J. Nonlinear Var. Anal. 7 (2023), No. 4, pp. 505-526
Available online at http://jnva.biemdas.com
https://doi.org/10.23952/jnva.7.2023.4.04

# DROP-DIP: A SINGLE-IMAGE DENOISING METHOD BASED ON DEEP IMAGE PRIOR

ZHANG XUEDING, LI ZHEMIN, WANG HONGXIA*

*Department of Mathematics, College of Sciences, National University of Defense Technology, China*

**Abstract.** Over the past few years, deep learning methods have emerged as powerful image denoising tools. Among them, unsupervised deep learning without external training data is more practical and challenging. Reducing noisy overfitting is challenging due to single-image unsupervised learning is prone to overfitting. In this paper, we propose a method named drop-DIP combing Deep Image Prior (DIP) with drop-out for the first time to solve the above problems. In our method, we construct new network training pairs by performing drop-out training on the Bernoulli sampling of the input and output, and then construct a regularization term by using the corrected bias of the output and the generated prior. Finally, update the parameters through the Alternating Direction Method of Multipliers (ADMM) algorithm. Experiments demonstrate that drop-DIP can alleviate the overfitting difficulty in DIP, facilitate the early stopping of the network, and is applicable to different noise models. Furthermore, our method has good performance on Peak Signal to Noise Ratio (PSNR), Structural Similarity (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS) metrics validated by two different datasets.

**Keywords.** ADMM; Deep image prior; Drop-out; Image denoising; Unsurpervised learning.

## 1. INTRODUCTION

Image denoising is a fundamental problem in computer vision. There are many common noise types from the physics-based noise simulation model, including Gaussian noise, Poisson noise, Pulse noise, and so on [1]. A general noise model can be expressed in the form of

$$\mathbf{y} = \mathbf{x} + \delta, \tag{1.1}$$

where $\mathbf{y}$ is the observed image, $\mathbf{x}$ is the original clean image, and $\delta$ is the noise term. For example, when $\delta$ is multiplicative noise, there is $\delta = (\delta_{\mathbf{x}} - \mathbf{I}) \cdot \mathbf{x}$.

Traditional denoising algorithms are usually modeled for noise types, and the corresponding noise types are single [2]. In recent years, with the development of deep learning, there have been many methods using convolutional neural networks (CNN) for denoising. Deep learning can adapt to multiple noise types. For example, Dn-CNN [3] uses CNN to learn the noise part in the image. Subsequent work builds a supervised learning mechanism that uses noisy-clean training pairs constructed from many images to train the network [4, 5, 6]. However, there are two main difficulties with supervised learning. First, CNNs are not good at generalizing

*Corresponding author.
E-mail address: wanghongxia@nudt.edu.cn (W. Hongxia).
Received July 4, 2022; Accepted September 6, 2022.

©2023 Journal of Nonlinear and Variational Analysis

to images that are quite different from the training dataset. Second, collecting a large training dataset is often challenging in reality.

In order to solve the difficulties, researchers have developed a series of unsupervised learning methods, which only need to use a single noisy image to achieve denoising. In the following work, we assume that CNN is denoted by $f_\theta$ with parameters $\theta$. According to different inputs, unsupervised learning methods can be divided into two categories. The first category uses images as inputs to denoise in the image domain [7, 8, 9, 10, 11, 12]. Such as Noise2Noise [13], which takes two independent noisy observations $\mathbf{y_1}, \mathbf{y_2}$ with the same ground-truth $\mathbf{x}$ as training pairs (in some works $\mathbf{y_1}, \mathbf{y_2}$ are the samplings of a single image $\mathbf{y}$), and then train the network by minimizing the loss $\mathscr{L}(f_\theta(\mathbf{y_1}) - \mathbf{y_2})$. The final denoised image is $f_{\theta^*}(\mathbf{y})$. The second category uses parameters in the generation domain as input[14, 15, 16], such as DIP [17]. The network input is a randomly generated tensor or a fixed $\mathbf{z}$ such as image coordinates. At this time, the loss function $\mathscr{L}(f_\theta(\mathbf{z}) - \mathbf{y})$ is minimized based on a single noisy image $\mathbf{y}$. It is equivalent to using the network to obtain a generated image similar to $\mathbf{y}$. The generated image $f_{\theta^*}(\mathbf{z})$ with early stopping is taken as the denoised image because fitting a single image with a network is prone to overfitting.

In this paper, we try to sample both the input and output of the DIP network to obtain $(g_1(\mathbf{z}), g_2(\mathbf{y}))$ data pairs, where $g_1, g_2$ are specific sampling methods given below. The operation of sampling a single image is called Image drop-out. This paper proposes an improved denoising method called drop-DIP, combining DIP with Image drop-out. The method sets training data pairs $(g_1(\mathbf{z}), g_2(\mathbf{y}))$ by sampling a single image and uses an explicit regularization to correct for training biases between the network output and the DIP generation prior. Finally, the network is iteratively updated using the ADMM algorithm [18].

