Bayesian Image Super-Resolution With Deep Modeling of Image Statistics

Shangqi Gao and Xiahai Zhuang

Abstract—Modeling statistics of image priors is useful for image super-resolution, but little attention has been paid from the massive works of deep learning-based methods. In this work, we propose a Bayesian image restoration framework, where natural image statistics are modeled with the combination of smoothness and sparsity priors. Concretely, first we consider an ideal image as the sum of a smoothness component and a sparsity residual, and model real image degradation including blurring, downsampling, and noise corruption. Then, we develop a variational Bayesian approach to infer their posteriors. Finally, we implement the variational approach for single image super-resolution (SISR) using deep neural networks, and propose an unsupervised training strategy. The experiments on three image restoration tasks, i.e., ideal SISR, realistic SISR, and real-world SISR, demonstrate that our method has superior model generalizability against varying noise levels and degradation kernels and is effective in unsupervised SISR. The code and resulting models are released via https://zmiclab.github.io/projects.html.

Index Terms—Image super-resolution, variational inference, neural network, generative learning

1 INTRODUCTION

Single image super-resolution (SISR), aiming to recover high-resolution (HR) images from low-resolution (LR) observations, is a typical task of image restoration. Image restoration (IR) has many significant applications, such as low-level image processing [1], medical imaging [2], and remote sensing [3]. Thanks to the advance of deep learning, studying IR becomes more popular in computer vision. Particular efforts have been made to explore the end-to-end IR frameworks for many applications [4], [5], [6], [7], [8]. Although the approaches deliver promising performance on synthetic data, directly transferring them to real-world images often undergoes a great decrease in performance, meaning the resulting models could suffer from poor generalization ability. In reality, the ground truth of images is unavailable, and thus unsupervised learning is more challenging.

Current methods could be categorized into two groups, i.e., the model-based and the learning-based schemes [9]. Model-based IR represents image degradation as analytical or statistical models [10], [11], and it aims to restore a degraded image without using any further data. This problem is known as being ill-posed. Therefore, many image priors were proposed to model the domain knowledge of natural images, such as Gaussian priors [12], Markov random field (MRF) [13], sparsity priors [14], and low-rank priors [15]. Many of them could not perfectly model image priors due to the complex structure of real-world images. Therefore, modeling image structure is still active and challenging.

Learning-based IR aims to learn the mappings from degraded spaces to the original space [4], [5]. Deep neural networks (DNNs) are widely used to learn the mappings due to their powerful ability in modeling complex functions. One of such networks is the deep convolutional neural networks (CNNs), which were widely adopted in image denoising [5], [16], deblurring [17] as well as super-resolution [4], [6], and achieved promising performance. For example, the residual networks (ResNet) were first proposed for the task of classification [18], which were then successfully applied in SISR and achieved superior performance against previous works [6].

The great majority of SISR models trained on ideal data [4], [6], [19], e.g., synthesized by bicubic interpolation, cannot generalize well when LR images include noise. To rectify the weakness, one can model image priors explicitly, and then restore them via Bayesian inference. Bigdeli et al. [20] proposed to estimate image priors using pre-trained denoising autoencoders and restored images via maximum a posteriori (MAP). Their restoration problem was solved iteratively, which could be computationally expensive. The multivariate Gaussian prior was adopted to model clean images in the recent four denoising works, including self-supervised Bayesian image denoising [21], variational denoising network [22], blind universal Bayesian image denoising [23], and patch-based non-local Bayesian networks [24]. However, the methods cannot deal with the problem of SISR, since they did not involve blurring and downsampling in their modeling.

Many of SISR models were developed for supervised SISR [25], [26], [27], and thus cannot be used in real-world scenarios where the ground truth is unavailable. To tackle the difficulty, Shocher et al. [28] used the information of a single image itself for internal learning, but the method requires long inference time due to thousands of gradient
This framework infers variational posterior distributions given observations, and can restore stochastic images by randomly sampling from the resulting distributions.

- Second, we build the BayeSR embedded with downsampling, upsampling, and inferring modules for SISR. The downsampling module is aimed to learn image degradation; the upsampling module is developed to upscale image space; and the inferring module is built to infer the variational parameters of posteriors.

- Finally, we develop an unsupervised learning strategy of training BayeSR when only LR images are available, and extend it for pseudo-supervised and supervised learning if unpaired and paired HR images are provided, respectively. Moreover, we show the generalization ability and unsupervised performance of BayeSR via three tasks, i.e., ideal, realistic, and real-world SISR.

The rest of our paper is organized as follows. In Section 2, we introduce the related works about model-based and learning-based IR. Section 3 presents the framework of BayeSR, including the network architecture and the training strategies. Section 4 provides the implementation details of BayeSR and the experimental results on three SISR tasks. We finally conclude this work in Section 5.

2 RELATED WORKS

2.1 Model-Based Image Restoration

Conventional IR is based on mathematical and statistical models which are designed to model the domain knowledge of images [11], [13]. Both of them aim to explicitly model domain knowledge, and therefore are often referred to as model-based IR [9]. A typical image degradation model could be expressed as \( y = Av + n \), where, \( y \), \( A \), \( u \), and \( n \) respectively denote the degraded image, degradation operator, natural image, and the additive noise [35]. IR could be categorized into specific tasks based on the forms of \( A \). For example, \( A \) is an identity matrix for image denoising [34], a blurring operator for image deblurring [35], and a downsampling operator for SISR [36].

From a mathematical perspective, IR aims to solve an inverse problem, e.g., \( \min_u \| Av - y \|^2 + \lambda R(u) \), where, \( R(u) \) denotes a regularization term, and \( \lambda \) is a hyperparameter [10]. Many efforts have been made to explore appropriate regularization terms. Tikhonov et al. [10] proposed the classical regularization for solving ill-posed inverse problems, and showed its application in IR. Rudin et al. [11] introduced the total variation (TV) regularization to keep images piece-wisely smooth, and it was widely applied in image denoising [34], [37], [38]. Figueiredo et al. [39] and Chan et al. [40] explored the sparsity of images based on wavelet transform, and demonstrated the effectiveness of sparsity regularization in reconstructing HR images. Kolchinskii et al. [41] and Candès et al. [42] showed the low-rank property of images, and developed efficient algorithms of recovering low-rank matrix. These methods solve inverse problems iteratively, which can be computationally expensive for large-scale images. Besides, manually selecting regularization parameters can be a practical issue.
From a statistical perspective, IR aims to infer the distribution of an image, \( u \), given an observation, \( y \), by maximizing the posterior probability \( p(u|y) \propto p(y|u)p(u) \), where, \( p(u) \) represents the prior knowledge of images [13]. Many works have been done to model image priors. Hunt et al. [12] used Gaussian prior to keep images smooth. Qin et al. [13] introduced Markov random field (MRF) to preserve the edges of textural images. Molina et al. [43] proposed a hierarchical Bayesian approach to model the structural form of the noise and local characteristics of images. After that, they introduced the compound Gaussian MRF [44] to model the multichannel image prior. Jalobeanu et al. [35] proposed the inhomogeneous Gaussian MRF to model the spatially variant characteristics of real satellite images. Pan et al. [45] developed the Huber-MRF to preserve the edges of images and improved the computational efficiency. Guerrero et al. [46] proposed the space-variant Gaussian scale mixtures to provide an effective local statistical description of images. Babacan et al. [47] adopted TV prior to describe statistical characteristics of images, and used a hierarchical Bayesian model to estimate the hyperparameter of the prior. Ayasso et al. [48] adopted the Markovian prior and Student’s-t prior to model the smooth part and point sources of astrophysical images, respectively. Many of these models are iteratively solved, which can be computationally expensive for large-scale images. However, they have the advantages of sampling a stochastic restoration instead of a deterministic one and quantifying the uncertainty of restorations.

### 2.2 Learning-Based Image Restoration

Modern IR aims to learn mappings from degraded image spaces to the original image space via dictionaries [49] or neural networks [4], [5], [6]. Different from the conventional IR, the methods use data for learning, and therefore are referred to as learning-based IR [9].

Many works have been done to learn deterministic mappings, i.e., the outputs of IR models are deterministic [4], [5]. In image denoising, Burger et al. [50] adopted a multi-layer perceptron and achieved comparable performance with the conventional methods. Zhang et al. [5], [16] trained CNN-based residual networks, and their method delivered a promising performance in removing Gaussian noise. Lehtinen et al. [51] only used noisy image pairs to train networks without requiring clean targets. Krull et al. [52] developed a blind-spot masking scheme to train networks using a single noisy image. Ulyanov et al. [30] proposed to directly extract image prior by randomly initializing CNNs, and then used the deep image prior for unsupervised denoising. Batson et al. [53] proposed a self-supervised method for blind denoising by exploiting noise independence between pixels. Chen et al. [54] proposed to first estimate the distribution of noise from noisy images by GAN, and then to generate noise samples to construct paired training data. In SISR, Yang et al. [36], [55] proposed to learn the sparse representation of images patches via dictionaries, and the resulting model showed good performance in reconstructing details. Dong et al. [4] developed three-layer CNNs to super-resolve images, and the resulting models delivered much better performance than the conventional methods. Following with this, deep neural networks, such as residual networks [6], [56], [57], [58], recursive networks [19], dense networks [7], [59], and pyramid networks [8], were studied to improve the Peak Signal-to-Noise Ratio (PSNR) value of SR images. Besides, Ledig et al. [26] developed a super-resolution GAN (SRGAN) and included perceptual loss [60] for training, which could improve the visual quality of SR images. Wang et al. [61] further enhanced the performance of SRGAN by improving its network architecture. Recently, Chen et al. [62] developed an image processing transformer by introducing self-attention, which delivers superior performance in image denoising and SISR.

To be the best of our knowledge, limited works have been reported to learn stochastic mappings, i.e., the outputs of IR models could be random samples [27]. Bigdeli et al. [20] built a Bayesian deep learning framework using a deep mean-shift prior, but the approach of restoring images is iterative and can be computationally expensive. Laine et al. [21] proposed a self-supervised Bayesian denoising framework using the multivariate Gaussian prior. Yue et al. [22] developed a variation denoising network using the conjugate Gaussian prior. Helou et al. [23] built a blind image denoiser using the Gaussian prior and a fusion network architecture. Izadi et al. [24] developed non-local Bayesian networks using the multivariate Gaussian prior and the non-local mean filtering. These denoising methods cannot deal with the problem of SISR, since the blurring and down-scaling are not involved in their modeling. Recently, Lugmayr et al. [27] explored the SR space using normalizing flow to reconstruct diverse SR images given an observation. However, the method does not explicitly model image priors, and therefore the description of statistical characteristics is unclear. Different from deterministic learning, stochastic learning could produce diverse restorations from an observation by random sampling, which may follow the property of ill-posed inverse problems that the number of solutions could be infinite.

### 3 Methodology

This work is aimed to build a Bayesian image restoration framework, and implement it by DNNs for SISR. Image restoration is particularly challenging when only a few degraded and noisy observations are available. To tackle the difficulty, we first impose on smoothness and sparsity priors to describe statistical characteristics of the original images, and then estimate the smoothness component and the sparsity residual by MAP. Although the iterative variational Bayesian approaches could be used to infer the posteriors [47], [48], [63], they are computationally expensive, due to many steps of iteration for SR images with large size. Motivated by the advance of deep learning which has great potential for real-time SISR, in this work we develop a Bayesian image super-resolution method via deep modeling of image priors.

For convenience, raw tensor data of images are vectorized in this paper, unless stated otherwise. Fig. 2a shows the probabilistic graphical model, which is also known as Bayesian belief network, of modeling an observation \( y \). Concretely, \( y \) can be modeled as the composition of a smoothness component \( x \), a sparsity residual \( z \), a Gaussian noise \( n \), and a deterministic downsampling operator \( A \), where the sum of \( x \) and
z is considered as the restoration of y, denoted as u. Besides, x depends on a variable of spatial correlation v, z depends on sparsity precision w, and n depends on mean m and noise strength σ. Fig. 2b shows the pipeline of Bayesian image super-resolution. To be specific, we first develop DNNs to infer the variational posterior distributions of x, z, and m. For example, μ̂ and ̂ denote the mean and standard deviation of the variational Gaussian distribution of x. We further explicitly compute the variational parameters of v, w, and σ. For instance, μ̂ denotes the mean of the variational Gamma distribution of v. Finally, we sample a smoothness component and a sparsity residual following their variational posterior distributions, and the sum of them is considered as a restoration of y.

Table 1 summarizes the notions and notations used in this paper. Besides, ||·||1 denotes the ℓ1 norm of vectors; ||·||2 denotes the ℓ2 norm of vectors; ||·||M, where M is a symmetric positive definite matrix, denotes the M-norm of vectors, i.e., ||x||M = 1Mx1M; and ⟨·, ·⟩ denotes the inner-product of vectors. The rest of this section is organized as follows. We specify the graphical model of modeling image degradation in Section 3.1, and develop the approach of inferring variational distributions in Section 3.2. After that, we interpret the variational loss in Section 3.3. Section 3.4 illustrates the details of building neural networks. Section 3.5 describes the training and test strategies.

