Accurate and Lightweight Image Super-Resolution with Model-Guided Deep Unfolding Network

Qian Ning, Student Member, IEEE, Weisheng Dong, Member, IEEE, Guangming Shi, Senior Member, IEEE, Leida Li, Member, IEEE and Xin Li, Fellow, IEEE

Abstract—Deep neural networks (DNNS) based methods have achieved great success in single image super-resolution (SISR). However, existing state-of-the-art SISR techniques are designed like black boxes lacking transparency and interpretability. Moreover, the improvement in visual quality is often at the price of increased model complexity due to black-box design. In this paper, we present and advocate an explainable approach toward SISR named model-guided deep unfolding network (MoG-DUN) by integrating deep denoising and nonlocal regularization as trainable modules within a deep learning framework. In contrast to existing model-based SISR methods including RCAN, SRMDNF, and SRFBN by involving dense/skip connections as well as nonlocal implementation. In addition to explainability, MoG-DUN is accurate (producing fewer aliasing artifacts), computationally efficient (with reduced model parameters), and versatile (capable of handling multiple degradations). The superiority of the proposed MoG-DUN method to existing state-of-the-art image SR methods including RCAN, SRMDNF, and SRFBN is substantiated by extensive experiments on several popular datasets and various degradation scenarios.

I. INTRODUCTION

The field of deep learning for single image super-resolution (SISR) has advanced rapidly in recent years. Super-resolution by convolutional neural network (SRCNN) [1] represented one of the pioneering works in this field. Since then, many follow-up works have been developed including Super-resolution via Generative Adversarial Network (SRGAN) [2], SR via very deep convolutional networks (VDSR) [3], Trainable Nonlinear Reaction Diffusion Networks (TNRD), Deeply-recursive convolutional network (DRCN) [4], Enhanced Deep Residual Networks (EDSR) [5], Laplacian Pyramid Super-Resolution Network (LapSRN) [6], and Deep Back-Projection Networks (DBPN) [7]. Most recently, SISR has benefited from the advances in novel design of network architectures such as densely-connected networks (e.g., RDN [8]), attention mechanism (e.g., RCAN [9] and SAN [10]), multiple degradations (e.g., SRMDNF [11]), feedback connections (e.g., SRFBN [12] and feature aggregation [13]).

Despite the rapid progress, one of the long-standing open issues is the lack of interpretability. Most existing networks for SISR are designed based on the black-box principle - i.e., little is known about their internal workings regardless of the desirable input-output mapping results. The difficulty with understanding the internal mechanism of deep learning-based SISR has become even more striking when the network gets deeper and more sophisticated (e.g., due to attention and feedback). For instance, the total number of parameters of EDSR [5] has reached over 40M, which makes it a less feasible for practical applications. By contrast, it is more desirable to seek alternative glass-box design (a.k.a. clear-box) where the inner components are readily available for inspection. A hidden benefit of such a transparent approach is that it might lead to more efficient design because any potential redundancy (in terms of model parameters) can be cautiously avoided. Can we solve SISR under an emerging framework of interpretable machine learning [14]? Does a transparent design lead to computationally more efficient solution to SISR facilitating practical applications (e.g., lightweight architecture [15])?

We provide affirmative answers to the above questions in this paper. An important new insight brought to the field of SISR by this work is model-guided (MoG) design for deep neural networks. The basic idea behind MoG design is to seek a mathematically equivalent implementation of existing model-based solution by deep neural networks. Similar ideas have scattered in the literature of so-called deep unfolding networks (e.g., [16], [17], [18]). It has been well-established that classic model-based tools including sparse coding, Markov Random Field, belief propagation, and non-negative matrix factorization can be unfolded to a network implementation [18], [16], [19], [20]. Note that in the field of SISR, there are plenty of model-based methods [21], [22], [23], [24], which could provide a rich source of inspiration for MoG design. Zhang et al. proposed USRNNet to handle the classical degradation model via a single model. Based on our previous [25] and recent [19] works, we propose a MoG Deep Unfolding Network (MoG-DUN) to SISR that is not only explainable and efficient but also accurate and versatile.

We will show that most model-based image priors or regularization functions, including both convex and nonconvex formulation, can be leveraged as the guidance for the design of deep neural networks. In fact, the class of nonconvex regularization models do not pose additional difficulty to learning-based approaches despite their intractability from an
We present a model-guided explainable approach toward SISR by striking an improved tradeoff between the cost (as measured by the number of model parameters) and performance (reflected by the PSNR metric on Set5 with ×4 bicubic down-sampling). Our methods are highlighted by the red color.

analytical perspective. This is because that the training of deep neural networks (DNN) has a natural and intrinsic connection with the optimization of cost functions by numerical methods (e.g., gradient descent methods [26]). Aiming at breaking the coherence barrier [27] (a long-standing open problem in SISR), we have chosen a reference model, nonlocal autoregressive model (NARM) with improved incoherence properties [25], to showcase the process of Model-Guided (MoG) design. We will unfold this model with a nonconvex cost function into network implementations consisting of multistage U-net modules [19]. This work further extends our previous work [19] by incorporating a nonlocal module (for AR modeling computation) and a reconstruction module (for AR modeling correction). Moreover, long connections are introduced across different stages to copy the hidden states from previous stages to the current, which facilitates the information flow. Even though MoG-DUN has achieved outstanding performance (in terms of both subjective and objective qualities), its model complexity has been kept much lower than that of DBPN [7], RDN [8], and RCAN [9] and only slightly higher than that of SRMDNF [11] and SRFBN [12], as shown in Fig. 1. Meantime, MoG-DUN is explainable and versatile thanks for its transparent design, which has the potential of leveraging to other image restoration applications.

The key contributions of this paper are summarized below.