The main contributions of this paper are as follows:

(1) The training data pairs are obtained by applying the random mask to a single image, and the regularization term that represents the difference of the ground truth value under different sampling is derived from alleviating the overfitting of the network. This work combining DIP with image-level drop-out is innovative.
(2) $g_1(\mathbf{z})$ and $g_2(\mathbf{y})$ constitute new training pairs for the network. This image drop-out sampling scheme is a similar but different method to Self2Self [9] and Neighbor2Neighbor [12].
(3) By improving the training strategy and updating the network by the ADMM algorithm iteratively, our method can alleviate the characteristics of rapid overfitting of DIP [17] and facilitate early termination. Moreover, drop-DIP theoretically explains the associated fitting error. Compared with now existing unsupervised denoising algorithms, this method has better denoising performance. At the same time, it performs better under different noise types than state-of-arts unsupervised learning methods and has specific potential value in practical application scenarios.

## 2. Related Work

Many traditional image denoising methods exist, including spatial or frequency domain denoising, sparse image approximation, and so on [19]. Some of them, such as non-local means [20] and BM3D [21], make full use of the non-local self-similarity of the image. In order to fit different noise types, people developed deep learning for denoising. Jain et al. first used CNN

for image denoising [22]. Zhang et al. estimated residuals of noisy inputs and corresponding clean images by Dn-CNN [3]. Their Dn-CNN achieves higher metrics on standard benchmarks. Inspired by the success of Dn-CNN, Gu et al. proposed the fast denoising network (FDnet) to seek a better trade-off between denoising performance and speed [23]. Supervised learning is to train CNNs using noisy-clean image groups constructed from many datasets. When considering single image denoising, we need to use unsupervised learning. Next we introduce two categories of related works about unsupervised learning.

2.1. **Image drop-out.** The first category we call Image drop-out. The difficulty of unsupervised learning is obtaining training data, and we can collect training pairs on the image domain through Image drop-out. Noise2Noise [13] uses independent and identically distributed noise observations with the same ground truth to construct training pairs, but obtaining noisy images in practical applications is challenging. Therefore, researchers developed data augmentation methods such as Self2Self [9] and Neighbor2Neighbor [12]. Moreover, these methods obtain training pairs $(g_1(\mathbf{y}), g_2(\mathbf{y}))$ from a single image through different sampling methods (denoted by $g_1, g_2$, which will be given below).

Self2Self(S2S) selects the Bernoulli sampling of a single image as the training set, and the corresponding complementary sampling as the test set to minimize the loss function:

$$\begin{aligned}\mathscr{L}_{S2S}(\theta) &= \mathbb{E}_{g_1,g_2} \left\| f_\theta\left(g_1(\mathbf{y})\right) - g_2(\mathbf{y}) \right\|_{g_2}^2 \\ &\approx \mathbb{E}_{g_1,g_2} \left\| f_\theta\left(g_1(\mathbf{x}+\delta)\right) - \mathbf{x} \right\|_{g_2}^2 .\end{aligned} \tag{2.1}$$

The input-target training pair is $(g_1(\mathbf{y}), g_2(\mathbf{y}))$, where $g_1(\mathbf{y}) = g_1 \odot \mathbf{y}$ is the partial mask of the single noisy image $\mathbf{y}$,

$$g_1[i,j] = \begin{cases} 1, & \text{with probability } p, \\ 0, & \text{with probability } 1-p. \end{cases} \tag{2.2}$$

And $g_2(\mathbf{y}) = (\mathbf{1} - g_1) \odot y$ is the complementary sample corresponding to $g_1$, and $\|\cdot\|_{g_2}^2 = \|(\mathbf{1} - g_1) \odot \cdot\|_2^2$.

Neighbor2Neighbor(N2N) samples the adjacent pixels of a single image to form training pairs $(g_1(\mathbf{y}), g_2(\mathbf{y}))$. Then train the network by minimizing the loss function:

$$\begin{aligned}\mathscr{L}_{N2N}(\theta) = &\mathbb{E}_{g_1,g_2} \left\| f_\theta\left(g_1(\mathbf{y})\right) - g_2(\mathbf{y}) \right\|_2^2 \\ &+ \lambda \mathbb{E}_{g_1,g_2} \left\| f_\theta\left(g_1(\mathbf{y})\right) - g_2(\mathbf{y}) - g_1\left(f_\theta(\mathbf{y})\right) + g_2\left(f_\theta(\mathbf{y})\right) \right\|_2^2 .\end{aligned} \tag{2.3}$$

The first term is implicit fidelity term, and the second regularization term is to correct the training bias of different samples. This operation for sub-sampling a single image is called Image drop-out, which essentially takes advantage of the insensitivity of the neural network to high-frequency information to reduce the noise representation of the network output. Then the network output is close to the ground truth of the image during the parameter update process. Image drop-out provides a way to build training pairs from a single image. However, Neighbor2Neighbor utilizes external datasets in nearest neighbor sampling. Finding a new way to construct training pairs is essential. Next, we provide a new method to construct by random sampling using DIP generation prior.

