Revisiting $\ell_1$ Loss in Super-Resolution: A Probabilistic View and Beyond

Xiangyu He, Jian Cheng

Abstract—Super-resolution as an ill-posed problem has many high-resolution candidates for a low-resolution input. However, the popular $\ell_1$ loss used to best fit the given high-resolution image fails to consider this fundamental property of non-uniqueness in image restoration. In this work, we fix the missing piece in $\ell_1$ loss by formulating super-resolution with neural networks as a probabilistic model. It shows that $\ell_1$ loss is equivalent to a degraded likelihood function that removes the randomness from the learning process. By introducing a data-adaptive random variable, we present a new objective function that aims at minimizing the expectation of the reconstruction error over all plausible solutions. The experimental results show consistent improvements on mainstream architectures, with no extra parameter or computing cost at inference time.

Index Terms—Image Super-Resolution, $\ell_1$ Loss, Probabilistic Model.

I. INTRODUCTION

The behavior of single image super-resolution (SISR) networks is primarily driven by the choice of the objective function. Two common examples in SISR are the mean squared error (MSE) loss and the mean absolute error (MAE) loss. Since the long-tested measure peak signal-to-noise ratio (PSNR) is essentially defined on the MSE, pioneer works [1]–[6] have largely focused on minimizing the mean squared reconstruction error to achieve high performances.

Due to the ill-posed nature of SISR problem, a low-resolution image $x$ can be well described as the down sampling result of many high-resolution images $\tilde{y}_{1:k}$. This implies that, in the inverse image reconstruction, the neural network trained by MSE loss, i.e., $E(x,\tilde{y})[\sum ||f(x) - \tilde{y}||^2_2]$, will learn to output the mean value of all HR targets, i.e., $f(x) \rightarrow \frac{1}{k} \sum \tilde{y}_t$, which best fits the image pairs $\{x, \tilde{y}_{1:k}\}$ [7]. However, the average of different HR images results in the absence of distinct edges and the blurring effect, which suffers from low image fidelity despite relatively high PSNR. Recently, $\ell_1$ loss has received much attention from low-level vision researchers, since it is in practice more robust to the regression-to-the-mean problem in MSE loss [8], [9]. Despite its empirical success in state-of-the-arts super-resolution methods [10]–[15], $\ell_1$ loss itself has been rarely discussed in the context of image processing.

In probabilistic setting, it is well known that a median is the minimizer of $\ell_1$ loss with respect to given targets [16]. In the case of training neural networks over LR-HR image pairs, the situation is quite similar to MSE loss where the network will learn to recover the median of all perceptually convincing HR images $\tilde{y}_{1:k}$. However, both $\ell_1$ and $\ell_2$ loss only capture the median/mean value of the whole solution space, which can be inadequate for estimating any specific solution in high-dimensional predictions, super-resolution in particular.

In this paper, we begin with a simple idea that it is intuitive to learn a one-to-many mapping by introducing $P(y|x; W)$ explicitly: 1) first perform a random sampling from $P(y|x; W)$ to obtain a underlying natural image $y$; 2) then maximize the likelihood of $y, \tilde{y}$ co-occurrence. Since we focus on the expected value averaged across all $y$ instead of any specific sampling result, the learning process serves as a multiple-valued mapping.

Fig. 1: Illustration of the learning process given an LR-HR image pair. $\ell_1$ loss only considers the fitting error between model output $f(x)$ and HR image $\tilde{y}$. We propose to learn an one-to-many mapping by introducing $P(y|x; W)$ explicitly: 1) first perform a random sampling from $P(y|x; W)$ to obtain a underlying natural image $y$; 2) then maximize the likelihood of $y, \tilde{y}$ co-occurrence. Since we focus on the expected value averaged across all $y$ instead of any specific sampling result, the learning process serves as a multiple-valued mapping.

X. He and J. Cheng are with the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing 100190, China (e-mail: xiangyu.he@nlpr.ia.ac.cn; jcheng@nlpr.ia.ac.cn).
Manuscript received April 19, 2005; revised August 26, 2015.

arXiv:2201.10084v2 [cs.CV] 9 Apr 2023
The learned standard deviation of $P(y|x; W)$ facilitates the HR image regression and, as a by-product, allows us to make meaningful estimates of the model uncertainty in SISR, i.e., a way to reflect its SR correctness likelihood. This can be useful in settings where errors carry serious repercussions such as medical image super-resolution.

The experimental results show consistent improvements over popular $\ell_1$ loss on mainstream networks, without the extra cost at runtime.

II. RELATED WORK

Super-resolution Over the past decades, super-resolution is among the most fundamental low-level vision problems. Early works often formulate the task as an interpolation [17]–[20]. Latter studies start to reconstruct high-resolution images in a data-driven way, by estimating the natural image statistics [21], [22] (including deep prior [23], [24]) and exploiting the neighbor embedding [25]–[27]. Another group of methods utilizes the self-similarities to learn the inverse mapping [28]–[31]. Benefiting from the development of sparse coding, [32]–[34] achieve reasonable results on benchmark datasets.

Advances in deep learning have further enhanced the state-of-the-art performance since [1], [3] first introduce a CNN-based SR method. Various CNN architectures have been proposed such as residual networks [2], [9], [35], densely connected networks [36], [37], ode-inspired design [38] and U-Net encoder-decoder networks [39]–[41]. In particular, the attention mechanism has become a popular tool for image super-resolution. Both channel attention [10] and non-local technique [11], [12] achieve notable improvements over baseline methods. Recently, self-correlation/self-attention that models the long-distance relationship among similar patches have led to successful results [13], [15], [42], as the large training dataset can be used as a means to capture the global information [43]–[45]. Our approach can facilitate the training of deep learning-based methods in a plug-in and play manner.