- We present a model-guided explainable approach toward SISR. Our approach is capable of leveraging existing NARM-based SISR into a network implementation in a transparent manner. Thanks to the improved incoherence property of NARM, the corresponding MoG-DUN has better capability of alleviating long-standing problem in SISR such as aliasing artifacts. The framework of MoG design is applicable to all existing models including nonconvex and nonlocal regularization.
- We demonstrate how to unfold an existing NARM [25] model into the corresponding network implementation.

The unfolding result consists of the multi-stage concatenation of Unet-like deep denoising module along with a nonlocal-AR module and a reconstruction module. Both long and short skip connections are introduced within and across different stages to facilitate the information flow. The model complexity of our MoG-DUN is shown to be comparable to that of [19].

- We report extensive experimental results for the developed MoG-DUN, which justify its achieving an improved tradeoff between the modeling complexity and the SR reconstruction performance. Our MoG-DUN has achieved better performance than existing state-of-the-art SISR such as RDN [8] and RCAN [9] with a lower cost as measured by the number of model parameters. In particular, visual quality improvement in terms of sharper edges and more faithful reconstruction of texture patterns is mostly striking.

II. MODEL-BASED IMAGE INTERPOLATION AND RESTORATION

We first briefly review previous works on model-based image interpolation (e.g., NARM [25]) and image restoration (e.g., DPDNN [19]), which sets up the stage for model-guided network design. The NARM model has achieved state-of-the-art performance in model-based image interpolation (a special case of SISR involving down-sampling only and without any anti-aliasing low-pass filter). Denoising Prior driven DNN (DPDNN) [19] represents the most recent MoG design of deep unfolding network whose variation has been adopted as the baseline method in our research.

A. Nonlocal Auto-regressive Model (NARM)

The basic idea behind nonlocal auto-regressive modeling (refer to Fig. 2) is to extend the traditional auto-regressive (AR) models by redefining the neighborhood. For a given patch \( x_i \), NARM seeks its sparse linear decomposition over a set of nonlocal (instead of local) neighborhood. Following the notation in [25], we have

\[
x_i \approx \sum_j \omega^j_i x^j_i
\]  

Fig. 1: This work advances the state-of-the-art in SISR by striking an improved tradeoff between the cost (as measured by the number of model parameters) and performance (reflected by the PSNR metric on Set5 with ×4 bicubic down-sampling). Our methods are highlighted by the red color.

Fig. 2: The illustration of NARM [25] - a nonlocal extension of classic auto-regressive (AR) model for image signals.
where \(x_i^j\) denotes the \(j\)-th similar patch found in the nonlocal neighborhood. A natural way of extending the classic AR modeling is to formulate the following regularized Least-Square problem:

\[
\mathbf{w}_i = \arg\min_{\mathbf{w}_i} ||x_i - \mathbf{X} \mathbf{w}_i||_2^2 + \gamma ||\mathbf{w}_i||_2^2 \tag{2}
\]

where \(\mathbf{X} = [x_1^1, x_1^2, \ldots, x_i^j, \ldots]\), \(\mathbf{w}_i = [\omega_{i1}, \omega_{i2}, \ldots, \omega_{i1}^j]^T\), and \(\gamma\) is the regularization parameter. The closed-form solution to Eq. (2) is given by

\[
\mathbf{w}_i = (\mathbf{X}^T \mathbf{X} + \gamma \mathbf{I})^{-1} (\mathbf{X}^T \mathbf{x}_i) \tag{3}
\]

where \(\mathbf{I}\) is an identity matrix with the same size as \(\mathbf{X}^T \mathbf{x}\). Based on newly determined AR coefficients \(\mathbf{w}_i\), we can represent the nonlocal autoregressive model (NARM) of image \(x\) by

\[
x = \mathbf{S} x + \mathbf{e}_x \tag{4}
\]

where \(\mathbf{e}_x\) is the modeling error, and the NARM matrix \(\mathbf{S}\) is

\[
\mathbf{S}_{ij} = \begin{cases} \omega_i^j, & \text{if } x_i^j \text{ is a nonlocal neighbor of } x_i \\ 0, & \text{otherwise} \end{cases} \tag{5}
\]

The NARM matrix \(\mathbf{S}\) can be embedded into a standard image degradation model by modifying Eq. (7) into

\[
y = \hat{\mathbf{A}} x + \mathbf{n} \tag{6}
\]

where \(\hat{\mathbf{A}} = \mathbf{A} \mathbf{S}\) is the new degradation operator. It has been shown in [25] that such nonlocal extension is beneficial to improving the incoherence between sampling matrix and sparse dictionaries under the framework of model-based image restoration. In this work, we will show how to unfold this NARM into a DNN-based implementation.

### B. Model-based Image Restoration

The objective of model-based image restoration (IR) is to estimate an unknown image \(x\) from its degraded observation \(y\). The degradation process can be formulated by:

\[
y = \mathbf{A} x + \mathbf{n} \tag{7}
\]

where \(\mathbf{A}\) denotes the degradation operators (e.g., blurring kernels, down-sampling operations) and \(\mathbf{n}\) denotes the additive noise. Accordingly, model-based IR can be formulated into the following optimization problem:

\[
x = \arg\min_x ||y - \mathbf{A} x||_2^2 + \lambda \Omega(x) \tag{8}
\]

where \(\lambda\) is the Lagrangian multiplier and \(\Omega(x)\) the regularization function. The choice of various regularization functions \(\Omega(x)\) reflects different ways of incorporating a priori knowledge about the unknown HR image \(x\).