2.2. **Generative prior based on DIP.** The second category we call generative prior. Unlike the methods in Section 2.1, the network $f_\theta$ becomes a function from generating parameters to generating images in DIP. Generative priors, also known as implicit fidelity terms, refer to the generated images in the process of fitting the image with the neural network. Unsupervised learning methods that train networks on generative domains construct training pairs $(\mathbf{z},\mathbf{y})$ from random tensors $\mathbf{z}$. DIP [17] proposed the concept of generative prior, which uses a $d$-layer generator network to represent the parameterized function $f_\theta(\mathbf{z}):\mathbb{R}^k\to\mathbb{R}^n$ that maps the code vector $\mathbf{z}\in\mathbb{R}^k$ to the image $\mathbf{y}\in\mathbb{R}^n$, $k<n$ in the following way:

$$f_\theta(\mathbf{z})=\mathrm{ReLU}\left(W_d(\mathbf{c})\ldots\mathrm{ReLU}\left(W_2(\mathbf{c})\,\mathrm{ReLU}\left(W_1(\mathbf{c})\mathbf{z}\right)\right)\ldots\right), \tag{2.4}$$

where $\mathrm{ReLU}(\mathbf{z})=\max(\mathbf{z},0)$ applies entrywise, $W_i\in\mathbb{R}^{n_i\times n_{i-1}}$ are the operators performing a convolution with the kernel $\mathbf{c}$ in the $i$-th layer, $n_i$ is the number of neurons in the $i$-th layer, $k=n_0$, $n=n_d$. Next, train the weights $W_i$ of the network by minimizing the loss function

$$\theta^*=\underset{\theta}{\mathrm{argmin}}\mathscr{L}_{imp}\left(f_\theta(\mathbf{z}),\mathbf{y}\right),\hat{\mathbf{x}}=f_{\theta^*}(\mathbf{z}), \tag{2.5}$$

where $\mathscr{L}_{imp}$ is the distance between the generated prior $f_\theta(\mathbf{z})$ to $\mathbf{y}$. It is implicit because the generative prior of denoised image is captured by the network $f_\theta$.

In addition to Ulyanov's DIP [17], there are currently several ways to exploit generative prior. Heckel et al. proposed Deep Decoder (DD) [15], which chooses to use the decoder to build an under-parameterized network based on the DIP work, thereby limiting the network's ability to express noise in the image and then realizing the denoising goal. Sitzman et al. [14] proposed that SIREN constructed implicit neural representations to establish a function mapping from coordinates to pixel values. The activation function selected a variant of the trigonometric sin function. Compared with the commonly used ReLU function, it has a better-detailed expression ability and preserves the output high-order gradient information. Fan et al. proposed DMF [16], a matrix factorization model based on a deep neural network structure. The input is a low-dimensional unknown latent variable, and the output is a partially observed variable. This method has significant advantages in matrix completion and image inpainting. Figure 1 shows the PSNR variation curve during the iterative process of several generating prior methods.

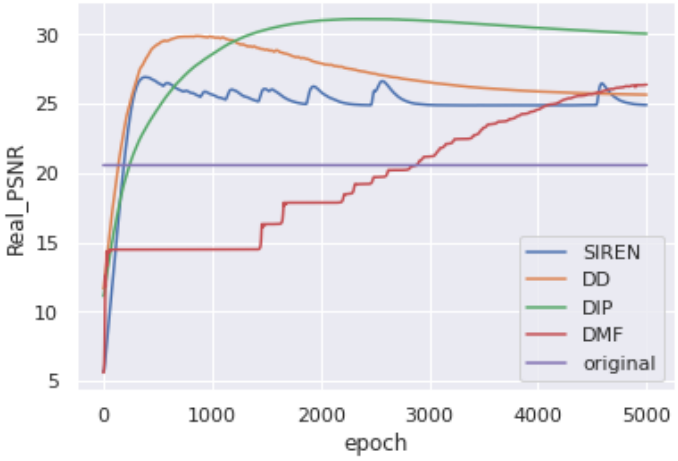



FIGURE 1. Comparison of some recent proposed denoising methods by generative priors, where 'original' is the PSNR of $\mathbf{y}$.

In Figure 1, we demonstrate how several methods mentioned earlier implicitly represent images. All methods can achieve denoising goals and improve image quality with the increase of iteration steps, among which the DIP method is the best. Based on this fact, we choose DIP as the generation before achieving a better denoising effect.

