Abstract—In this paper, we propose a simple but powerful idea to improve super-resolution (SR) methods based on convolutional neural networks (CNNs). We consider a linear manifold, which is the set of all SR images whose downsampling results are the same as the input image, and apply the orthogonal projection onto this linear manifold in the output layers of the CNNs. The proposed method can guarantee the consistency between the SR image and the input image and reduce the mean squared error. The proposed method is especially effective for SR methods based on generative adversarial networks (GANs), composed of one generator and one discriminator, since the generator can learn high-frequency components while maintaining low-frequency ones. Experiments show the effectiveness of the proposed technique for a GAN-based SR method. Finally we introduce an idea of extension to noisy images.

Index Terms—Single image super-resolution, generative adversarial network, orthogonal projection, constrained learning.

I. INTRODUCTION

Super-resolution (SR) is a reconstruction problem of high-resolution (HR) images including various high-frequency components from low-resolution (LR) images including only low-frequency components [1]–[13]. In SR, it is important not only to increase the number of pixels but also to recover the original high-frequency components. In this paper, we focus on single image SR, which is an under-determined inverse problem since we have to recover a HR image from a single LR image having a smaller number of pixels. The simplest ways to increase the number of pixels are algebraic interpolations, e.g., the nearest-neighbor, bilinear, and bicubic interpolations. Although these algebraic methods are very fast, they cannot recover the high-frequency components at all. Therefore, SR results, called SR images in this paper, of the algebraic methods are very blurred.

To accurately recover the high-frequency components, most SR methods learn the transform from LR images to HR images by using training data. SR methods based on dictionary learning [2], [3] were studied before, but recently SR methods based on convolutional neural networks (CNNs) [4]–[11] are mainly used in terms of both reconstruction accuracy and processing time. Dong et al. proposed the first end-to-end CNN for SR, named SRCNN [4]. SRCNN transforms interpolated LR images, which are enlarged to the HR image size by the bicubic interpolation, into SR images through three convolution layers. There exist many improved versions of SRCNN [5]–[11]. For example, VDSR [6] increased the number of the convolution layers by introducing the residual learning to avoid the gradient vanishing. ESPCN [7] proposed the sub-pixel convolution layer called pixel shuffler, which enlarges LR images at various magnification ratios and removes the bicubic interpolation utilized in the input layer of SRCNN. In each SR method [5]–[11], a single CNN, generator, is trained by minimizing mainly the mean squared error (MSE) or the mean absolute error (MAE). Therefore, Ledig et al. proposed SRGAN [12] that uses a generative adversarial network (GAN) [14]. SRGAN is composed of two CNNs, i.e., a generator and a discriminator. The discriminator is trained to judge whether the input image is a true HR image or a generated SR one. Since the generator tries to induce the discriminator’s misjudgment, SR images including realistic high-frequency components can be generated. However, the generated SR images have some artifacts which do not exist in true HR images, and values of PSNR decrease.

In this paper, we consider the set of all SR images whose downsampling results correspond to the input LR image. This set becomes a linear manifold, and the orthogonal projection onto this set can be easily computed as in [15] under a simple downsampling model in (1) (see Sect. II.A). We propose to use the orthogonal projection in the output layer of the generator. The proposed method can be applied to any generator and SR images having the perfect consistency with the input images are always generated. Therefore, the generator can learn high-frequency components while keeping true low-frequency ones. Numerical experiments show that the proposed method significantly reduces the artifacts of SR images generated by a GAN-based method while reconstructing high-frequency components at high accuracy. Finally, we conclude this paper and introduce an extension of the proposed method to noisy LR images.

II. IMAGE SUPER-RESOLUTION BY NEURAL NETWORKS

A. Formulation of Downsampling

Let \( Y := (Y_{i,j,c}) \in \mathbb{R}^{I \times J \times 3} \) be a low-resolution (LR) RGB color image to be enlarged, and \( Y_{i,j,c} \in \mathbb{R} \) be the \((i,j)\)th pixel value \((i = 1, 2, \ldots, I \text{ and } j = 1, 2, \ldots, J)\) of color channel \( c \) \((c = R, G, B)\). Suppose that \( Y \) is the downsampling result of a high-resolution (HR) color image \( X := (X_{i,j,c}) \in \mathbb{R}^{IK \times JL \times 3} \) with slight anti-aliasing (i.e., non-overlapped slight blurring) as