### TABLE 1

| Notion                  | Notation                                                                 |
|-------------------------|---------------------------------------------------------------------------|
| Scalar                  | lowercase letter, e.g., a                                                 |
| Vector                  | boldface lowercase letter, e.g., a                                        |
| Matrix                  | boldface capital letter, e.g., A                                           |
| Observation/Reference   | y ∈ ℝd, u ∈ ℝd, x ∈ ℝd, z ∈ ℝd, n ∈ ℝd, v ∈ ℝd, m ∈ ℝd, w ∈ ℝd, σ ∈ ℝd, w ∈ ℝd |
| Restoration             | x denotes a variable of spatial smoothness; z, n                        |
| Smoothness component    | x denotes another variable of spatial sparsity; n                        |
| Sparsity residual       | m denotes a deterministic downsampling matrix related to a convolutional kernel k ∈ ℝd and a downsampling factor s. For example, Ax equals to the vectorization of (X * K) ⊥ for SISR, where, X and K are the matrix forms of x and k, respectively, and ⊥ (s > 1) denotes a downsampling operator. Next, we will select detailed statistical models for these variables. |
| Gaussian noise          | Ax equals to the vectorization of (X * K) ⊥ for SISR, where, X and K are the matrix forms of x and k, respectively, and ⊥ (s > 1) denotes a downsampling operator. Next, we will select detailed statistical models for these variables. |
| Spatial correlation w.r.t. x | Ax equals to the vectorization of (X * K) ⊥ for SISR, where, X and K are the matrix forms of x and k, respectively, and ⊥ (s > 1) denotes a downsampling operator. Next, we will select detailed statistical models for these variables. |
| Sparsity precision w.r.t. z | Ax equals to the vectorization of (X * K) ⊥ for SISR, where, X and K are the matrix forms of x and k, respectively, and ⊥ (s > 1) denotes a downsampling operator. Next, we will select detailed statistical models for these variables. |
| Mean/Strength w.r.t. n | m denotes a deterministic downsampling matrix related to a convolutional kernel k ∈ ℝd and a downsampling factor s. For example, Ax equals to the vectorization of (X * K) ⊥ for SISR, where, X and K are the matrix forms of x and k, respectively, and ⊥ (s > 1) denotes a downsampling operator. Next, we will select detailed statistical models for these variables. |
| Mean/Deviation of VDs   | μ/σ                                                                    |
| Normal/Gamma distribution | N(⟨·, ·⟩/G(·, ·)                                                   |
| Hyperparameters         | s, k, μ0, σ0, φ, γ, λ, r                                               |

Here, VDs denote variational distributions.

### 3.1 Statistical Modeling of Image Degradation

#### 3.1.1 Formulation of Degradation

Modeling smoothness and sparsity is crucial to IR. In real-world, noise is inevitably introduced by imaging systems. Therefore, denoising could be a basic task. Estimating the piece-wisely smooth components based on the TV prior has shown to be effective in denoising [34], but image details can be missed. Recent works showed that the sparsity prior has the potential of capturing more details [63]. Motivated by this, we propose to infer the smoothness component and sparsity residual of images for restoration.

Suppose corrupted observations are sampled from some variable y ∈ ℝd, where d denotes the dimension of y, and clean images are sampled from a variable, u*, where u* ∈ ℝd, then the degradation process of images could be models as,

$$
y = A(x + z) + n,
$$

where, x ∈ ℝd denotes a variable of smoothness prior; z ∈ ℝd represents another variable of sparsity prior; n ∈ ℝd is a Gaussian noise; and A ∈ ℝd×d denotes a deterministic downsampling matrix related to a convolutional kernel k ∈ ℝd and a downsampling factor s. For example, Ax equals to the vectorization of (X * K) ⊥ for SISR, where, X and K are the matrix forms of x and k, respectively, and ⊥ (s > 1) denotes a downsampling operator. Next, we will select detailed statistical models for these variables.

#### 3.1.2 Modeling of Priors in Detail

The observation likelihood of y can be expressed as

$$
p(y|A, x, z, m, σ) = N(y|A(x + z) + m, \text{diag}(σ)^{-1}).
$$

Here, we model n as a spatially-variant Gaussian noise with a mean m ∈ ℝd and a variance diag(σ)^{-1} ∈ ℝd×d, namely,

$$
p(n|m, σ) = N(n|m, \text{diag}(σ)^{-1}).
$$

Moreover, we assign Gaussian prior to m and Gamma prior to σ, i.e.,

$$
p(m|μ0, σ0) = N(m|μ0, σ0^{-1}I),
$$

$$
p(σ|φ, γ) = \prod_{i=1}^{d} \Gamma(σ|φ_i, γ_i),
$$

where, I denotes an identity matrix; μ0, σ0, φ, γ are user-defined hyperparameters, and Γ(·, ·) denotes Gamma distribution.
To account for the piecewise smoothness of $x$, we adopt the TV or Markovian prior which could be expressed as follows,

$$p(x|u) = \mathcal{N}(x|0, |D_h^T \text{diag}(v) D_h + D_v^T \text{diag}(v) D_v|^{-1}).$$  \hspace{1cm} (6)

where, $D_h$ and $D_v$ denote the finite-difference matrix in the horizontal and vertical directions, respectively, and $v$ is a variable describing the spatial correlation of $x$, which follows the Gamma prior,

$$p(v|\phi_v, \gamma_v) = \prod_{i=1}^{d_v} \mathcal{G}(v_i|\phi_v, \gamma_v).$$  \hspace{1cm} (7)

where, $\phi_v$ and $\gamma_v$ are hyperparameters.

To account for the sparsity of $z$, we adopt the Student’s $t$ prior which could be obtained by marginalizing a three-parameter Normal-Gamma distribution as follows,

$$p(z|\phi_z, \gamma_z) = \int_{\mathbb{R}^{d_z}} p(z|\omega)p(\omega|\phi_z, \gamma_z) d\omega = \prod_{i=1}^{d_z} \int_{\mathbb{R}} \mathcal{N}(z_i|0, \omega_{i1}^{-1}) \mathcal{G}(\omega_i|\phi_z, \gamma_z) d\omega_i, \hspace{1cm} (8)$$

where, $\phi_z$ and $\gamma_z$ are hyperparameters; and $\omega \in \mathbb{R}^{d_z}$ is a variable of conducting the sparsity precision of $z$, which follows the Gamma prior,

$$p(\omega|\phi_z, \gamma_z) = \prod_{i=1}^{d_z} \mathcal{G}(\omega_i|\phi_z, \gamma_z).$$  \hspace{1cm} (9)

Specially, $p(z_i|\omega) = \mathcal{N}(z_i|0, \omega_{i1}^{-1}) \mathcal{G}(\omega_i|\phi_z, \gamma_z)$ is known as Normal-Gamma distribution. One can refer to Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2022.3163307, of Appendices for the details of the sparsity prior.

Fig. 2a illustrates the architecture of a probabilistic graphical model, which represents the observation $y$ as the composition of a deterministic linear operator $A$ and three variables $n$, $x$, and $z$, where the downsampling operator $A$ is determined by the blur kernel $k$ and the downsampling factor $s$. Moreover, the noise $n$ depends on the mean $m$ and the variance $\text{diag}(\rho)^{-1}$, where $m$ is related to the Gaussian hyperparameters $\mu_0$ and $\sigma_0$, and $\rho$ is related to the Gamma hyperparameters $\phi_\rho$ and $\gamma_\rho$: the smoothness component $x$ depends on the spatial correlation $v$, where, $v$ is related to the Gamma hyperparameters $\phi_v$ and $\gamma_v$; and the sparsity residual $z$ depends on the sparsity precision $\omega$, where, $\omega$ is related to the Gamma hyperparameters $\phi_\omega$ and $\gamma_\omega$. Next, we will estimate the distributions of these variables given $y$ via variational Bayesian inference.

### 3.2 Variational Inference of Posterior Distributions

Our aim is to infer the distributions of latent variables given an observation $y$, i.e., to estimate the posterior distribution of each variable in $\psi = \{m, \rho, x, v, z, \omega\}$. One could compute the posteriors via the Bayesian rule, i.e., $p(\psi|y) \propto p(y|\psi)p(\psi)$, and the marginalization, which is however intractable since some of the variables are conditionally dependent. To tackle the difficulty, we propose to use the variational Bayesian (VB) approach. The VB method approximates $p(\psi|y)$ via a variational posterior distribution $q(\psi)$. Generally, the variables in $\psi$ are often enforced to be independent, namely,

$$q(\psi) = q(m)q(\rho)\prod_{i=1}^{d_x} q(x_i)q(v)\prod_{i=1}^{d_z} q(z_i)q(\omega).$$  \hspace{1cm} (10)

One method of obtaining the variational approximations is to minimize the Kullback-Leibler (KL) divergence between $q(\psi)$ and $p(\psi|y)$, as follows,

$$\hat{q}(\psi) \in \arg \min_{\hat{q}(\psi)} \text{KL}(q(\psi)||p(\psi|y)).$$  \hspace{1cm} (11)

Since we assigned the conjugate priors [64] to all variables, the variational posterior approximations of the marginal distributions of $m$, $\rho$, $x$, $v$, $z$, and $\omega$ could be successively expressed as follows,

$$\hat{q}(m) = \mathcal{N}(m|\mu_m, \text{diag}(\sigma_m^2)),$$  \hspace{1cm} (12)

$$\hat{q}(\rho) = \prod_{i=1}^{d_x} \mathcal{G}(\rho_i|\bar{\beta}_m, \bar{a}_m),$$  \hspace{1cm} (13)

$$\hat{q}(x) = \mathcal{N}(x|\mu_x, \text{diag}(\bar{\sigma}_x^2)),$$  \hspace{1cm} (14)

$$\hat{q}(v) = \prod_{i=1}^{d_v} \mathcal{G}(v_i|\bar{\beta}_v, \bar{a}_v),$$  \hspace{1cm} (15)

$$\hat{q}(z) = \mathcal{N}(z|\mu_z, \text{diag}(\sigma_z^2)),$$  \hspace{1cm} (16)

$$\hat{q}(\omega) = \prod_{i=1}^{d_\omega} \mathcal{G}(\omega_i|\bar{\beta}_\omega, \bar{a}_\omega),$$  \hspace{1cm} (17)

where, $\mu$, $\sigma$, $\bar{\alpha}$ and $\bar{\beta}$ respectively denote the parameters of the variational distributions to be further computed, and the variational posteriors $\hat{q}(\cdot)$ in (12)-(17) are corresponding to the priors $p(\cdot)$ in (4), (5), (6), (7), (8), and (9).

**Algorithm 1. Training and Test of BayeSR**

**Input:** Training and test datasets

**Output:** A stochastic restoration $u$

1: if Preliminary stage then
2: Estimate the downsampling $A$ from training data.
3: Generate a pool of noise patches from training data.
4: end if
5: if Unsupervised training stage then
6: Freeze the parameters of the downsampling module.
7: Stop the back propagation through $\mu_\omega$, $\mu_\omega$, and $\mu_\rho$.
8: while not up to total training steps do
9: Sample patches $y_i$ and $u_i^\omega$ from training data.
10: Sample patches $n_i$ from noise pool.
11: Generate pseudo degradations $y_i^\omega$ via $A u_i^\omega + n_i$.
12: Update the parameters of BayeSR by (28).
13: Update the parameters of $D_x$ and $D_y$ by (32).
14: end while
15: end if
16: if Test stage then
17: Sample an LR image $y$ from test data.
18: Infer the variational distribution of $\hat{q}(x)$ and $\hat{q}(z)$.
19: Sample $x$ and $z$ from their variational distributions.
20: return $u = x + z$
21: end if