## 3. Proposed Method

We utilize random sampling of DIP generated parameters to construct training data. Ulyanov [17] pointed out that DIP network $f_{\theta^*}(\mathbf{z})$ is stable respect to a small perturbation of input $\mathbf{z}$. Jo et al. [24] proposed that Stochastic temporal ensembling (STE) can be used to improve the fitting performance of DIP by adding a disturbance to the input of the network [25]. In fact, experimental results demonstrate that, for a trained DIP $f_{\theta^*}(\mathbf{z})$, $f_{\theta^*}(g(\mathbf{z}))$ is less different from $f_{\theta^*}(\mathbf{z})$, where $g$ is a random mask, $p = 0.9$. And the PSNR of $f_{\theta^*}(g(\mathbf{z}))$ is 0.15db higher than $f_{\theta^*}(\mathbf{z})$ in experiment. So we can build pairs of data by random sampling.

We replace the single $(\mathbf{z},\mathbf{y})$ with $(g_1(\mathbf{z}),g_2(\mathbf{y}))$ pairs, where $g_1,g_2$ are different Bernoulli samples to achieve multiple sets of random masks. Next, we combine the DIP generation prior with the loss function (2.3) in Neighbor2Neighbor, and obtain the network loss function with respect to the parameter $\theta$ as:

$$\mathscr{L}(\theta) = \underbrace{\mathbb{E}_{g_1,g_2}\|f_\theta(g_1(\mathbf{z})) - g_2(\mathbf{y})\|_{g_2}^2}_{\mathscr{L}_{imp}(\theta)} + \lambda \underbrace{\mathbb{E}_{g_1,g_2}\|g_2(f_\theta(g_1(\mathbf{z}))) - g_2(\mathbf{y}) - (f_\theta(g_1(\mathbf{z})) - f_\theta(\mathbf{z}))\|_2^2}_{\mathscr{L}_{reg}(\theta)}. \tag{3.1}$$

where $\|\cdot\|_{g_2}^2 = \|g_2 \odot \cdot\|_2^2$. The first item is an implicit fidelity term, the second item is an explicit regularization, and $f_\theta(\mathbf{z})$ is the generation network of DIP after freezing the network parameters. Let us denote by $\mathscr{N}(\cdot)$ an image normalization operation. We will normalize the final denoised estimate. We have

$$\mathscr{N}(\mathbb{E}_{g_1,g_2} g_2(f_\theta(g_1(\mathbf{z})))) = \mathscr{N}(p_2 \mathbb{E}_{g_1} f_\theta(g_1(\mathbf{z}))) \approx \mathscr{N}(p_2 \mathbf{x}) = \mathbf{x},$$

so we can use $\mathbb{E}_{g_1,g_2} g_2(f_\theta(g_1(\mathbf{z})))$ as the final denoising estimate.

### 3.1. Implicit fidelity term and Explicit regularization term.

Below we introduce the role of the two terms in (3.1) during training. First, compute $\mathscr{L}_{imp}(\theta)$. Even though the loss for each training pair is only measured on those pixels sampled by $g_2$, $\mathbb{E}_{g_2}$ measures the difference across all image pixels since the random pixel mask $g_2$ is randomly selected using a Bernoulli process.

Training with the pairs $(g_1(\mathbf{z}),g_2(\mathbf{y}))$ is closely related to training with $(g_1(\mathbf{z}),\mathbf{x})$. Furthermore, the following proposition is established.

**Proposition 3.1.** *Assume $\mathbf{y} = \mathbf{x} + \delta$ with noise $\delta$. The implicit fidelity term $\mathscr{L}_{imp}(\theta)$ satisfies*

$$\begin{aligned} \mathscr{L}_{imp}(\theta) &= \mathbb{E}_{g_1,g_2}\|f_\theta(g_1(\mathbf{z})) - g_2(\mathbf{y})\|_{g_2}^2 \\ &= \mathbb{E}_{g_1,g_2}\|f_\theta(g_1(\mathbf{z})) - \mathbf{x}\|_{g_2}^2 + \mathbb{E}_{g_1,g_2}\|\delta\|_{g_2}^2 - 2\,\mathrm{tr}(\mu^T \mathbf{r}), \end{aligned} \tag{3.2}$$

*where $\mu = \mathbb{E}_{g_1,g_2}(g_2 \odot \delta)$ and $\mathbf{r} = \mathbb{E}_{g_1,g_2}(g_2 \odot (f_\theta(g_1(\mathbf{z})) - \mathbf{x}))$.*

*Proof.* See Appendix A . □

Assume that the mean of noise $\delta$ is zero, and the expectation of the loss function with respect to $\delta$ is the same as

$$\mathbb{E}_\delta \mathscr{L}_{imp} = \mathbb{E}_{g_1,g_2} \|f_\theta(g_1(\mathbf{z})) - \mathbf{x}\|_{g_2}^2 + \mathbb{E}_{g_1,g_2} \|Var(\delta)\|_{g_2}^2 + \mathbb{E}_{g_1,g_2} \|\mathbb{E}_\delta \delta\|_{g_2}^2 - 2\,\mathrm{tr}(\mathbb{E}_\delta(\mu)^T \mathbf{r}),$$

where the last two terms are 0, and the second term is a fixed value when the noise distribution is determined. So minimize $\mathscr{L}_{imp}(\theta)$ will enforce $f_\theta(g_1(\mathbf{z})) \approx \mathbf{x}$.