\[
Y_{i,j,c} = \sum_{k=1}^{K} \sum_{l=1}^{L} w_{k,l} X_{(i-1)K+k,(j-1)L+l,c} \tag{1}
\]
where \( K \) and \( L \) are supposed to be integers larger than 1, and 
\( w_{k,l} \in \mathbb{R} \) are downsampling weights \(^1\) s.t. 
\[ \sum_{k=1}^{K} \sum_{l=1}^{L} w_{k,l} = 1. \]
Define \( y \in \mathbb{R}^{3\times J \times L} \) and \( x \in \mathbb{R}^{3\times J \times K \times L} \) as the vectorized versions 
of the LR image \( Y \) and the HR image \( X \), respectively. Then 
by using a block-diagonal-like matrix \( A \in \mathbb{R}^{3\times J \times L \times K \times L} \), 
the downsampling model in (1) is expressed as a matrix form \(^2\)
\[ y = Ax. \quad (2) \]

\(^1\)If downsampling is the nearest-neighbor interpolation or the bilinear 
interpolation, then (1) is always satisfied. If downsampling is the bicubic polynomial 
interpolation and both \( K \) and \( L \) are equal to or larger than 4, then (1) is 
also satisfied, but some weights are negative. However, if downsampling is the 
bicubic polynomial interpolation and \( K \) or \( L \) is equal to or smaller than 3, then 
(1) is not satisfied. If downsampling is the bicubic spline interpolation, then (1) 
is never satisfied. If standard anti-aliasing, i.e., overlapped blurring as in [16], 
is done right before interpolation, then (1) becomes an approximation model.

\(^2\)Although most downsampling methods, including the bicubic polynomial 
and spline interpolations, are expressed as linear operators \( A \) in (2), the condition 
in (1) is important to easily compute the orthogonal projection as in (10).

\[ \mathbb{R}^{3\times J} \]. As a loss function to be minimized for training the SR 
network, the mean squared error (MSE) 
\[ l_{\text{MSE}} = \frac{1}{N} \sum_{n=1}^{N} \| \hat{x}_n - x_n \|_2 \quad (3) \]
is usually adopted, where \( \| \cdot \|_2 \) denotes the \( \ell_2 \) norm of a vector. 
Some papers claim that the mean absolute error (MAE) 
\[ l_{\text{MAE}} = \frac{1}{N} \sum_{n=1}^{N} \| \hat{x}_n - x_n \|_1 \quad (4) \]
leads to slightly better results, where \( \| \cdot \|_1 \) is the \( \ell_1 \) norm. The 
difference between (3) and (4) has little effect on human eyes.

\[ l_{\text{AD}} = \frac{1}{N} \sum_{n=1}^{N} \log(D(x_n)) + \frac{1}{N} \sum_{n=1}^{N} \log(1 - D(\hat{x}_n)), \quad (5) \]
where \( \hat{x}_n = G(y_n) \) is the SR image generated from the input 
LR image \( y_n \) by the current generator \( G \). In GAN-based tech-
niques, the second term \( \frac{1}{N} \sum_{n=1}^{N} \log(1 - D(\hat{x}_n)) \) in (5) is 
usually used as a loss function for \( G \), but in the context of SR, 
\[ l_{\text{ADV}} = -\frac{1}{N} \sum_{n=1}^{N} \log(D(\hat{x}_n)) \quad (6) \]
is often used as the adversarial loss due to its better gradient 
behavior [12]. As a result, the total loss function for \( G \) can be 
expressed as 
\[ l = l_{\text{C}} + \kappa l_{\text{A}}, \quad (7) \]
where \( l_C \) is the content loss evaluating the consistency between \( \hat{x}_n \) and \( x_n \), and \( \kappa > 0 \) is a weight for the adversarial loss \( l_A \) in (6). In the simplest cases, the content loss \( l_C \) is defined as \( l_{\text{MSE}} \) in (3) or \( l_{\text{MAE}} \) in (4). In more complicated cases [12], [13], the content loss \( l_C \) is defined, for example, as, for \( c \) MSE of the VGG feature maps [18] or MAE of the differential images.

In general, the exact computation of \( l_C \) mainly evaluates the differences between SR images \( \hat{x}_n \) and true HR images \( x_n \). However, it is not considered whether re-downsampling results \( A\hat{x}_n \) are close to given LR images \( y_n \) or not. On the other hand, the true HR images \( x_n \) always satisfy \( A\hat{x}_n = y_n \). To make the most of training data, we should also consider the re-downsampling results \( A\hat{x}_n \). In this paper, we propose a modification technique which enables any generator to generate SR images \( \hat{x}_n \) satisfying \( A\hat{x}_n = y_n \). The proposed technique is expected effective especially for GAN-based SR methods since the generator learns high-frequency components while maintaining the original low-frequency components, i.e., information on the input LR images.