In practice, we do not directly compute the KL divergence, but convert it to an easily derived formula,

$$\text{KL}(\hat{q}(\psi)||p(\psi|y)) = \mathbb{E}[\log \hat{q}(\psi)] - \mathbb{E}[\log p(\psi|y)]$$

$$= \mathbb{E}[\log \hat{q}(\psi)] - \mathbb{E}[\log p(\psi, y)] + \log p(y),$$  \hspace{1cm} (18)

where, all expectations are taken with respect to $\hat{q}(\psi)$, and the evidence $p(y)$ only depends on the priors. This formula shows that minimizing KL divergence is equivalent to
TABLE 2
Computational Details and Interpretation of the Variational Terms in (27)

| Notation | Formula | Adaptive weight | Interpretation |
|----------|---------|-----------------|---------------|
| \( \mathcal{L}_y \) | \( \frac{1}{2} \| y - A(x + z) - m \|_{M_y}^2 \) | \( M_y = \text{diag}(\hat{\mu}_y) \) | Ensures the consistency between observations and restorations |
| \( \mathcal{L}_{\hat{\mu}_y} \) | \( \frac{1}{2} \| D_1 \hat{\mu}_y \|_{M_{\hat{\mu}_y}}^2 + \| D_2 \hat{\mu}_y \|_{M_{\hat{\mu}_y}}^2 \) | \( M_{\hat{\mu}_y} = \text{diag}(\hat{\mu}_y) \) | Encourages \( \hat{\mu}_y \) to be piece-wise smooth |
| \( \mathcal{L}_{\sigma_y} \) | \( \frac{1}{2} \left[ (4\hat{\sigma}_y, \hat{\sigma}_y) - (1, \log(\hat{\sigma}_y^2)) \right] \) | - | Prevents \( \hat{\sigma}_y(x) \) from degrading to a one-point distribution |
| \( \mathcal{L}_{\hat{\sigma}_y} \) | \( \frac{1}{2} \| \hat{\sigma}_y \|_{M_{\hat{\sigma}_y}}^2 \) | \( M_{\hat{\sigma}_y} = \text{diag}(\hat{\sigma}_y) \) | Encourages \( \hat{\sigma}_y \) to be sparse |
| \( \mathcal{L}_{\hat{\mu}_m} \) | \( \frac{2}{\alpha_m} \| \hat{\mu}_m \|_{2} \) | - | Prevents \( \hat{\mu}_m(z) \) from degrading to a one-point distribution |
| \( \mathcal{L}_{\sigma_m} \) | \( \frac{1}{2} \left[ (\sigma_m, \sigma_m) - (1, \log(\sigma_m^2)) \right] \) | - | Constrains the energy of \( \hat{\mu}_m(z) \). |

Here, 1 denotes a vector with all elements to be ones.

\[
\min_{\Theta} \mathbb{E}[\log \tilde{q}(\psi)] - \mathbb{E}[\log p(\psi, y)] = \min_{\Theta} \text{KL}(\tilde{q}(\psi)||p(\psi)) - \mathbb{E}[\log p(y|\psi)].
\]

The second term of (19) could be expressed as

\[
-\mathbb{E}[\log p(y|\psi)] = -\mathbb{E}_q[\mathbb{E}_q[\log p(y|\psi)]].
\]

Since directly computing \( \mathbb{E}_q[\log p(y|\psi)] \) is difficult, we adopt the widely used reparameterization technique [65]. Concretely, let \( \epsilon \) denote white Gaussian noise sampled from \( \mathcal{N}(0, 1) \), then we have \( x = \hat{x} + \epsilon + \hat{\mu}_m \), \( z = \hat{z} + \epsilon + \hat{\mu}_m \), and \( m = \sigma_m \epsilon + \hat{\mu}_m \), where \( \hat{\mu}_m \) denotes the element-wise multiplication. Moreover, we consider \( \log p(y|\psi) \) as an approximation of \( \mathbb{E}_q[\log p(y|\psi)] \), and therefore the Formula (20) could be converted to

\[
-\mathbb{E}_q[\mathbb{E}_q[\log p(y|\psi)] = -\mathbb{E}_q[\log p(y|\psi)].
\]

Finally, we infer variational posteriors by optimizing the following problem,

\[
\min_{\Theta} \mathbb{E}_q[\log p(y|\psi)] - \mathbb{E}_q[\log p(y|\psi)].
\]

3.3 Interpretation of the Objective Function

In this section, we decompose the objective function in (22) into computational details according to the modeling variables for intuitive interpretation. The variational posteriors of \( u, \omega, \) and \( \rho \) can be explicitly formulated using that of \( x, z, \) and \( m \). Concretely, the first term of (22) could be expressed as,

\[
\text{KL}(\tilde{q}(\psi)||p(\psi)) = \text{KL}(\tilde{q}(x||p(x||p(\psi))) + \text{KL}(\tilde{q}(z||p(z||p(\omega))) + \text{KL}(\tilde{q}(m||p(m))) + \text{KL}(\tilde{q}(\rho||p(\rho))).
\]

Minimizing (23a), related to \( x, u, \) and \( v \), could induce the formula of computing \( \hat{\mu}_y, \) namely,

\[
\hat{\mu}_y = \frac{\alpha_u}{\beta_u} = \frac{2\gamma_u + 1}{(D_1\hat{\mu}_y)^2 + (D_2\hat{\mu}_y)^2 + 4\sigma_x^2 + 2\Phi_u},
\]

where, the operations in the above formula are element-wise.

Minimizing (23b), related to \( z \) and \( \omega, \) results in the formula of computing \( \hat{\mu}_\omega, \)

\[
\hat{\mu}_\omega = \frac{\alpha_\omega}{\beta_\omega} = \frac{2\gamma_\omega + 1}{\hat{\mu}_z^2 + \sigma_z^2 + 2\Phi_\omega}.
\]

Finally, minimizing (23c) and the second term of (22), related to \( m \) and \( \rho, \) leads to the formula of computing \( \hat{\mu}_\rho, \) as follows,

\[
\hat{\mu}_\rho = \frac{\alpha_\rho}{\beta_\rho} = \frac{2\gamma_\rho + 1}{(y - A(x + z) - m)^2 + 2\Phi_\rho}.
\]

The variational posteriors of \( x, z, \) and \( m \) can be inferred from \( y, \) given \( \hat{\mu}_y, \hat{\mu}_\omega, \hat{\mu}_\rho, \) \( \mu_0 = 0, \) and \( \sigma_\omega. \) Concretely, the formulas in (22) induces a variational loss function with respect to \( \{\hat{\mu}_y, \hat{\sigma}_y\}, \{\hat{\mu}_\omega, \hat{\sigma}_\omega\}, \{\hat{\mu}_\rho, \hat{\sigma}_\rho\} \) as follows,

\[
\mathcal{L}_{\text{var}}(y) = \mathcal{L}_y + \mathcal{L}_{\hat{\mu}_y} + \mathcal{L}_{\hat{\sigma}_y} + \mathcal{L}_{\hat{\mu}_\omega} + \mathcal{L}_{\hat{\sigma}_\omega} + \mathcal{L}_{\hat{\mu}_\rho} + \mathcal{L}_{\hat{\sigma}_\rho}.
\]

The computational details and interpretation of each term are summarized in Table 2. Note that the variational loss in (27) is a derivation from (22). Therefore, all these terms are adaptively balanced by MAP. This is different from conventional regularization methods, which use multiple terms and thus require to manually set the balancing weights for different terms. For details of the derivation of Formulas (24)-(27), please refer to Appendix C, available in the online supplemental material, of Appendices.

3.4 Deep Learning of Variational Parameters

We develop deep neural networks to implement the Bayesian image restoration framework described in Section 3.2 for SISR. In practice, the posterior parameters cannot be explicitly formulated since solving the resulting nonlinear equations is intractable, as shown in Appendix B, available in the online supplemental material, of Appendices. The previous work showed that iterative VB algorithms could be applied to tackle the difficulty [47], but they are computationally expensive due to the need for many iterations on high-dimensional parameters. Thanks to the promising performance of DNNs in learning non-linear mappings and the efficient platforms of deploying DNNs in parallel, we build deep neural networks to achieve Bayesian image super-resolution.

Fig. 3 illustrates the architecture of BayeSR, which mainly consists of three types of modules, i.e., CNN, upsampling,
and downampling. The CNN module could be designed using the backbone of ResNet [25] or UNet [66]. For ResNet, the upsampling module comprises two convolutional layers for \( s = 1 \), and one (two) transpose convolutional layer(s) followed by two convolutional layers for \( s = 2, 3 \) (\( s = 4 \)). For UNet, since we have adopted bilinear interpolation to upscale its inputs, the transpose convolutional layers of the upsampling module will be removed. The downampling module, which is removed for \( s = 1 \), consists of six convolutional layers, and the strides of the last layer are equal to \( 2 \). In our experiments, \( \text{CNN}_m \) will be fixed as ResNet, while \( \text{CNN}_c \) and \( \text{CNN}_z \) can be either ResNet or UNet.

The three modules, i.e., \( \text{CNN}_m \), \( \text{CNN}_c \), and \( \text{CNN}_z \), are successively developed to estimate the distribution parameters of \( \hat{q}(m), \hat{q}(z), \) and \( \hat{q}(x) \). Given an observation \( y \), we first use \( \text{CNN}_m \) followed by two convolutional layers to estimate \( \mu_m \) and \( \sigma_m \) from \( y \). Then, we compute the residual \( y - m \), and use \( \text{CNN}_c \) followed by an upsampling module to infer \( \mu_z \) and \( \sigma_z \) from the residual. Finally, we downsample \( z \) by a downampling module, which is developed to implement the downampling operator \( A \), and compute another residual \( y - m - Az \). Similarly, we use \( \text{CNN}_z \) followed by another upsampling module to estimate \( \mu_z \) and \( \sigma_z \) from the residual. Once these parameters have been estimated from the observation, we could explicitly compute \( \mu_{\omega}, \mu_{\rho}, \) and \( \mu_p \) via the Formulas (24)-(26), respectively. Note that the distribution parameters \( \mu \) and \( \sigma \) are feature maps parameterized by the network parameters, \( \theta_G \), of BayeSR.

3.5 Training and Test Strategies

3.5.1 Preliminary Stage

We pre-train the downampling operator \( A \) before training BayeSR. If the ground truth \( u^i \) of an observation \( y \) is available, we will train the downampling module via minimizing MSE, \( \frac{1}{N} \sum_{i=1}^{N} \| Au^i - y^i \|^2 \), where \( N \) denotes the number of training samples. Otherwise, we will adopt KernelGAN [67] to train the module. Concretely, we discriminate the patch distributions between observations \( y \) and their degradations \( Ay \) via a discriminator, to make the downampling module learn image degradation from \( y \). Once the downampling module is pre-trained, its parameters will be fixed in the following training.

We extract noise patches from observations before training BayeSR. Similar to the noise block extraction in [54], if the mean and variance of any sub-patch, \( y^p_i \), of an observation \( y \), satisfy \( \| \text{mean}(y_i) - \text{mean}(y^p_i) \| \leq 0.05 \cdot \| \text{mean}(y_i) \| \) and \( \| \text{var}(y_i) - \text{var}(y^p_i) \| \leq 0.1 \cdot \| \text{var}(y_i) \| \), we will add \( n_i = y_i - \text{mean}(y_i) \) into the pool of noise patches, noted as \( S_n = \{ n_i \} \).

3.5.2 Unsupervised Training

BayeSR could be training by combining generative learning (GL), discriminative learning (DL), and generative adversarial learning (GAL). Concretely, training BayeSR by GL induces a variational loss notated as \( L_{\text{var}}(\theta_G) \): training BayeSR by DL induces a self-supervised loss notated as \( L_{\text{self}}(\theta_G) \), and training BayeSR by GAL induces a generative loss notated as \( L_{\text{gen}}(\theta_G) \). Therefore, our unsupervised strategy of training BayeSR is

\[
\min_{\theta_G} L_{\text{var}}(\theta_G) + \tau L_{\text{self}}(\theta_G) + \lambda L_{\text{gen}}(\theta_G),
\]

where \( \tau \) and \( \lambda \) are hyperparameters. The details of this strategy are showed as follows.

BayeSR could be trained via GL when only LR images are available. Suppose \( y \) and \( u^i \) are two randomly cropped patches from LR images, we could generate a pseudo degradation from \( u^i \), i.e., \( y^i = Au^i + n_i \), where \( n_i \) denotes a sample from \( S_n \). After that, we consider the concatenation of \( y \) and \( y^i \) as an input, and infer the distortion parameters as shown in Fig. 3. Finally, we compute the variational loss as shown in (27) for \( y \) and \( y^i \), and the resulting loss of training BayeSR is

\[
L_{\text{var}}(\theta_G) = \frac{1}{N} \sum_{i=1}^{N} \left[ L_{\text{var}}(y_i) + L_{\text{var}}(y^i) \right].
\]

BayeSR could be trained via DL when the observation likelihood \( p(u^i|y^i, \theta_G) \) is given. If \( p(u^i|y^i, \theta_G) = N(x^i + x^i_0, 1) \), maximum log-likelihood will induce the squared \( \ell_2 \) norm, \( \| u^i - x^i - z^i_0 \|^2 \), where \( x^i_0 \) and \( z^i_0 \) are smoothness component and sparsity residual parameterized by \( \theta_G \). If \( p(u^i|y^i, \theta_G) = \prod_{r=1}^{R} \mathcal{L}(x^i_{2r} + z^i_{2r-1}) \), where \( \mathcal{L} \) denotes the Laplace distribution, maximum log-likelihood will induce the \( \ell_1 \) norm, \( \| u^i - x^i - z^i_0 \|_1 \). Overall, the self-supervised loss of training BayeSR can be expressed as

\[
L_{\text{self}}(\theta_G) = \frac{1}{N} \sum_{i=1}^{N} \| u^i - x^i - z^i_0 \|_p^p.
\]
where, $p = 2$ if the downsampling factor $s$ equals to 1, and $p = 1$ otherwise. Note that the “self” means we use the LR image dataset itself for discriminative learning, instead of using an LR image itself for internal learning [28].

