Received August 13, 2021, accepted August 25, 2021, date of publication August 27, 2021, date of current version September 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3108396

One-to-One Mapping-Like Properties of DCN-Based Super-Resolution and Its Applicability to Real-World Images

CHULHEE LEE, J. YOON, J. KIM, AND S. PARK
School of Electrical and Electronic Engineering, Yonsei University, Seoul 03722, Republic of Korea

Corresponding author: Chulhee Lee (chulhee@yonsei.ac.kr)

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology under Grant NRF-2020R1A2C1012221.

ABSTRACT Although super-resolution techniques based on deep neural networks (SRDNN) have drawn significant interest and numerous algorithms have been proposed, they still have reliability problems and produce artefacts when applied to new datasets. In this paper, the working mechanisms of SRDNN techniques are analyzed in terms of data mapping. Since most SRDNN techniques can be viewed as dynamic linear projections, we analyzed a large number of projection vectors (over 70 million) and found that the SRDNN method performs one-to-one mapping-like operations and may be vulnerable to unknown data patterns. Then, we applied several SRDNN techniques to real-world images and analyzed the output images. The current SRDNN methods failed to distinguish the blurred edges/lines due to low resolutions from coding artefacts and enhanced both, even though the SRDNN methods were trained using compressed low-resolution (LR) images. These analyses and results indicate that current SRDNN methods may not be able to provide robust performance and new structures may be necessary for reliable super-resolution performance.

INDEX TERMS DNN, super-resolution, artefacts, real world images, compression error enhancement.

I. INTRODUCTION

Recently, super-resolution based on deep neural networks (SRDNN) has drawn significant interest and numerous algorithms have been proposed [1]–[15]. Although it has been reported that these SRDNN algorithms produce much better performance compared to traditional interpolation methods such as bi-cubic interpolation, SRDNN methods still tend to produce unexpected artefacts in some cases and this reliability issue can restrict the use of SRDNN methods.

In typical SRDNN methods, a reduced image is enlarged by an integer factor (e.g., 2, 3 or 4). Recently, the perceptual quality of SRDNN methods has been studied [16]–[19]. These SRDNN methods aim to improve perceptual image quality instead of the conventional PSNR. Furthermore, applying SRDNN methods to real-world images has been investigated by assuming that paired HR (high-resolution) and LR images are unavailable [21]–[23]. Most existing SR (super-resolution) methods use degradation models that are not related to real images. Typically, bicubic down-sampling is used to generate low-resolution images to train the model. Using bicubic down-sampling is similar to applying a low-pass filter, which reduces high-frequency components in low-resolution images. Consequently, performance may be degraded when applied to real images. To address this problem, Ji *et al.* [25] proposed a degradation method using an estimation kernel and noise injection. Zhang *et al.* [26] proposed a degradation model consisting of randomly blended blur, down-sampling, and noise.

Super-resolution deals with an extremely ill-posed problem. Once the resolution of an image is reduced, some information can be permanently lost and never recovered. In 4x super-resolution, a pixel in the reduced image can be viewed as an average of 16 pixels (as a 4×4 block). For example, the images shown in Figure 1 will be identical when they are reduced by 1/4 assuming the average of 16 pixels is used.

When using 8-bit images, the 4×4 block shows a very large number of combinations, which are mapped into the same 8-bit value (0-255). Although this is an irreversible process,

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wei.

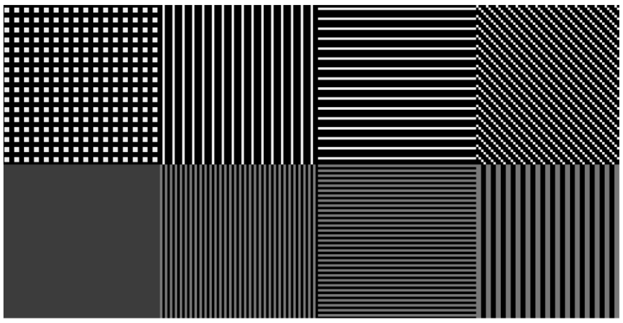

FIGURE 1. Various image patterns whose averages (4 × 4) are identical. The square of the top left consists of four pixels (2 by 2).

it is conjectured that SRDNN methods can utilize contextual information to recover the lost detail information. However, since most SRDNN methods produce unexpected artefacts, this conjecture should be examined and tested.

In this paper, we examined the working mechanism of SRDNN methods and showed that SRDNN methods show one-to-one mapping-like properties as they can be viewed as a dynamic linear transformation. Then, we analyzed the images enlarged by some SRDNN techniques as applied to compressed real-world images.

II. ONE-TO-ONE MAPPING-LIKE PROPERTIES OF SRDNN

A. DYNAMIC LINEAR TRANSFORMATION

A basic building block for SRDNN is a convolution layer followed by a ReLU layer, though some SRDNN methods use residual blocks, sigmoid functions, channel attention layers, etc. [6]. The depth of neural networks determines the receptive field size. For example, the receptive field of the VDSR [2] is 41 × 41. For some deep SRDNN methods, the receptive field can be the entire image. In other words, a pixel value in the output layer can be theoretically affected by the entire image. However, most SRDNN methods are trained using a large number of patches (K × K images). The typical value of K is 41 to 48.

In [20], it was shown that a convolution layer followed by a ReLU layer can be expressed by matrix operations. The patch was expressed as a vector (N × 1) with $N = K^2$. If there are 64 filters in the convolution layer, the convolution layer followed by the ReLU layer was expressed as follows [20]:

$$X^{j+1}_{64N\times 1} = \mathrm{ReLU}(A^j_{64N\times N}X^j_{N\times 1} + b^j_{64N\times 1}) \tag{1}$$

where the superscript represents the layer index. $A^j_{64N\times N}$ is a filter matrix of the j-th layer and is a bias vector of the j-th layer:

$$A^j_{64N\times N} = \begin{bmatrix} A^{j,0}_{N\times N} \\ A^{j,1}_{N\times N} \\ \vdots \\ A^{j,63}_{N\times N} \end{bmatrix}, \quad b^j_{64N\times 1}$$
$$= [b^j_0, b^j_0, \ldots, b^j_0, \ldots, b^j_{63}, b^j_{63}, \ldots, b^j_{63}] \tag{2}$$

As shown in [20], the output ($X^{j+1}_{64N\times 1}$) can be also expressed as a vector (64N × 1). The ReLU operator replaces negative elements with zeros, which is equivalent to setting the corresponding row of $A^j_{64N\times N}$ and the corresponding element of $b^j_{64N\times 1}$ to zero. This operation makes SRDNN with ReLU a non-linear function. Thus, (1) can be rewritten as follows:

$$X^{j+1}_{64N\times 1} = {}_RA^j_{64N\times N}X^j_{N\times 1} + {}_Rb^j_{64N\times 1} \tag{3}$$

In (3), ${}_RA^j_{64N\times N}$ is a matrix that reflects the ReLU operations and ${}_Rb^j_{64N\times 1}$ a vector that reflects the ReLU operations. In other words, a convolution layer followed by a ReLU layer can be modelled as a dynamic linear transformation.

After the DNN is trained, all the filter and bias coefficients are fixed. Without the ReLU operations, the transformation matrices would be identical for all input images. However, the ReLU operator produces a different transformation matrix depending on the images. In particular, the sign of the elements of the vector ($A^j_{64N\times N}X^j_{N\times 1} + b^j_{64N\times 1}$) determines the transformation matrix (${}_RA^j_{64N\times N}$) and the bias vector (${}_Rb^j_{64N\times 1}$). If all the element signs are identical, the transformation matrix and the bias vector will be the same.

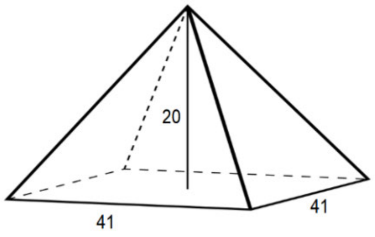

FIGURE 2. The receptive field of VDSR.

In VDSR [2], there are 20 layers and the filter size is 3 × 3. Thus the receptive field is 41 × 41. In [20], it was shown that an output pixel can be expressed as follows:

$$\begin{aligned} y &= W_{1\times N}X^0_{N\times 1} + \sum_{k=0}^{L-1}\sum_{j=0}^{n^k_{bias}} \alpha^k_j b^k_j \\ &= W_{1\times N}X^0_{N\times 1} + W_{1\times N_b}B_{N_b\times 1} \end{aligned} \tag{4}$$

where $X^0_{N\times 1}$ represents an input patch (*21 × 21* image), $W_{1\times N}$ and $W_{1\times N_b}$ represents weight vectors. Theoretically, 1681 pixels of the input image affects the output pixel (Figure 2).

On the other hand, the first layer (convolution and ReLU) will produce 64 images and 97344 (39 × 39×64) pixels of the 64 images of the first layer affect the output pixel. Before the ReLU operator, some of the pixel values may be negative and the ReLU operator will set the values to zero, which in turn will produce a different transformation matrix (${}_RA^j_{64N\times N}$) and a bias vector (${}_Rb^j_{64N\times 1}$). Therefore, a total of 682177 pixels of layer output images can affect an output pixel in the VDSR method (Table 1). In particular, the signs of the 682177 pixels

TABLE 1. Output images sizes and numbers of pixels of the 20 layers of VDSR.

layer	width	no. pixels	no. filters	total pixels
1	39	1521	64	97344
2	37	1369	64	87616
3	35	1225	64	78400
4	33	1089	64	69696
5	31	961	64	61504
6	29	841	64	53824
7	27	729	64	46656
8	25	625	64	40000
9	23	529	64	33856
10	21	441	64	28224
11	19	361	64	23104
12	17	289	64	18496
13	15	225	64	14400
14	13	169	64	10816
15	11	121	64	7744
16	9	81	64	5184
17	7	49	64	3136
18	5	25	64	1600
19	3	9	64	576
20	1	1	1	1
			total	682177

determine the final linear transformation matrix. In other words, the output pixel can be modelled as follows:

$$y = W_{1\times N}(P)X^0_{N\times 1} + b_{1\times 1}(P) \quad (5)$$

where P is the set of the 682177 pixels included in the pyramid, as illustrated in Figure 2. In this paper, P is defined as a *pyramid pixel set*, which is a set of pixels of the output images of the layers. The output images of each layer are 64 in VDSR except for the last layer whereas the input is a single-channel image. If the signs of the 682177 pixels before the ReLU operation are identical, the linear transformation ($W_{1\times N}(P)$ and $b_{1\times 1}(P)$) will be identical. This property can be applied to any SRDNN method that uses the ReLU function.

Using this paradigm, most SRDNN methods can be modelled as a dynamic linear transformation. The input patch (K x K) and the filters determine the pyramid pixel set (P), which in turn determines $W_{1\times N}(P)$ and $b_{1\times 1}(P)$. Although $W_{1\times N}(P)$ is a function P, P is also a function of the input patch ($X^0_{N\times 1}$). Thus, $W_{1\times N}$ and $b_{1\times 1}$ can be viewed as functions of $X^0_{N\times 1}$:

$$y = W_{1\times N}(X^0_{N\times 1})X^0_{N\times 1} + b_{1\times 1}(X^0_{N\times 1}) \quad (6)$$

Figure 3 illustrates this mapping procedure. Thus, the SRDNN method can be understood as first generating the weight vector and bias term ($W_{1\times N}$ & $b_{1\times 1}$) and then applying a linear transformation. In the case of VDSR, the network generates $W_{1\times N}$ and $b_{1\times 1}$ from $X^0_{N\times 1}$, and then uses (6) to compute the output pixel, though these operations are simultaneously performed in the VDSR network.