Similar to Neighbor2Neighbor [12], $\mathscr{L}_{reg}(\theta)$ is to correct the deviation between the DIP generation prior and the network output. Define $\varepsilon = f_\theta(g_1(\mathbf{z})) - f_\theta(\mathbf{z})$. $\mathscr{L}_{reg}(\theta)$ can be reduced to

$$\begin{aligned} \mathscr{L}_{reg}(\theta) =& \mathbb{E}_{g_1,g_2} \|g_2(f_\theta(g_1(\mathbf{z}))) - g_2(\mathbf{y}) - (f_\theta(g_1(\mathbf{z})) - f_\theta(\mathbf{z}))\|_2^2 \\ =& \mathbb{E}_{g_1,g_2} \left\|(f_\theta(g_1(\mathbf{z})) - \mathbf{y})_{g_2} - \varepsilon\right\|_2^2. \end{aligned} \tag{3.3}$$

If $\varepsilon = 0$, the deviation between the network output $f_\theta(g_1(\mathbf{z}))$ and DIP generation $f_\theta(\mathbf{z})$ is zero, and the algorithm is a special case of DIP. However, if $\varepsilon \neq 0$ and $f_\theta(g_1(\mathbf{z})) = \mathbf{x} + \varepsilon_1$, then $f_\theta(g_1(\mathbf{z})) - \mathbf{y} = \varepsilon_1 - \delta$. At this time, (3.3) is equal to

$$\begin{aligned} &\mathbb{E}_{g_1,g_2} \|g_2(\varepsilon_1 - \delta) - \varepsilon\|_2^2 \\ &= \mathbb{E}_{g_1,g_2} \|g_2(\varepsilon_1) - \varepsilon\|_2^2 + \mathbb{E}_{g_1,g_2} \|\delta\|_{g_2}^2 - 2tr(\mu^T \mathbb{E}_{g_1,g_2}(g_2(\varepsilon_1) - \varepsilon)). \end{aligned} \tag{3.4}$$

The expectation of the loss function (3.4) with respect to noise is the same as

$$\begin{aligned} &\mathbb{E}_\delta \mathscr{L}_{reg} = \mathbb{E}_\delta \mathbb{E}_{g_1,g_2} \|g_2(\varepsilon_1 - \delta) - \varepsilon\|_2^2 \\ &= \mathbb{E}_{g_1,g_2} \|g_2(\varepsilon_1) - \varepsilon\|_2^2 + \mathbb{E}_{g_1,g_2} \|Var(\delta)\|_{g_2}^2 + \mathbb{E}_{g_1,g_2} \|\mathbb{E}_\delta \delta\|_{g_2}^2 - 2\,\mathrm{tr}(\mathbb{E}_\delta(\mu)^T \mathbb{E}_{g_1,g_2}(g_2(\varepsilon_1) - \varepsilon)), \end{aligned}$$

where the last two terms are 0, and the second term is a fixed value when the noise distribution is determined. Likewise, due to the normalization operation, we have

$$\mathscr{N}(\mathbb{E}_{g_1,g_2} g_2(\varepsilon_1)) = \mathscr{N}(p_2 \varepsilon_1) = \varepsilon_1.$$

So minimizing (3.4) will enforce $\varepsilon \approx \varepsilon_1$, and $f_\theta(g_1(\mathbf{z})) \to x, \varepsilon_1 \to 0$ when we minimize implicit fidelity term (3.2). Then $\varepsilon \to 0$. In summary, minimizing $\mathscr{L}_{reg}(\theta)$ can ensure that $f_\theta(g_1(\mathbf{z})) \to f_\theta(\mathbf{z})$ during network training.

3.2. **The fitting error of drop-DIP.** We gave the drop-DIP training settings and loss function in the previous sections. In this section, we give the fitting error of drop-DIP and then verify the theoretical basis of neural network denoising and prove that the drop-DIP denoising effect is better than DIP.