BayeSR could be trained via GAL. Concretely, we use a discriminator, referred to as $D_u$, to discriminate the patch distributions between the restoration $x_i^r + z_i^r$ and the reference $u_i^r$. Moreover, we use another discriminator, referred to as $D_y$, to discriminate the patch distributions between $A(x_i^r + z_i^r)$ and $Au_i^r$. Therefore, a generative loss of training BayeSR could be expressed as,

$$
\mathcal{L}_{\text{gen}}(\theta_G) = \frac{1}{N} \sum_{i=1}^{N} \log [1 - D_u(x_i^r + z_i^r)] \\
+ \frac{1}{N} \sum_{i=1}^{N} \log [1 - D_y(A(x_i^r + z_i^r))]. 
$$

(31)

Besides, the discriminator $D_u$ and $D_y$ are trained by

$$
\max_{\theta_u} \frac{1}{N} \sum_{i=1}^{N} [\log D_u(u_i^r) + \log [1 - D_u(x_i^r + z_i^r)]] \\
\max_{\theta_y} \frac{1}{N} \sum_{i=1}^{N} [\log D_y(Au_i^r) + \log [1 - D_y(A(x_i^r + z_i^r))]] 
$$

(32)

where, $\theta_u$ and $\theta_y$ denote the parameters of $D_u$ and $D_y$, respectively.

### 3.5.3 Pseudo-Supervised and Supervised Training

BayeSR could be trained via pseudo-supervised learning, if unpaired LR and HR images are available. Suppose $y_i$ and $u_i^{hr}$ are two randomly cropped patches from LR and HR images, respectively, then we generate a pseudo degradation $y_i^{hr}$ from $u_i^{hr}$ using the same strategy as the unsupervised case, and replace $y_i^{hr}$ with $y_i^{hr}$ to compute the losses for GL, DL, and GAL. The only difference is that the loss for DL becomes a pseudo-supervised one noted as $\mathcal{L}_{\text{pseudo}}(\theta_G)$, instead of the self-supervised loss. Therefore, the pseudo-supervised strategy of training BayeSR is

$$
\min_{\theta_G} \mathcal{L}_{\text{var}}(\theta_G) + \tau \mathcal{L}_{\text{pseudo}}(\theta_G) + \lambda \mathcal{L}_{\text{gen}}(\theta_G). 
$$

(33)

BayeSR could be trained via supervised learning, if paired LR and HR images are available. Suppose $y_i$ is randomly cropped patches from LR images, and $u_i$ is its ground truth, then we replace $y_i^{hr}$ and $u_i^{hr}$ of the unsupervised case with $y_i$ and $u_i$, and compute the losses for GL and DL. Being different from the unsupervised case, $\mathcal{L}_{\text{var}}(\theta_G)$ is computed only for $y_i$, and the loss for DL becomes a supervised one noted as $\mathcal{L}_{\text{sup}}(\theta_G)$. Therefore, the supervised strategy of training BayeSR is

$$
\min_{\theta_G} \mathcal{L}_{\text{var}}(\theta_G) + \tau \mathcal{L}_{\text{sup}}(\theta_G). 
$$

(34)

### 3.5.4 Test Stage

In the test stage, we could obtain many HR restorations from one LR observation by the proposed BayeSR. Concretely, an LR image $y_i$ (not vectorized) is fed into BayeSR, and the distribution parameters, $\{\mu_x, \sigma_x\}$, w.r.t. the smoothness component $x$ and the distribution parameters, $\{\mu_z, \sigma_z\}$, w.r.t. the sparsity residual $z$ are inferred from $y_i$ for evaluating the performance of restoration, we directly consider $\mu_x + \mu_z$ as the deterministic restoration of $y$. For quantifying the diversity of restorations, we repeatedly sample $x$ and $z$ from their variational distributions for 10 times to generate a set of stochastic restorations $\{u_i = x_i + z_i\}_{i=1}^{10}$.

### 4 Experiments

In this section, we first performed preliminary studies to obtain proper architectures and settings for the proposed BayeSR, and to interpret the functionality of BayeSR. After that, we validated the generalization ability of BayeSR, and evaluated the unsupervised performance using three tasks, i.e., ideal SISR, realistic SISR, and real-world SISR.

#### 4.1 Implementation Details

Three datasets were used to train BayeSR, i.e., DIV2K, Flickr2K, and DPED, thanks to their high-resolution (2K) and diversity. DIV2K$^1$ was first released from the NTIRE 2017 challenge on SISR, which consists of 800 training images, 100 validation images, and 100 test images. Flickr2K$^2$ consists of 2650 diverse HR images, whose clean HR images were used as the unpaired references for pseudo-supervised training. DPED$^3$ consists of photos taken synchronously in the wild by three smartphones and one professional camera. We used the DPED-iPhone from the NTIRE 2020 challenge for training and test. This dataset consists of 5614 training images, 113 validation images, and 100 test images. For ideal SISR, we used the bicubic DIV2K where LR images were synthesized via bicubic interpolation. For realistic SISR, we utilized the mild DIV2K where LR images were corrupted by unknown Poisson noise and random shifts. For real-world SISR, we adopted the DPED-iPhone where LR images were corrupted by real noise.

Seven metrics were used to evaluate the performance of SISR, including five full-reference image quality assessments (IQA), i.e., the standard Peak Signal to Noise Ratio (PSNR) in HR space, the Structural Similarity (SSIM) index, the PSNR in LR space (LRPSNR), the Learned Perceptual Image Patch Similarity (LPIPS), and the Diversity (Div.) Score, and two no-reference IQAs, i.e., the Natural Image Quality Evaluator (NIQE) and the Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE). To evaluate the performance of BayeSR in deterministic restoration, we obtained a restoration $\tilde{\mu}_x + \tilde{\mu}_z$, as shown in Section 3.5.4, from an observation $y_i$ and computed the PSNR, SSIM, LRPSNR, NIQE, and BRISQUE, where LRPSNR was computed between the degradation $A(\tilde{\mu}_x + \tilde{\mu}_z)$ and the observation $y$. To evaluate the performance of BayeSR in stochastic restoration, we obtained 10 stochastic restorations as shown in section 3.5.4, and computed the LPIPS of each restoration to calculate the average LPIPS and Div. Score. For the task of ideal SISR, being consistent with the previous works [4], we converted the super-resolved RGB images to YCbCr image, and computed PSNR and SSIM only on the Y channel by ignoring $s + 4$ pixels from boundaries, where $s$ denotes the downsampling factor. For the task of realistic

---

1. https://data.vision.ee.ethz.ch/cvl/DIV2K/
2. http://cv.snu.ac.kr/research/EDSR/Flickr2K.tar
3. http://people.ee.ethz.ch/ihnatova/index.html
SISR, being consistent with the evaluation criterion of NTIRE 2018 SR challenge on realistic SR, we computed maximal PSNR and SSIM by cropping a 60 × 60 patch from the center of an RGB image and shifting it up to 40 pixels in four directions. Since the unknown random shifts between LR images and their references make the computation of LPIPS and Div. Score inaccurate, we adopted NIQE and BRISQUE instead of LPIPS and Div. Score to evaluate models. For the task of real-world SISR, we only adopted LRPCSNR, NIQE, and BRISQUE to evaluate models, since the ground truth of degraded images is inaccessible.

For downsampling modules, the architecture is the same as KernelGAN. For other modules, the kernel size of convolutional layers (Convs) is 5 × 5, and that of transpose Convs is 5 × 5. The discriminator $D_{\ell}$ consists of four Convs followed with batch normalization (BN) and Leaky ReLU, and one Conv as the output layer. The kernel sizes of five Convs are 4. The strides of the first three Convs are 2, and that of the last two Convs are 1. The numbers of kernels of five Convs are 64, 128, 256, 512, and 1, respectively. The discriminator $D_{\ell}$ has the same structures as $D_{\ell}$, except that only the strides of the first Conv are 2.

For the graphical model, as shown in Fig. 2a, the elements of $y_{\ell}$ and $y_{\ell-1}$ were set to 2; the elements of $\phi_{\ell}$, and $\phi_{\ell-1}$ were set to $10^{-3}$; the elements of $\phi_{\ell}$ were set $10^{-5}$ in (28) and (33), and that were set to $10^{-3}$ in (34). In the training stage, the hyperparameters $\tau$ and $\lambda$ in (28) were set to 1 and $10^{-3}$, respectively. Besides, we adopted the ADAM optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-8}$ to train BayeSR. The total training steps were set to be $1 \times 10^6$. The initial learning rate was set to be $1 \times 10^{-4}$, and it was decreased by a factor of 0.5 every $2 \times 10^5$ updates. BayeSR was implemented with TensorFlow, and all models were trained and tested on a TITAN RTX GPU with 24 GB memory.

### 4.2 Preliminary Study

In this section, we studied appropriate settings for BayeSR and interpreted the functionality of BayeSR.

#### 4.2.1 Degradation Kernel Study

To study network architectures, we trained BayeSR by using the supervised strategy as shown in Section 3.5.1, and obtained a standard kernel noted as $k_{\text{Bicubic}}$. After that, we used the unsupervised strategy to train the module on the task of ideal, realistic, and real-world SISR, and obtained three kernels noted as $k_{\text{IdSR}}$, $k_{\text{ReSR}}$, and $k_{\text{RWSR}}$, respectively. We visualized the four kernels in Fig. 4. This figure shows that $k_{\text{IdSR}}$ is very similar to $k_{\text{Bicubic}}$, which demonstrates the effectiveness of KernelGAN in estimating degradation kernels. Moreover, the realistic kernel is similar to the ideal kernel, but the real-world one is greatly different, which shows the challenges and necessity of estimating degradation kernels for real-world images.

#### 4.2.2 Ablation Study

To study basic modules, we fixed the depth of CNN modules, CNN$_{\ell}$, and CNN$_{\ell-1}$ to be $d_n = 2$, $d_\ell = 2$, and $d_{\ell-1} = 2$, and used the supervised strategy in (34) to train BayeSR on the task of realistic SISR by setting the basic modules to be different structures, as shown in Table 3. We first trained two models to test the effect of whether to use BN or not. After that, we trained an additional model to test the effect of using RCAB. The comparisons between model #1 and #2 show that using BN does not improve the performance of BayeSR, and therefore we remove BN in the following studies. The comparisons between model #2 and #3 show that using RCAB does not improve the performance of BayeSR, but we adopt RCAB in the following studies, since the weight of skip connection in RCAB is learnable while that in ResBlock is fixed to 0.2.

To study network architectures, we trained BayeSR by increasing its depth and using different backbones. We first trained two models to test the effect of increasing the depth of CNNs from 4 to 8. Then, we trained an additional model, i.e., model #6, to test the effect of using different networks. The comparisons among model #3, #4, and #5 show that deeper models deliver better performance. Table 3 shows that model #5 built with ResNet is lighter in terms of the number of parameters and computationally cheaper according to the training and test time. Moreover, model #5 delivers the best performance among supervised models, and thus its settings are used in the following supervised BayeSR models.

To study generative adversarial learning (GAL), we trained four groups of unsupervised models by using different strategies, and each group contained two models with different backbones. First, we trained a group of models without using any GAL strategies. Then, we trained the second group of models, i.e., model #9 and #10, using the original GAL strategy, which is known as generative adversarial networks (GAN). Besides, we trained the third group of models, i.e., model #11 and #12, using the strategy of Wasserstein GAN. Finally, we trained the fourth group of models, i.e., model #13 and #14, using the strategy of the least square GAN (LSGAN). The comparisons between model #7 and #8 show that model #7 using ResNet delivers better performance in SISR and is computationally more efficient, which is consistent with the supervised case. As the same as
the first group, the internal comparisons of other groups confirm that the architecture of ResNet is more appropriate for our framework. Moreover, the external comparisons among four groups demonstrate that model #9 with GAN delivers the best performance in SISR among all unsupervised models.

To study generative learning (GL) and discriminative learning (DL), we trained two models using different learning methods. First, we trained model #15 by GL, i.e., minimizing the variation loss \( \mathcal{L}_{\text{var}} \) in (28). Then, we trained model #16 by DL, i.e., minimizing the self-supervised loss \( \mathcal{L}_{\text{self}} \) in (28). The comparisons among model #7, #15, and #16 show that training BayeSR by combining GL and DL could obtain particularly better PSNR and SSIM values. Moreover, model #9 shows that training BayeSR by combining GL, DL, and GAL could further improve PSNR and SSIM values. Thus, we adopt the settings of model #9 for unsupervised learning in the following sections.