Loss functions Most learning objects in the literature fall into two main categories: pixel-wise loss and perceptual loss. The former aims to optimize the full reference metric such as PSNR and SSIM. MSE loss, in particular, is equivalent to PSNR. However, [46]–[48] show that $\ell_2$ loss results in overly-smooth SR images and the following works tend to use $\ell_1$ during training. In light of the success of knowledge distillation in image recognition, [49], [50] use loss functions in the feature space to enhance the performance of student networks. Our work also falls into this category. We propose to learn from a posterior distribution instead of best-fitting ground-truth images.

Perceptual loss tackles the problem by employing generative adversarial networks [48], [51], especially in super-resolution with large upscaling factors (e.g., $4 \times$, $8 \times$) [6], [52]. The ImageNet pre-trained networks are commonly used as a discriminator loss [24], [46], [47], which leads to high-frequency details and perceptually satisfying results in the sense. Our approach can be compatible with those adversarial settings as the content loss term in a perceptual loss.

The most related work is Noise2noise [53], which suffers from inaccurate gradient directions (details in Section III-C).

We solve this problem by a well-bounded gradient step and point out that [53] indeed serves as a special case of our approach. Noise2Self [54] assumes that noise in each patch is independent of other patches, which can not hold responsible when it comes to the content of images instead of noise. Self2Self [55] presents dropout in both image and feature space to reduce prediction variances, which is also compatible with our loss functions.

Model uncertainty Despite their unprecedented power to solve the inverse problem, deep learning-based SR methods are prone to make mistakes. As deep models are widely used in computer-aided diagnosis [56], [57], it is crucial to better understand the confidence of neural networks in their predictions. That is, the SR network may provide uncertainty about the generated super-resolved images: which part is “real” and when we should be cautious.

[58] first develops a framework for modeling uncertainty, which is hard to describe the modern deep neural networks. Recently, Monte Carlo Dropout [59] presents a practical solution to estimate uncertainty via the dropout layer. [60] further shows that networks trained with Batch-Normalization [61] (BN) is an approximate Bayesian model and allows an uncertainty estimation by sampling the network’s BN parameters. Unfortunately, neither of them was used in recent SR networks, which disables the variational inference in those approximate Bayesian models. Instead, we model the posterior distribution $P(y|x; W)$ explicitly, as a by-product, the estimated standard deviation facilitates the uncertainty estimation for SISR.

III. ANALYSIS

A. Probabilistic Modeling

From a statistical viewpoint, we are interested in the point estimation of the model parameters $W$, used in the following likelihood function:

$$\max_W \mathcal{L}(W|\hat{y}) = P(\hat{y}|x; W)$$

where $W$, $\hat{y}$ and $x$ indicate weights, observed ground-truth labels and inputs respectively. Super-resolution using convolutional neural networks also fall into this form, that learning parameters $W$ w.r.t. an inference function $f(x)$ from LR-HR image pairs directly.

In contrast to finding a one-to-one mapping between corrupted images and targets implied by $f(x)$, we begin with a subtle point: in reality, the mapping from $\hat{y}$ to $x$ is surjective. For example, a single LR image $x$ may correspond to the downsampling result of multiple natural images. Hence, it is intuitive to recover or approximate the local image manifold instead of simply fitting the high-resolution image $\hat{y}$.

Objective function The idea is to explicitly estimate a posterior distribution $P(y|x; W)$ where $y$ refers to (multiple) underlying natural images with the size of $H \times W$. Then, the observed HR image $\hat{y}$ should look like its counterpart $y$ sampled from $P(y|x; W)$, with high probability. Formally, our goal is to maximize the following conditional joint density function:

$$P(\hat{y}, y|x; W) = P(\hat{y}|y)P(y|x; W)$$
where \( P(\hat{y}|y) \) is a measure of the probability that \( \hat{y} \) being observed given \( y \). Note that, there are plenty of \( y \) can serve as the perceptually convincing solution to Equation (2) then, as is common in machine learning, we take the expectation of that formula to be optimized:

\[
\mathbb{E}_{\hat{y} \sim P(\hat{y}|y; W)}[P(\hat{y}|y)].
\]

To solve Equation (3), we will first need to deal with \( P(y|x; W) \), the distribution of underlying natural images \( y \) consistent with the low-resolution \( x \).

**Optimizing the objective** Based on the finding of non-local means [62], [63] that each pixel is obtained as a weighted average of pixels centered at regions similar to the reference and the central limit theorem (CLT), we shall model \( P(y|x; W) \) as a multivariate normal distribution:

\[
P(y|x; W) \sim \mathcal{N}(\mu|x; W); \Sigma(x; W) \tag{4}
\]

where mean \( \mu \) and covariance \( \Sigma \) are outputs of multi-layer neural networks with parameters \( W \). \( \sigma^2 \) corresponds to the diagonal elements in \( \Sigma \). However, the expectation in Equation (3) still involves sampling \( y \) from \( P(y|x; W) \) which is a non-differentiable operation. To allow the backprop through \( y \), we use one of the most common tricks in machine learning, the “reparameterization” [64], to move the sampling to the random variable \( z \sim \mathcal{N}(0, 1) \):

\[
\mathbb{E}_{\hat{y} \sim P(\hat{y}|y; W)}[P(\hat{y}|y)] = \mathbb{E}_{z \sim \mathcal{N}(0, 1)}[P(\hat{y}|\mu + \sigma * z)]
\]

where \( z \) is reformulated as \( \mu + \sigma * z \) and “\( * \)” indicates elementwise product. Then, we can conduct the standard gradient descent and the gradient averaged over \( z \) should converge to the actual gradient of Equation (3).