First, we define the set of all SR images matching the input LR image by

\[
A_n := \{ x \in \mathbb{R}^{3JKL} \mid Ax = y_n \} = \{ x_n + z \in \mathbb{R}^{3JKL} \mid Az = 0 \} = x_n + \mathcal{N}(A), \tag{8}
\]

where \( \mathcal{N}(A) \) denotes the null space of \( A \). From (8), it is found that the set \( A_n \) is a linear manifold and the true HR image \( x_n \) always belongs to \( A_n \). By applying the orthogonal projection \( P_{A_n} \) onto \( A_n \), in the output layer (see the red box of Fig. 1), to the conventional SR image \( \hat{x}_n \), we obtain the proposed SR image \( \hat{x}_n := P_{A_n}(\hat{x}_n) \) which matches the input LR image \( y_n \).

The proposed SR image \( \hat{x}_n \) is concretely expressed as

\[
\hat{x}_n = P_{A_n}(\hat{x}_n) = \arg\min_{x \in A_n} \| \hat{x}_n - x \|_2 \tag{9}
\]

\[
= \hat{x}_n - A^T(AA^T)^{-1}(A\hat{x}_n - y_n). \tag{9}
\]

In general, the exact computation of \( (AA^T)^{-1} \) is difficult when the image size becomes huge. In this paper, since we assumed the blockwise downsampling model as in (1), \( AA^T \) is always a diagonal matrix and hence (9) is easily computed by

\[
\hat{x}_n = P_{A_n}(\hat{x}_n) = \hat{x}_n - \frac{1}{\sum_{k=1}^{K} \sum_{l=1}^{L} w_{k,l}^2} A^T(A\hat{x}_n - y_n). \tag{10}
\]

As shown in Fig. 2, MSE between \( \hat{x}_n \) and \( x_n \) in (3) (brown line) can be divided into vertical components to \( A_n \) (blue line) and horizontal components to \( A_n \) (green line), and we have

\[
\| \hat{x}_n - x_n \|_2^2 = \| \hat{x}_n - P_{A_n}(\hat{x}_n) \|_2^2 + \| P_{A_n}(\hat{x}_n) - x_n \|_2^2 \geq \| P_{A_n}(\hat{x}_n) - x_n \|_2^2 = \| \hat{x}_n - x_n \|_2^2. \tag{11}
\]

From (11), if \( A\hat{x}_n \not= y_n \), then MSE always becomes smaller, i.e., PSNR always improves, by the orthogonal projection \( P_{A_n} \). Since the output image is changed from \( \hat{x}_n \) to \( P_{A_n}(\hat{x}_n) \), MSE content loss is changed to

\[
l_{C'} = \frac{1}{N} \sum_{n=1}^{N} \| P_{A_n}(\hat{x}_n) - x_n \|_2^2, \tag{12}
\]

where we use not MAE but MSE as the content loss for stable training. When we only consider a weighted sum of \( l_{C'} \) in (12) and \( l_A \) in (6) as the total loss function \( l \) for the generator \( G \), actually SR results are not good because error information of the vertical components to \( A_n \) is lost. Hence, to generate SR images as close as possible to the linear manifold \( A_n \) before the orthogonal projection, we further propose to consider MSE of the vertical components to \( A_n \) as the projection loss

\[
l_P = \frac{1}{N} \sum_{n=1}^{N} \| x_n - P_{A_n}(\hat{x}_n) \|_2^2. \tag{13}
\]

Note that, under the downsampling model in (1), the projection loss \( l_P \) is also expressed as

\[
l_P = \frac{1}{N(K)} \sum_{k=1}^{K} \sum_{l=1}^{L} \frac{w_{k,l}^2}{2} \sum_{n=1}^{N} \| A^T(A\hat{x}_n - y_n) \|_2^2. \tag{14}
\]

From (14), the projection loss is essentially equivalent to MSE between the re-downsampling results \( A\hat{x}_n \) and the LR images \( y_n \). Finally the proposed total loss function for \( G \) is defined as

\[
l = l_{C'} + \lambda l_P + \kappa l_A, \tag{15}
\]

where \( \lambda > 0 \) and \( \kappa > 0 \). Since the content loss \( l_{C'} \) in (12) is more important than the projection loss \( l_P \) in (13), we recommend to use \( \lambda \) which is smaller than 1. In (15), we evaluate the horizontal and vertical MSEs at a ratio of \( 1 : \lambda \) while in (7) the conventional methods evaluated them at a ratio of \( 1 : 1 \).

