] Donald Olding Hebb. *The organization of behavior*. J. Wiley; Chapman & Hall, 1949.
- [70] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- [71] Michal Hradiš, Jan Kotera, Pavel Zemčík, and Filip Šroubek. Convolutional neural networks for direct text deblurring. In *The British Machine Vision Conference (BMVC)*, volume 10, page 2, 2015.
- [72] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5197–5206, 2015.
- [73] Viren Jain and Sebastian Seung. Natural image denoising with convolutional networks. In *Advances in Neural Information Processing Systems*, pages 769–776, 2009.
- [74] Jeremy Jancsary, Sebastian Nowozin, and Carsten Rother. Loss-specific training of non-parametric image restoration models: A new state of the art. In *European Conference on Computer Vision (ECCV)*, pages 112–125. Springer, 2012.
- [75] Hui Ji, Chaoqiang Liu, Zuowei Shen, and Yuhong Xu. Robust video denoising using low rank matrix completion. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1791–1798, 2010.
- [76] Hui Ji, Sibin Huang, Zuowei Shen, and Yuhong Xu. Robust video restoration by joint sparse and low rank matrix approximation. *SIAM Journal on Imaging Sciences*, 4(4): 1122–1142, 2011.
- [77] Cheolkon Jung, Licheng Jiao, Hongtao Qi, and Tian Sun. Image deblocking via sparse representation. *Signal Processing: Image Communication*, 27(6):663–677, 2012.

- [78] Sergei Igorevich Kabanikhin. Definitions and examples of inverse and ill-posed problems. *Journal of Inverse and Ill-Posed Problems*, 16(4):317–357, 2008.
- [79] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014.
- [80] Jozef Kelemen. From artificial neural networks to emotion machines with marvin minsky. *Acta Polytechnica Hungarica*, 4(4):1–12, 2007.
- [81] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1646–1654, 2016.
- [82] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1637–1645, 2016.
- [83] Kwang In Kim and Younghee Kwon. Single-image super-resolution using sparse regression and natural image prior. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 32(6):1127–1133, 2010.
- [84] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [85] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [86] Jinsa Kuruvilla and K Gunavathi. Lung cancer classification using neural networks for ct images. *Computer methods and programs in biomedicine*, 113(1):202–209, 2014.
- [87] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 624–632, 2017.

- [88] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Fast and accurate image super-resolution with deep laplacian pyramid networks. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 41(11):2599–2613, 2018.
- [89] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–104, 2004.
- [90] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.
- [91] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P 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)*, volume 2, pages 4681–4690, 2017.
- [92] Seong Won Lee and Joon Ki Paik. Image interpolation using adaptive fast b-spline filtering. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 177–180, 1993.
- [93] Anat Levin, Rob Fergus, Frédo Durand, and William T Freeman. Image and depth from a conventional camera with a coded aperture. *ACM transactions on graphics (TOG)*, 26(3):70, 2007.
- [94] Xin Li and Michael T Orchard. New edge-directed interpolation. *IEEE transactions on image processing*, 10(10):1521–1527, 2001.
- [95] Zhen Li, Jinglei Yang, Zheng Liu, Xiaomin Yang, Gwanggil Jeon, and Wei Wu. Feedback network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3867–3876, 2019.
- [96] AW-C Liew and Hong Yan. Blocking artifacts suppression in block-coded images using overcomplete wavelet representation. *IEEE transactions on circuits and systems for video technology (TCSVT)*, 14(4):450–461, 2004.

- [97] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPRW)*, pages 1132–1140, 2017.
- [98] Peter List, Anthony Joch, Jani Lainema, Gisle Bjontegaard, and Marta Karczewicz. Adaptive deblocking filter. *IEEE transactions on circuits and systems for video technology (TCSVT)*, 13(7):614–619, 2003.
- [99] Ding Liu, Zhaowen Wang, Bihan Wen, Jianchao Yang, Wei Han, and Thomas S Huang. Robust single image super-resolution via deep networks with sparse prior. *IEEE Transactions on Image Processing*, 25(7):3194–3207, 2016.
- [100] Kui Liu, Jieqing Tan, and Benyue Su. An adaptive image denoising model based on tikhonov and tv regularisations. *Advances in Multimedia*, 2014:8, 2014.
- [101] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3431–3440, 2015.
- [102] Lu Lu, Weiqi Jin, and Xia Wang. Non-local means image denoising with a soft threshold. *IEEE Signal Processing Letters*, 22(7):833–837, 2015.
- [103] Kede Ma, Zhengfang Duanmu, Qingbo Wu, Zhou Wang, Hongwei Yong, Hongliang Li, and Lei Zhang. Waterloo exploration database: New challenges for image quality assessment models. *IEEE Transactions on Image Processing*, 26(2):1004–1016, 2016.
- [104] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, page 3, 2013.
- [105] Julien Mairal, Michael Elad, and Guillermo Sapiro. Sparse representation for color image restoration. *IEEE Transactions on image processing*, 17(1):53–69, 2007.
- [106] Julien Mairal, Francis R Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *IEEE 12th international conference on computer vision (ICCV)*, volume 29, pages 54–62. Citeseer, 2009.