Similarly, as [24], we consider the conditional model defined by the Gibbs distribution (also known as Boltzmann distribution):

\[
P(\hat{y}|y) \propto \prod_{i} \exp\left(\frac{-|\hat{y}_{i} - y_{i}|}{kT}\right) \tag{5}
\]

where the corresponding Gibbs energy is in the form of \( ||\hat{y} - y||_1 \) and \( kT \) is the fixed partition function. The model (5) assumes that \( y \), looks like the observed ground-truth image \( \hat{y} \), is likely to be a strong evidence for \( \hat{y} \) occurring. We can now get both \( P(y|x; W) \) and \( P(\hat{y}|y) \) into Equation (3) then apply negative log-likelihood to the final objective for computational convenience:

\[
\min \mathbb{E}_{z}\left[\frac{1}{kT} \sum_{i}^{H \times W} |\hat{y}_{i} - (\mu_{i} + \sigma_{i} * z)|\right]. \tag{6}
\]

This equation implies a two-branch network (\( \mu \) and \( \sigma \)) that requires randomness during training. The expectation on the data-independent random variable \( z \) allows us to move the gradient symbol into the expectation to finish the standard stochastic gradient descent [65]. For notational simplicity, we shall in the remaining text denote Equation (6) as \( \mathbb{E}_{z}[||\hat{y} - (\mu + \sigma * z)||_1] \).

1 The distribution of \( P_{y|x} \) is not required to be Gaussian: for instance, if \( y \) is in practice more-Laplace, then \( P_{y|x} \) might be re-parameterized as \( \mu - \sigma * \text{sgn}(z) * \ln(1 - 2|z|) \), \( z \sim \mathcal{N}(0, 1) \).

**B. Revisiting \( \ell_1 \) Loss**

According to Jensen’s inequality, for any measurable convex function \( f(\cdot) \), we have \( \mathbb{E}[f(z)] \geq f(\mathbb{E}[z]) \). Since every p-norm is convex and \( z \sim \mathcal{N}(0, 1) \), then we have

\[
\mathbb{E}_{z}[||\hat{y} - (\mu + \sigma * z)||_1] \geq ||\hat{y} - (\mu + \sigma * \mathbb{E}[z])||_1 \tag{7}
\]

Considering \( \mathbb{E}[z] = 0 \), then

\[
\mathbb{E}_{z}[||\hat{y} - (\mu + \sigma * z)||_1] \geq ||\hat{y} - \mu||_1. \tag{8}
\]

Notice that the proposed learning target (6) actually serves as an upper bound of the popular \( \ell_1 \) loss function. In other words, \( \ell_1 \) loss also suffers from the degradation that loses the randomness during training and is confined to the given HR images \( \hat{y} \). Though the optimal solution for the original function and for the upper bound are indeed different, the probabilistic learning object can relax the deterministic learning process at each step by focusing on the expectation of the loss function.

Owing to the reparameterization trick [64], learning probabilistic model via Equation (6) seems straightforward. However, there exists a trivial solution to easily bridge the Jensen gap (i.e., \( \mathbb{E}[f(z)] = f(\mathbb{E}[z]) \)) such that \( \sigma \in \hat{0} \). As shown in Figure 2, the output of the \( \sigma \) branch gradually approaches zero which ultimately eliminates the randomness in our learning target. Similar degradation also occurs in Variational Auto-Encoder (VAE). In a nutshell, VAE is an auto-encoder when latent variable \( z \) that determines encodings distribution lose randomness. VAE regularises its random latent variable \( z \) towards Normal distribution that enables generative process.

Otherwise, the latent space will converge to a single point instead of a distribution. That is, VAE degrades into an one-to-one mapping given input images, i.e., a (deterministic) auto-encoder. In light of this, we focus on avoiding the degradation of \( \sigma \) in the following section.

**C. Beyond \( \ell_1 \) Loss**

**Data-independent \( \sigma \)**

The simplest way is to remove the trainable \( \sigma \) branch and let \( \sigma_i \) be a small constant \( k \) invisible to the human eye. Then, \( P(y|x; W) \sim \mathcal{N}(\mu; k^2 \mathbb{I}) \) where \( \mathbb{I} \) refers to an identity matrix. Contracting this term into Equation (6) and rearranging yields:

\[
\mathbb{E}_{z}[||\hat{y} - (\mu + k \cdot z)||_1] = \mathbb{E}_{z}[||\hat{y} + z' - \mu||_1 \tag{9}
\]

where \( z' \) is an additive Gaussian noise \( \mathcal{N}(0, k) \) and \( k \neq 0 \). The left hand side of this equation is something we hope
to optimize via backward propagation. The right hand side serves is the core of Noise2Noise [53] that restores corrupted images without clean data, i.e., learning super-resolution by only looking at noised HR images.

Though the performance of training on corrupted observations may approach its clean counterpart [53], the noise term can do harm to the optimization process. To be specific, we expect \( \mu \) to approach \( \hat{y} \) via stochastic gradient descent. However, for some \( z' \) satisfying \((\hat{y}_i - \mu_i) + z'\) \(|\hat{y}_i - \mu_i| < 0\), we have
\[
\text{sgn}(\frac{\partial}{\partial \mu_i} \|\hat{y} + z'\| - \|\hat{y} - \mu\|) = -1 \cdot \text{sgn}(\frac{\partial}{\partial \mu_i} \|\hat{y}_i - \mu_i\|)
\]
where \( z' \) results in gradient ascent instead of descent, which ultimately hinders \( \mu \) from fitting \( \hat{y} \). To get a good estimate of the proper gradient, it would require sampling massive noised observations given a single clean image, which would be expensive. Hence, we hope to find some \( \sigma \) that run correct backprop efficiently and retain randomness.