To solve model-based SR problem in Eq. (3), half-quadratic splitting method [28] converts an equally-constrained optimization problem into an equivalent non-constrained optimization problem, which can be written as

\[
(x, v) = \arg\min_{x,v} ||y - \mathbf{A} x||^2_2 + \eta ||x - v||^2_2 + \lambda \Omega(v) \tag{9}
\]

where \(v\) is an auxiliary splitting variable. It follows that Eq. (9) boils down to alternatively solving two sub-problems associated with the fidelity and regularization terms respectively,

\[
x^{(t+1)} = \arg\min_{x} ||y - \mathbf{A} x||^2_2 + \eta ||x - v^{(t)}||^2_2 \tag{10a}
\]

\[
v^{(t+1)} = \arg\min_{v} \eta ||x^{(t+1)} - v||^2_2 + \lambda \Omega(v) \tag{10b}
\]

The main idea behind deep unfolding network is that conventional iterative soft-thresholding algorithm (ISTA) in sparse coding can be implemented equivalently by a stack of recurrent neural networks [16]. Such correspondence has inspired a class of convolutional neural network (CNN)-based image denoising techniques [29]. CNN-based denoising prior was later extended into other model-based image restoration problems in Image Restoration via CNN (IRCNN) [30]. In IRCNN, a CNN module is adopted to solve Eq. (10b) while Eq. (10a) is solved in close-form by

\[
x^{(t+1)} = \mathbf{W}^{-1} \mathbf{b}, \tag{11}
\]

where \(\mathbf{W}\) represents the matrix related to the degradation matrix \(\mathbf{A}\). Note that it is often time-consuming to calculate an inverse matrix. Based on such observation, a computationally more efficient unfolding strategy was developed in DPDNN [19]. DPDNN addressed the problem of matrix inverse by solving Eq. (10a) in a different way - i.e., they propose to compute \(x^{(t+1)}\) with a single step of gradient descent by

\[
x^{(t+1)} = x^{(t)} - \delta [\mathbf{A}^T (\mathbf{A} x^{(t)} - y) + \eta (x^{(t)} - v^{(t)})]
\]

\[
= \hat{\mathbf{A}} x^{(t)} + \delta \mathbf{A}^T y + \delta \eta v^{(t)} \tag{12}
\]

where \(\hat{\mathbf{A}} = [(1 - \delta \eta) \mathbf{I} - \delta \mathbf{A}^T \mathbf{A}]\) can be pre-computed (so the updating of \(x^{(t)}\) can be done efficiently). As shown in [19], we only need to obtain an approximated solution to...
the \(x\)-subproblem, which can be done in a computationally efficient manner. Although DPDNN \[19\] has found several image restoration applications including SISR, its network architecture remains primitive (e.g., lacking dense and skip connections) and there is still room for further optimization.

### III. Model-Guided Deep Unfolding Network

In this paper, we take another step forward by showing a generalized unfolding strategy applicable to almost any model-based image restoration including both nonlocal and nonconvex regularization. It is well known that the class of nonconvex optimization problems do not admit computationally efficient solutions, which calls for convex relaxation or approximation \[31\]. However, we note that nonconvex cost functions have become almost the default option in the field of machine learning \[32\] especially deep learning \[33\]. This is likely due to the fact that the training of deep neural networks (DNN) has a natural and intrinsic connection with the optimization of cost functions by numerical methods such as gradient descent methods \[29\] (e.g., the popular Adam optimization represents a stochastic gradient descent method). Here, we have chosen a previous reference model (NARM originally designed for image interpolation \[25\]) to showcase the process of Model-Guided (MoG) design. Thanks to the improved coherence property of NARM, unfolding this model has the potential of alleviating some long-standing open problems in SISR such as the suppression of aliasing artifacts.

#### A. Deep Unfolding Network for NARM Model

When compared with the image prior model in Eq. (3), NARM is more sophisticated because it involves two regularization terms: one specified by the denoising prior (a variation of DPDNN \[19\]) and the other related to the nonlocal AR model \[25\] (a brand-new design). Formally, we consider the following nonconvex optimization problem

\[
x = \arg\min_x ||y - Ax||_2^2 + \mu ||y - ASx||_2^2 + \lambda \Omega(x)
\]  

where \(\Omega(x)\) and \(Sx\) denote denoising-based prior and NARM-defined prior respectively. Using Half-Quadratic Splitting \[28\], we can rewrite the above optimization problem into

\[
(x, v) = \arg\min_{x,v} ||y - Ax||_2^2 + \mu ||y - ASx||_2^2 + \eta ||x - v||_2^2 + \lambda \Omega(v)
\]

where \(v\) is the auxiliary variable. It is well known that the above optimization problem can be solved by alternatively solving two subproblems related to \(x\) and \(v\) as shown in Eqs. (10a) and (10b) \[19\].

Based on the NARM model in Eq. (3), we have \(Sx = x + e\) where \(e = -e_x\) is the modeling error (regardless of sign flip). It follows that the NARM optimization problem in Eq. (13) can be translated into

\[
(x, v, e) = \arg\min_{x,v,e} ||y - Ax||_2^2 + \mu ||y - A(x + e)||_2^2 + \gamma ||Sx - (x + e)||_2^2 + \eta ||x - v||_2^2 + \lambda \Omega(v)
\]  

As an extension of previous result \[19\], we note that the newly-formulated NARM-based optimization problem can be solved by alternatively solving the following three sub-problems

\[
x^{(t+1)} = \arg\min_x ||y - Ax||_2^2 + \mu ||y - A(x + e^{(t)})||_2^2 + \gamma ||Sx - (x + e^{(t)})||_2^2 + \eta ||x - v^{(t)}||_2^2
\]

\[
e^{(t+1)} = \arg\min_e \mu ||y - A(x^{(t)} + e)||_2^2 + \gamma ||x^{(t)} + e - Sx^{(t)}||_2^2
\]

\[
v^{(t+1)} = \arg\min_v \eta ||x^{(t)} - v||_2^2 + \lambda \Omega(v)
\]  

In traditional model-based approaches \[34\], \[35\], alternatively solving the above three equations requires many iterations to converge leading to prohibitive computational cost. Meantime, the regularization functions and the hyperparameters cannot be jointly optimized in an end-to-end manner. To address those issues, we propose to unfold the NARM-based optimization in Eq. (16) into a concatenation of repeating network modules as shown in Fig. 3. The overall network architecture of our MoG-DUN can be viewed as an extension of previous work DPDNN \[19\] by incorporating two new modules (non-local-AR and reconstruction) in addition to the denoising module. Note that both short and long skip connections \[36\] are incorporated to facilitate the information flow across the stages.