- [107] Maurits Malfait and Dirk Roose. Wavelet-based image denoising using a markov random field a priori model. *IEEE Transactions on image processing*, 6(4):549–565, 1997.
- [108] Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, (7):674–693, 1989.
- [109] Xiaojiao Mao, Chunhua Shen, and Yu-Bin Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. In *Advances in neural information processing systems (NIPS)*, pages 2802–2810, 2016.
- [110] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423, 2001.
- [111] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
- [112] Keith Miller. Least squares methods for ill-posed problems with a prescribed bound. *SIAM Journal on Mathematical Analysis*, 1(1):52–74, 1970.
- [113] Bryan S Morse and Duane Schwartzwald. Image magnification using level-set reconstruction. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1:333–340, 2001.
- [114] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML)*, pages 807–814, 2010.
- [115] Rajashree Nayak, S Monalisa, and Dipti Patra. Spatial super resolution based image reconstruction using hibp. In *India Conference (INDICON), 2013 Annual IEEE*, pages 1–6, 2013.
- [116] Shree K Nayar and Moshe Ben-Ezra. Motion-based motion deblurring. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 26(6):689–698, 2004.

- [117] Aria Nosratinia. Embedded post-processing for enhancement of compressed images. In *Proceedings Data Compression Conference (DCC)*, pages 62–71. IEEE, 1999.
- [118] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *IEEE signal processing magazine*, 20(3):21–36, 2003.
- [119] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.
- [120] Stuart W Perry and Ling Guan. Weight assignment for adaptive image restoration by neural networks. *IEEE Transactions on neural networks*, 11(1):156–170, 2000.
- [121] Javier Portilla, Vasily Strela, Martin J Wainwright, and Eero P Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Trans Image Processing*, 12(11), 2003.
- [122] Matan Protter, Michael Elad, Hiroyuki Takeda, and Peyman Milanfar. Generalizing the nonlocal-means to super-resolution reconstruction. *IEEE Transactions on image processing*, 18(1):36–51, 2008.
- [123] Yajun Qiu, Ruxin Wang, Dapeng Tao, and Jun Cheng. Embedded block residual network: A recursive restoration model for single-image super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4180–4189, 2019.
- [124] Ramesh Raskar, Amit Agrawal, and Jack Tumblin. Coded exposure photography: motion deblurring using fluttered shutter. In *ACM transactions on graphics (TOG)*, volume 25, pages 795–804, 2006.
- [125] Howard C Reeve and Jae S Lim. Reduction of blocking effects in image coding. *Optical Engineering*, 23(1):230134, 1984.
- [126] Gernot Riegler, Samuel Schuler, Matthias Ruther, and Horst Bischof. Conditioned regression models for non-blind single image super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 522–530, 2015.

- [127] Yaniv Romano, John Isidoro, and Peyman Milanfar. Raisr: Rapid and accurate image super resolution. *IEEE Transactions on Computational Imaging*, 3(1):110–125, 2017.
- [128] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [129] Stefan Roth and Michael J Black. Fields of experts: A framework for learning image priors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 860–867, 2005.
- [130] Leonid I Rudin and Stanley Osher. Total variation based image restoration with free local constraints. In *Proceedings of 1st International Conference on Image Processing*, volume 1, pages 31–35. IEEE, 1994.
- [131] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.
- [132] David Rumelhart, DRGHR Williams, and Geoffrey Hinton. Learning representations by back-propagating errors. *Nature*, 323(6088):533–538, 1986.
- [133] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press, Cambridge, 1985.
- [134] Uwe Schmidt, Qi Gao, and Stefan Roth. A generative perspective on mrfs in low-level vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1751–1758, 2010.
- [135] Samuel Schulter, Christian Leistner, and Horst Bischof. Fast and accurate image upscaling with super-resolution forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3791–3799, 2015.
- [136] Qi Shan, Zhaorong Li, Jiaya Jia, and Chi-Keung Tang. Fast image/video upsampling. In *ACM Transactions on Graphics (TOG)*, volume 27, page 153. ACM, 2008.