**Data-adaptive \( \sigma \)** Fortunately, a delicate \( \sigma \) gives a definite answer to both. The full equation we want to optimize is:
\[
\min_\mu \mathbb{E}_z \left[ \|\hat{y} - (\mu + \hat{y} - \mu) \cdot z\| \right] \tag{10}
\]
where \( |\hat{y} - \mu| \) corresponds to the \( \sigma \) term and \( \| \cdot \| \) refers to element-wise absolute function. We shall denote Equation (10) as \( \ell_\Sigma \) in the remaining text for simplicity.

From the view of Equation (9), \( \ell_\Sigma \) is also in the form of adding some (data-adaptive) noise to HR images. However, for Equation (10), we allow the backprop through \( |\hat{y} - \mu| \) instead of regarding it as a scaling factor, which contributes to the following property:

**Lemma III.1.** Given a random variable \( z \sim \mathcal{N}(0,1) \) and fixed \( \hat{y} \), the gradient direction of \( \mu \) with respect to the following object
\[
\min_\mu \mathbb{E}_z \left[ \|\mu + |\hat{y} - \mu| \cdot z\| \right]
\]
is always consistent with \( \|\mu - \hat{y}\| \) and \( \mathbb{E}_z \left[ \frac{\partial \ell_\Sigma}{\partial \mu_i} \right] \) converges to 1.17.

**Proof.** Since the expectation of random variable \( z \) does not depend on model parameters, we can safely move the gradient symbol into \( \mathbb{E}_z[\cdot] \) while maintaining equality:
\[
\frac{\partial \ell_\Sigma}{\partial \mu_i} = \text{sgn}(\mu_i + |\mu_i - \hat{y}_i| \cdot z_i - \hat{y}_i)(1 + z_i \cdot \text{sgn}(\mu_i - \hat{y}_i)).
\]
Assuming that \( \mu_i > \hat{y}_i \), the above equation can be further simplified as:
\[
\frac{\partial \ell_\Sigma}{\partial \mu_i} = \text{sgn}((\mu_i - \hat{y}_i)(1 + z_i))(1 + z_i).
\]
Leaving out the non-differentiable \( z_i = -1 \), it is easy to prove that
\[
\frac{\partial \ell_\Sigma}{\partial \mu_i} = \begin{cases} 1 + z_i, & \forall z_i > -1 \\ -1 - z_i, & \text{otherwise} \end{cases}
\]
In both cases, we always have \( \frac{\partial \ell_\Sigma}{\partial \mu_i} > 0 \). Similarly, for \( \mu_i < \hat{y}_i \), we obtain \( \frac{\partial \ell_\Sigma}{\partial \mu_i} < 0 \) given any \( z_i \sim \mathcal{N}(0,1) \). Recall that \( \ell_1 \) norm is not differentiable with respect to the origin. Elsewhere, the partial derivative \( \text{sgn}(\mu_i - \hat{y}_i) \) is consistent with the gradient direction of \( \frac{\partial \ell_\Sigma}{\partial \mu_i} \) above.

For the estimation of \( \mathbb{E}_z \left[ \frac{\partial \ell_\Sigma}{\partial \mu_i} \right] \), we shall begin with \( \mu_i > \hat{y}_i \) to simplify the derivation:
\[
\mathbb{E}_z \left[ \frac{\partial \ell_\Sigma}{\partial \mu_i} \right] = \int_{-\infty}^{+\infty} \phi(z_i)dz_i - \int_{-\infty}^{-1} \phi(z_i)dz_i - \int_{-1}^{+\infty} \phi(z_i)dz_i \approx 1.1663
\]
where \( \Phi(\cdot) \) and \( \phi(\cdot) \) are the cumulative distribution function (CDF) and probability density function (PDF) of the standard normal distribution respectively. Similarly, we have \( \mathbb{E}_z \left[ \frac{\partial \ell_\Sigma}{\partial \mu_i} \right] = -1.1663 \) for any \( \mu_i \approx \hat{y}_i \) for any \( z \sim \mathcal{N}(0,1) \), we have
\[
\text{sgn}(\frac{\partial \ell_\Sigma}{\partial \mu_i}) = \text{sgn}(\frac{\partial \|\hat{y} - \mu\|}{\partial \mu_i}) \tag{11}
\]
\[
\mathbb{E}_z \left[ \frac{\partial \ell_\Sigma}{\partial \mu_i} \right] \approx 1. \tag{12}
\]
Note that the gradient direction is always making \( \hat{y} \) and the step-size converges to nearly one. Therefore, we can safely average the gradient of \( \nabla_{W,\ell_\Sigma} \) over arbitrarily many samplings of \( z \) and training images \( x \), without the risk of gradient ascent or the problem of vanishing/exploding gradient. Due to the ill-posed nature of super-resolution, \(|\hat{y} - \mu|\) can never be a zero vector then the randomness of (10) will be preserved.