B. ONE-TO-ONE MAPPING-LIKE PROPERTIES

Recently, numerous SRDNN methods have been proposed and they have shown impressive performance. Figure 4 shows

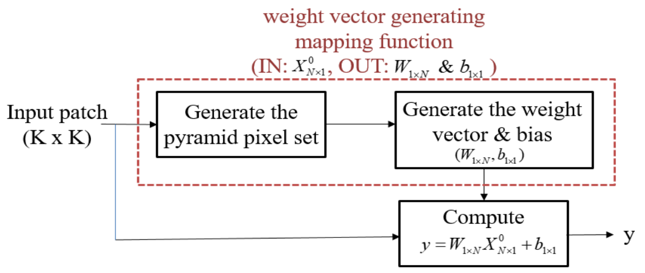
weight vector generating mapping function
(IN: $X^0_{N\times 1}$, OUT: $W_{1\times N}$ & $b_{1\times 1}$)
Input patch (K x K)
Generate the pyramid pixel set
Generate the weight vector & bias ($W_{1\times N}$, $b_{1\times 1}$)
Compute $y = W_{1\times N}X^0_{N\times 1} + b_{1\times 1}$
y

FIGURE 3. Weight vector generating mapping function.

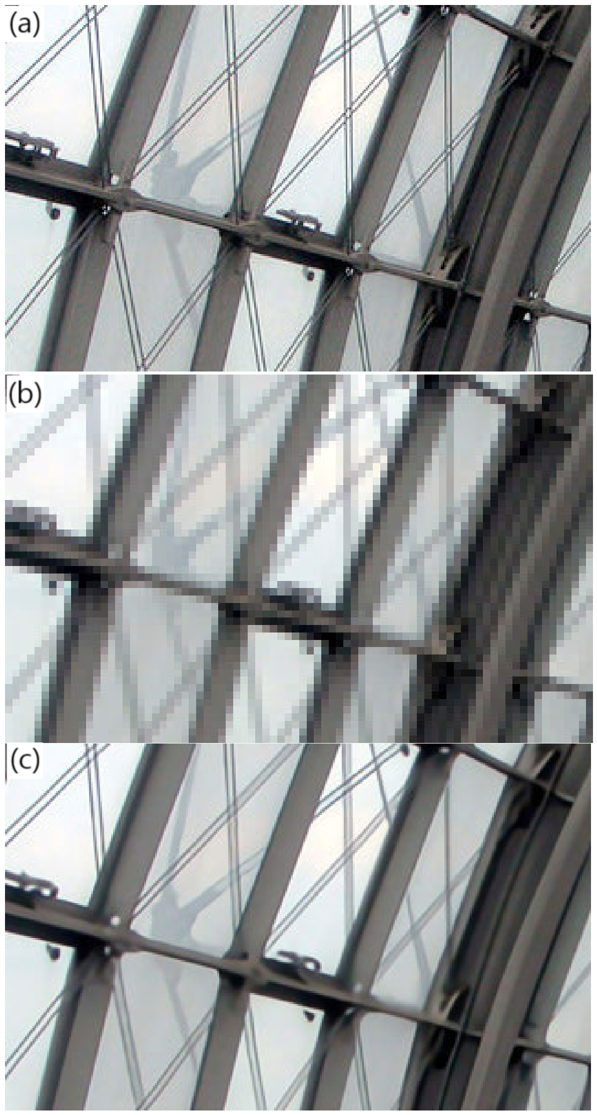

FIGURE 4. Examples of SRDNN methods. (a) HR (high resolution), (b) LR (low resolution), (c) SR enlarged by 4 using RRDB [12].

some examples of SRDNN methods (RRDB [12]). These methods produced impressive enlarged images (4x) from a reduced image. From what appears as blurred lines at the center of the LR image, the method successfully reconstructed the two fine line structures. Also, the SRDNN method impressively reconstructed the detailed structures of the beam on the right. Since SRDNN methods can be modelled as shown in Figure 3, one may claim that SRDNN methods effectively utilize surrounding structures to successfully restore lost detailed information. However, erratic behaviors for new input images also suggest that SRDNN methods may fail to use relevant information and they might suffer from reliability problems.

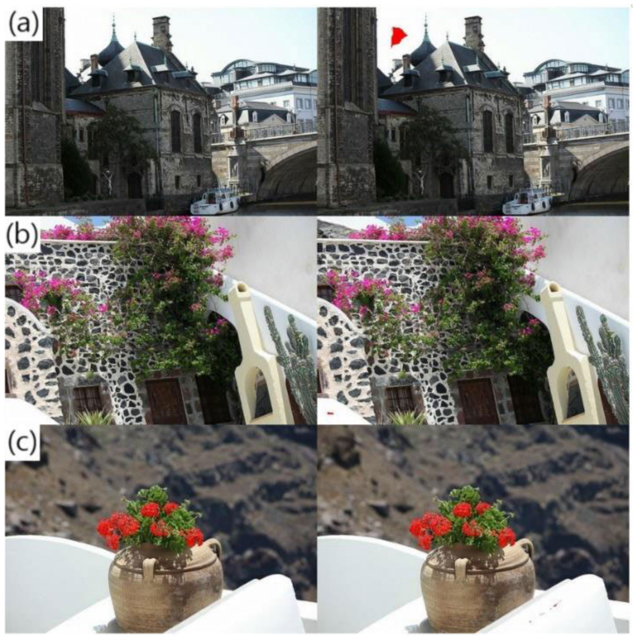

FIGURE 5. Output pixels with identical sign sequences (DSLR images). The left images are the original images and the red pixels of the right images represent the pixels with identical sign sequences. (a) red pixels in the upper-left, (b) red pixels in the lower-left, (c) red pixels in the lower-right.

In this paper, we examined the signs of the pyramid pixel set (P) for over 70 million output pixels when the VDSR was used. From each pyramid pixel set, we generated a sign sequence of 682177 numbers. If two output pixels have the same sign sequence, then the two output pixels will use the same linear transformation of (6). Another interpretation can be made from a point of input space division. The convolution filters will divide the input space (1681 dimensions) into a very large number of hyper-polygons. Then, all the pixels within the same hyper-polygon will use the identical linear transformation ($W_{1\times N}(P)$ and $b_{1\times 1}(P)$).

Figures 5-6 show some output pixels with identical sign sequences. The left image is an SR image and the right image shows where the output pixels with the same sign sequence are marked as red pixels. It can be seen that all those output pixels correspond to almost constant regions (either white or black). These results indicate that almost every input patch produced a different linear transformation of (6). Almost every hyper-polygon was occupied by a single pixel and the number of hyper-polygons may significantly exceed the number of training patches.

In other words, the *weight vector generating mapping function* shown in Figure 3 shows behaviors like a one-to-one mapping operator. Since this one-to-one mapping operator is designed from finite training samples, the mapping may not reflect valid logic using relevant information. More complex SRDNN methods (e.g., RCAN, RRDB, SAN, etc.), which have a larger number of parameters than VDSR, might suffer from the same problem since they may divide the input space into a much larger number of hyper-polygons.

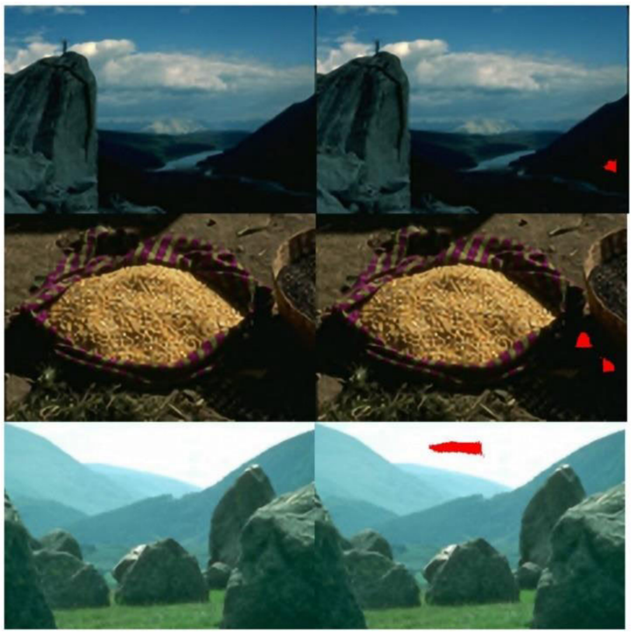

FIGURE 6. Output pixels with identical sign sequences for some public image databases (red pixels of the right images).

Consequently, SRDNN methods may have a fundamental reliability problem due to this one-to-one mapping property. In other words, when SRDNN methods are applied to real-world images that are not used for training, they may show erratic behaviors. SRDNN has been reported to produce clear and sharp high-resolution images from low-resolution images with blurred edges and lines. However, since most SRDNN methods produce one-to-one mapping solutions, they may not be able to distinguish blurred edges and lines from the compression artefacts that can be produced by coding. To investigate this vulnerability, we applied various SRDNN methods to some real-world images in the next section.

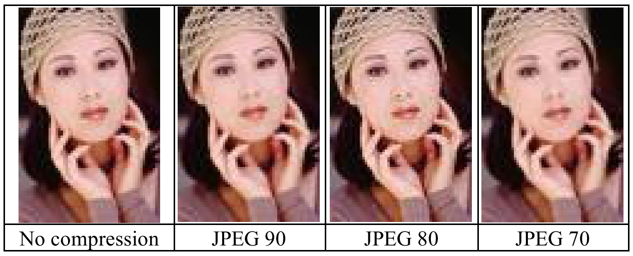

FIGURE 7. Compressed LR images (JPEG at various quality levels).

III. APPLICATIONS TO REAL-WORLD IMAGES

A. APPLICATION TO COMPRESSED IMAGES

In general, SRDNN performance is evaluated using standard databases. Usually, low resolution (LR) images are generated

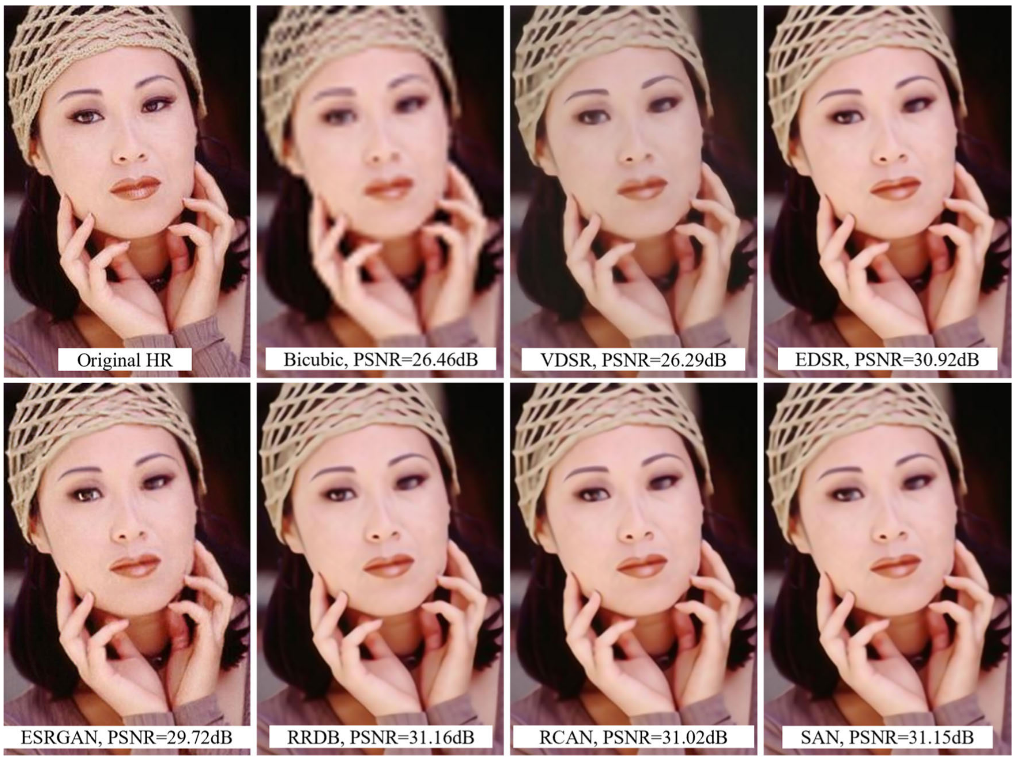

FIGURE 8. SR images produced by various SRDNN methods when the LR image was not compressed.