The \(T\) repeating stages in MoG-DUN exactly executes \(T\) iterations of Eq. (16). In our current implementation, a
total of $T = 4$ stages was adopted. Each stage of MoG-DUN consists of three basic building blocks: U-net based deep denoising module, a fast nonlocal-AR module, and a versatile reconstruction module. The deep denoising module is responsible for the updating of auxiliary variable $v^{(t+1)}$ as described in Eq. (16); the fast nonlocal-AR module calculates the NARM matrix $S$ as defined by Eq. (5) and the corresponding $Sx^{(t)}$ in a computationally efficient manner. The versatile reconstruction module takes auxiliary variable $v^{(t+1)}$ and nonlocal AR model $Sx^{(t)}$ as inputs and output reconstructed image $x^{(t+1)}$. Then, the updated $x^{(t+1)}$ is fed into the next stage to refine the estimate of $v$ and $S$ again. The denoising module, nonlocal-AR module and reconstruction module are alternatively updated $T$ times until reaching the final reconstruction. We will elaborate on each module next.

B. Deep Denoising Module via Dense RNN

In general, any existing image denoising network can be used as the denoising module here. Inspired by the success of U-net in semantic segmentation [37] and object refinement [38], we have adopted a variant of U-net [39] - dense recurrent neural network (RNN) [40] - as the backbone of our denoising module. Let $h_1, ..., h_t$ denote the hidden states of $t$ stages, which will be used in the next stages. Different from exiting RNN methods [41], [42], [43], for SISR receiving only one state $h^{t-1}$ of former stage, we propose to leverage multiple states $h^1, ..., h^{t-1}$ of former stages through long connections. As shown in Fig. 3(b) the processed information can be leveraged to refine the current image reconstruction at the $(t+2)$-th stage by receiving former states $h^3, ..., h^{t-1}, h^t, h^{t+1}$ at previous $t$ + 1 stages [40]. Exploiting the hidden states of previous stages allows us to more faithfully reconstruct the missing high-frequency information for SISR (please refer to Figs. 3[4] for concrete image examples).

As shown in Fig. 3(a), the encoder part consists of four encoding blocks (EB) and four decoding blocks (DB). Except the last EB, each EB is followed by a downsampling layer that sub-samples the feature maps with scaling factor of two along both axes to increase the receptive field of neurons. As shown in Fig. 3(b), each DB consists of three convolutional layers with $3 \times 3$ kernels, a residual layer and ReLU nonlinearity to generate 64 channel feature maps. The decoder reconstructs the image with four DBs, each of which contains three convolutional layers and a residual layer as shown in Fig. 3(c). Except the last DB, each DB is followed by a deconvolution layer to increase the spatial size of feature maps by a scaling factor of two. To compensate the lost spatial information, upsampled feature maps are concatenated with the feature maps of the same spatial dimension from the encoder. Thanks to the transparency of our design, all dense-RNN module in the T stages share the same network parameters.

C. Fast Nonlocal-AR Module

The nonlocal-AR module corresponds to the unfolding of NARM matrix $S$ into a network implementation. Based on the observation that natural images often contain rich repetitive structures, nonlocal similarity has shown effective for recovering missing high-frequency information in SISR. In model-based implementation, finding similar patches is often the computational bottleneck because nearest-neighbor search is an NP-hard problem [42]. By contrast, calculating the nonlocal relationship among image patches can be implemented efficiently in parallel by nonlocal neural networks [43].

Inspired by the design of nonlocal operation in [43] and its application into image restoration [44], we have designed a fast nonlocal operation module for computing NARM matrix $S$ here. Fig. 4 illustrates the block diagram of implementing a non-local operation (highlighted by orange color) designed for computing the similarity for a given image. Following the formulation in nonlocal-mean filtering [21] and bilateral filtering [45], a non-local operation can be defined as

$$y_i = \left(\sum_{v_j} f(x_i, x_j) g(x_j)\right) / \sum_{v_j} f(x_i, x_j) \quad (17)$$

where $i$ is the index of an output position (e.g., in space or time), the $j$ is the enumeration of all possible positions, and the pairwise similarity function $f$ calculates the relationship between $i$ and all $j$. Similar to [44], we only calculate the $q \times q$
block centered at position $i$ instead of the whole image. For a balanced trade-off between cost and performance, we have chosen $q = 15$.

The design of similarity function $f$ has been considered in [43]. For example, an embedded Gaussian function can be used to calculate similarity

$$f(x_i, x_j) = e^{\theta(x_i)T \phi(x_j)}$$ (18)

where $\theta(x_i) = W_0 x_i$, $\phi(x_j) = W_\theta x_j$, and $W_0, W_\theta$ are the weight matrices. For simplicity, we opt to employ a linear embedding for $g(x_i) = W_g x_j$, where $W_g$ is a learnable weight matrix. Then the output of nonlocal-AR block $S_x_i$ is calculated by

$$Sx_i = W_\sigma y_i + x_i = W_\sigma [\theta(x_i)T \phi(x_j)] g(x_j) + x_i$$ (19)

where $\sigma$ denotes the softmax operator and $W_\sigma$ is the embedding weight matrix. Note that the pairwise computation of a non-local block enjoys the benefit of being lightweight because its computational cost implemented by matrix multiplication is comparable to a typical convolutional layer in standard networks. Moreover, pairwise computation of nonlocal blocks can be used in high-level, sub-sampled feature maps (e.g., using the subsampling trick as described in [43]). As shown in [46], nonlocal module admits even more efficient implementations by considering a compact representation for multiple kernel functions with Taylor expansion.