It is shown that \(|\hat{y} - \mu|\) plays a core role in learning \( \mu \) and serves as \( \sigma \) in \( \ell_\Sigma \) loss, which inspires us to re-fetch the \( \sigma \) branch to perform the following multi-task learning except \( \ell_\Sigma \) loss:
\[
\min_\sigma \beta \cdot \|\|\hat{y} - \mu - \sigma\|\|_1 \tag{13}
\]
where \( \beta \) is a penalty factor and \( \mu \) is detached from backward propagation. This equation also, as a by-product, allows us to
make a rough estimate of the model uncertainty in SISR via the prediction of $\sigma$, e.g., small $\sigma_i$ means we can be certain of the super-resolved image produced by neural networks, otherwise should be less confident (details in Section IV-G). Finally, Figure 3 further illustrates the overall structure of the proposed scheme.

D. Revisiting Naive Setting

It is intuitive to directly model the joint probability density function $P(y_i, \hat{y}_i | x, W)$ in the form of the PDF of Laplacian distribution $\frac{1}{\sigma_i} \exp(-\frac{|y_i - \hat{y}_i|}{\sigma_i})$ without considering $P(\hat{y}_i | y_i)$. In this subsection, we argue that this naive setting where $y_i$ and $\sigma_i$ combine into a unified objective function is not suitable for either image restoration or uncertainty estimation.

Firstly, we are interested in estimating $P(y_i, \hat{y}_i | x, W)$ using maximum likelihood estimation

\[
\max \log P(y_i, \hat{y}_i | x, W) = \max \log \sigma_i - \frac{|y_i - \hat{y}_i|}{\sigma_i} \quad (14)
\]

\[
\Leftrightarrow \min \log \sigma_i + \frac{|y_i - \hat{y}_i|}{\sigma_i} \quad (15)
\]

where $y_i$ is the predicted SR image. It is easy to prove that Equation (15), as a two-differentiable function of $\sigma_i$, is convex on an interval $\sigma_i \in (0, 2|y_i - \hat{y}_i|]$. The local optimum is surprisingly

\[
\sigma_i^* = |y_i - \hat{y}_i| \quad (16)
\]

which is consistent with the learning target of Equation (13). However, as illustrated in Figure 4, the optimization process can be quite challenging when $|y_i - \hat{y}_i|$ is relatively large due to an ill-behaved (more precisely, flat) loss landscape of $\sigma_i$. Note that, the averaged $\ell_1$ training loss on DIV2K dataset is roughly $3.3$ in $2 \times$ SR (more than $5$ for $3 \times$ SR), which implies we could not reach $\sigma_i^*$ since the derivative approaches zero along with $\sigma_i \rightarrow \sigma_i^*$, as shown in Figure 4b. In contrast, Equation (13) enjoys a constant derivative almost everywhere with the same $\sigma_i^*$.

Besides, the behavior of Equation (15) can be counter-intuitive when $\sigma_i \rightarrow \sigma_i^*$, which leads to a weighted MAE loss for the prediction of $y_i$. The gradient then becomes

\[
\frac{\partial \mathcal{L}}{\partial y_i} = \frac{1}{\sigma_i} \text{sgn}(y_i - \hat{y}_i) \quad (17)
\]

Assuming that $\sigma_i \approx \sigma_i^*$, we may rewrite the above equation into

\[
\frac{\partial \mathcal{L}}{\partial y_i} \approx \frac{1}{|y_i - \hat{y}_i|} \text{sgn}(y_i - \hat{y}_i) \quad (18)
\]

which indicates that pixels with larger regression error should contribute less to the gradient of $y$. To be specific, since the magnitude of $\text{sgn}(y_i - \hat{y}_i)$ remains $+1$ for all pixels, the gradient will be dominated by $\frac{1}{\sigma_i}$. It is common sense that learning based methods should focus on edges with large $|y_i - \hat{y}_i|$ instead of signals in low-frequency with small $|y_i - \hat{y}_i|$. As for Equation (15), it is just the other way around. Therefore, even if $\sigma_i \rightarrow \sigma_i^*$, one may encounter unexpected results.

IV. EXPERIMENTS

A. Dataset

For Bicubic image SR, following the common setting in [9], [11], [12], [15], [38], [66], we conduct all experiments on the 1st-800th images from DIV2K [67] training set then evaluate our models on five benchmarks: Set5 [68], Set14 [69], B100 [70], Urban100 [71] and Manga109 [72] with scale factors $2 \times$, $3 \times$, $4 \times$. To make a fair comparison with early works trained on the tiny dataset [32], we also cite the reproduced DIV2K results [49] in Table IV. For real-world SR, we use the same dataset and training settings proposed in RealSR benchmark [73]. We report PSNR and SSIM on the Y channel in YCbCr space and ignore the same amount of pixels as scales from the border.

Since $\ell_1$ loss is also known-to-work for Additive White Gaussian Noise (AWGN) denoising, we evaluate our methods on another representative task: color (Table VIII)/grayscale (Table VII) image denoising.

B. Training Details

We use low-resolution RGB image patches with the size of $48 \times 48$ for training. All HR-LR image pairs are randomly rotated by $90^\circ$, $180^\circ$, $270^\circ$ and flipped horizontally [9]. We set the mini-batch size to 16. Our models are trained by Adam optimizer [74] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The initial learning rate is set to $10^{-4}$ and decreases to half at every $2 \times 10^5$ mini-batch updates. The total training cost is $10^6$ iterations [9]. For RealSR, as described in [73], the training patch size is $192 \times 192$ and learning rate is fixed at...
10^{-4}. Other details are the same as classical SR. We use the PyTorch framework [75] to implement our methods with an NVIDIA Tesla V100 GPU.