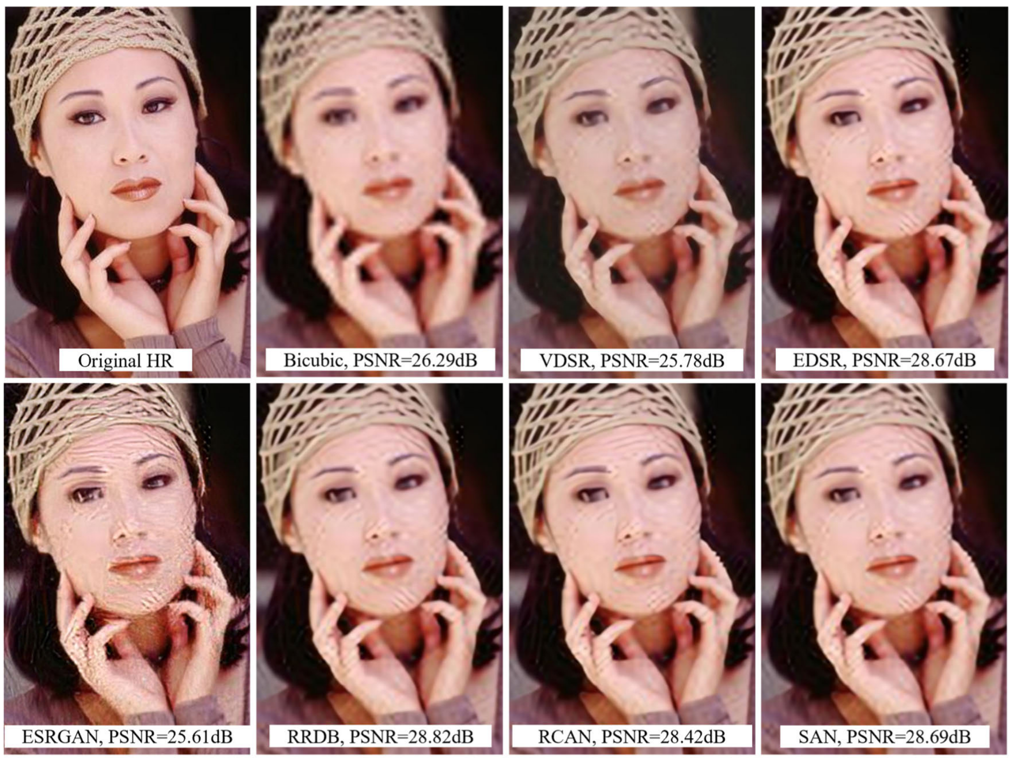

FIGURE 9. SR images produced by various SRDNN methods when the LR image was compressed (JPEG 90). The SRDNN methods were trained using uncompressed LR images.

by reducing high resolution (HR) images with bi-cubic interpolations. This operation acts like low pass filtering and the resulting LR images are usually free of artefacts and noise. However, in real-world applications, the input images may contain noise and almost all input images have compression artefacts.

TABLE 2. Performance comparison when the SRDNN methods trained using uncompressed LR images were applied to uncompressed LR test images.

Dataset	BICUBIC	VDSR	EDSR	ESRGAN	RRDB	RCAN	SAN	HAN
BDS100	25.96	27.29	27.81	25.31	27.85	27.75	27.84	27.80
BDS200	26.41	27.84	28.36	25.89	28.41	28.27	28.39	28.34
manga109	24.89	28.80	31.45	28.48	31.64	31.20	31.56	31.44
Set5	28.42	31.44	32.64	30.46	32.73	32.64	32.72	32.61
Set14	26.08	28.15	28.95	26.28	29.00	28.85	29.00	28.90
urban100	23.14	25.23	26.87	24.35	27.03	26.75	27.02	26.85
Average	25.82	28.13	29.35	26.79	29.44	29.24	29.42	29.32

TABLE 3. Performance comparison when the SRDNN methods trained using uncompressed LR images were applied to compressed LR test images (JPEG 90).

Dataset	BICUBIC	VDSR	EDSR	ESRGAN	RRDB	RCAN	SAN	HAN
BDS100	25.79	26.56	26.60	24.00	26.62	26.51	26.64	26.61
BDS200	26.24	27.04	27.03	24.32	27.08	26.93	27.09	27.06
manga109	24.77	27.57	28.36	24.29	28.31	27.96	28.41	28.27
Set5	28.12	29.79	30.04	27.15	30.06	29.90	30.00	30.00
Set14	25.91	27.20	27.38	24.00	27.41	27.22	27.41	27.36
urban100	23.06	24.52	25.01	21.92	25.08	24.82	25.10	25.12
Average	25.65	27.11	27.40	24.28	27.43	27.22	27.44	27.40

TABLE 4. Performance comparison when the SRDNN methods trained using uncompressed LR images were applied to compressed LR test images (JPEG 80).

Dataset	BICUBIC	VDSR	EDSR	ESRGAN	RRDB	RCAN	SAN	HAN
BDS100	25.52	25.87	25.78	23.85	25.79	25.65	25.81	25.76
BDS200	25.96	26.33	26.18	24.17	26.21	26.03	26.22	26.16
manga109	24.50	26.23	25.88	23.16	25.73	25.40	25.87	25.59
Set5	27.65	28.45	28.26	25.87	28.22	28.06	28.24	28.19
Set14	25.60	26.25	26.15	23.83	26.09	25.91	26.13	26.08
urban100	22.89	23.77	23.87	21.00	23.88	23.58	23.94	23.90
Average	25.35	26.15	26.02	23.65	25.99	25.77	26.03	25.95

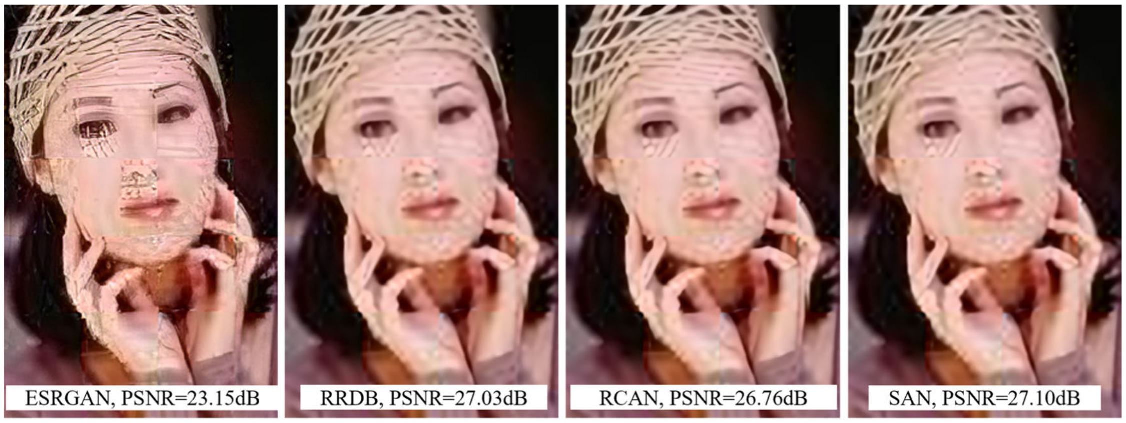

FIGURE 10. SR images produced by various SRDNN methods when the LR image was compressed (JPEG 80). The SRDNN methods were trained using uncompressed LR images.

In the next experiment, we applied several SRDNN methods (VDSR [2], EDSR [5], RRDB [12], ESRGAN [12], RCAN [6], SAN [7], CAR [27], HAN [28]) to the compressed LR images. The pre-trained models were downloaded from the authors' sites [29]. After the HR images were reduced to LR images, we compressed them using JPEG coding at various quality levels. When creating the compressed images, we used a JPEG function (built-in function

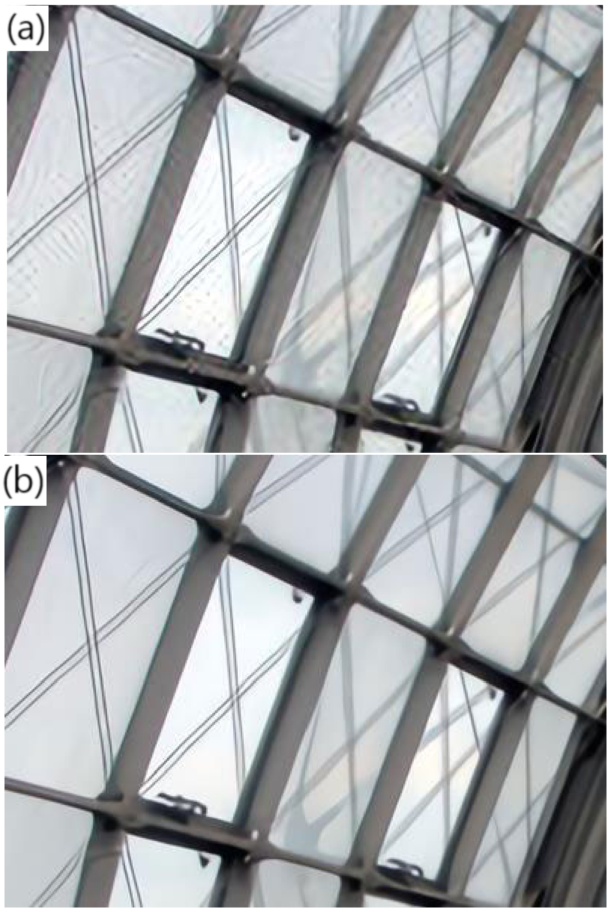
FIGURE 11. **EDSR output images when applied to a compressed image (JPEG 90). (a) EDSR trained using uncompressed LR images enhanced coding artefacts, (b) EDSR trained using both uncompressed and compressed LR produced an improved output image without enhancing coding artefacts.**

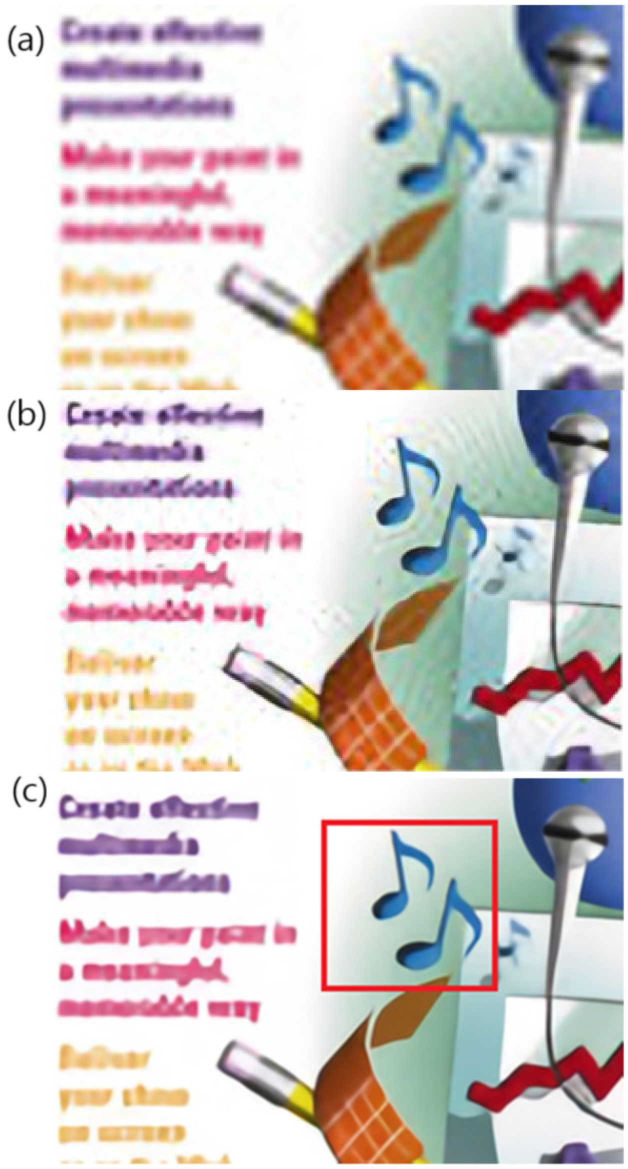
FIGURE 12. **Enlarged images of a compressed LR image. (a) bi-cubic, (b) EDSR trained using uncompressed LR images enhanced coding artefacts, (c) EDSR trained using both uncompressed and compressed LR images produced an overly smooth image.**

in Ubuntu, ver. 18.04). Figure 7 shows the compressed LR images at various quality levels. Then, we used the SRDNN methods to enlarge the compressed LR images to the original high resolution.