4.2.3 Interpretation of BayeSR

Fig. 5 visualizes the posteriors inferred by model #15, #16, and #7 in Table 3, respectively denoted as GL, DL, and GL+DL for convenience. Note that \( \mathbf{n}, \mathbf{x}, \) and \( \mathbf{z} \) are sampled from their variational distributions, e.g., \( \mathbf{n} \sim \mathcal{N}(\mathbf{m}, \text{diag}(\mathbf{\mu}_n^{-1})) \) and \( \mathbf{m} \sim \mathcal{N}(\mathbf{m}_z, \text{diag}(\sigma_m^2)). \) Besides, we normalize all inferences by first taking absolute values and then being divided by their maximal values, except for \( \mathbf{x}, \mathbf{\mu}_x, \) and \( \mathbf{u}. \)

To understand the advantage of combining generative learning (GL) and discriminative learning (DL), we compared the posteriors of GL and DL in Fig. 5. One can see that \( \mathbf{z} \) and \( \mathbf{x} \) of GL are prone to be sparse and smooth, respectively, this is because we explicitly modeled the priors of \( \mathbf{z} \) and \( \mathbf{x} \) by (8) and (6). However, since the noise \( \mathbf{n} \) of corrupting \( \mathbf{y} \) was not properly estimated, maximizing the observation likelihood of \( \mathbf{y} \), as shown by \( \mathcal{L}_y \) in (27), induced a particularly small error \( \mathbf{e} \) in LR space, and thus the restoration of GL contains a lot of artifacts induced by noise. By

Fig. 5. Visualization of variational posteriors inferred by three typical models in Table 3, i.e., #15 (GL), #16 (DL), and #7 (GL+DL). The five columns split by dotted lines represent LR observations, posteriors w.r.t. the noise \( \mathbf{n}, \) the sparsity residual \( \mathbf{z}, \) and the smoothness component \( \mathbf{x}, \) and outputs. Here, \( \mathbf{e} = \mathbf{y} - \mathbf{A}(\mathbf{x} + \mathbf{z}) \) denotes the residual error in LR space, GL represents generative learning, and DL is discriminative learning. Please zoom in the online electronic version for more details.

| Model | Network | BasicBlock | BN \((d_m, d_z, d_x)\) | Strategy | GL/DL/GAL | PSNR | SSIM | NIQE | BRISQUE | #Para | Training time | Test time |
|-------|---------|------------|----------------|---------|----------|------|-----|------|--------|-------|--------------|-----------|
| #1    | ResNet  | ResBlock   | Y \((2, 2, 2)\) | Sup     | Y/Y/N    | 23.83| 0.5455| 9.26 | 66.78  | 1.29M | 1.55h        | 5.18s     |
| #2    | ResNet  | RCAB       | N              |         |          | 23.84| 0.5452| 9.26 | 66.66  | 1.29M | 1.68h        | 5.15s     |
| #3    | RCAB    |            | N              |         |          | 23.83| 0.5452| 9.31 | 66.72  | 1.29M | 1.68h        | 5.78s     |
| #4    | ResNet  | RCAB       | N \((4, 4, 4)\) |         | Y/Y/N    | 23.92| 0.5491| 9.20 | 67.13  | 1.74M | 1.77h        | 6.03s     |
| #5    | ResNet  | RCAB       | (8, 8, 8)     |         |          | 24.10| 0.5560| 7.88 | 66.79  | 2.63M | 2.35h        | 6.21s     |
| #6    | U-Net   | ConvBlock  | (8, 6, 6)     |         |          | 23.92| 0.5510| 8.93 | 63.05  | 4.59M | 2.95h        | 8.57s     |
| #7    | ResNet  | RCAB       | (8, 8, 8)     | Unsup   | Y/Y/N    | 23.43| 0.5284| 7.23 | 59.32  | 2.63M | 3.23h        | 6.61s     |
| #8    | U-Net   | ConvBlock  | (8, 6, 6)     |         |          | 23.01| 0.5063| 8.13 | 59.46  | 4.59M | 3.47h        | 8.04s     |
| #9    | ResNet  | RCAB       | N \((8, 8, 8)\) |         | Y/Y/GAN  | 23.67| 0.5334| 7.40 | 57.83  | 2.63M | 5.38h        | 7.22s     |
| #10   | U-Net   | ConvBlock  | (8, 6, 6)     |         |          | 23.45| 0.5138| 7.40 | 57.84  | 2.63M | 4.59h        | 7.58s     |
| #11   | ResNet  | RCAB       | N \((8, 8, 8)\) |         | Y/Y/WGAN | 23.41| 0.5134| 5.06 | 22.78  | 2.63M | 5.18h        | 6.17s     |
| #12   | U-Net   | ConvBlock  | (8, 6, 6)     |         |          | 23.12| 0.4956| 4.55 | 17.81  | 4.59M | 5.58h        | 8.40s     |
| #13   | ResNet  | RCAB       | N \((8, 8, 8)\) |         | Y/Y/LSGAN| 23.54| 0.5224| 7.53 | 29.72  | 2.63M | 5.38h        | 7.22s     |
| #14   | U-Net   | ConvBlock  | (8, 6, 6)     |         |          | 23.36| 0.5083| 4.96 | 14.55  | 4.59M | 3.47h        | 8.04s     |
| #15   | ResNet  | RCAB       | N \((8, 8, 8)\) |         | Y/N/N    | 21.74| 0.4707| 8.09 | 67.73  | 2.63M | 3.35h        | 6.61s     |
| #16   | N/N/N   |            | (8, 8, 8)     |         |          | 22.20| 0.4849| 8.14 | 56.29  | 2.63M | 2.55h        | 6.50s     |

Here, the depths of CNN\(_m\), CNN\(_z\), and CNN\(_x\) are denoted as \((d_m, d_z, d_x)\). The bold font indicates the optimal settings in each of the sub-studies, while the underline font denotes the best model across the sub-studies.
contrast, e of DL is prone to approximate the noise of corrupting y, since we minimized the distance between the restoration and its reference by the self-supervised loss $L_{self}$ in (28). However, $z$ and $x$ of DL are feature maps with unknown statistics, since we did not explicitly model their priors. Moreover, the restoration of DL is prone to be oversmooth. Therefore, we trained BayeSR by combining GL and DL. That could generate the smooth $x$, the sparse $z$, and the best restoration $u$, as shown in Fig. 5.

To understand why BayeSR can produce interpretable components, we explained its functionality based on the posteriors of GL+DL in Fig. 5. $L_{\mu_x}$ in (27) quantifies the smoothness of $\mu_x$ weighted by $\mu_x$. The visualized results show that small values of $\mu_x$ are aligned to boundaries of $\mu_x$, while large values are aligned to smooth areas. Therefore, minimizing $L_{\mu_x}$ could produce a piece-wisely smooth $x$ with sharp edges; $L_{\mu_x}$ quantifies the sparsity of $\mu_z$ weighted by $\mu_z$. The visualized results show that small values are aligned to pixels of $\mu_z$ representing image details, while large values are aligned to pixels located in smooth areas. Therefore, minimizing $L_{\mu_z}$ could generate a sparse $z$ including image details; $L_y$ quantifies the weighted error by $\mu_y$ and $\text{diag}(\mu_y)^{-1}$ denoted the variance of $n$. The comparisons between $\mu_y$ and $e$ show that small values are aligned to large errors, in other words, strong noise is used to approximate a large error that could not be fitted by $A(x + z)$. Therefore, minimizing $L_y$ could generate a spatially variant noise $n$.

To understand why BayeSR could generate diverse restorations, we analyzed its uncertainties based on the posteriors of GL+DL in Fig. 5. Concretely, small values of $\sigma_y^2$ correspond to the smooth areas of $x$, while large values correspond to rich textures. Therefore, $\sigma_y^2$ represents the uncertainty of smoothness, namely, a large value of $\sigma_y^2$ indicates the pixel is more likely to be located in a non-smooth area of $x$. Similarly, $\sigma_y^2$ represents the uncertainty of sparsity, namely, a large value of $\sigma_y^2$ indicates the pixel of $z$ is more likely to be non-zero. Using the two uncertainty maps, one can generate diverse stochastic restorations. Moreover, $\mu_y^{-1}$ represents the uncertainty of observations, namely, a large value of $\mu_y^{-1}$ indicates the pixel of $y$ is more likely to be corrupted by strong noise.

4.3 Explicit Modeling and Generalization Ability
This section studies the robustness of explicit modeling via training BayeSR as an auto-encoder, and shows the generalization ability of BayeSR by supervised learning.

4.3.1 Robustness of Explicit Modeling
To study the effect of explicit modeling of priors, we trained BayeSR to be an auto-encoder. Concretely, we first set up BayeSR using the same settings as model #5 in Table 3 Section 4.2, but removed the unnecessary upsampling and downsampling modules. Then, we fed clean HR patches (of size $128 \times 128$) from DIV2K to BayeSR, and trained it using (34). For comparisons, we set up a baseline using the same architecture, but the network was trained without using the variational loss, $L_{vari}$ in (34), i.e., without explicit prior modeling. Finally, we tested the performance of baseline and BayeSR on the public BSDS68 [71]. Although the baseline and BayeSR were trained only using clean images, the test images were corrupted by adding white Gaussian noise (AWGN) with the noise level ranging in $[0, 20]$. Fig. 7 shows the curves of PSNR and SSIM of the two models. One can see that BayeSR achieves lower scores than the baseline when the noise level is smaller than 2, but significantly higher PSNR and SSIM values when the noise level is bigger than 4. This demonstrates that explicit modeling of priors could improve the robustness of BayeSR against unseen noise,
### 4.3.2 Generalization Ability

To study the generalization ability of BayeSR, we trained BayeSR on the task of ideal SISR ×4 via supervised learning. In the training stage, we first randomly cropped HR patches \( u_i \) (of size 128 × 128) and LR patches \( y_i \) (of size 32 × 32) from the bicubic DIV2K to generate the data \( \{ y_i, u_i \}_{i=1} \) for supervised training, as shown in Section 3.5.3. Then, we set up BayeSR using the same settings as model #5 in Table 3 Section 4.2, and initialized the downsampling module using the pre-trained model with respect to \( k_{\text{Bicubic}} \). Finally, we froze the downsampling module, and fed randomly selected batches (of size 4) to train BayeSR up to 1 × 10^6 steps. This BayeSR supervisedly trained using (34) was referred to as S-BayeSR. For comparisons, we trained a supervised baseline model referred to as S-Baseline, which had the same network architecture, but only minimized the supervised loss \( L_{\text{sup}} \) without \( L_{\text{var}} \) in (34).

In the test stage, we used four public datasets, i.e., Set5 [69], Set14 [70], BSDS100 [71], and Urban100 [72], to evaluate the performance of S-Baseline and S-BayeSR, and comparisons with six supervised methods, i.e., SRCNN [4], VDSR [57], LapSRN [8], EDSR [6], RCAN [25], and OISR [68], in terms of PSNR, SSIM, LRPSNR, LPIPS, and Div. Score. Besides, we tested the methods on noisy datasets, which were corrupted by the AWGN with the noise level of \( \sigma = 10, 20 \), to show the generalization ability of them.

Table 4 summarizes the results on the task of supervised ideal SISR ×4. The results in the cases of \( \sigma = 10 \) and \( \sigma = 20 \) show that S-BayeSR significantly outperforms the compared models in PSNR, SSIM, and LPIPS, which confirms that the explicit modeling of image priors could improve the generalization ability. The comparisons between S-Baseline and S-BayeSR in the case of \( \sigma = 0 \) show that the imperfect modeling of image priors, i.e., by the combination of smoothness and sparsity priors, decreases the PSNR, SSIM, and LRPSNR values of S-BayeSR, but increases its LPIPS. Due to the uncertainties of S-BayeSR as shown in section 4.2.3, it could generate diverse stochastic restorations instead of a deterministic reconstruction, and thus achieves non-zero Div. Scores. Fig. 6 visualizes three typical examples. This figure shows that S-BayeSR maintains better local-similarity in \( \sigma = 10 \) and \( \sigma = 20 \), thanks to the explicit modeling of image priors.