### D. Versatile Reconstruction Module

With the output of denoising module $v^{(t+1)}$ and nonlocal-AR module $Sx^{(t)}$, we can reconstruct the updated image $x^{(t+1)}$ in two steps. First, we need to calculate $e^{(t+1)}$ by solving Eq. (16b) using a single step of gradient descent

$$e^{(t+1)} = e^{(t)} - \delta [\mu A^T (A e^{(t)} - y) + \gamma (x^{(t)} + e^{(t)} - Sx^{(t)} - v)]$$ (20)

where $\delta$ is a parameter controlling the step size of convergence. Then, we can reconstruct new $x^{(t+1)}$ with updated $e^{(t+1)}$ and $v^{(t+1)}$ by solving Eq. (16a) using another single step of gradient descent

$$x^{(t+1)} = x^{(t)} - \delta' [A^T (A x^{(t)} - y) + \mu A^T (A (x^{(t)} - e) - y) + \gamma (x^{(t)} + e - Sx^{(t)} + \eta(x^{(t)} - v))]$$ (21)

where $\delta'$ is another relaxation parameter.

Note that Eqs. (20),(21) still involve sophisticated degradation matrix $A$ that is expensive to calculate. In DPDNN [19], the pair of operators $A$ and $A^T$ were replaced by downsampling and upsampling operators but at the price of limited modeling capability (e.g., they can not handle multiple degradation kernels [11]). Different from DPDNN, we propose a more versatile design here - i.e., to simulate the forward and inverse process of degradation by a shallow four-layer convolutional network. Specifically, the degradation process $A$ is simulated by a network called Down-net consisting of three convolutional layers with $3 \times 3$ kernels and 64 channels and one convolutional layer to decrease the spatial resolution with corresponding scale as shown in Fig. 4(d). In a similar way, the network called Up-net representing $A^T$ consists of three convolutional layers with $3 \times 3$ kernels and 64 channels and one deconvolution layer to increase the spatial resolution with corresponding scale as shown in Fig. 4(e).

The versatility of our newly-design reconstruction module is further demonstrated by its capability of handling multiple blur kernels [11]. Dealing with multiple degradations is highly desirable in practical SISR scenarios where image degradation is complex and even spatially-varying. In order to handle multiple blur kernels, one can expand the blur kernel to the same spatial dimension as the input images using a strategy called “dimensionality stretching” [11]. Specifically, assuming the blur kernel is sized by $k \times k$, the blur kernel is first stretched into a vector of size $k^2 \times 1$ and then projected onto a $d$-dimensional($d < k^2$) linear space by the principal component analysis (PCA) to reduce the computation. This way, the blur kernel map consisting of size $d \times H \times W$ is stretched from the original blur kernel of size $k \times k$, where $H$ and $W$ denote the height and width of input images.
A. Experimental Settings

Benchmark Datasets and Performance Metrics. We have used 800 high-quality (2K resolution) images from the DIV2K dataset [47] for training. Following [11], [12], [5], [8], [14], five standard benchmark datasets: Set5 [48], Set14 [49], BSD100 [50], Urban100 [51], Manga109 [52] are used for testing. Performance evaluation in terms of PSNR and SSIM metrics is conducted on the luminance (Y) channel only.

Degradation Models. In order to demonstrate the robustness of our model in varying degradation scenarios, we have designed the following experiments with different parameter settings with the degradation model.

- Default setting. This is the scenario considered in most previous SISR studies - i.e., the low-resolution (LR) image is obtained by bicubic downsampling of the high-resolution (HR) image. The downsampling ratio is usually a small positive integer (×2, ×3, ×4).
- Interpolation setting. This is the scenario consistent with the NARM study [25] in which a LR image is directly down-sampled from the HR image without any anti-aliasing filtering involved. Due to the presence of aliasing, this scenario is generally believed to be more difficult than the default setting.
- Realistic setting. To more faithfully characterize the degradation in the real world, this setting aims at simulating multiple degradation situations caused by different Gaussian kernels [11]. Similar to SRMDNF [11], we have obtained a single trained network through reconstruction module taking multiple degradation kernels. The set of degradation kernels include isotropic Gaussian blur kernels whose width ranges are set to [0.2, 3], [0.2, 3] and [0.2, 4] with scale factor ×2, ×3 and ×4 respectively. The kernel width is uniformly sampled in the above ranges and the kernel size is fixed to 21 × 21 and the projected d-dimensional linear space is set to 6, 8, 10 with scale factor ×2, ×3, ×4 respectively. The mean values of kernel width are 0.5, 1.3, 2.6 as shown in Table IV.

Training Setting. Thanks to the parameter sharing across T stages, the overall MoG-DUN can be trained in an end-to-end manner. In order to further reduce the number of parameters and avoid over-fitting, we enforce deep denoising module (dense-RNN) and reconstruction module to share the same parameters. Unlike DPDNN [19] adopting MSE loss, we have found $L_1$ loss function works better for training the proposed MoG-DUN (e.g., it can facilitate the recovery of more high-frequency information). The $L_1$-based loss function can be expressed as:

$$\Theta = \arg\min_{\theta} \sum_{i=1}^{N} ||F(y_i; \Theta) - x_i||_1,$$

where $y_i$ and $x_i$ denote the $i$th pair of degraded and original image patches respectively, and $F(y_i; \Theta)$ denotes the reconstructed image patch by the network with the parameter set $\Theta$. We randomly select 16 RGB LR patches sized by 48 × 48 as the inputs and stretch the blur kernels when dealing with multiple degradations (called “dimensionality stretching” in [11]). The image patches are randomly rotated by 90°, 180°, 270° and flipped horizontally as standard data augmentation techniques do. The ADAM algorithm [54] with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$ is adopted to optimize the network. The initial learning rate is $10^{-4}$ and decreases by half for every 300 epochs. Our network is implemented under the Pytorch framework and run on a machine with 4 NVIDIA 1080Ti GPUs.