Tables 2-4 show the PSNR performance of the SRDNN methods along with the bi-cubic interpolation. The PSNR of the SRDNN methods considerably decreased for the compressed LR images.

Figure 8 shows the enlarged images when uncompressed LR images were used. Compared to the bi-cubic method, the SRDNN methods produced much better quality. Figure 9 shows the enlarged images when the LR image was compressed (JPEG 90). Although the bi-cubic method shows a similar output, the SRDNN methods showed more artefacts. In particular, ESRGAN, which aimed to maximize perceptual image quality, showed severe artefacts. It can be seen that the SRDNN methods immediately produced artefacts when the LR images were compressed. The VDSR generated the least number of artefacts for the compressed LR images, though its enlarged images were not as good as the outputs of the other SRDNN methods for the uncompressed LR images.

Figure 9 shows that the SRDNN methods produced artefacts similar to some 2D cosine transforms. For example, RCAN and SAN showed diagonal artefact patterns on the chin. Along strong horizontal edges such as eyebrows, a horizontal artefact pattern appeared. Figure 10 shows the enlarged images when the LR image was compressed using JPEG (quality level: 80). The artefact patterns appeared similar to 2D cosine transforms of lower frequency. This issue will be discussed in detail later.

Although the SRDNN methods successfully restored fine details from the blurred LR images in Figure 4, they also enhanced the coding artefacts of the compressed LR images and produced poor perceptual image quality (Figure 11(a)).

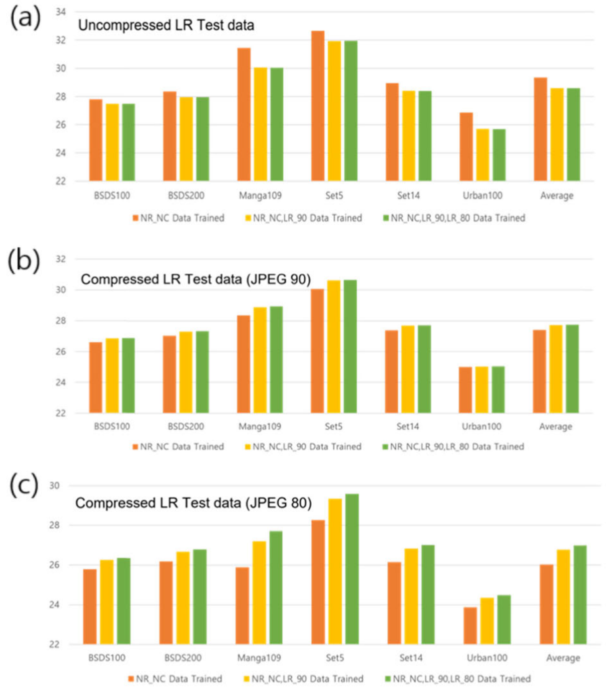

FIGURE 13. Performance comparison of the EDSR trained with various datasets when applied to compressed LR images. LR_NC: trained with uncompressed LR; LR_NC, LR_90: trained with LR_NC and compressed LR (JPEG 90); LR_NC, LR_90, LR_80: trained with LR_NC and compressed LR (JPEG 90 & 80).

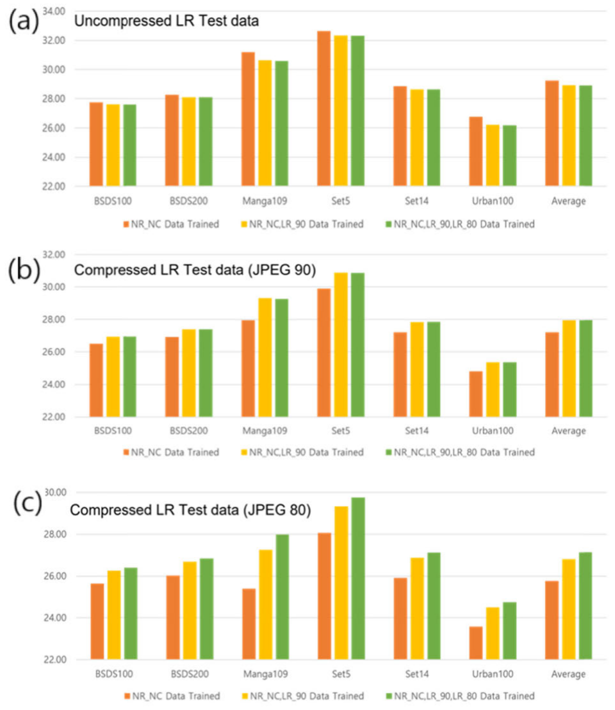

FIGURE 14. Performance comparison of RCAN trained with various datasets when applied to compressed LR images. LR_NC: trained with uncompressed LR; LR_NC, LR_90: trained with LR_NC and compressed LR (JPEG 90); LR_NC, LR_90, LR_80: trained with LR_NC and compressed LR (JPEG 90 & 80).

These results indicate that the one-to-one mapping-like properties of SRDNN methods may not be able to use relevant information that reflects the true nature of the target images.

To solve this reliability problem of SRDNN methods, new structures may be developed.

B. TRAINING USING COMPRESSED LR IMAGES

In the next experiments, we trained the EDSR and RCAN methods using uncompressed and compressed LR images. In other words, in addition to uncompressed LR images (LR_NC: LR no compression), we also used compressed LR images coded by JPEG for training. We encoded the training set using JPEG (quality level: 90) and decoded the JPEG LR images to produce the compressed LR training data. We used the DIV2K dataset for training and validation. In general, the 800 DIV2K images have been used for training and the 100 DIV2K images have been used for validation. To train the EDSR model, 32 residual blocks and 256 features were used. To train the RCAN model, we used 10 residual groups, 20 residual blocks, and 64 features. The models were trained for 300 epochs. When the EDSR was trained using only the uncompressed LR images, it produced annoying artefacts when it was applied to the compressed LR images (Figure 11(a)).

However, the EDSR trained using both uncompressed and compressed LR images did not produce these artefacts for the

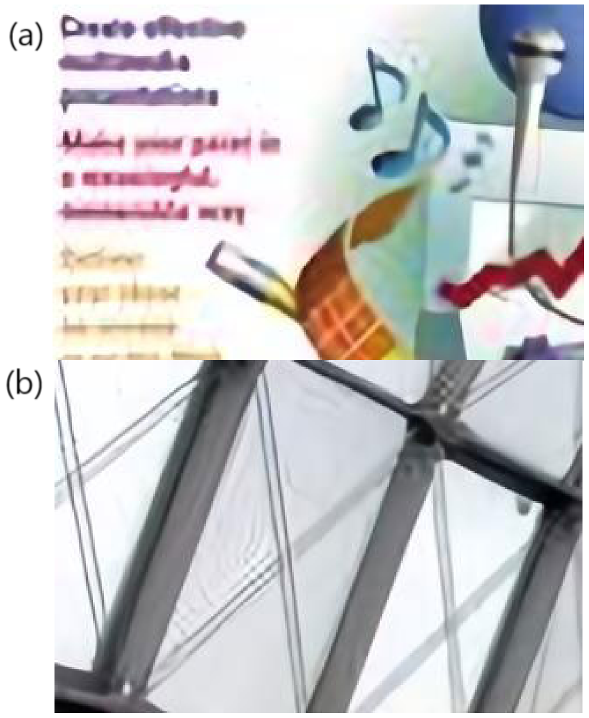

FIGURE 15. Enlarged images of a compressed LR image (JPEG80). (a) EDSR, (b) RCAN. EDSR and RCAN enhanced coding artefacts.

compressed LR images (Figure 11(b)). However, there were PSNR decreases (Figure 13). Similar performance patterns were observed for the RCAN (Figure 14).

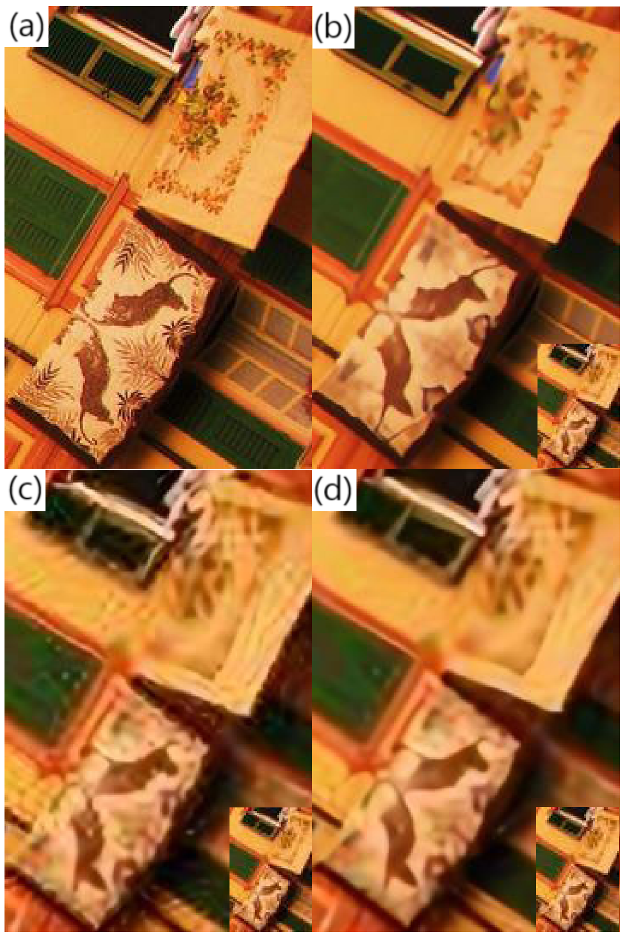

FIGURE 16. EDSR performance. (a) original HR, (b) SR (trained using NC) for uncompressed LR, (c) SR (trained using NC) for compressed LR (80) (d) SR (trained using NC, 80, 90) for compressed LR (80). NC: no compression, 80/90: JPEG 80/90. EDSR trained using uncompressed and compressed LR images produced an overly smooth image.

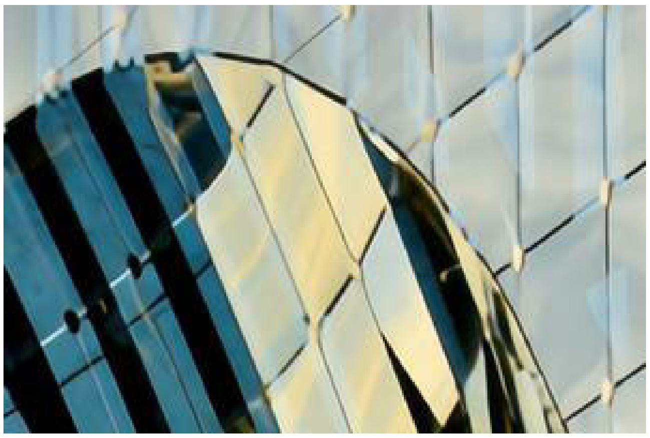

FIGURE 17. EDSR (train: uncompressed data LR images; test: uncompressed data LR images) produced a good output image.

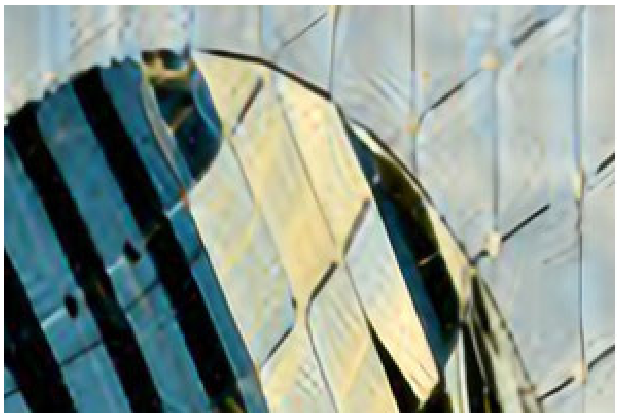

FIGURE 18. EDSR (train: uncompressed data LR; test: JPG 90) enhanced coding artefacts.