Consider image denoising problem (1.1). To explore the network's fitting error, we approximate the CNN's nonlinear training process as a linear process [26]. It is assumed that the network parameter $\theta$ changes relatively little concerning the initialization parameter $\theta_0$ during training. Then the parameter update process can be approximated by a linearization process. In the theoretical analysis, we consider the simplest gradient descent method to solve the following regularization problems

$$\min_\theta \mathbb{E}_{g_1,g_2} \|f_\theta(g_1(\mathbf{z})) - g_2(\mathbf{y})\|_{g_2}^2 + \lambda \|g_2(f_\theta(g_1(\mathbf{z}))) - g_2(\mathbf{y}) - (f_\theta(g_1(\mathbf{z})) - f_\theta(\mathbf{z}))\|_2^2. \tag{3.5}$$

To simplify the analysis, we only focus on the training process by one training pair. The upper bound of the error is obtained by any single sampling, and the error corresponding to the

sampling expectation should still be less than this upper bound. Consider

$$\|f_\theta(g_1(\mathbf{z}))-g_2(\mathbf{y})\|_{g_2}^2=\|f_\theta(g_1(\mathbf{z}))-\mathbf{y}\|_{g_2}^2\le\|f_\theta(g_1(\mathbf{z}))-\mathbf{y}\|_2^2,$$

we can find an upper bound on the error on the right-hand side of the inequality. So (3.5) can be simplified to

$$\min_\theta\|f_\theta(g_1(\mathbf{z}))-\mathbf{y}\|_2^2+\lambda\left\|g_2(f_\theta(g_1(\mathbf{z})))-g_2(\mathbf{y})-(f_\theta(g_1(\mathbf{z}))-f_\theta(\mathbf{z}))\right\|_2^2. \tag{3.6}$$

Assume that the network parameters of the $k$th step in the training process are $\theta_k$, and the Jacobian matrix of the network is $\mathscr{J}(\theta_k)$, where $[\mathscr{J}(\theta_k)]_{i,j}=\frac{\partial f_i(\theta_k)}{\partial\theta_{kj}}$. It can be obtained that the Taylor expansion of $f_{\theta_k}(g_1(\mathbf{z}))$ in $\theta_0$ is $f_{\theta_k}(g_1(\mathbf{z}))\approx f_{\theta_0}(g_1(\mathbf{z}))+\mathscr{J}(\theta_0)(\theta_k-\theta_0)$. Let $\mathbf{J}:=\mathscr{J}(\theta_0)$. For $f_\theta(\mathbf{z})$, we also have $f_{\theta_k}(\mathbf{z})\approx f_{\theta_0}(\mathbf{z})+\mathbf{J}(\theta_k-\theta_0)$. It is because freezing the network parameters can use $\mathscr{J}(\theta_k)$ to represent the Jacobian matrix of $f_\theta(\mathbf{z})$. Due to the randomness of Bernoulli sampling, we perform theoretical analysis on an arbitrary sampling. Further, (3.5) can be transformed into the following linear approximation problem

$$\begin{aligned}\min_\theta\mathscr{L}_{\text{lin}}(\theta)=&\left\|f_{\theta_0}(g_1(\mathbf{z}))-y+\mathbf{J}(\theta-\theta_0)\right\|_2^2\\&+\lambda\left\|g_2(f_{\theta_0}(g_1(\mathbf{z})))-g_2(\mathbf{y})-(f_{\theta_0}(g_1(\mathbf{z}))-f_{\theta_0}(\mathbf{z}))+g_2(\mathbf{J})(\theta-\theta_0)\right\|_2^2,\end{aligned} \tag{3.7}$$

At this point, we have transformed the drop-DIP loss function into an approximately linear problem. Next, we simplify the neural network so that we can use the gradient descent method to analyze the change of $\theta$ in the iterative process and give the fitting error of the network for $\mathbf{x}$ during the $k$ step iteration.

According to Heckel's Theorem 2 [26], we can obtain an upper bound of the error between $f_\theta(g_1(\mathbf{z}))$ and $\mathbf{x}$. Consider a simplified two-layer network $f_\theta(g_1(\mathbf{z}))=ReLU(\mathbf{U}g_1(\mathbf{z})\theta)\mathbf{v}$ with an activation function $ReLU$ and upsampling layer $\mathbf{U}$ and one convolution operator $\theta$, where $\mathbf{z}\in\mathbb{R}^{n\times n}$is the fixed input, $\theta\in\mathbb{R}^{n\times k}$is the convolution opertor, $\mathbf{U}$ is the unsampling opertor, $\mathbf{v}\in\mathbb{R}^{k\times1}=[1,\dots,1,-1,\dots,-1]/\sqrt{k}$. Then perform singular value decomposition on the Jacobi matrix $\mathbf{J}$, $\mathbf{J}=\mathbf{W}\Sigma\mathbf{W}^T$, where$\mathbf{W}=[\mathbf{w}_1,\dots,\mathbf{w}_n],\Sigma=diag\{\sigma_1,\dots,\sigma_n\}$ with $\sigma_1\ge\sigma_2\ge\dots\ge\sigma_n$, $\sigma\in(0,1)$ decays sharply near the $p$ singular value. Assume that the ratio $\sigma_p/\sigma_{p+1}$ is sufficiently large. Then minimize (3.7) using gradient descent $\theta_{k+1}=\theta_k-\eta\nabla\mathscr{L}_{lin}(\theta_k)$, where $\eta$ is a fixed learning rate. The fitting error of drop-DIP at the k-th iteration is obtained as follows.