B. Ablation Study

To further verify the effectiveness of nonlocal-AR module, we have conducted an ablation study to compare the PSNR performance of MoG-DUN with and without nonlocal-AR module. In our ablation study, we have used directly downsampling degradation with different Gaussian kernel size of 0.5, 1.0 and ×3 directly downsampling. As shown in Tab. [11]...
the nonlocal-AR module does make a contribution to the overall performance of MoG-DUN.

**TABLE II: Average PSNR results with and without nonlocal-AR module for directly downsampling degradation ×3 with different Gaussian kernels width on four frequently-used benchmark datasets.**

| Methods       | Kernel Width | Set5 [48] | Set14 [49] | BSD68 | Urban100 | Manga109 [52] |
|---------------|--------------|-----------|------------|-------|----------|---------------|
|               |              | PSNR      | SSIM       | PSNR  | SSIM     | PSNR          |
| w/o NL-AR     | 0.5          | 32.86     | 28.99      | 28.00 | 27.27    |               |
| w NL-AR       | 0.5          | 32.98     | 29.11      | 28.12 | 27.51    |               |
| w/o NL-AR     | 1.0          | 34.15     | 30.15      | 28.30 | 28.37    |               |
| w NL-AR       | 1.0          | 34.34     | 30.23      | 28.99 | 28.63    |               |

To explore the impact of the number of unfolded stages on the SISR performance, we have conducted another experiment with varying the parameter $T$, Fig. 8 shows the average PSNR results of different stages $T$ from two to six (with bicubic-downsampling and scaling factor of $\times 2, \times 3, \times 4$). It can be seen that the PSNR increases as the number of stages increases. However, the PSNR improvement rapidly saturates when $T \geq 4$, which justifies the choice of $T = 4$ in our implementation to balance the performance and computational complexity.

**Fig. 8:** The average PSNR performance as a function of parameter $T$ (the total number of Unet stages) of proposed MoG-DUN with $\times 2, \times 3, \times 4$ bicubic-downsampling on Set5 [48].

**C. Experimental Results for the Default Setting**

For bicubic downsampling, we have compared MoG-DUN with nine state-of-the-art image SR methods: EDSR [5], DPDNN [19], DSRN [55], RDN [8], RCAN [9], D-DBPN.
SRMDNF [11], SRFBN [12], USRNet [20]. Similar to [5, 8], we have also employed seven boosting strategies [50] to further improve our results (denoted by MoG-DUN+). The average PSNR and SSIM results of eight benchmark methods in Tab. II are cited from corresponding papers. It is easy to see that our method is superior to most of competing methods in terms of PSNR and SSIM values. When compared with a much deeper network RCAN [9] involving over 400 convolutional layers, we can achieve highly comparable and sometimes even better results.

The image comparison results for a scale factor of $\times 4$ are reported in Fig. 5. For this specific example, our SR-resolved result of ‘Img_046’ from Urban100 are recovered with fewer visible artifacts (e.g., the glassy surface on the right side of the building) than other competing methods. Note that our PSNR result is also noticeably higher than the previous state-of-the-art RCAN at a lower cost. In another challenging example (‘Img_078’ from Urban100 dataset), our method can recover much more faithful textured details as shown in Fig. 6 while all other competing methods suffer from severe aliasing artifacts (i.e., distorted tile patterns). The visual quality improvement achieved by MoG-DUN is mainly due to the fact that our proposed model makes full use of the feature maps from former stages to refine the final result. Taking one more classic example known for its notorious aliasing distortion, Fig. 7 shows the results of bicubic $\times 2$ degradation of ‘Img_002’ from the Set14 dataset. In addition to visually more pleasant SR reconstruction results (in terms of fewer artifacts), our work significantly outperforms other methods in terms of PSNR performance.

V. Conclusion

In this paper, we have demonstrated how to unfold the existing NARM into a multi-stage network implementation that is both explainable and efficient. The unfolded network consists of a concatenation of multi-stage building blocks each of which is decomposed of a deep denoising module, a fast non-local-AR module, and a versatile reconstruction module. This work extends the previous work DPDNN [19] in the following aspects. First, the new regularization term characterized by NARM leads to a three-way (instead of double-headed) alternating optimization, which in principle is applicable to other forms of regularization functions. Meanwhile, the improved incoherence property of NARM makes it suitable for SISR applications particularly on suppressing aliasing artifacts. Second, the unfolded network allows the hidden states of previous stages to be exploited by the later stage. Such densely connected recurrent network architecture is shown important to the recovery of missing high-frequency information in SISR. Extensive experimental results have been reported to show that our MoG-DUN is capable of achieving an improved trade-off between the cost (in terms of network parameter size) and the performance (in terms of both subjective and objective qualities of reconstructed SR images).

References

[1] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in European Conference on Computer Vision, 2014, pp. 184–199.

[2] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. 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, 2017, pp. 4681–4690.