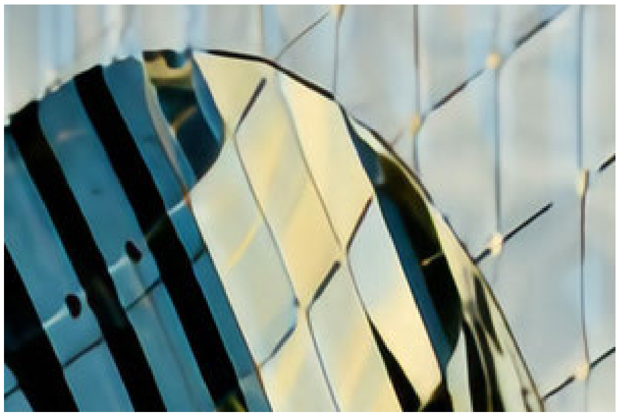

FIGURE 19. EDSR (train: uncompressed LR, JPG 90; test: JPG 90) produced an improved output image without enhancing coding artefacts.

Figure 12 shows enlarged images of a compressed LR image (JPEG 90). Although the bi-cubic interpolation method produced an image with some coding artefacts, it retained some naturalness (Figure 12(a)). The EDSR trained using only uncompressed LR images enhanced the coding artefacts (Figure 12(b)). Although the EDSR trained using both uncompressed and compressed LR images produced a clean image (Figure 12(c)), the output image was overly smooth and fine details were lost. For example, the gradation within the music notes was lost and the text lines look unnatural.

Next, we encoded the LR images with higher compression (JPEG 80). With larger coding impairments, the EDSR and RCAN enhanced the coding artefacts (Figure 15), though they were trained using uncompressed and compressed LR images (JPEG 90 & 80).

It appears that the SRDNN methods failed to distinguish the lost details from the coding impairments when those impairments were larger than a threshold. Figure 16 shows the SR images produced by the EDSR when the LR images were compressed. When the EDSR trained using only uncompressed LR images was applied to uncompressed LR images, it produced good outputs (Figure 16(b)). However, it produced severe artefacts when applied to compressed LR images (Figure 16(c)). Even when the EDSR was trained using uncompressed and

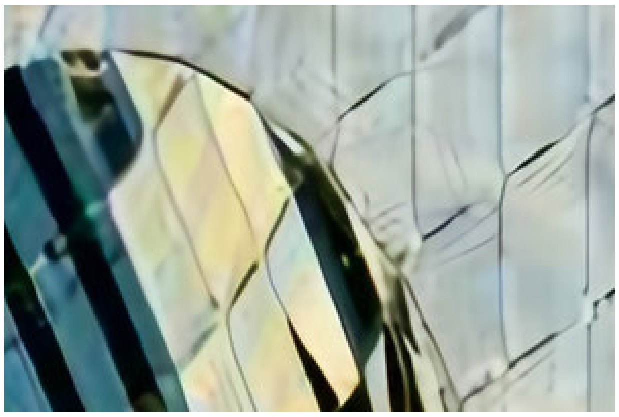

FIGURE 20. EDSR (train: uncompressed LR, JPG 90; test: JPG 80) failed to suppress coding artefacts.

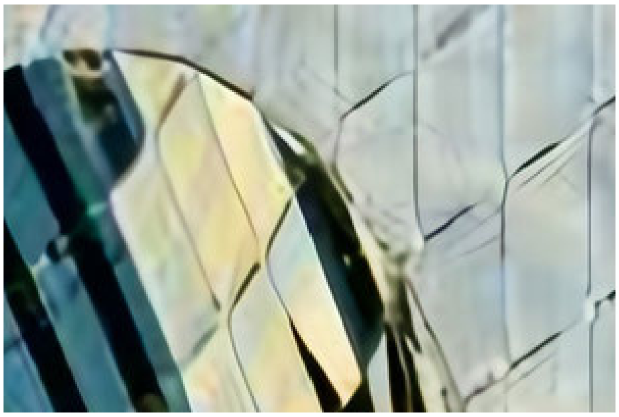

FIGURE 21. EDSR (train: uncompressed LR, JPG 90&80; test: JPG 80) still failed to suppress coding artefacts.

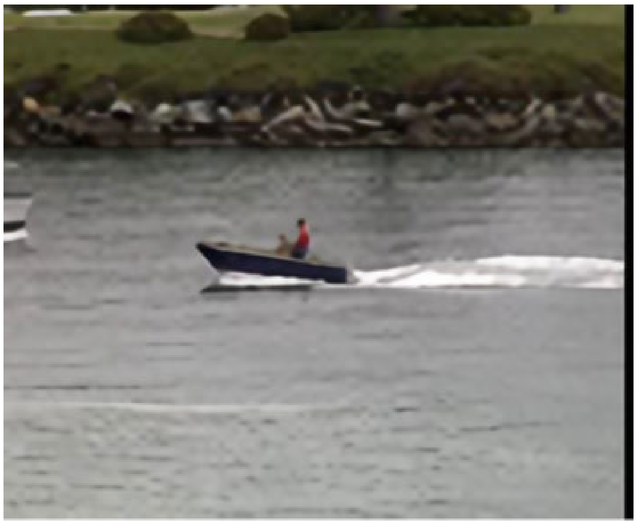

FIGURE 22. Output image when the EDSR trained using uncompressed LR images was applied to an uncompressed LR test image.

compressed LR images, it still produced some artefacts (Figure 16(d)). The corresponding LR images were shown at the lower-right.

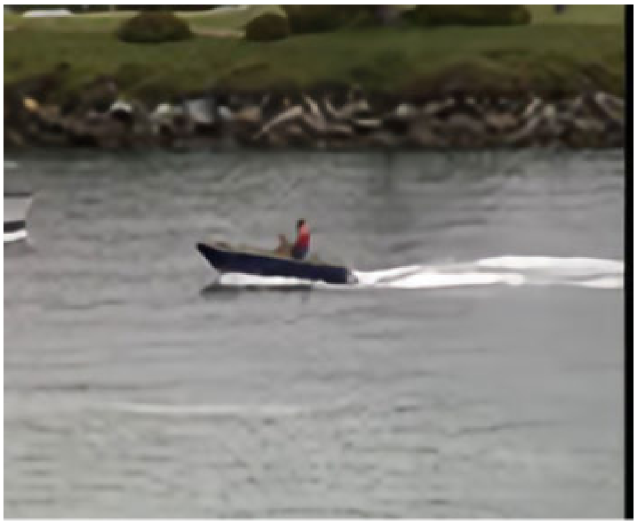

FIGURE 23. Output image when the EDSR trained using uncompressed LR images and compressed LR images (JPG 90&80) was applied to an uncompressed LR test image. It produced an overly smooth image.

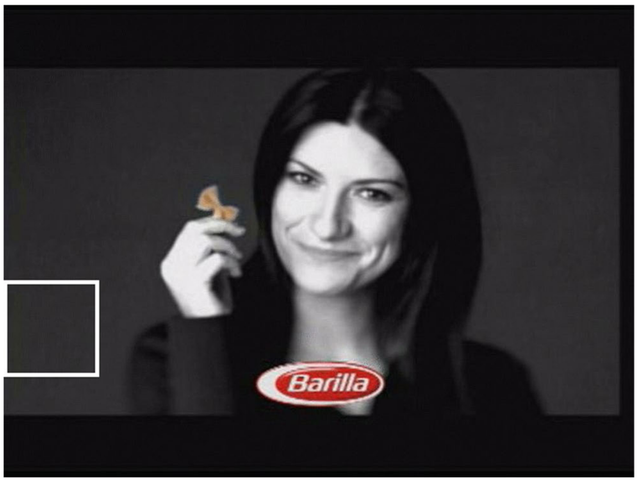

FIGURE 24. Original JPG image and the selected sub-image.

Furthermore, when the images are highly compressed, the SRDNN methods enhanced the coding artefacts even when the compressed training data of the same level was used. For example, when the EDSR trained using the uncompressed LR images (NC LR: no compression low-resolution images) was applied to compressed LR images (JPEG 90), it produced coding artefacts (Figure 18) whereas it produced a good image when applied to the uncompressed LR image (Figure 17). When trained using both uncompressed and compressed LR images, the EDSR reduced the coding artefacts (Figure 19). However, it still produced coding artefacts (Figures 20 & 21) when the image was more compressed (JPEG 80), though compressed images (JPEG 80) were also used for training. Also, when the EDSR was trained using both uncompressed and compressed LR images, it produced overly smooth images (Figures 22 & 23). Restoring the detailed information without amplifying the coding

FIGURE 25. Sub-image enlarged by the bi-cubic interpolation. The coding artefacts (DCT patterns) are visible.

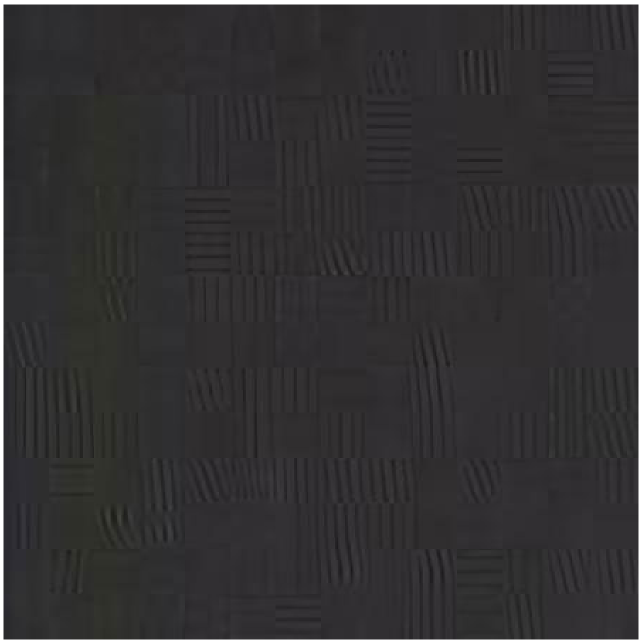

FIGURE 26. Sub-image enlarged by the EDSR. The enhanced coding artefacts (DCT patterns) are clearly visible.

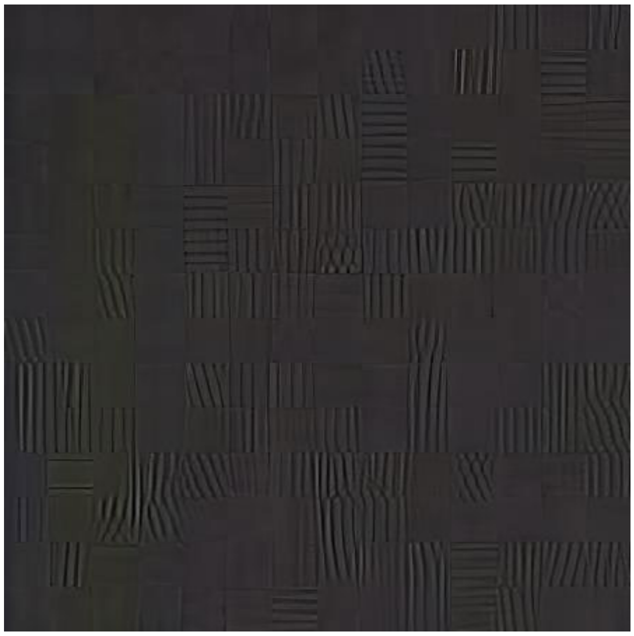

FIGURE 27. Sub-image enlarged by ESRGAN (perceptual). The enhanced coding artefacts (DCT patterns) are clearly visible.