**Theorem 3.1.** *Assume we have the same setting as Heckel's Theorem 2* [26]. *The fitting error between $f_\theta(g_1(\mathbf{z}))$ and $\mathbf{x}$ is upper bounded by*

$$\begin{aligned}\left\|f_{\theta_k}(g_1(\mathbf{z}))-\mathbf{x}\right\|_2\le&\underbrace{\|\left(1-\eta\sigma_p^2\right)^k\bar{g}_2(\mathbf{x})+\left(1-\eta(\lambda+1)\sigma_p^2\right)^k g_2(\mathbf{x})\|_2}_{\textit{error in fitting signal}}\\&+\underbrace{\left(\sum_{i=1}^n\left(\left(1-\eta\sigma_i^2\right)^k-1\right)^2\langle\mathbf{w}_i,\bar{g}_2(\delta)\rangle^2+\sum_{i=1}^n\left(\left(1-\eta(\lambda+1)\sigma_i^2\right)^k-1\right)^2\langle\mathbf{w}_i,g_2(\delta)\rangle^2\right)^{1/2}}_{\textit{noise fitting}}\\&+\xi\|\mathbf{y}\|_2\end{aligned} \tag{3.8}$$

*with high probability, where $\bar{g}_2=\boldsymbol{1}-g_2,\xi\in(0,\sigma_p/\sigma_1]$.*

*Proof.* See Appendix B. □

(3.8) describes the training process of the simplified drop-DIP model by gradient descent update. In particular, it explains why drop-DIP adapts to noise-free signal $x$ much faster than noise $\delta$, so we can achieve image denoising through CNN training. The first term in (3.8) is the fitting error with respect to the signal $x$, the second term is the fitting of the noise $\delta$, and the third term is the fixed value of the observed signal $y$. When $x$ is located in the space spanned by the eigenvectors corresponding to the top $p$ largest singular values, after a few iterations, most of $x$ is fitted ($\left(1-\eta\sigma_p^2\right)^k$ and $\left(1-\eta(\lambda+1)\sigma_p^2\right)^k$ is small). And most of the noise has not been fitted since

$$\left(\left(1-\eta\sigma_i^2\right)^k-1\right)^2\approx 0,$$

and

$$\left(\left(1-\eta(\lambda+1)\sigma_i^2\right)^k-1\right)^2\approx 0$$

for $i=p+1,\ldots,n$. Therefore, we obtain a theoretical validation of the drop-DIP network for denoising.

Next Let us compare (3.8) with the original DIP fitting error.The loss function in the original DIP is $\|f_\theta(\mathbf{z})-\mathbf{y}\|_2^2$. The fitting error of DIP is

$$\left\|f_{\theta_k}(\mathbf{z})-\mathbf{x}\right\|_2\leq \underbrace{\left(1-\eta\sigma_p^2\right)^k\|\mathbf{x}\|_2}_{\text{error in fitting signal}}+\underbrace{\sqrt{\sum_{i=1}^n\left(\left(1-\eta\sigma_i^2\right)^k-1\right)^2\langle\mathbf{w}_i,\delta\rangle^2}+\xi\|\mathbf{y}\|_2}_{\text{noise fitting}}. \tag{3.9}$$

Comparing (3.8) with (3.9), we see that the two bounds are equal when $\lambda=0$ because $\bar{g_2}+g_2=\mathbf{1}$. For the error in fitting signal, we have

$$\|\left(1-\eta\sigma_p^2\right)^k\bar{g_2}(\mathbf{x})+\left(1-\eta(\lambda+1)\sigma_p^2\right)^k g_2(\mathbf{x})\|_2\leq\left(1-\eta\sigma_p^2\right)^k\|\mathbf{x}\|_2.$$

For the noise fitting, we have

$$\left(\sum_{i=1}^n\left(\left(1-\eta\sigma_i^2\right)^k-1\right)^2\langle\mathbf{w}_i,\bar{g_2}(\delta)\rangle^2+\sum_{i=1}^n\left(\left(1-\eta\sigma_i^2\right)^k-1\right)^2\langle\mathbf{w}_i,g_2(\delta)\rangle^2\right)^{1/2}$$
$$\leq\sqrt{\sum_{i=1}^n\left(\left(1-\eta\sigma_i^2\right)^k-1\right)^2\langle\mathbf{w}_i,\delta\rangle^2}.$$

We can find that when $\lambda>0$,

$$\left\|f_{\theta_k}(g_1(\mathbf{z}))-\mathbf{x}\right\|_2\leq\left\|f_{\theta_k}(\mathbf{z})-\mathbf{x}\right\|_2.$$

Therefore, our drop-DIP method after adding the regularization term can reduce the fitting errors of DIP. The result in Experiment 4.3 also proves that as the number of iterations increases, the fitting error of drop-DIP is less than that of DIP.