For image denoising, here we use the state-of-the-art DRUNet [76] as a strong baseline method. Following the same settings in [76], we include 400 BSD images [70], 4744 images of Waterloo Exploration Database [77], 900 images from DIV2K [67], and 2750 images from Flick2K dataset [9] as the training dataset. The standard AWGN with noise level $\sigma$ is added to the clean image. A specific noise level is randomly chosen from [0.05] during training. The learning rate starts from $1 \times 10^{-4}$ and finally ends at $1.56 \times 10^{-6}$. We set the patch size to 192 $\times$ 192 and the batch size is 16 for 4 Nvidia RTX GPUs.

### C. Ablation Studies

We first present an ablation analysis on each object function in our learning process and report quantitative results in terms of average PSNR and SSIM on Set14 [69] with the scale factor of 2. The baseline results in Table I (i.e., first row) are obtained using EDSR pre-trained model $3$. In the case of training neural networks by Equation (13) (i.e., third row), we run the backpropagation through both $\mu$ and $\sigma$ to allow the prediction of super-resolved images via $\mu$ branch. The following rows illustrate that both $\ell_g$ loss (Equation (10)) and the multi-task learning (Equation (13)) contribute to higher performances $\ell_1$ baseline. When applying both settings to the original network, we can further improve the PSNR and SSIM results. In the following experiments, we use “Equation (10)+Equation (13)” as the learning target.

Table II indicates that a proper $\beta$ matters in the balance between reconstruction loss $\ell_g$ and the auxiliary term. Fortunately, 0.01 seems to perform well in most cases. For simplicity, we set the channel number of the sigma branch to 160 and introduce a sigmoid function to normalize its output. In our experiments, we notice that nearly half of the estimated $\sigma$ approach zero, which correspond to the low-frequency/smooth parts of the high-resolution images (as shown in Figure 9). In light of this, we set $|\hat{y}_i - \mu_i|$ that is less than the average to zero and focus on the hard samples. Due to the limited computing resource, we did not conduct an ablation study on the structure of the sigma branch, instead simply borrowing the idea of Conv+PReLU proposed in [3], shown in Figure 3.

### D. Convergence rate

Figure 5 presents a comparison of the convergence rate between EDSR-baseline [9] and standard EDSR model [9], which indicates that deep SR models are still easy to train compared with lightweight counterparts, using the proposed learning object. Besides, the convergence speed of the proposed

---

1. [https://cv.snu.ac.kr/research/EDSR/models/edsr_baseline_x2-1bc95232.pt](https://cv.snu.ac.kr/research/EDSR/models/edsr_baseline_x2-1bc95232.pt)
TABLE IV: PSNR comparison with recently proposed loss functions for SISR. PISR [49] is a Knowledge Distillation framework where a “Teacher” network is required during training. “DIV2K” means models reproduced by [49] using DIV2K dataset with \( \ell_1 \) loss. “EDSR-baseline” refers to the baseline model in [9].

![Table IV](image)

TABLE V: Average PSNR (dB) for different models on RealSR testing set. We directly cite SRResNet and RCAN results reported in RealSR benchmark [73]. RCAN used in [73] is smaller than the original model. The running time is measured on a 1200 \( \times \) 2200 SR image.

![Table V](image)

E. Quantitative Results

Figure 6 shows the performance comparison of our approach coupled with various SR methods to state-of-the-arts in terms of the number of operations and parameters. We notably improve the Pareto frontier with no extra parameter or computing cost at inference time. When compared with recent learning strategies for image restoration, ours performs the best at most cases (Table IV), even surpasses the knowledge distillation method PISR [49]. Listed SR methods generally benefit from the proposed scheme except for IDN [78] with the scale factor of 2 on B100 [70].

We further detail the performance of our approach in contrast to recent lightweight networks [38], [66], [79] in Table III. To well describe different methods, we report the number of parameters, operations required to reconstruct a 720P image, and its average runtime speed measured on an NVIDIA Titan RTX GPU. From Table III, we observe that VDSR [35] trained by the proposed learning process outperforms baselines by a large margin, consistently for all scaling factors. Former state-of-the-art EDSR-baseline [9] with our approach also shows competitive results with recent well-designed lightweight SR methods. We further conduct experiments on the large BI degradation model EDSR [9]. As shown in Table VI, the original model has achieved high PSNR/SSIM on each dataset, while ours can further enhance the performance in most cases.
IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. XX, NO. XX, 2015

Fig. 7: Visual comparison for $2 \times$, $3 \times$ and $4 \times$ SR with BI models on benchmark datasets.

Fig. 8: Qualitative comparison of $3 \times$, $4 \times$ RealSR results.

TABLE VI: Quantitative comparison with computing-intensive models. We report the multiply-accumulate operations (MAC) and real runtime cost, required to reconstruct a $1280 \times 720$ HR image. RCAN [10] coupled with our methods can be competitive with recent leading works.