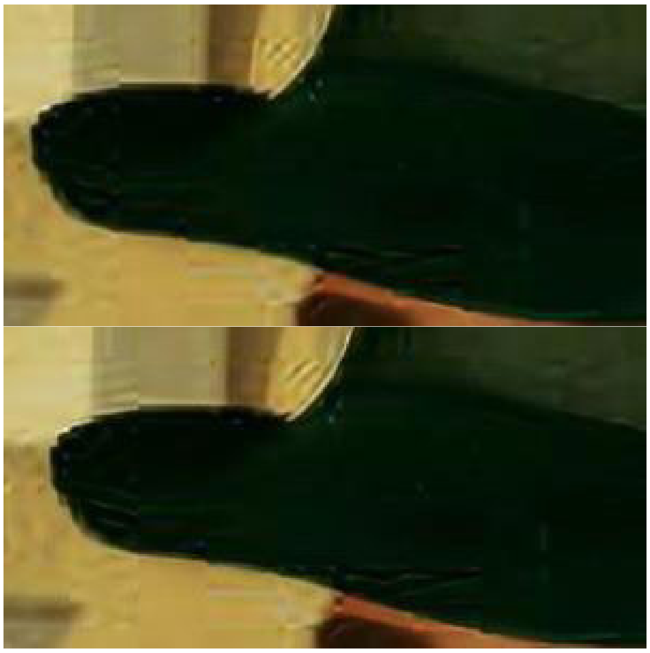

FIGURE 28. Sub-image enlarged by EDSR (top) and RCAN (bottom). The enhanced coding artefacts (DCT patterns) are clearly visible.

artefacts and noise is still an unsolved challenge for SRDNN methods.

C. ENHANCING CODING ARTEFACTS

In the next experiment, we applied the SRDNN methods (VDSR, EDSR, RRDB, ESRGAN, RCAN, and SAN) to the images of the MICC logo database [21]. Figure 24 shows an original JPG image, which was enlarged (4x) by using the SRDNN methods. Figure 25 shows a sub-image (enlarged box area of Figure 24) of the enlarged image produced by the bi-cubic interpolation. One can clearly see the DCT patterns.

Figure 26 shows a sub-image of the enlarged image produced by the EDSR, which noticeably enhanced the coding artefacts (DCT patterns). Figure 27 shows a sub-image

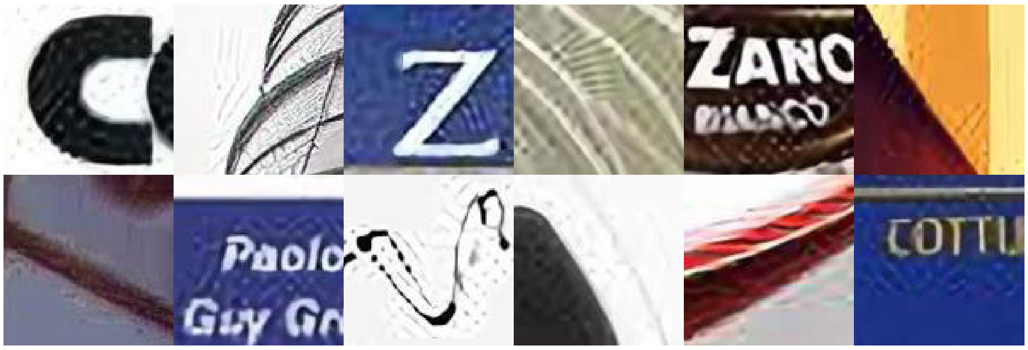

FIGURE 29. More examples of enhanced mosquito noise produced by the SRDNN methods.

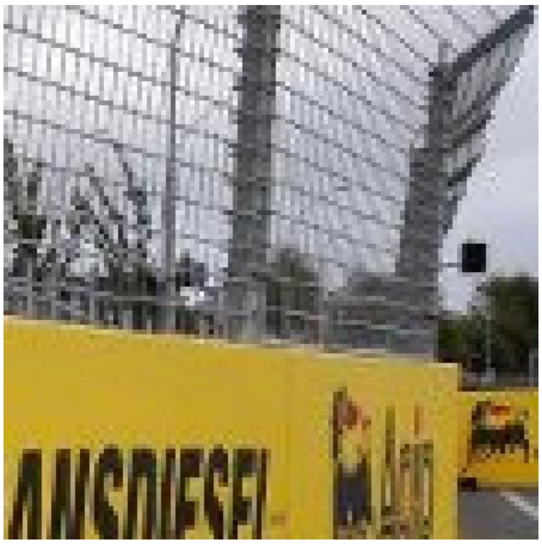
FIGURE 30. Sub-image enlarged by the nearest neighbor.

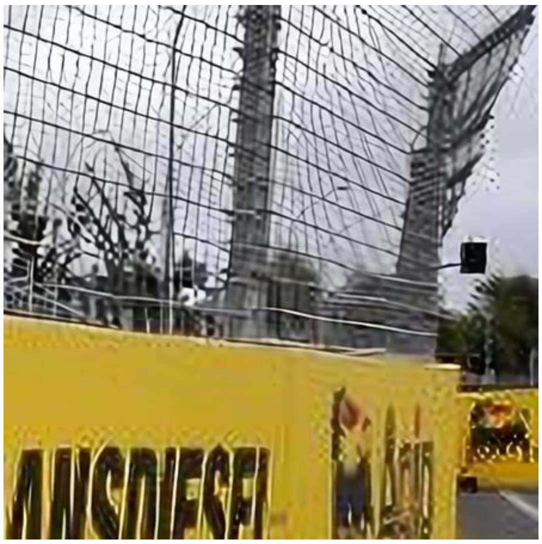
FIGURE 31. Sub-image enlarged by RCAN. The enhanced coding artefacts (e.g., mosquito noise) are clearly visible.

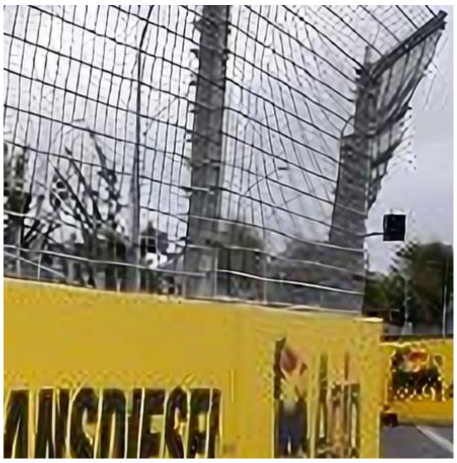
FIGURE 32. Sub-image enlarged by HAN [28]. The enhanced coding artefacts (e.g., mosquito noise) are clearly visible.

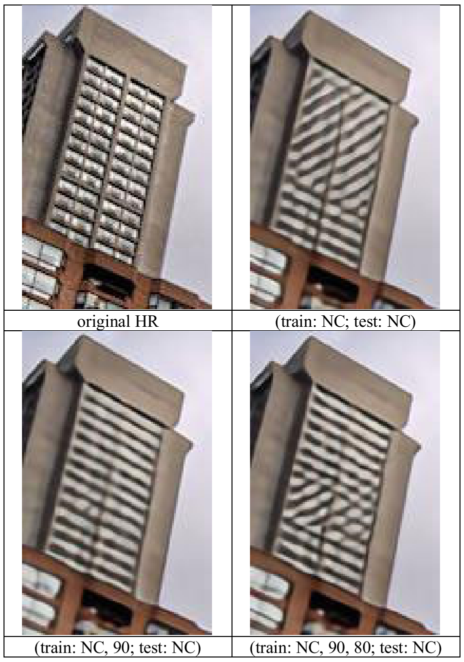

FIGURE 33. Various aliasing artefacts produced by EDSR that was trained different training datasets and applied to uncompressed LR images (NC: uncompressed LR, 90: compressed with JPEG 90, 80: compressed with JPEG 80).

enlarged by ESRGAN, which produced severe coding artefact enhancement.

Figures 28-32 show more examples of enhanced coding artefacts (e.g., mosquito noise due to encoding) produced by the SRDNN methods. These kinds of enhanced coding artefacts were observed in many of the MICC logo images and other JPEG images. In particular, the ESRGAN method,

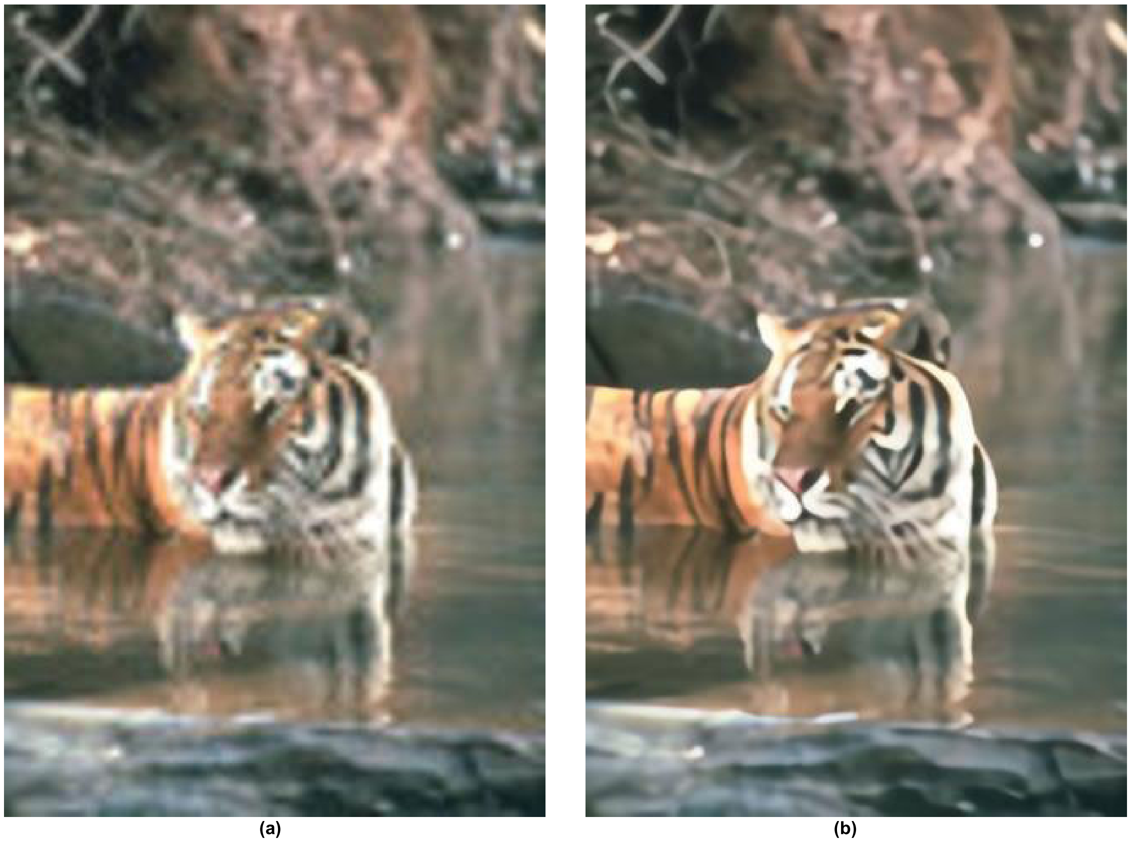

FIGURE 34. (a) Bicubic (test: JPG 90), (b) RCAN (train: NC, JPG 90; test: JPG90). RCAN trained with the uncompressed and compressed LR images produced an overly smooth output image that looks unnatural compared to the conventional bi-cubic method.

which is a perceptual model, produced the most severe artefacts.

D. ALIASING AND NATURALNESS

It is well known that aliasing can occur when images are reduced. Many SRDNN methods have produced such aliasing artefacts (Figure 33). Depending on the training data, SRDNN methods might produce different aliasing artefacts. In Figure 33, the EDSR trained with different training datasets produced different aliasing patterns.

In some cases, the SRDNN methods tend to produce overly smooth output images that look unnatural compared to conventional methods (Figure 34). Figures 37-42 show more of these examples. This unnaturalness was more easily perceived for human faces and characters.