3Some downsampling losses similar to (14) have been proposed in [19]–[21]. Differently from [19]–[21], we applied the orthogonal projection in the output layer and clarified the equivalence of the projection and downsampling losses.
IV. NUMERICAL EXPERIMENTS

We compare SR results of EDSR, EDSR-Projection, EDSR-GAN, and EDSR-GAN-Projection, where ‘Projection’ means that the orthogonal projection is added in the output layer of the generator and the content loss \( l_c = l_{MSE} \) is replaced with \( l_c + \lambda l_{P} \), and ‘GAN’ means that the discriminator of SRGAN [12] is trained and used in the adversarial loss \( l_A \). We set the weights \( \lambda = 10^{-3} \) and \( \kappa = 10^{-3} \). DIV2K dataset is adopted as training data. We set the size of training HR patches as 96 × 96 for ×2 scale (i.e., \( K = L = 2 \)) and 144 × 144 for ×3 scale. For each scale, all LR patches of size 48 × 48 are given by down-sampling of the HR patches with an arithmetic mean matrix \( A \). We evaluate the SR performance of each method on four test LR images. One is downsampled with the matrix \( A \), and the other is given with the bicubic spline interpolation.

We utilize ADAM [22] as the optimizer, where we set \( \alpha = 10^{-4} \) in the training of the CNNs, \( \alpha = 10^{-5} \) in the training of the GANs, and \( \beta = 0.9 \). For stable GAN training, a pre-trained EDSR is used as the initial value of the generator. We set the minibatch size as 96 to speed up training, and use PSNR and FSIMc [23] as image quality indices for comparison. It takes about 2 days to train the networks with GeForce GTX 1080ti.

Tables I and II summarize the SR results for the two kinds of the LR images. Among the single-CNN-based methods, there are no significant differences, which means that a MSE-based CNN almost satisfies \( A \hat{x}_n = y_n \) without the orthogonal projection. On the other hand, among the GAN-based methods, the proposed method improves PSNR by the orthogonal projection in Tables I and II. Moreover, the proposed method improves FSIMc mainly for ×3 scale in Table I, and for the both scales in Table II (though the downsampling is different from training data), which means that the proposed method has a certain robustness against the different down-sampling. Figures 3 and 4 show two SR results. The SR images of the MSE-based CNNs are over-smoothed, and those of EDSR-GAN seem clearer but include some artifacts. On the other hand, the proposed GAN-based SR (EDSR-GAN-Projection) generates high-frequency components accurately while greatly reducing the artifacts.

V. CONCLUSION AND FUTURE WORK

In this paper, for image SR using CNNs, we proposed to use an orthogonal projection in the output layer for generating SR images having the perfect consistency with the input LR images. In particular, the proposed method is effective for GAN-based SR methods since generators learn high-frequency components while keeping the original low-frequency ones. Note that the proposed method can be applied to other constrained learning if the projection onto the constraint is available.

As future work, by considering noise and model error of \( A \), we plan to use the set of all images of noise level under \( \epsilon > 0 \):

\[
A_{\epsilon} := \{ x \in \mathbb{R}^{MJKL} | ||Ax - y_n||_2 \leq \epsilon \}
\]

Under the condition in (1), the projection onto \( A_{\epsilon} \) is given by

\[
P_{A_{\epsilon}}(\hat{x}_n) = \hat{x}_n - \sum_{k=1}^{K} \sum_{l=1}^{L} w_{k,l} A^T (A\hat{x}_n - y_n),
\]

where \( \delta_{\epsilon}^n : \mathbb{R}^{MJKL} \rightarrow \mathbb{R} \) is a differentiable function defined as

\[
\delta^n_{\epsilon}(\hat{x}_n) := \begin{cases} 0 & \text{if } ||A\hat{x}_n - y_n||_2 \leq \epsilon, \\ \frac{||A\hat{x}_n - y_n||_2 - \epsilon}{||A\hat{x}_n - y_n||_2} & \text{otherwise.} \end{cases}
\]