#### TABLE 4

| \( \sigma \) Method | #Paras | Set5 | Set14 | BSDS100 | Urban100 | Div. Score |
|-----------------|--------|------|-------|---------|----------|------------|
|                  |        | PSNR | SSIM | LRPSNR | LPIPS | PSNR | SSIM | LRPSNR | LPIPS | Score |
| 0 Bicubic        | -      | 28.42 | 0.8105 | 35.62 | 0.3357 | 26.10 | 0.7048 | 34.62 | 0.4320 | 25.96 | 0.6676 | 35.86 | 0.5175 | 23.15 | 0.6579 | 32.73 | 0.4677 |
| SRCNN 0.07M      | 30.49  | 0.8629 | 40.74 | 0.1954 | 27.61 | 0.7535 | 40.37 | 0.3096 | 26.91 | 0.7104 | 41.99 | 0.4041 | 24.53 | 0.7230 | 39.67 | 0.3123 |
| VDSR 0.67M       | 31.35  | 0.8838 | 40.14 | 0.1798 | 28.02 | 0.7678 | 40.90 | 0.3002 | 27.28 | 0.7250 | 42.85 | 0.3920 | 25.18 | 0.7523 | 39.95 | 0.2730 |
| LapSRN 0.90M     | 31.52  | 0.8854 | 41.19 | 0.1813 | 28.08 | 0.7687 | 41.02 | 0.3014 | 27.30 | 0.7253 | 42.89 | 0.3947 | 25.20 | 0.7544 | 40.12 | 0.2731 |
| EDSR 43.1M       | 32.46  | 0.8767 | 42.89 | 0.1707 | 28.80 | 0.7872 | 42.66 | 0.2742 | 27.72 | 0.7414 | 43.91 | 0.3613 | 26.64 | 0.8029 | 41.22 | 0.2040 |
| RCAN 15.6M       | 36.08  | 0.8991 | 42.93 | 0.1692 | 28.71 | 0.7851 | 42.56 | 0.2727 | 27.75 | 0.7426 | 43.79 | 0.3569 | 26.81 | 0.8079 | 41.53 | 0.1953 |
| OISR 44.3M       | 32.51  | 0.8993 | 42.85 | 0.1698 | 28.88 | 0.7872 | 42.69 | 0.2755 | 27.75 | 0.7423 | 43.93 | 0.3617 | 26.78 | 0.8066 | 41.48 | 0.2027 |
| S-Baseline 2.63M | 32.07  | 0.8923 | 42.94 | 0.1731 | 28.38 | 0.7764 | 42.66 | 0.2841 | 27.51 | 0.7336 | 43.85 | 0.3757 | 25.98 | 0.7802 | 41.58 | 0.2322 |
| S-BayeSR 2.63M   | 31.30  | 0.8850 | 40.92 | 0.1223 | 28.08 | 0.7561 | 38.15 | 0.2229 | 27.21 | 0.7091 | 38.36 | 0.3216 | 25.10 | 0.7528 | 37.01 | 0.2366 |

We test all methods on Set5 [69], Set14 [70], BSDS100 [71], and Urban100 [72], and report the average PSNR (1), SSIM (1), LRPSNR (1), LPIPS (1), and Div. Score (1). Here, Div. Score is only used to indicate whether a model is deterministic or stochastic. The bold value denotes the best performance, and the italic value represents the second-best performance.
Baseline, we refer to unsupervised BayeSR as U-BayeSR, pseudo-supervised BayeSR as Ps-BayeSR, and unsupervised baseline as U-Baseline to avoid confusion.

### 4.4.1 Ideal Image Super-Resolution

To study the unsupervised performance of BayeSR in ideal SISR × 4, we trained BayeSR on the bicubic DIV2K. In the training stage, we first randomly cropped large LR patches \( u_{lr} \) (of size 128 × 128) and small LR patches \( y_{lr} \) (of size 32 × 32) from the LR images of the bicubic DIV2K. Then, we obtained pseudo LR patches \( y'_{lr} \) from \( u_{lr} \) using the strategy as shown in Section 3.5.2, and generated the data \( \{y_i, y'_{lr}, u_{lr}\}_{i=1}^{N} \) for unsupervised training. Besides, we set up BayeSR using the same settings as model #9 in Table 3 Section 4.2, and initialized the downsampling module of BayeSR using the pre-trained model with respect to \( k_{PSISR} \). For comparisons, we first trained U-Baseline by minimizing the self-supervised loss \( \mathcal{L}_{self} \) in (28). After that, we trained U-BayeSR using (28).

In the test stage, we included two unsupervised methods, i.e., ZSSR [28] and MZSR [29], trained via internal learning for comparisons, and four supervised methods, i.e., EnhanceNet [6], SRGAN [26], ESRGAN [61], and SRFlow [27], oriented by perceptual quality for reference. Moreover, we used the same test datasets and criteria as the previous section to evaluate the performance of compared methods.

Table 5 summarizes the results on the task of ideal SISR × 4. U-BayeSR achieves the best PSNR, SSIM, and LPIPS in all studies, and gets the second-best LRPSNR. Besides, U-BayeSR outperforms U-Baseline in 12 studies (out of 16), which shows that train BayeSR by combining GL, DL, and GAL is more effective. Moreover, U-Baseline outperforms ZSSR and MZSR in all studies, which means the self-supervised learning on the LR dataset could be better than the internal learning on a single LR image. Fig. 8 visualizes three typical examples. This figure shows that U-BayeSR could restore more image details than the other unsupervised models. Also, the supervised models could generate some image artifacts, while U-BayeSR maintains better local similarity due to the explicit modeling of image priors.

### 4.4.2 Realistic Image Super-Resolution

To study the unsupervised performance of BayeSR in realistic SISR × 4, we trained BayeSR on the mild DIV2K. For unsupervised training, we adopted the similar strategy as the ideal SISR to generate training data \( \{y_i, y'_{lr}, u_{lr}\}_{i=1}^{N} \) from the LR images of the mild DIV2K. For pseudo-supervised training, we cropped HR patches \( u_{hr} \) (of size 128 × 128) from the HR images of the mild DIV2K to generate training data \( \{y_i, y'_{lr}, u_{lr}\}_{i=1}^{N} \) using the strategy as shown in Section 3.5.3. For supervised training, we cropped the ground truth of \( y_i \) from the HR images of the mild DIV2K to generate training data \( \{y_i, u_{lr}\}_{i=1}^{N} \). In the training stage, we first set up BayeSR using the same settings as model #9 (5) in Table 3 Section 4.2 for unsupervised or pseudo-supervised (supervised) training, and initialized its downsampling module using the pre-trained model with respect to \( k_{PSISR} \). After that, we trained U-Baseline and U-BayeSR as the ideal SISR, and trained Ps-BayeSR using (33). For reference, we trained EDSR [6], WDSR [58], RCAN [25], and S-Baseline by minimizing \( \mathcal{L}_{sup} \) in (34), and trained S-BayeSR using (34). In the test stage, since the test dataset of the mild DIV2K is not public, we evaluated the performance of all methods on the validation dataset by PSNR, SSIM, LRPSNR, NIQE, and BRISQUE. Note that higher LRPSNR does not mean better performance for realistic SISR, since LR images were corrupted by noise.

Table 6 summarizes the quantitative results on the task of realistic SISR × 4. Ps-BayeSR achieves the best performance in terms of PSNR and SSIM, and U-BayeSR gets the second best. Besides, U-BayeSR significantly outperforms the U-Baseline in PSNR and SSIM, which shows that training BayeSR by combining GL, DL, and GAL is better than only by DL. Among transferred models, S-BayeSR achieves the highest PSNR and SSIM values, due to its better generalization ability as shown in Section 4.3.2. Compared with other methods, the transferred models achieve higher LRPSNR, and thus more noise artifacts are included in their restoration. Fig. 9 visualizes three typical examples. This figure shows that U-BayeSR and Ps-BayeSR are prone to produce images with fewer noise artifacts, while U-Baseline generates color artifacts. Among the transferred models, S-BayeSR maintains better local
similarity due to the explicit modeling of image priors. Although SRFlow achieves the best NIQE and BRISQUE in Table 6, it generates more noisy artifacts, as one can observe from exemplar cases in Fig. 9.

4.4.3 Real-World Image Super-Resolution

To study the unsupervised performance of BayeSR in real-world SISR $\times 4$, we trained BayeSR on the DPED-iPhone. In the training stage, we used a similar strategy as the realistic SISR to train U-Baseline, U-BayeSR, and Ps-BayeSR, except for replacing the LR dataset and degradation kernel with DPED-iPhone and $k_{RWSR}$, respectively. In the test stage, we evaluated the performance of all models on the test dataset of DPED-iPhone by reporting LRPSNR, NIQE, and BRISQUE. Here, EDSR [6], RCAN [25], SRGAN [26], ESRGAN [61], and SRFlow [27] were transferred from ideal SISR, while RealSR [73] was trained on real-world SISR. Note that we mainly evaluated the visual quality of restorations, due to the lack of ground truth.

Table 7 summarizes the results on the task of real-world SISR $\times 4$. Due to the lack of ground truth, we combine the quantitative results with visualized examples to evaluate each model. Fig. 10 shows three typical examples from the test dataset of DPED-iPhone. Compared with the transferred models, U-BayeSR and Ps-BayeSR could generate clean images with more details. Besides, U-BayeSR outperforms U-Baseline in qualitative and quantitative results, which means training BayeSR by combining GL, DL, and GAL is more effective. Since RealSR was oriented by perceptual quality, it could generate more details than U-BayeSR oriented by PSNR, but some of them are fake. For example, the letters “ts” are inaccurately super-resolved by RealSR in the second row. Similarly, RealSR could generate unrealistic branches and wheels in the first and third rows, respectively. In contrast, U-BayeSR was trained by the

![Fig. 8. Visualization on the task of ideal SISR $\times 4$: three typical examples from Set14, B100, Urban100, respectively. The red boundary denotes the supervised method, while the green boundary represents the model trained without ground truth. Please refer to Supplementary Material, available online, for high-resolution images.](image-url)

### Table 6

| Model         | PSNR | SSIM | LRPSNR | NIQE | BRISQUE | # Paras |
|---------------|------|------|--------|------|---------|---------|
| HR            | $\infty$ | 1    | 18.63  | 3.07 | 14.66   | –       |
| Bicubic       | 23.16 | 0.5178 | 38.98  | 8.20 | 62.65   | –       |

Model transferred from ideal SISR

| Model | PSNR | SSIM | LRPSNR | NIQE | BRISQUE | # Paras |
|-------|------|------|--------|------|---------|---------|
| EDSR  | 22.83 | 0.4958 | 44.46  | 7.32 | 58.56   | 43.1M   |
| RCAN  | 22.84 | 0.4962 | 44.51  | 7.39 | 57.45   | 15.6M   |
| SRFlow| 21.41 | 0.3688 | 43.38  | 7.40 | 56.29   | 2.63M   |
| S-BayeSR | **23.09** | **0.5090** | **40.29** | **7.40** | **60.71** | **2.63M** |

Model trained without ground truth

| Model               | PSNR | SSIM | LRPSNR | NIQE | BRISQUE | # Paras |
|---------------------|------|------|--------|------|---------|---------|
| U-Baseline          | 22.20 | 0.4849 | 27.44  | 8.14 | 56.29   | 2.63M   |
| Ps-BayeSR           | **23.86** | **0.5422** | **32.84** | **8.28** | **61.18** | **2.63M** |
| U-BayeSR            | 23.67 | 0.5334 | 33.37  | 7.40 | 57.83   | 2.63M   |

Supervised model for reference

| Model   | PSNR | SSIM | LRPSNR | NIQE | BRISQUE | # Paras |
|---------|------|------|--------|------|---------|---------|
| EDSR    | 24.38 | 0.5800 | 24.44  | 7.85 | 61.35   | 43.1M   |
| WDSR    | 24.45 | 0.5824 | 24.08  | 7.88 | 61.89   | 9.9M    |
| RCAN    | 24.55 | 0.5831 | 25.13  | 8.09 | 63.77   | 15.6M   |
| S-Baseline | 24.55 | 0.5827 | 25.97  | 8.15 | 63.49   | 2.63M   |
| S-BayeSR | 24.10 | 0.5560 | 26.15  | 8.78 | 66.79   | 2.63M   |

We test all methods on the validation dataset of the mild DIV2K, and report the average PSNR ($\uparrow$), SSIM ($\uparrow$), LRPSNR, NIQE, and BRISQUE. The bold value denotes the best PSNR and SSIM for the models transferred from ideal SISR or trained without ground truth, and the italic value represents the second-best performance.
maximum likelihood of observations, and thus could generate restorations more consistent with LR images. Overall, real-world SISR is still challenging due to diverse degradation and lack of reliable references.

5 DISCUSSION

In this section, we discuss the generalizability of BayeSR to diverse noise levels and degradation kernels, and present our perspective regarding BayeSR in real-world applications. Similar to Section 4.4, we refer to supervised BayeSR as S-BayeSR, pseudo-supervised BayeSR as Ps-BayeSR, unsupervised BayeSR as U-BayeSR, and unsupervised Baseline as U-Baseline.