Figures 35-36 show some examples of enlarged text. It appears that the bi-cubic method provided the best naturalness and readability whereas the SRDNN methods produced unrecognizable/disturbing characters along with enhanced coding artefacts. Figure 43 shows enlarged tree branches. The SRDNN methods (RCAN, ESRGAN, CAR, HAN) produced very unnatural tree branch patterns. Also, they created visible vertical stripes at the left side ((c)-(f)).

Super-resolution is an ill-posed problem. Although it is claimed SRDNN aims to restore lost detail information using

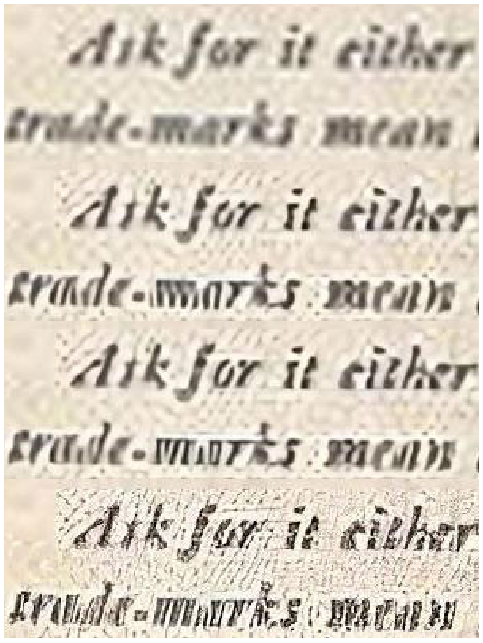

FIGURE 35. Examples of enlarged text: (from top) bi-cubic, EDSR, RCAN, ESRGAN. The SRDNN methods enhanced coding artefacts, thereby producing unrecognizable/disturbing characters.

surrounding structures, it is a challenging task and the current SRDNN methods may not able to successfully handle all types of impairments.

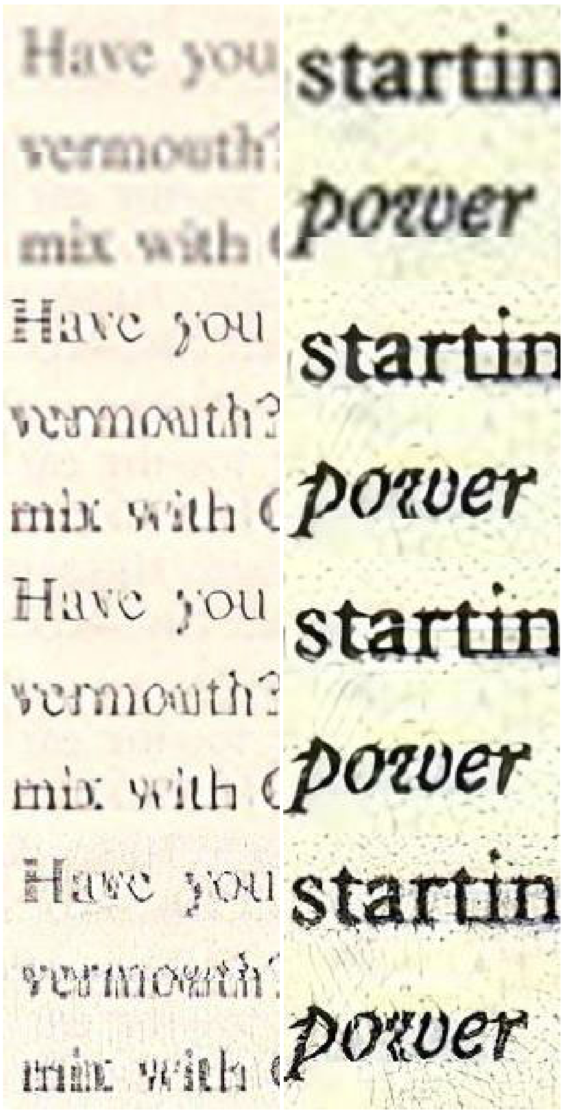

FIGURE 36. Examples of enlarged characters: (from top) bi-cubic, EDSR, RCAN, ESRGAN. The SRDNN methods enhanced coding artefacts (e.g., mosquito noise).

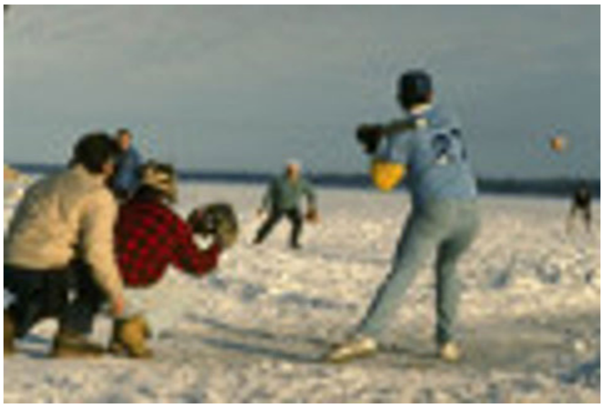

FIGURE 37. Output image produced by the bi-cubic method when applied to a compressed LR test image (JPG 90).

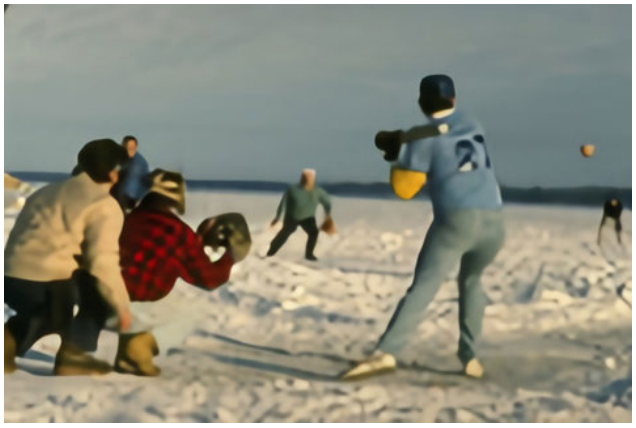

FIGURE 38. Overly smooth output image produced by RCAN trained with uncompressed and compressed (JPG90) LR images when applied to a compressed LR test image (JPG 90).

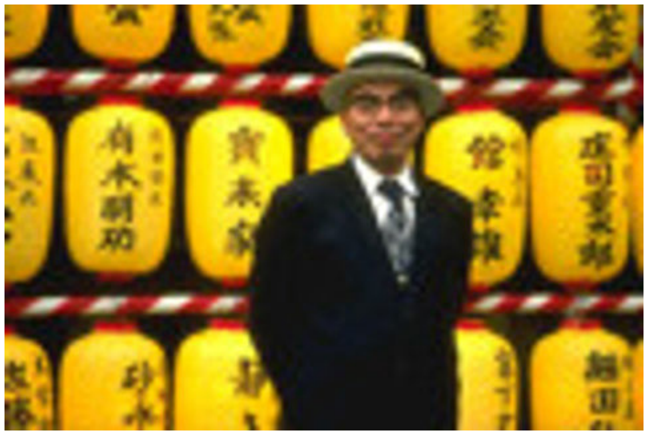

FIGURE 39. Output image produced by the bi-cubic method when applied to a compressed LR test image (JPG 90).

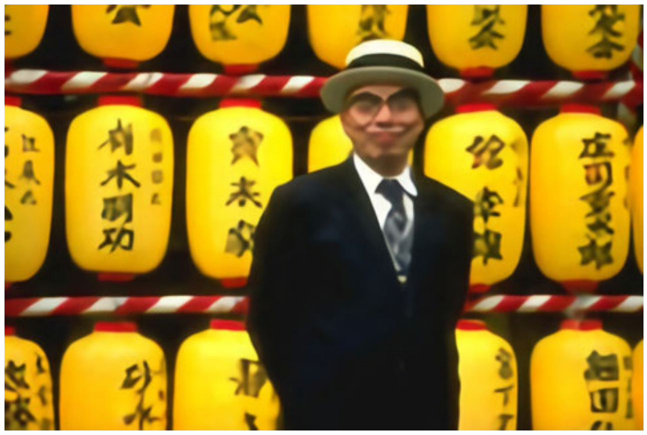

FIGURE 40. Overly smooth output image produced by RCAN trained with uncompressed and compressed (JPG90) LR images when applied to a compressed LR test image (JPG 90).

Table 5 summarizes some impairment types of real-world images and the goal/target of SRDNN along with the results/side-effects. SRDNN aims to reconstruct lost high-frequency information mainly by enhancing blurred edges/lines. However, blurred edge/line-like structures can also occur when images are compressed. The current SRDNN

TABLE 5. Impairment types in real-world images, goal/target of SRDNN and results/side effects.

impairment types	note	goals/targets	results/side effects
high frequency loss (blurred images)	blurred edges/lines, loss of details due to low resolution	sharp edges/lines	partially successfully (Fig. 4), but enhance other impairments/noise (Figs. 9-10)
coding artefacts (DCT patterns)	occur in highly compressed images	suppressed coding artefacts	enhanced DCT patterns (Figs. 24-28)
coding artefacts (mosquito noise, etc.)	occur in many compressed images	suppressed mosquito noise; sharp edges/lines without enhancing noise	enhanced mosquito noise (Figs. 12(b), Figs. 18-21, Fig. 29); failed to distinguish blurred edges/lines from noisy components (Figs. 31-32, Fig. 36)
permanent information loss	impossible to restore (Fig. 1)	restoration using neighborhood information	undesirable artefacts/aliasing (Fig. 31-32, Fig. 33, Figs. 45-46); produced unrecognizable/disturbing characters (Figs. 35-36)
blurred LR images and coding artefacts	solution: train the network using uncompressed and compressed images	sharp edges/lines without enhancing compressing impairments	partially successfully (Fig. 11(a), Fig. 12(c)), but failed for highly compressed images (Fig. 21) and produced overly smooth unnatural images (Figs. 16, 34, 38, 40, 42)

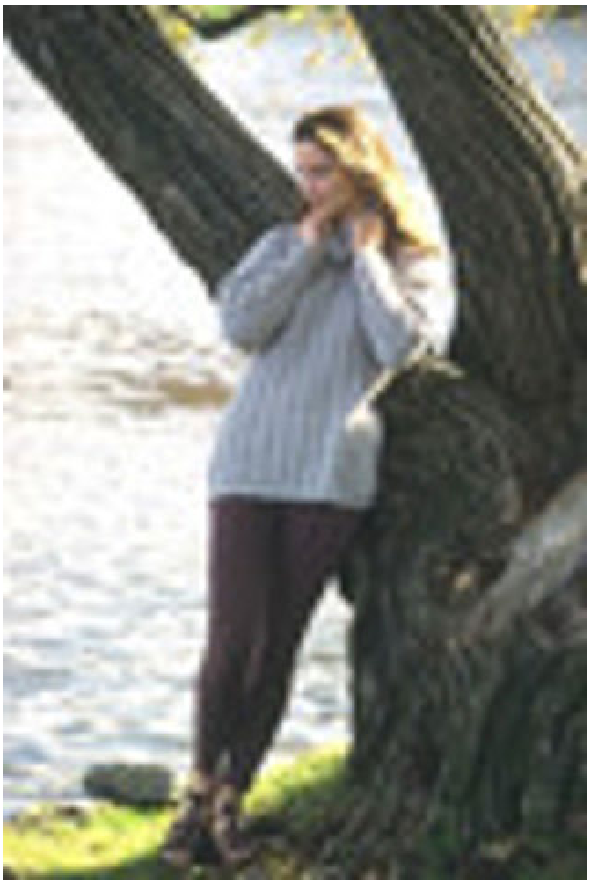

FIGURE 41. Output image produced by the bi-cubic method when applied to a compressed LR test image (JPG 90).