- [137] Hamid R Sheikh, Alan C Bovik, and Gustavo De Veciana. An information fidelity criterion for image quality assessment using natural scene statistics. *IEEE Transactions on image processing (TIP)*, 14(12):2117–2128, 2005.
- [138] Hamid R Sheikh, Muhammad F Sabir, and Alan C Bovik. A statistical evaluation of recent full reference image quality assessment algorithms. *IEEE Transactions on image processing (TIP)*, 15(11):3440–3451, 2006.
- [139] Wenzhe Shi, Jose Caballero, Ferenc Huszár, 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. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1874–1883, 2016.
- [140] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.
- [141] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [142] Richard Socher. *Recursive Deep Learning for Natural Language Processing and Computer Vision*. PhD thesis, Stanford University, 2014.
- [143] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.
- [144] Henry Stark. *Image recovery: theory and application*. Elsevier, 2013.
- [145] Deqing Sun and Wai-Kuen Cham. Postprocessing of low bit-rate block dct coded images based on a fields of experts prior. *IEEE Transactions on Image Processing (TIP)*, 16(11):2743–2751, 2007.
- [146] Jian Sun, Zongben Xu, and Heung-Yeung Shum. Image super-resolution using gradient profile prior. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

- [147] Jian Sun, Zongben Xu, and Heung-Yeung Shum. Gradient profile prior and its applications in image super-resolution and enhancement. *IEEE Transactions on Image Processing (TIP)*, 20(6):1529–1542, 2010.
- [148] Jian Sun, Wenfei Cao, Zongben Xu, and Jean Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
- [149] Pavel Svoboda, Michal Hradiš, David Bařina, and Pavel Zemčık. Compression artifacts removal using convolutional neural networks. *Journal of WSCG*, 24(2):63–72, 2016. ISSN 1213-6972. URL [http://www.fit.vutbr.cz/research/view\\_pub.php?id=11176](http://www.fit.vutbr.cz/research/view_pub.php?id=11176).
- [150] Pavel Svoboda, Michal Hradiš, Lukáš Maršık, and Pavel Zemcık. CNN for license plate motion deblurring. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3832–3836, 2016.
- [151] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. Memnet: A persistent memory network for image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4539–4547, 2017.
- [152] Yaniv Taigman, Ming Yang, Marc’Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1701–1708, 2014.
- [153] Yapeng Tian. *SRCNN-Keras Code*. <https://github.com/YapengTian/SRCNN-Keras>, 2017.
- [154] Radu Timofte, Vincent De Smet, and Luc Van Gool. Anchored neighborhood regression for fast example-based super-resolution. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1920–1927, 2013.
- [155] Radu Timofte, Vincent De Smet, and Luc Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In *Asian Conference on Computer Vision*, pages 111–126. Springer, 2014.

- [156] Tom Tirer and Raja Giryes. Image restoration by iterative denoising and backward projections. *IEEE Transactions on Image Processing (TIP)*, 28(3):1220–1234, 2018.
- [157] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *IEEE Sixth international conference on computer vision (ICCV)*, volume 98, pages 839–846, 1998.
- [158] Tong Tong, Gen Li, Xiejie Liu, and Qinquan Gao. Image super-resolution using dense skip connections. In *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, pages 4799–4807, 2017.
- [159] RY Tsai. Multiframe image restoration and registration. *Advances in Computer Vision and Image Processing.*, 1(2):317–339, 1984.
- [160] Matej Ulicny, V Krylov, and Rozenn Dahyot. Harmonic networks for image classification. In *British Machine Vision Conference (BMVC)*, 2019.
- [161] Matej Ulicny, Vladimir A Krylov, and Rozenn Dahyot. Harmonic networks with limited training samples. In *IEEE 27th European Signal Processing Conference (EU-SIPCO)*, pages 1–5, 2019.
- [162] Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.
- [163] Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. Plug-and-play priors for model based reconstruction. In *IEEE Global Conference on Signal and Information Processing*, pages 945–948, 2013.
- [164] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning (ICML)*, pages 1096–1103, 2008.
- [165] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3156–3164, 2015.