5.1 Generalizability to Diverse Noise

This study investigates the performance of BayeSR when there is difference between the pre-extracted noise and true noise in the training stage. Concretely, we simulated real-world camera sensor noise by a signal-dependent Gaussian distribution [74], i.e., \( N(\mu, \sigma_r^2 + \sigma_s^2 y_i) \), where, \( \sigma_r \) and \( \sigma_s \) respectively denote the levels of read noise and shot noise [74], and \( y_i \) denotes the \( i^{th} \) pixel of an image \( y \). After that, we degraded HR images of DIV2K by bicubic interpolation, and added the Gaussian noise to generate realistic LR images, with \( \sigma_r \) and \( \sigma_s \) uniformly ranging in \([0; 25]\) and \([0; 8]\), respectively. The resulting LR and HR image pairs were used to train S-BayeSR. Besides, we adopted the same strategy as shown in Section 3.5.1 to extract pseudo noise from the real-world dataset DPED-iPhone to ensure the difference of distributions between the extracted noise and Gaussian noise. Moreover, we used the same strategy showed in Section 3.5.3 (and Section 3.5.2) to generate pseudo LR images for training Ps-BayeSR (and U-Baseline and U-BayeSR) by degrading the HR images from Flickr2K (the realistic LR images generated from DIV2K) with bicubic interpolation and the pre-extracted pseudo noise. Due to the diversity of Gaussian noise, we increased each element of the hyperparameters, \( \gamma \), which controls the shape of Gamma prior, for BayeSR from 2 to 8 to ensure flatter-shaped Gamma distributions. Finally, we trained U-Baseline, Ps-BayeSR, U-BayeSR, and S-BayeSR using the same settings as shown in Section 4.4.2.

In the test stage, we degraded the HR images from Set5, Set14, BSD100, and Urban100 by bicubic interpolation and the Gaussian noise with three different noise levels as shown in Table 8, to generate test LR images. Note that the third level, i.e., 30/9 for \( \sigma_r/\sigma_s \), was out-of-scope noise from the training stage. We evaluated the performance of all models using the same strategy of computing PSNR and SSIM as the ideal SISR.

Table 8 presents the performance of U-Baseline, Ps-BayeSR, U-BayeSR, and S-BayeSR. One can see that the difference between the pre-extracted noise and true noise could greatly weaken the performance of U-Baseline. By contrast, the proposed generative learning could improve the generalizability of BayeSR to this difference, and therefore the performance of Ps-BayeSR and U-BayeSR did not degrade much. Owing to the generalizability of BayeSR to unseen noise, as shown in Table 4, the performance of BayeSR dropped less than Baseline when the test noise level, i.e., 30/9 for \( \sigma_r/\sigma_s \), was out of the scope of training.

![Fig. 9. Visualization on the task of realistic SISR ×4: three typical examples from DIV2K. The red boundary denotes the model transferred from ideal SISR, while the green boundary represents the model trained without ground truth. Please refer to Supplementary Material, available online, for high-resolution images.](image)

| Model | Bicubic | EDSR | RCAN | SRGAN | ESRGAN | SRFlow | S-BayeSR | U-Baseline | Ps-BayeSR | U-BayeSR |
|-------|---------|------|------|-------|--------|--------|----------|------------|-----------|----------|
| LRPSNR | 36.74   | 37.78 | 37.80 | 27.89 | 36.83  | 37.52  | 36.93    | 33.08      | 38.59     | 35.65    |
| NIQE  | 7.99    | 6.89  | 6.91  | 3.83  | 4.02   | 3.84   | 7.26     | 4.85       | 7.69      | 6.96     |
| BRISQUE | 60.41   | 55.74 | 55.12 | 15.68 | 27.50  | 25.66  | 58.60    | 16.42      | 60.45     | 47.67    |
| #Paras | –       | 43.1M | 15.6M | 2.03M | 16.7M  | 39.5M  | 2.63M    | 16.7M      | 2.63M     | 2.63M    |

We test all methods on the test dataset of DPED-iPhone, and report the average LRPSNE, NIQE, and BRISQUE.

TABLE 7
Evaluation on the Task of Real-World SISR ×4

We test all methods on the test dataset of DPED-iPhone, and report the average LRPSNE, NIQE, and BRISQUE.
Overall, the supervised model achieved superior performance, and further improving unsupervised models of BayeSR yet remains to be explored in future work.

### 5.2 Generalizability to Kernel Estimation

This section studies the performance of BayeSR when there is evident difference between the estimated degradation kernels and the true ones in the test stage. To this end, we trained a new BayeSR model, referred to as K-BayeSR, using the similar network architecture in Fig. 3 and training strategy of S-BayeSR in Section 4.3.2.

Concretely, we set the downsampling operator of K-BayeSR, i.e., $A$ in (1), to be an explicit one dependent on the input, instead of a trainable module as S-BayeSR used. This was implemented by replacing the downsampling module of the network in Fig. 3 by the input degradation operation. Therefore, in the training stage of K-BayeSR we adopted random Gaussian kernel and noise degradation, referred to as $A_{\text{Gaussian}}$, and used $A_{\text{Gaussian}}$ to degrade HR images from DIV2K to generate realistic LR images. This Gaussian degradation $A_{\text{Gaussian}}$ used two parameters ranging within $\{0.7, 4\}$ for generating Gaussian blur kernels, and valued $\sigma_r$ and $\sigma_s$ respectively ranging within $\{0.12\}$ and $\{0.4\}$ for Gaussian noise. Note that the training images of K-BayeSR were different from that of S-BayeSR which adopted solely bicubic interpolation $K_{\text{Bicubic}}$ as the degradation kernel to generate training images. For comparisons, we also trained the Baseline model adopting the same settings as K-BayeSR but without using the proposed generative learning loss.

In the test stage, to be consistent with USRNet [76], we used twelve kernels, including four for isotropic Gaussian, four for anisotropic Gaussian, and four motion blur kernels, to generate test LR images from Set14. Then, four groups of methods were evaluated for comparisons. The first group included RCAN [25] and S-BayeSR. They were directly transferred from the resulting models in Section 4.3.2 and did not need an explicit input of blur kernels. The second group, i.e., Baseline, K-BayeSR, and USRNet [76], were tested by feeding bicubic interpolation $K_{\text{Bicubic}}$ as the degradation input for super-resolving LR images. The third group consisted of four methods, i.e., DIP-FKP [77], DIP-FKP+Baseline, DIP-FKP+K-BayeSR, and DIP-FKP+USRNet. DIP-FKP is a state-of-the-art blind SR method for jointly estimating kernels and super-resolving LR images, and the latter three took the estimated kernels from DIP-FKP as inputs for super-resolving LR images. Finally, the fourth group, i.e., GT+Baseline, GT+K-BayeSR, and GT+USRNet,
TABLE 9

Evaluation When Images are Blurred by Diverse Kernels for SISR \( \times 4 \)

| Method          | Test images are degraded by diverse blur kernels without noise | \( \times 4 \) |
|-----------------|-------------------------------------------------------------|-------------|
| RCAN            | 20.70/0.4209 | 20.27/0.4794 | 21.77/0.5259 | 21.37/0.5221 |
| K-BayeSR        | 21.80/0.5097 | 20.84/0.5016 | 21.92/0.5459 | 21.60/0.5327 |
| DIP-FKP         | 23.10/0.5290 | 22.45/0.4948 | 23.20/0.5290 | 22.80/0.5155 |
| DIP-FKP + K-BayeSR | 22.90/0.5220 | 22.15/0.4834 | 22.80/0.5155 | 22.40/0.4999 |
| USRNet          | 22.70/0.5394 | 22.25/0.5046 | 22.90/0.5394 | 22.50/0.5248 |
| K-BayeSR        | 22.90/0.5220 | 22.15/0.4834 | 22.80/0.5155 | 22.40/0.4999 |
| DIP-FKP         | 23.10/0.5290 | 22.45/0.4948 | 23.20/0.5290 | 22.80/0.5155 |
| DIP-FKP + K-BayeSR | 22.90/0.5220 | 22.15/0.4834 | 22.80/0.5155 | 22.40/0.4999 |
| USRNet          | 22.70/0.5394 | 22.25/0.5046 | 22.90/0.5394 | 22.50/0.5248 |
| K-BayeSR        | 22.90/0.5220 | 22.15/0.4834 | 22.80/0.5155 | 22.40/0.4999 |

Here, we report the average PSNR/SSIM on Set14. The bold values denote the best performance in each group. Here, the right arrow (\( \rightarrow \)) indicates the input of degradation kernels; \( K_m \) denotes the bicubic interpolation degradation and GT means the ground truth kernel. Note that the results of the last group, indicated as gray values, are directly cited from VIRNet [75] for reference, as the evaluation criteria are different.

were tested by feeding the true degradation kernel of each LR image. As the evaluation criteria in VIRNet [75] are different from ours, we solely cited their test results in the paper for reference.

Table 9 presents the results for SISR \( \times 4 \). One can see that the BayeSR-based methods demonstrated better generalizability than others when the input kernels were different from the ground truth (GT). Note that when the GT kernels were given, USRNet set superior performance in all categories of the fourth group; by contrast when the input changed to the estimated ones or bicubic interpolation, its performance dropped down dramatically, to much poorer results compared to K-BayeSR. This confirmed neither bicubic nor DIP-FKP could represent or estimate the kernels of test LR images accurately enough for USRNet. By contrast, K-BayeSR performed consistently in the second, third and fourth groups with three sources of kernel inputs. K-BayeSR demonstrated good robustness to the estimated kernels, while Baseline and USRNet could be more sensitive. The robustness could be attributed to the advantageous statistical modeling. Concretely, given the observation \( k \), the degraded term, \( y = m - Az \) is deterministic for Baseline, since the distributions of \( z \) and \( m \) are degraded into one-point distributions without the constraints in Table 2. By contrast, \( y = m - Az \) is stochastic for K-BayeSR, since \( z \) and \( m \) follow their own priors. In other word, Baseline is aimed to learn a point-to-point mapping from \( y = m - Az \) to \( x \), where \( x \) is also deterministic, but K-BayeSR is conducted to learn a distribution-to-distribution mapping from \( q(y - m - Az) \) to \( q(x) \). Since the prior corresponding to \( q(x) \) is kernel-independent, as shown in (6), K-BayeSR is less sensitive to given kernels. Nevertheless, it is worth mentioning that K-BayeSR delivered much poor results when the LR images were degraded by the motion kernels, a group of different degradation to the Gaussian kernels. Therefore, how to improve the generalizability when the distributions of degradation kernels are different remains to be further explored. Furthermore, K-BayeSR did not match the best results when the GT kernels were given, due to the limitation of explicit modeling and generative learning. Nevertheless, in real-world image super resolution the GT kernels could not be available, and improving the modeling capacity and accuracy should be considered in future work.

5.3 Super-Resolution on Real-World Images

Here, we studied the real-world image SR, where the degradation procedures of images are unknown. We used three examples, i.e., chip, frog and stars, and compared the results from RCAN [25], K-BayeSR, and USRNet [76]. Note that K-BayeSR and USRNet require a blur kernel as the input, which is simply set as the bicubic interpolation. K-BayeSR overcame the difference of degradations and performed well. Besides, the real-world noise in frog and stars was mapped into artifacts by RCAN and USRNet due to the difference between simulated noise and real-world noise. By contrast, K-BayeSR could super resolve images with less artifacts, demonstrating better generalizability in real-world scenarios.

6 CONCLUSION

In this work, we proposed a Bayesian image restoration framework, and implemented it for SISR by neural networks. Concretely, we first modeled image statistics using the smoothness and sparsity priors, and presented the variational inference framework of estimating the smoothness component and sparsity residual from an observation. Then, we built neural networks to implement the framework for SISR, and proposed the unsupervised strategies to train the networks. Finally, we showed the superior
generalization ability of our method, and demonstrated its effectiveness in unsupervised SISR.

In our future work, we can jointly infer blur kernels and restorations by simultaneously modeling kernel and image priors. Besides, modeling image priors and quantifying uncertainties of IR models is opening. We adopted the smoothness and sparsity priors to model image features, but this method cannot represent particularly complex image structures. How to accurately model image priors is still opening and worth further exploring. Moreover, quantifying uncertainties of deep learning models has arisen as one of the new requirements in many applications [78]. As low-level computer vision, IR could be further considered as an estimation of stochastic mappings, to explore any possible solutions of this ill-posed inverse problem. After that, one can evaluate uncertainties of deep IR models, which is helpful for AI safety [78] in computer vision systems.

ACKNOWLEDGMENTS

The authors are grateful to all anonymous reviewers for their insightful comments, which improve many aspects of this work, especially the discussion of BayeSR in Section 5. We also would like to thank Fuping Wu, Hangqi Zhou, and Ke Zhang for useful comments and proofread of the manuscript.