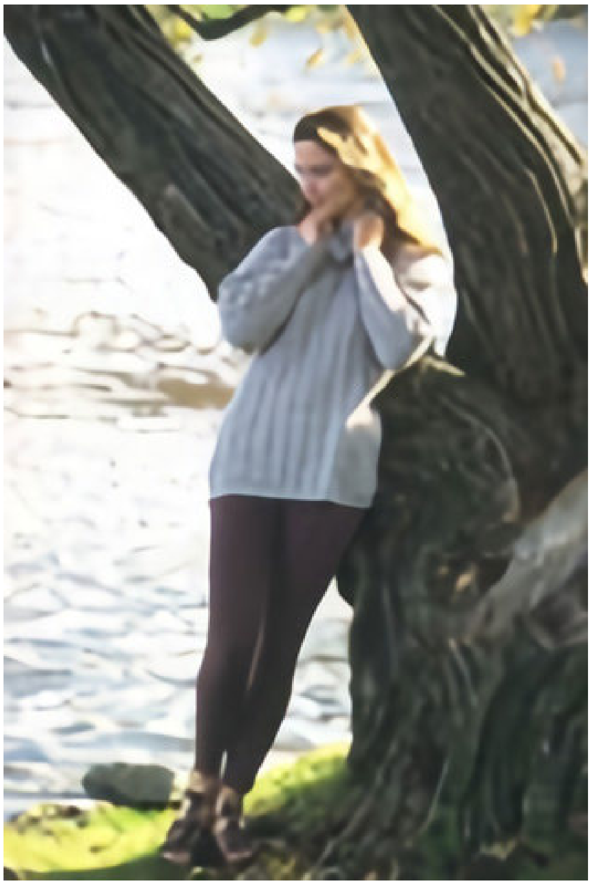

FIGURE 42. Overly smooth output image produced by RCAN trained with uncompressed and compressed (JPG90) LR images when applied to a compressed LR test image (JPG 90).

methods failed to distinguish the blurred edges/lines due to low resolutions from the blurred edge/line-like structures produced by encoding, even though the SRDNN methods were trained with compressed LR images. In some cases, when the SRDNN methods were trained with compressed LR images, they produced overly smooth unnatural images. It appears that the current SRDNN methods may not able to provide global solutions to all super-resolution

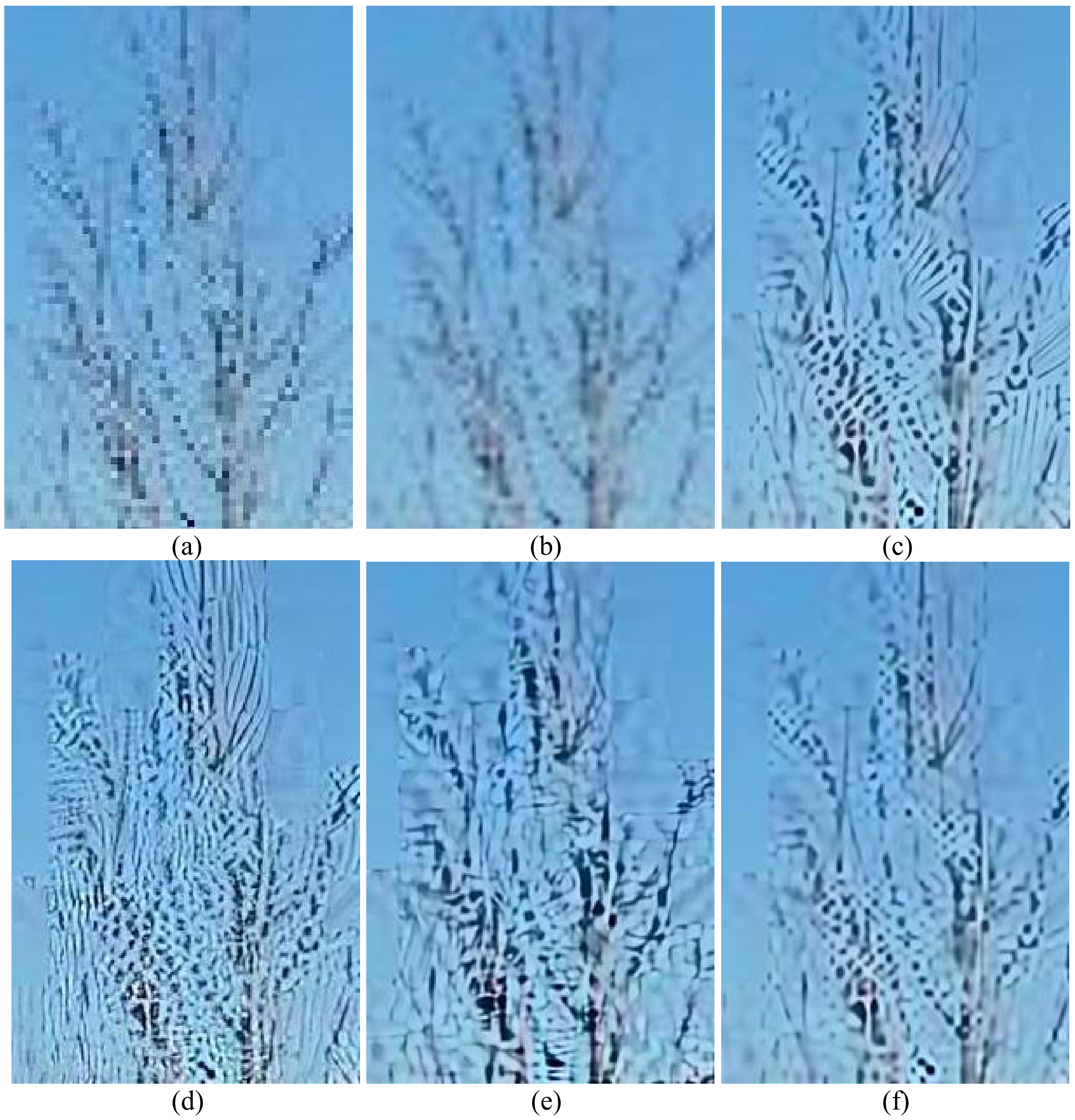

FIGURE 43. Enlarged tree branches, (a) nearest neighbor, (b) bilinear, (c) RCAN, (d) ESRGAN, (e) CAR [27], (f) HAN [28]. The tree branch patterns of the SRDNN methods are very unnatural. All of them created visible vertical stripes at the left side.

problems, though they may be a good solution for specific applications.

IV. CONCLUSION

In this paper, we analyzed the working mechanisms of SRDNN techniques and investigated the one-to-one mapping nature. After we modelled SRDNN methods in terms of weight vector generating and dynamic linear transformation, we analyzed a large number of projection vectors (over 70 million) and found that the SRDNN method showed one-to-one mapping-like properties. After further analyses of real-world images, it appears that current SRDNN techniques are vulnerable to unknown data patterns since the one-to-one mapping function is designed with a limited number of training samples, though the mapping space is enormous. Although it is desirable to restore the blurred edges/lines due to low resolutions without enhancing the blurred edge/line-like structures (e.g., coding artefacts, mosquito noise, etc.) produced by encoding, the current SRDNN methods failed to distinguish them and enhanced both, thereby generating undesirable artefacts. Using compressed LR images as training data failed to completely solve the problem and produced overly smooth unnatural images in some cases. Super-resolution is an ill-posed problem, though recent SRDNN methods have shown promising results. However, to provide robust performance for real-world compressed images, new structures may be necessary for successful SRDNN applications.

REFERENCES

[1] C. Dong, C. Change Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[2] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1646–1654.

[3] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4681–4690.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[5] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 136–144.

[6] Y. Zhang, "Image super-resolution using very deep residual channel attention network," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 286–301.

[7] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, "Second-order attention network for single image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11065–11074.

[8] M. S. M. Sajjadi, B. Scholkopf, and M. Hirsch, "EnhanceNet: Single image super-resolution through automated texture synthesis," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4491–4500.

[9] A. Dosovitskiy and T. Brox, "Generating images with perceptual similarity metrics based on deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 658–666.

[10] J. Johnson, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 694–711.

[11] I. Goodfellow, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1–9.

[12] N. C. Rakotonirina and A. Rasoanaivo, "ESRGAN+ : Further improving enhanced super-resolution generative adversarial network," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 3637–3641.

[13] A. Jolicoeur-Martineau, "The relativistic discriminator: A key element missing from standard GAN," 2018, *arXiv:1807.00734*. [Online]. Available: http://arxiv.org/abs/1807.00734

[14] T. Shang, Q. Dai, S. Zhu, T. Yang, and Y. Guo, "Perceptual extreme super resolution network with receptive field block," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 440–441.

[15] X. Wang, K. Yu, C. Dong, and C. Change Loy, "Recovering realistic texture in image super-resolution by deep spatial feature transform," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 606–616.

[16] M. S. Rad, B. Bozorgtabar, U.-V. Marti, M. Basler, H. K. Ekenel, and J.-P. Thiran, "SROBB: Targeted perceptual loss for single image super-resolution," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2710–2719.

[17] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[18] Y. Blau, R. Mechrez, R. Timofte, T. Michaeli, and L. Zelnik-Manor, "The 2018 PIRM challenge on perceptual image super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 334–355.

[19] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 586–595.

[20] C. Lee, J. Park, S. Woo, J. Kim, and J. Yoon, "Mathematical analysis of DCN-based super-resolution," *IEEE Access*, vol. 8, pp. 90420–90429, 2020.

[21] A. Lugmayr, M. Danelljan, and R. Timofte, "NTIRE 2020 challenge on real-world image super-resolution: Methods and results," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 494–495.

[22] A. Lugmayr, M. Danelljan, and R. Timofte, "Unsupervised learning for real-world super-resolution," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 3408–3416.

[23] M. Fritsche, S. Gu, and R. Timofte, "Frequency separation for real-world super-resolution," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 3599–3608.

[24] H. Sahbi, L. Ballan, G. Serra, and A. Del Bimbo, "Context-dependent logo matching and recognition," *IEEE Trans. Image Process.*, vol. 22, no. 3, pp. 1018–1031, Mar. 2013.

[25] X. Ji, Y. Cao, Y. Tai, C. Wang, J. Li, and F. Huang, "Real-world super-resolution via kernel estimation and noise injection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 466–467.

[26] K. Zhang, J. Liang, L. Van Gool, and R. Timofte, "Designing a practical degradation model for deep blind image super-resolution," 2021, *arXiv:2103.14006*. [Online]. Available: http://arxiv.org/abs/2103.14006

[27] W. Sun and Z. Chen, "Learned image downscaling for upscaling using content adaptive resampler," *IEEE Trans. Image Process.*, vol. 29, pp. 4027–4040, 2020.

[28] B. Niu, W. Wen, W. Ren, X. Zhang, L. Yang, S. Wang, K. Zhang, X. Cao, and H. Shen, "Single image super-resolution via a holistic attention network," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 191–207.

[29] *VDSR, EDSR*. [Online]. Available: https://github.com/sanghyun-son/EDSR-PyTorch *ESRGAN, RRDB*. [Online]. Available: https://github.com/xinntao/ESRGAN; *SAN*. [Online]. Available: https://github.com/daitao/SAN; *RCAN*. [Online]. Available: https://github.com/yulunzhang/RCAN; *CAR*. [Online]. Available: https://github.com/sunwj/CAR; *HAN*. [Online]. Available: https://github.com/wwlCape/HAN

CHULHEE LEE received the B.S. and M.S. degrees in electronic engineering from Seoul National University, in 1984 and 1986, respectively, and the Ph.D. degree in electrical engineering from Purdue University, in 1992. From 1993 to 1996, he worked with the National Institutes of Health, Bethesda, MD, USA. In 1996, he joined the Faculty of the Department of Electrical and Computer Engineering, Yonsei University, Seoul, South Korea. His research interests include image/signal processing, pattern recognition, and neural networks.

J. YOON received the B.S. degree in electrical and electronics engineering from Yonsei University, Seoul, South Korea, in 2016, where he is currently pursuing the Ph.D. degree. His research interests include DNN-based signal processing and pattern recognition.

J. KIM received the B.S. degree in electronics and electrical engineering from Dongguk University, Seoul, South Korea, in 2019, and the M.S. degree in electrical and electronics engineering from Yonsei University, Seoul, in 2021. His research interests include DNN-based super-resolution and dependable deep learning.

S. PARK received the B.S. degree in electrical and electronics engineering from Yonsei University, Seoul, South Korea, in 2021. His research interests include computer vision, pattern recognition, and deep learning.