[3] J. Kim, J. Kwon Lee, and K. Mu Lee, “Accurate image super-resolution using very deep convolutional networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 1646–1654.
TABLE III: Average PSNR and SSIM results for directly downsampling degradation on five benchmark datasets. The best performance is shown in bold.

| Method          | Scale | Set5 [48] | Set14 [49] | BSD100 [50] | Urban100 [51] | Manga109 [52] |
|-----------------|-------|-----------|-----------|-------------|---------------|---------------|
|                 |       | PSNR      | SSIM      | PSNR        | SSIM          | PSNR          | SSIM          | PSNR          | SSIM          | PSNR          | SSIM          |
| EDSR [5]        | ×2    | 35.57     | 0.9444    | 31.30       | 0.8864        | 30.21         | 0.8666        | 28.82         | 0.8971        | 35.25         | 0.9446        |
| DPDPNN [19]     | ×2    | 35.51     | 0.9465    | 31.26       | 0.8865        | 30.13         | 0.8648        | 28.79         | 0.8972        | 35.15         | 0.9641        |
| RDN [8]         | ×2    | 35.77     | 0.9458    | 31.38       | 0.8870        | 30.36         | 0.8692        | 29.54         | 0.9071        | 35.67         | 0.9658        |
| RCAN [9]        | ×2    | 35.63     | 0.9449    | 31.34       | 0.8867        | 30.30         | 0.8677        | 29.19         | 0.9024        | 35.59         | 0.9651        |
| SRBFN [12]      | ×2    | 35.66     | 0.9450    | 31.35       | 0.8866        | 30.28         | 0.8670        | 29.04         | 0.9002        | 35.42         | 0.9648        |
| MoG-DUN (ours)  | ×2    | 35.99     | 0.9474    | 31.77       | 0.9333        | 30.53         | 0.8724        | 30.44         | 0.9200        | 36.33         | 0.9689        |
| EDSR [5]        | ×3    | 31.50     | 0.8992    | 27.79       | 0.7976        | 27.19         | 0.7651        | 25.41         | 0.8038        | 29.13         | 0.9109        |
| DPDPNN [19]     | ×3    | 31.56     | 0.9004    | 27.82       | 0.7991        | 27.13         | 0.7638        | 25.47         | 0.8057        | 29.23         | 0.9121        |
| RDN [8]         | ×3    | 31.91     | 0.9039    | 28.00       | 0.8017        | 27.38         | 0.7716        | 26.02         | 0.8204        | 29.91         | 0.9189        |
| RCAN [9]        | ×3    | 31.81     | 0.9026    | 27.93       | 0.7994        | 27.33         | 0.7677        | 25.98         | 0.8188        | 29.68         | 0.9165        |
| SRBFN [12]      | ×3    | 31.89     | 0.9033    | 27.96       | 0.8012        | 27.32         | 0.7686        | 25.85         | 0.8174        | 29.66         | 0.9172        |
| MoG-DUN (ours)  | ×3    | 32.36     | 0.9095    | 28.21       | 0.8059        | 27.64         | 0.7806        | 27.15         | 0.8483        | 30.65         | 0.9279        |
| EDSR [5]        | ×4    | 28.94     | 0.8528    | 25.98       | 0.7326        | 25.66         | 0.6961        | 23.47         | 0.7303        | 26.04         | 0.8567        |
| DPDPNN [19]     | ×4    | 28.94     | 0.8539    | 25.97       | 0.7341        | 25.57         | 0.6937        | 23.44         | 0.7309        | 25.96         | 0.8585        |
| RDN [8]         | ×4    | 29.24     | 0.8608    | 26.16       | 0.7383        | 25.86         | 0.7053        | 24.05         | 0.7527        | 26.59         | 0.8688        |
| RCAN [9]        | ×4    | 29.13     | 0.8580    | 26.18       | 0.7374        | 25.82         | 0.7002        | 23.95         | 0.7486        | 26.35         | 0.8643        |
| SRBFN [12]      | ×4    | 29.09     | 0.8583    | 26.07       | 0.7354        | 25.72         | 0.6996        | 23.71         | 0.7402        | 26.32         | 0.8648        |
| MoG-DUN (ours)  | ×4    | 29.65     | 0.8706    | 26.35       | 0.7448        | 25.98         | 0.7107        | 24.70         | 0.7744        | 26.99         | 0.8776        |

Fig. 9: SISR visual quality comparisons of different methods on 'Img_109' from Manga109 [52]. The degradation is directly downsampling with scale factor ×4.

Fig. 10: SISR visual quality comparisons of different methods on 'Img_039' from Urban100 [51]. The degradation involves Gaussian kernel with kernel width 1.3 and direct downsampling with scale factor ×3.
TABLE IV: Average PSNR results for multiple degradation on four benchmark datasets. The best performance is shown in bold.