REFERENCES

[1] M. E. Celebi and B. Smolka, Advances in Low-Level Color Image. Dordrecht, Netherlands: Springer, 2014.
[2] J. Jan, Medical Image Processing, Reconstruction and Restoration. Boca Raton, FL, USA: CRC Press, 2006.
[3] J. R. Jensen, Introductory Digital Image Processing: A Remote Sensing Perspective. Upper Saddle River, NJ, USA: Prentice Hall, 1995.
[4] C. Dong, C. 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.
[5] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, “Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising,” IEEE Trans. Image Process., vol. 26, no. 7, pp. 3142–3155, Jul. 2017.
[6] B. Lim, S. Son, H. Kim, S. Nah, and K. Lee, “Enhanced deep residual networks for single image super-resolution,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, 2017, pp. 136–144.
[7] T. Tong, G. Li, X. Liu, and Q. Gao, “Image super-resolution using dense skip connections,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 4799–4807.
[8] W. Lai, J. Huang, N. Ahuja, and M. Yang, “Deep laplacian pyramid networks for fast and accurate super-resolution,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 624–632.
[9] W. Dong, P. Wang, W. Yin, G. Shi, F. Wu, and X. Lu, “Denoising prior driven deep neural network for image restoration,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 41, no. 10, pp. 2305–2318, Oct. 2019.
[10] A. Tikhonov and V. Arsenin, Solutions of Ill-Posed Problems. Berlin, Germany: Springer, 1986.
[11] L. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” Physica D, vol. 60, no. 1–4, pp. 259–168, 1992.
[12] B. R. Hunt, “Bayesian methods in nonlinear digital image restoration,” IEEE Trans. Comput., vol. C-26, no. 3, pp. 219–229, Mar. 1977.
[13] W. Qian and D. M. Titterington, “Bayesian image restoration: An application to edge-preserving surface recovery, IEEE Trans. Pattern Anal. Mach. Intell., vol. 15, no. 7, pp. 748–752, Jul. 1993.
[14] J. Portilla, A. Ttran-Vega, and I. W. Selesnick, “Efficient and robust image restoration using multiple-feature l2-relaxed sparse analysis priors,” IEEE Trans. Image Process., vol. 24, no. 12, pp. 5046–5059, Dec. 2015.
[15] E. J. Candès and Y. Plan, “Tight oracle inequalities for low-rank matrix recovery from a minimal number of noisy random measurements,” IEEE Trans. Inf. Theory, vol. 57, no. 4, pp. 2324–2359, Apr. 2011.
[16] K. Zhang, W. Zuo, and L. Zhang, “FFDNet: Toward a fast and flexible solution for CNN-based image denoising,” IEEE Trans. Image Process., vol. 27, no. 9, pp. 4608–4622, Sep. 2018.
[17] S. Nah, T. H. Kim, and K. M. Lee, “Deep multi-scale convolutional neural network for dynamic scene deblurring,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 257–265.
[18] K. He, X. Zhang, J. Sun, and S. Ren, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.
[19] J. Kim, J. Lee, and A. Lee, “Deeply-recursive convolutional network for image super-resolution,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 1637–1645.
[20] S. A. Bigdeli, M. Jin, P. Favaro, and M. Zwick, “Deep mean-shift priors for image restoration,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 763–772.
[21] S. Laine, T. Karras, J. Lehtinen, and T. Aila, “High-quality self-supervised image denoising,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 6970–6980.
[22] Z. Yue, H. Yong, Q. Zhao, D. Meng, and L. Zhang, “Variational denoising network: Toward blind noise modeling and removal,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 1690–1701.
[23] M. E. Helou and S. Süssstrunk, “Blind universal Bayesian image denoising with Gaussian noise level learning,” IEEE Trans. Image Process., vol. 29, no. 4, pp. 1687–1697, 2020.
[24] S. Izadi and G. Hamarneh, “Patch-based non-local Bayesian networks for blind confocal microscopy denoising,” 2020, arXiv: 2003.11177.
[25] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, “Image super-resolution using very deep residual channel attention networks,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 294–310.
[26] C. Ledig et al., “Photo-realistic single image super-resolution using a generative adversarial network,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 4681–4690.
[27] A. Lugmayr, M. Danelljan, L. V. Gool, and R. Timofte, SRFlow: Learning the super-resolution space with normalizing flow,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 715–732.
[28] A. Shochar, N. Cohen, and M. Irani, “Zero-shot super-resolution using deep internal learning,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2021, pp. 3103–3105.
[29] J. W. Soh, S. Cho, and N. I. Cho, “Meta-transfer learning for zero-shot super-resolution,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 3513–3522.
[30] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Deep image prior,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 1946–1954.
[31] A. Bulat, J. Yang, and G. Tzimiropoulos, “To learn image super-resolution, use a GAN to learn how to image degradation first,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 187–202.
[32] S. Maeda, “Unpaired image super-resolution using pseudo-supervision,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 288–297.
[33] A. Lugmayr, M. Danelljan, and R. Timofte, “Unsupervised learning for real-world super-resolution,” in Proc. IEEE Int. Conf. Comput. Vis. Workshops, 2019, pp. 3408–3416.
[34] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin, “An iterative regularization method for total variation-based image restoration,” Multiscale Model. Simul., vol. 4, no. 2, pp. 460–489, 2005.
[35] A. Jalobeanu, L. Blanc-Féraud, and J. Zerubia, “An adaptive Gaussian model for satellite image deblurring,” IEEE Trans. Image Process., vol. 17, no. 9, pp. 1677–1688, 2008.
[36] J. Yang, J. Wright, T. Huang, and Y. Ma, “Image super-resolution as sparse representation of raw image patches,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2008, pp. 1–8.
[37] A. Chambolle and P. Lions, “Image recovery via total variation minimization and related problems,” Numerische Mathematik, vol. 76, no. 2, pp. 167–188, 1997.
[38] J. Kim, J. Chung, F. Park, and A. Yip, “Total variation image restoration: Overview and recent developments,” in Handbook of Mathematical Models in Computer Vision, New York, NY, USA: Springer, 2006, pp. 17–31.
[39] M. Figueiredo and R. Nowak, “An em algorithm for wavelet-based image restoration,” IEEE Trans. Image Process., vol. 12, no. 8, pp. 906–916, Aug. 2003.
Gao and Zhuang: Bayesian image super-resolution with deep modeling of image statistics

[40] R. Chan, T. Chan, L. Shen, and Z. Shen, “Wavelet algorithms for high-resolution image reconstruction,” *SIAM J. Sci. Comput.*, vol. 24, no. 4, pp. 1408–1432, 2003.

[41] V. Koltchinskii, K. Lounici, and A. B. Tsybakov, “Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion,” *Ann. Stat.*, vol. 39, no. 5, pp. 2302–2329, 2011.

[42] E. Candès, X. Li, Y. Ma, and J. Wright, “Robust principal component analysis?,” *J. ACM*, vol. 58, no. 3, 2011, Art. no. 11.

[43] R. Molina, “On the hierarchical Bayesian approach to image restoration: applications to astronomical images,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 11, pp. 1122–1128, Nov. 1994.

[44] R. Molina, J. Mateos, A. K. Katsaggelos, and M. Vega, “Bayesian multichannel image restoration using compound gauss-markov random fields,” *IEEE Trans. Image Process.*, vol. 12, no. 12, pp. 1642–1654, Dec. 2003.

[45] R. Pan and S. J. Reeves, “Efficient huber-markov edge-preserving image restoration,” *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3728–3735, Dec. 2006.

[46] J. A. Guerrero-Colón, L. Mancera, and J. Portilla, “Image restoration using space-variant Gaussian scale mixtures in overcomplete pyramids,” *IEEE Trans. Image Process.*, vol. 17, no. 1, pp. 27–41, Jan. 2008.

[47] S. D. Babacan, R. Molina, and A. K. Katsaggelos, “Variational Bayesian blind deconvolution using a total variation prior,” *IEEE Trans. Image Process.*, vol. 18, no. 1, pp. 12–26, Jan. 2009.

[48] H. Ayasso, T. Rotet, and A. Abergel, “A variational Bayesian approach for unsupervised super-resolution using mixture model priors,” *Inverse Problems*, vol. 28, no. 12, 2012, Art. no. 125005.

[49] R. Timofte, V. De Smet, and L. Van Gool, “A+: Adjusted anchored neighborhood regression for fast super-resolution,” in *Proc. Asian Conf. Comput. Vis.*, 2014, pp. 111–126.

[50] H. Burger, C. Schuler, and S. Harmeling, “Image denoising: Can plain neural networks compete with BM3D?,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2392–2399.

[51] J. Lehtinen et al., “Noise2Noise: Learning image restoration without clean data,” in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2965–2974.

[52] A. Krull, T.-O. Buchholz, and F. Jug, “Noise2Void-learning denoising from single noisy images,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2129–2137.

[53] J. Batson and L. Royer, “Noise2self: blind denoising by self-supervision,” in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 524–533.

[54] J. Chen, J. Chan, H. Chao, and M. Yang, “Image blind denoising with generative adversarial network based noise modeling,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3155–3164.

[55] J. Yang, J. Wright, T. S. Huang, and Y. Ma, “Image super-resolution via sparse representation,” *IEEE Trans. Image Process.*, vol. 19, no. 11, pp. 2861–2873, Nov. 2010.

[56] M. Sajjadi, B. Scholkopf, and M. Hirsch, “EnhanceNet: Single image super-resolution through automated texture synthesis,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 4491–4500.

[57] J. Kim, J. Lee, and K. Lee, “Accurate image super-resolution using very deep convolutional networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1646–1654.

[58] J. Yu, Y. Fan, J. Yang, N. Xu, Z. Wang, X. Wang, and T. Huang, “Wide activation for efficient and accurate image super-resolution,” *arXiv:1808.08718*, 2018.

[59] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, “Residual dense network for image super-resolution,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2472–2481.

[60] J. Jonsson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2016, pp. 694–711.

[61] X. Wang et al., “ESRGAN: Enhanced super-resolution generative adversarial networks,” in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2018, pp. 63–79.

[62] H. Chen et al., “Pre-trained image processing transformer,” 2020, arXiv:2012.00364.

[63] G. Chantas, N. Galatsanos, A. Likas, and M. Saunders, “Variation Bayesian image restoration based on a product of t-distributions image prior,” *IEEE Trans. Image Process.*, vol. 17, no. 10, pp. 1795–1805, Oct. 2008.

[64] R. Diascon and D. Ylivisaker, “Conjugate priors for priorion families,” *Ann. Statist.*, vol. 7, no. 2, pp. 269–281, 1979.

[65] D. P. Kingma and M. Welling, “An introduction to variational autoencoders,” *Foundations Trends Mach. Learn.*, vol. 12, no. 4, pp. 307–392, 2019.

[66] O. Ronneberger, P. Fischer, and T. Brox, “U-Net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention*, Berlin, Germany: Springer, 2015, pp. 234–241.

[67] S. Bell-Klglser, A. Shcher, and M. Irani, “Blind super-resolution kernel estimation using an internal-GAN,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 284–293.

[68] X. He, Z. Mo, P. Wang, Y. Liu, M. Yang, and J. Cheng, “ODEN-inspired network design for single image super-resolution,” in *Proc. IEEE Int. Conf. Comput.Vis.*, 2019, pp. 1732–1741.

[69] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alber Morel, “Low-complexity single-image super-resolution based on non-negative neighbor embedding,” in *Proc. Brit. Mach. Vis. Conf.*, 2012, pp. 135.1–135.10.

[70] R. Zeyde, M. Elad, and M. Protter, “On single image scale-up using sparse-representations,” in *Proc. 7th Int. Conf. Curves Surfaces, 2012*, pp. 711–730.

[71] J. A. Guerrero-Colón, L. Mancera, and J. Portilla, “Image restoration using space-variant Gaussian scale mixtures in overcomplete pyramids,” *IEEE Trans. Image Process.*, vol. 17, no. 1, pp. 27–41, Jan. 2008.

[72] J. B. Huang, A. Singh, and N. Ahuja, “Single image super-resolution from transformed self-exemplars,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5197–5206.

[73] X. Li, Y. Cao, A. Yu, and T. Huang, “Real-world super-resolution via kernel estimation and noise injection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 1914–1923.

[74] G. Healey and R. Kondepudy, “Radiometric CCD camera calibration and noise estimation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 3, pp. 267–276, Mar. 1994.

[75] Z. Yue, H. Yong, Q. Zhao, L. Zhang, and D. Meng, “Variational image restoration network,” 2020, arXiv:2008.10796v1.

[76] K. Zhang, L. V. Gool, and R. Timofte, “Deep unfolding network for image super-resolution,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3217–3226.

[77] J. Liang, K. Zhang, S. Gu, L. V. Gool, and R. Timofte, “Flow-based kernel prior with application to blind super-resolution,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 10 596–10 605.

[78] Y. Gal, “Uncertainty in deep learning,” Ph.D. dissertation, Univ. Cambridge, Cambridge, U.K., 2016.

Shangqi Gao received the BS degree from the Faculty of Mathematics, Northwestern Polytechnical University, Xi’an, China, in 2015, the MSc degree from the School of Mathematics and Statistics, Wuhan University, Wuhan, China, in 2018. He is currently working toward the PhD degree with the School of Data Science, Fudan University, Shanghai, China. His current research interests include image reconstruction, tensor completion, and inverse problem.

Xiuhai Zhuang received the graduate degree from the Department of Computer Science, Tianjin University, the MS degree in computer science from Shanghai Jiao Tong University, and the doctorate degree from University College London. He is currently a professor with the School of Data Science, Fudan University, Shanghai, China. His current research interests include medical image analysis, image processing, and computer vision. His works have been nominated twice for the MICCAI Young Scientist awards (2008, 2012).

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.