REFERENCES


### Table I

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Scale</th>
<th>EDSR</th>
<th>EDSR-Projection</th>
<th>EDSR-GAN</th>
<th>EDSR-GAN-Projection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set5</td>
<td>×2</td>
<td>36.31/0.9728</td>
<td>36.31/0.9726</td>
<td>34.02/0.9664</td>
<td>35.01/0.9625</td>
</tr>
<tr>
<td></td>
<td>×3</td>
<td>34.27/0.9416</td>
<td>34.26/0.9376</td>
<td>33.02/0.9251</td>
<td>33.33/0.9300</td>
</tr>
<tr>
<td>Set14</td>
<td>×2</td>
<td>33.91/0.9538</td>
<td>33.93/0.9540</td>
<td>32.38/0.9027</td>
<td>31.50/0.8685</td>
</tr>
<tr>
<td></td>
<td>×3</td>
<td>32.59/0.9065</td>
<td>32.58/0.9059</td>
<td>31.50/0.8685</td>
<td>31.30/0.8567</td>
</tr>
<tr>
<td>BSD100</td>
<td>×2</td>
<td>33.76/0.9396</td>
<td>33.76/0.9399</td>
<td>32.38/0.9275</td>
<td>31.50/0.8685</td>
</tr>
<tr>
<td></td>
<td>×3</td>
<td>32.47/0.8807</td>
<td>32.64/0.8810</td>
<td>31.50/0.8685</td>
<td>31.71/0.8743</td>
</tr>
<tr>
<td>Urban100</td>
<td>×2</td>
<td>33.71/0.9468</td>
<td>33.73/0.9464</td>
<td>32.50/0.9332</td>
<td>32.91/0.9325</td>
</tr>
<tr>
<td></td>
<td>×3</td>
<td>32.29/0.8854</td>
<td>32.29/0.8856</td>
<td>31.30/0.8657</td>
<td>31.55/0.8729</td>
</tr>
</tbody>
</table>

SR results (PSNR/FSIM\(_C\)) for Set5, Set14, BSD100 and Urban100. Test images are downsampled by the matrix \(A\) used in training.

### Table II

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Scale</th>
<th>EDSR</th>
<th>EDSR-Projection</th>
<th>EDSR-GAN</th>
<th>EDSR-GAN-Projection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set5</td>
<td>×2</td>
<td>35.97/0.9697</td>
<td>35.98/0.9697</td>
<td>33.69/0.9637</td>
<td>35.15/0.9644</td>
</tr>
<tr>
<td></td>
<td>×3</td>
<td>33.88/0.9340</td>
<td>33.87/0.9339</td>
<td>33.21/0.9250</td>
<td>33.36/0.9285</td>
</tr>
<tr>
<td>Set14</td>
<td>×2</td>
<td>33.45/0.9370</td>
<td>33.48/0.9378</td>
<td>32.39/0.9302</td>
<td>32.98/0.9314</td>
</tr>
<tr>
<td></td>
<td>×3</td>
<td>32.40/0.8961</td>
<td>32.39/0.8957</td>
<td>31.83/0.8885</td>
<td>32.02/0.8939</td>
</tr>
<tr>
<td>BSD100</td>
<td>×2</td>
<td>33.60/0.9360</td>
<td>33.59/0.9361</td>
<td>32.41/0.9238</td>
<td>33.08/0.9306</td>
</tr>
<tr>
<td></td>
<td>×3</td>
<td>32.33/0.8704</td>
<td>32.33/0.8706</td>
<td>31.79/0.8671</td>
<td>31.92/0.8708</td>
</tr>
<tr>
<td>Urban100</td>
<td>×2</td>
<td>33.10/0.9336</td>
<td>33.11/0.9337</td>
<td>32.25/0.9241</td>
<td>32.70/0.9272</td>
</tr>
<tr>
<td></td>
<td>×3</td>
<td>31.94/0.8636</td>
<td>31.94/0.8636</td>
<td>31.48/0.8576</td>
<td>31.63/0.8624</td>
</tr>
</tbody>
</table>

SR results (PSNR/FSIM\(_C\)) for Set5, Set14, BSD100 and Urban100. Test images are downsampled by the bicubic spline interpolation.

![Fig. 3. SR results of ‘head’ in Set5 for ×3 scale (PSNR/FSIM\(_C\)).](a) HR image (a’) HR image (b) EDSR (32.38/0.9091) (c) EDSR-Projection (32.37/0.9092) (d) EDSR-GAN (31.64/0.8969) (e) EDSR-GAN-Projection (31.78/0.9017)

![Fig. 4. SR results of ‘img003’ in Urban100 for ×3 scale (PSNR/FSIM\(_C\)).](a) HR image (a’) HR image (b) EDSR (30.55/0.8582) (c) EDSR-Projection (30.55/0.8568) (d) EDSR-GAN (29.74/0.8349) (e) EDSR-GAN-Projection (29.96/0.8452)

664