| Scale | Methods     | #Param. | Runtime | Set5 [68] | Set14 [69] | B100 [70] | Urban100 [71] | Manga109 [72] |
|-------|-------------|---------|---------|-----------|------------|-----------|---------------|---------------|
|       |             |         |         | PSNR      | SSIM       | PSNR      | SSIM          | PSNR          | SSIM          |
| 2×    | EDSR [9]    | 40.73M  | 1003.6ms| 38.11     | 0.9602     | 33.92     | 0.9195        | 32.32         | 0.9013        |
|       | EDSR+Ours  |         |         | 38.24     | 0.9613     | 34.05     | 0.9209        | 32.36         | 0.9020        |
|       | RCAN [10]   | 15.44M  | 667.8ms | 38.27     | 0.9614     | 34.16     | 0.9231        | 32.41         | 0.9027        |
|       | RCAN+Ours  |         |         | 38.25     | 0.9615     | 34.16     | 0.9231        | 32.40         | 0.9025        |
|       | SAN [11]    | 15.71M  | 481.7ms | 38.31     | 0.9620     | 34.16     | 0.9217        | 32.42         | 0.9028        |
|       | HAN [12]    | 15.92M  | 743.3ms | 38.27     | 0.9614     | 34.16     | 0.9217        | 32.41         | 0.9027        |
|       | NLSA [42]   | 41.80M  | 1915.8ms| 38.34     | 0.9618     | 34.08     | 0.9231        | 32.43         | 0.9027        |
| 3×    | EDSR [9]    | 43.68M  | 493.8ms | 34.65     | 0.9295     | 30.52     | 0.8462        | 29.23         | 0.8093        |
|       | EDSR+Ours  |         |         | 34.70     | 0.9295     | 30.58     | 0.8467        | 29.28         | 0.8100        |
|       | RCAN [10]   | 15.63M  | 313.4ms | 34.74     | 0.9299     | 30.65     | 0.8482        | 29.32         | 0.8111        |
|       | RCAN+Ours  |         |         | 34.84     | 0.9304     | 30.70     | 0.8485        | 29.34         | 0.8117        |
|       | SAN [11]    | 15.90M  | 1587.9ms| 34.75     | 0.9390     | 30.59     | 0.8476        | 29.33         | 0.8112        |
|       | HAN [12]    | 16.11M  | 343.7ms | 34.75     | 0.9299     | 30.67     | 0.8483        | 29.32         | 0.8110        |
|       | NLSA [42]   | 44.75M  | 869.0ms | 34.84     | 0.9306     | 30.70     | 0.8485        | 29.34         | 0.8117        |
| 4×    | EDSR [9]    | 43.10M  | 322.6ms | 32.46     | 0.8968     | 28.80     | 0.7876        | 27.71         | 0.7420        |
|       | EDSR+Ours  |         |         | 32.48     | 0.8985     | 28.86     | 0.7883        | 27.74         | 0.7423        |
|       | RCAN [10]   | 15.59M  | 181.1ms | 32.63     | 0.9002     | 28.87     | 0.7889        | 27.77         | 0.7436        |
|       | RCAN+Ours  |         |         | 32.69     | 0.9005     | 28.89     | 0.7892        | 27.78         | 0.7437        |
|       | SAN [11]    | 15.86M  | 880.0ms | 32.64     | 0.9002     | 28.92     | 0.7888        | 27.78         | 0.7442        |
|       | HAN [12]    | 16.07M  | 196.8ms | 32.64     | 0.9002     | 28.90     | 0.7890        | 27.80         | 0.7444        |
|       | NLSA [42]   | 44.15M  | 523.6ms | 32.59     | 0.9000     | 28.87     | 0.7891        | 27.78         | 0.7444        |

To further verify the effectiveness of our approach, we conduct extensive experiments on image denoising datasets with a strong baseline method DRUNet [76]. Since DRUNet has achieved promising results, we believe the improvements can be nontrivial. Note that, our methods outperform baseline with $\sim 0.1$dB on Urban100 and consistently bypass the previous leading methods on all datasets. The results are listed in Table VII and VIII.

F. Qualitative Results

To further illustrate the analysis above, we show visual comparisons in Figure 7 and 8. Bicubic downsampling results
in the loss of texture details and structure information, which induces super-resolved images with blurring and artifacts. Even worse, EDSR generates perceptually convincing results such as “barbara” in Set14 (first row in Figure 7c). However, the SR image contains several lines with wrong directions. Instead, our models can recover them being more faithful to the ground truth. CARN [66] with ours also obtains much better results as “barbara” in Set14 (first row in Figure 7c). However, the SR image contains several lines with wrong directions. Instead, our models can recover them being more faithful to the ground truth. CARN [66] with ours also obtains much better results as “barbara” in Set14 (first row in Figure 7c).

Despite the promising results, networks can make mistakes when exposed to unseen data outside the trained distribution. In this case, we expect information about model uncertainty in addition to the SR images.

G. Model Uncertainty

By introducing a posterior distribution $P(\hat{y}|x; \mathbf{W})$, we can easily use the predicted $\sigma$ as the estimated model uncertainty (Figure 9e). Following previous Bayesian models [59], [60], [102], [103], we evaluate uncertainty quality based on the standard metric Predictive Log Likelihood (PLL):

$$PLL(f(x), (\hat{y}, x)) = \log P(\hat{y}|f(x))$$ (19)

where $P(\hat{y}|f(x))$ is the probability of target $\hat{y}$ generated by a probabilistic model $f(x)$. In this paper, $f(x)$ corresponds to the multivariate Gaussian $N(\mu, \sigma^2)$ with predicted mean $\mu$ and standard deviation $\sigma$. $P(\hat{y}|f(x))$ is the Gaussian probability density function (PDF) evaluated at $\hat{y}$.

We report the average PLL of our approach coupled with EDSR [9] and CARN [66] in Table IX. The original EDSR/CARN networks fail to consider the model uncertainty, hence we use their average fitting error $E_D[|\hat{y} - \mu|]$ calculated on the training dataset $D$ to create a constant uncertainty estimation, which corresponds to baseline results in Table IX. Any improvement over the baseline reflects a sensible estimate of uncertainty. It is shown that PLL for high upscaling factors is $\mathbf{33.39}$ (Figure 9e). Following previous Bayesian models [59], [60], we evaluate uncertainty quality based on the standard metric Predictive Log Likelihood (PLL): $\mathbf{33.39}$ (Figure 9e). Following previous Bayesian models [59], [60], [102], [103], we evaluate uncertainty quality based on the standard metric Predictive Log Likelihood (PLL):

$$PLL(f(x), (\hat{y}, x)) = \log P(\hat{y}|f(x))$$ (19)

where $P(\hat{y}|f(x))$ is the probability of target $\hat{y}$ generated by a probabilistic model $f(x)$. In this paper, $f(x)$ corresponds to the multivariate Gaussian $N(\mu, \sigma^2)$ with predicted mean $\mu$ and standard deviation $\sigma$. $P(\hat{y}|f(x))$ is the Gaussian probability density function (PDF) evaluated at $\hat{y}$.