| Method            | Kernel Width | Set5 [48] | Set14 [49] | BSD100 [50] | Urban100 [51] |
|-------------------|--------------|-----------|------------|-------------|--------------|
|                   | x2           | x3         | x4         | x2          | x3           | x4          |
| EDSR[5]           | 0.5          | 35.52      | 32.37      | 29.67       | 32.09        | 28.83       | 26.61       | 30.94       | 27.69       | 26.04       | 29.77       | 26.03       | 23.93       |
| SRMDNF[11]        | 0.5          | 35.01      | 31.30      | 28.83       | 31.11        | 27.98       | 26.04       | 30.12       | 27.14       | 25.55       | 27.66       | 24.76       | 22.93       |
| RDN[8]            | 0.5          | 36.46      | 32.72      | 30.00       | 32.07        | 28.87       | 26.94       | 30.95       | 27.82       | 26.31       | 29.89       | 26.90       | 24.49       |
| RCAN[9]           | 0.5          | 36.37      | 32.62      | 29.99       | 31.94        | 28.78       | 26.79       | 30.87       | 27.73       | 26.22       | 29.69       | 26.46       | 24.40       |
| SRFBN[12]         | 0.5          | 36.31      | 32.56      | 29.92       | 31.86        | 28.72       | 26.88       | 30.78       | 27.75       | 26.16       | 28.57       | 25.67       | 24.21       |
| MoG-DUN(ours)     | 0.5          | 38.60      | 32.94      | 30.13       | 32.42        | 29.01       | 26.97       | 31.09       | 28.01       | 26.41       | 30.53       | 27.04       | 24.75       |
|                   |              | 1.3        | 37.51      | 33.87      | 31.29        | 33.26       | 29.97       | 28.01       | 31.98       | 28.84       | 27.12       | 30.69       | 27.45       | 25.30       |
| SRMDNF[11]        | 1.3          | 36.67      | 32.58      | 30.17       | 31.89        | 29.24       | 27.31       | 31.00       | 28.32       | 26.71       | 28.60       | 26.20       | 24.30       |
| RDN[8]            | 1.3          | 37.45      | 34.26      | 31.65       | 32.31        | 30.27       | 28.36       | 32.05       | 29.05       | 27.35       | 30.94       | 28.24       | 25.97       |
| RCAN[9]           | 1.3          | 37.44      | 34.06      | 31.59       | 32.18        | 30.12       | 28.27       | 31.96       | 28.92       | 27.31       | 30.57       | 27.64       | 25.98       |
| SRFBN[12]         | 1.3          | 37.39      | 34.09      | 31.55       | 33.21        | 30.08       | 28.26       | 31.89       | 28.87       | 27.26       | 30.66       | 27.84       | 25.71       |
| MoG-DUN(ours)     | 1.3          | 38.11      | 34.51      | 32.00       | 33.87        | 30.40       | 28.46       | 32.19       | 29.15       | 27.38       | 31.73       | 28.42       | 26.22       |

Fig. 11: SISR visual quality comparisons of different methods on 'Img 092' from Urban100 [51]. The degradation involves Gaussian kernel with kernel width 2.6 and direct down-sampling with scale factor ×2.

IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 2359–2368.

IEEE Conference on Computer Vision and Pattern Recognition (CVPR’05), 2005.

IEEE Transactions on Computer Vision and Pattern Recognition, 2017, pp. 4346–4350.

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017.

IEEE Transactions on Image Processing, 2019.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018.

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018.

IEEE/ACM International Conference on Multimedia, 2018.

IEEE Transactions on Image Processing, 2019.

IEEE International Conference on Computer Vision and Pattern Recognition, 2020.

IEEE Transactions on Image Processing, 2019.

IEEE Transactions on Image Processing, 2013.

IEEE Transactions on Image Processing, 2020.

IEEE Transactions on Image Processing, 2018.

IEEE Transactions on Image Processing, 2016.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 2016.

IEEE Transactions on Image Processing, 2017.

IEEE Transactions on Image Processing, 2020.

IEEE Transactions on Image Processing, 2013.

IEEE Transactions on Image Processing, 2012.

IEEE Transactions on Image Processing, 2011.

IEEE Transactions on Image Processing, 2008.

IEEE Transactions on Image Processing, 2017.

IEEE Transactions on Image Processing, 2011.

IEEE Transactions on Image Processing, 2010.

IEEE Transactions on Image Processing, 2008.

IEEE Transactions on Image Processing, 2017.

IEEE Transactions on Image Processing, 2016.
connected convolutional networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[37] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3431–3440.

[38] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollar, “Learning to refine object segments,” in European conference on computer vision. Springer, 2016, pp. 75–91.

[39] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in International Conference on Medical image computing and computer-assisted intervention. Springer, 2015, pp. 234–241.

[40] Y. Zhao, Y. Shen, and J. Yao, “Recurrent neural network for text classification with hierarchical multiscale dense connections.” in IJCAI, 2019, pp. 5450–5456.

[41] Y. Tai, J. Yang, and X. Liu, “Image super-resolution via deep recursive residual network,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 3147–3155.

[42] P. Indyk and R. Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” in Proceedings of the thirtieth annual ACM symposium on Theory of computing, 1998, pp. 604–613.

[43] X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local neural networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 7794–7803.

[44] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, “Non-local recurrent network for image restoration,” in Advances in Neural Information Processing Systems, 2018, pp. 1673–1682.

[45] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in Sixth international conference on computer vision (IEEE Cat. No. 98CH36271). IEEE, 1998, pp. 839–846.

[46] K. Yue, M. Sun, Y. Yuan, F. Zhou, E. Ding, and F. Xu, “Compact generalized non-local network,” in Advances in Neural Information Processing Systems, 2018, pp. 6510–6519.

[47] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, and L. Zhang, “Ntire 2017 challenge on single image super-resolution: Methods and results,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2017, pp. 114–125.

[48] M. Bevilacqua, A. Roumy, C. Guilleminot, and M. L. Alberi-Morel, “Low-complexity single-image super-resolution based on nonnegative neighbor embedding,” in British Machine Vision Conference 2012, 2012, pp. 1–10.

[49] R. Zeyde, M. Elad, and M. Protter, “On single image scale-up using sparse-representations,” in Proceedings of the 7th international conference on Curves and Surfaces, 2010, pp. 711–730.

[50] D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, vol. 2, 2001, pp. 416–423.

[51] J.-B. Huang, A. Singh, and N. Ahuja, “Single image super-resolution from transformed self-exemplars,” in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 5197–5206.

[52] Y. Matsui, K. Ito, Y. Aramaki, A. Fujimoto, T. Ogawa, T. Yamasaki, and K. Aizawa, “Sketch-based manga retrieval using manga109 dataset,” Multimedia Tools and Applications, vol. 76, no. 20, pp. 21 811–21 838, 2017.

[53] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600–612, 2004.

[54] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[55] W. Han, S. Chang, D. Liu, M. Yu, M. Wirth, and T. S. Huang, “Image super-resolution via dual-state recurrent networks,” in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 1654–1663.

[56] R. Timofte, R. Rothe, and L. Van Gool, “Seven ways to improve example-based single image super resolution,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1865–1873.