- [166] Ci Wang, Jun Zhou, and Shu Liu. Adaptive non-local means filter for image deblurring. *Signal Processing: Image Communication*, 28(5):522–530, 2013.
- [167] Zhaowen Wang, Ding Liu, Jianchao Yang, Wei Han, and Thomas Huang. Deep networks for image super-resolution with sparse prior. In *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, pages 370–378, 2015.
- [168] Zhou Wang and Alan C Bovik. Mean squared error: Love it or leave it? A new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.
- [169] 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 (TIP)*, 13(4):600–612, 2004.
- [170] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.
- [171] Yue Wu, Brian Tracey, Premkumar Natarajan, and Joseph P Noonan. James–stein type center pixel weights for non-local means image denoising. *IEEE Signal Processing Letters*, 20(4):411–414, 2013.
- [172] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems (NIPS)*, pages 341–349, 2012.
- [173] Zhiwei Xiong, Xiaoyan Sun, and Feng Wu. Image hallucination with feature enhancement. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2074–2081, 2009.
- [174] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *Deep Learning Workshop, ICML 2015*, 2015.
- [175] Li Xu, Jimmy SJ Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1790–1798, 2014.

- [176] Hitoshi Yamauchi, Jörg Haber, and H-P Seidel. Image restoration using multiresolution texture synthesis and image inpainting. In *Proceedings of Computer Graphics International 2003*, pages 120–125. IEEE, 2003.
- [177] Chih-Yuan Yang, Jia-Bin Huang, and Ming-Hsuan Yang. Exploiting self-similarities for single frame super-resolution. In *Asian conference on computer vision*, pages 497–510. Springer, 2010.
- [178] Chih-Yuan Yang, Chao Ma, and Ming-Hsuan Yang. Single-image super-resolution: A benchmark. In *European Conference on Computer Vision (ECCV)*, pages 372–386. Springer, 2014.
- [179] Jianchao Yang, John Wright, Thomas Huang, and Yi Ma. Image super-resolution as sparse representation of raw image patches. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1–8, 2008.
- [180] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE transactions on image processing (TIP)*, 19(11):2861–2873, 2010.
- [181] Jianchao Yang, Zhaowen Wang, Zhe Lin, Scott Cohen, and Thomas Huang. Coupled dictionary training for image super-resolution. *IEEE transactions on image processing (TIP)*, 21(8):3467–3478, 2012.
- [182] Ruikang Yang, Lin Yin, Moncef Gabbouj, Jaakko Astola, and Yrjö Neuvo. Optimal weighted median filtering under structural constraints. *IEEE transactions on signal processing*, 43(3):591–604, 1995.
- [183] Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, Jing-Hao Xue, and Qingmin Liao. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12):3106–3121, 2019.
- [184] Yongyi Yang, Nikolas P Galatsanos, and Aggelos K Katsaggelos. Projection-based spatially adaptive reconstruction of block-transform compressed images. *IEEE Transactions on Image Processing (TIP)*, 4(7):896–908, 1995.

- [185] Changhoon Yim and Alan Conrad Bovik. Quality assessment of deblocked images. *IEEE Transactions on Image Processing (TIP)*, 20(1):88–98, 2011.
- [186] Ke Yu, Chao Dong, Chen Change Loy, and Xiaoou Tang. Deep convolution networks for compression artifacts reduction. *arXiv preprint arXiv:1608.02778*, 2016.
- [187] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision (ECCV)*, pages 818–833. Springer, 2014.
- [188] Matthew D Zeiler, M Ranzato, Rajat Monga, Min Mao, Kun Yang, Quoc Viet Le, Patrick Nguyen, Alan Senior, Vincent Vanhoucke, Jeffrey Dean, et al. On rectified linear units for speech processing. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3517–3521, 2013.
- [189] 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. Springer, 2010.
- [190] Haichao Zhang, Jianchao Yang, Yanning Zhang, and Thomas S Huang. Non-local kernel regression for image and video restoration. In *European Conference on Computer Vision (ECCV)*, pages 566–579. Springer, 2010.
- [191] 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 (TIP)*, 26(7):3142–3155, 2017.
- [192] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 3929–3938, 2017.
- [193] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing (TIP)*, 27(9):4608–4622, 2018.
- [194] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Learning a single convolutional super-resolution network for multiple degradations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 6, 2018.

- 
- [195] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2472–2481, 2018.
- [196] Qiang Zhou, Shifeng Chen, Jianzhuang Liu, and Xiaoou Tang. Edge-preserving single image super-resolution. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 1037–1040, 2011.