We report the average PLL of our approach coupled with EDSR [9] and CARN [66] in Table IX. The original EDSR/CARN networks fail to consider the model uncertainty, hence we use their average fitting error $E_D[|\hat{y} - \mu|]$ calculated on the training dataset $D$ to create a constant uncertainty estimation, which corresponds to baseline results in Table IX. Any improvement over the baseline reflects a sensible estimate of uncertainty. It is shown that PLL for high upscaling factors is $\mathbf{33.39}$ (Figure 9e). Following previous Bayesian models [59], [60], [102], [103], we evaluate uncertainty quality based on the standard metric Predictive Log Likelihood (PLL):

$$PLL(f(x), (\hat{y}, x)) = \log P(\hat{y}|f(x))$$ (19)

where $P(\hat{y}|f(x))$ is the probability of target $\hat{y}$ generated by a probabilistic model $f(x)$. In this paper, $f(x)$ corresponds to the multivariate Gaussian $N(\mu, \sigma^2)$ with predicted mean $\mu$ and standard deviation $\sigma$. $P(\hat{y}|f(x))$ is the Gaussian probability density function (PDF) evaluated at $\hat{y}$.
To improve the interpretability of the model uncertainty, we visualize the predicted $\sigma$ in Figure 9e. It is shown that the estimated uncertainty strongly correlates with the actual residual error $|\hat{y} - \mu_i|$. We also report the PSNR results between them to further evaluate the uncertainty quality fairly since PLL has been criticized for being vulnerable to outliers [104].

V. IMPLEMENTATION

In this section, we provide Python-style pseudo-codes to facilitate the implementation and further understanding of our approach by automatic differentiation, which has been well supported in mainstream deep-learning frameworks.

As discussed in Section III-B, the main limitation of $\ell_1$ loss is the degeneracy problem. By recalling Equation (10)

$$\min_{\mu} \mathbb{E}_z \left[ \| \hat{y} - (\mu + |\hat{y} - \mu| \ast z) \|_1 \right] ;$$

as $|\hat{y} - \mu| \rightarrow 0$, the variance term approaches zero then the proposed Equation (10) also gradually degrades into $\ell_1$ loss

$$\min_{\mu} \mathbb{E}_z \left[ \| \hat{y} - (\mu + 0^+ \ast z) \|_1 \right].$$

Note that large models lead to much better results than lightweight models, *i.e.*, the residual error $|\hat{y} - \mu|$ is much smaller in practice, which indicates that cumbersome models may not benefit as much as lightweight ones from Equation (10).

To resolve this issue, we present a simple heuristic method that only applies Equation (10) to “hard samples” with large $|\hat{y}_i - \mu_i|$. This setting shares a similar idea with Focal-Loss [105] and prevents the vast number of easy samples (*i.e.*, $|\hat{y}_i - \mu_i| \approx 0$) from overwhelming the sigma branch during training.

Algorithm 1 Pseudo-codes of Equation (10,13) in a Python style.

```python
# mu: super-resolved images [N, C, H, W]
n# hr: high resolution images (ground-truth) [N, C, H, W]
# sigma_pre: predicted standard deviations [N, C, H, W]
sigma_gt = abs(hr - mu) # real residual error
z = randn(hr.shape) # Gaussian random variables (4D tensor)
z = z*sigma_gt # Set z corresponding to small sigma_gt (i.e., easy samples)
# loss_main = mean(abs(mu + sigma_gt + z.detach() - hr)) # Equation (8)
# As in VAE, averaging the gradient over many samples of z during training can handle the expectation over z.
loss_main = mean(abs(mu + sigma_gt + z.detach())) # Equation (10)
loss_aux = mean(abs(sigma_pre - sigma_gt.detach())) # Equation (10)
return loss_main + beta*loss_aux # beta: penalty factor
```

*: element-wise product.

VI. CONCLUSIONS

In this paper, we present a novel learning process for single image super-resolution with neural networks. Instead of best fitting the ground-truth HR images, we introduce a posterior distribution $P(y|x; W)$ of underlying natural images $y$. By sampling from $P(y|x; W)$, the learning process serves as a multiple-valued mapping that better approximates the local manifold corresponding to the observed HR image. The experiments show that our method can produce SR images with minimal artifacts and the predicted standard deviation of $P(y|x; W)$, as a by-product, allows us to make meaningful estimations on the model uncertainty.

Though presented in the context of super-resolution, the formulated learning process can be extended to other image reconstruction scenarios with $y$ following different distributions, not limited to Gaussian. Finally, the performance may be further improved by a more complex sigma branch to predict the actual residual error, which helps to solve both HR image regression and uncertainty estimation simultaneously.

REFERENCES

[1] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV*, ser. Lecture Notes in Computer Science, vol. 8692. Springer, 2014, pp. 184–199. 1, 2

[2] J. Kim, J. K. Lee, and K. M. Lee, “Deeply-recursive convolutional network for image super-resolution,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV*, 2016.