However, (3.8) demonstrates that the fitting error will not disappear during training but will increase through fitting noise, so the network training process needs early stopping. In this paper, we set the network to stop early for the number of training steps when the distance between the output and the generated prior is less than a threshold $\varepsilon$. That is, stopping training at step $k$, we have $\mathbb{E}_{g_1,g_2}\|f_{\theta_k}(\mathbf{z})-f_{\theta_k}(g_1(\mathbf{z}))\|_{g_2}<\varepsilon$.

3.3. **Unsupervised training with ADMM.** We can find the optimal of (3.1) through ADMM algorithm [18], which is widely used in image restoration. (3.1) can be rewritten as:

$$\begin{aligned} &\arg\min_{\theta,\mathbf{t}} \frac{1}{2}\mathbb{E}_{g_1,g_2}\left\|f_\theta(g_1(\mathbf{z}))-g_2(\mathbf{y})\right\|_{g_2}^2+\frac{1}{2}\|\mathbf{t}\|_2^2 \\ &\text{s.t. } \mathbb{E}_{g_1,g_2}\left\|g_2(f_\theta(g_1(\mathbf{z})))-g_2(\mathbf{y})-(f_\theta(g_1(\mathbf{z}))-f_\theta(\mathbf{z}))-\mathbf{t}\right\|_2=0. \end{aligned} \tag{3.10}$$

The augmented Lagrangian function for (3.10) is:

$$\begin{aligned} \mathscr{L}(\theta,\mathbf{t},\mathbf{u}) = &\mathbb{E}_{g_1,g_2}\frac{1}{2}\left\|f_\theta(g_1(\mathbf{z}))-g_2(\mathbf{y})\right\|_{g_2}^2+\frac{1}{2}\|\mathbf{t}\|_2^2 \\ &+\frac{\lambda}{2}\left\|g_2(f_\theta(g_1(\mathbf{z})))-g_2(\mathbf{y})-(f_\theta(g_1(\mathbf{z}))-f_\theta(\mathbf{z}))-\mathbf{t}\right\|^2 \\ &+<\mathbf{u},g_2(f_\theta(g_1(\mathbf{z})))-g_2(\mathbf{y})-(f_\theta(g_1(\mathbf{z}))-f_\theta(\mathbf{z}))-\mathbf{t}>, \end{aligned} \tag{3.11}$$

where $\lambda>0$ is a scalar, and $u$ is the Lagrangian parameter related to the constraint. We fix two variables in the Lagrangian function, update the other variable, and then find its optimal one. Upon suitable initialization of the variables involved, the $k$th iteration of ADMM reads as follows:

$$\begin{aligned} \theta_{k+1} = \arg\min_{\theta}&\mathbb{E}_{g_1,g_2}\frac{1}{2}\left\|f_\theta(g_1(\mathbf{z}))-g_2(\mathbf{y})\right\|_{g_2}^2 \\ &+\frac{\lambda}{2}\left\|g_2(f_\theta(g_1(\mathbf{z})))-g_2(\mathbf{y})-(f_\theta(g_1(\mathbf{z}))-f_\theta(\mathbf{z}))-\mathbf{t}_k+\frac{\mathbf{u}_k}{\lambda}\right\|_2^2, \end{aligned} \tag{3.12}$$

$$\begin{aligned} \mathbf{t}_{k+1} = \arg\min_{\mathbf{t}}&\mathbb{E}_{g_1,g_2}\frac{1}{2}\|\mathbf{t}\|_2^2 \\ &+\frac{\lambda}{2}\left\|\mathbf{t}-\left(g_2(f_{\theta_{k+1}}(g_1(\mathbf{z})))-g_2(\mathbf{y})-(f_{\theta_{k+1}}(g_1(\mathbf{z}))-f_{\theta_{k+1}}(\mathbf{z}))+\frac{\mathbf{u}_k}{\lambda}\right)\right\|_2^2, \end{aligned} \tag{3.13}$$

and

$$\mathbf{u}_{k+1}=\mathbf{u}_k+\lambda\mathbb{E}_{g_1,g_2}g_2(f_{\theta_{k+1}}(g_1(\mathbf{z})))-g_2(\mathbf{y})-(f_{\theta_{k+1}}(g_1(\mathbf{z}))-f_{\theta_{k+1}}(\mathbf{z}))-\mathbf{t}_{k+1}. \tag{3.14}$$

(3.12) can be solved by the gradient-based method. In particular, we use the Radam iterative scheme [27] with an adaptive learning rate. The solution of (3.13) can be expressed explicitly. The second problem (3.13) is differentiable so that we can derive the corresponding minimum value from the formula (3.13) with respect to $\mathbf{t}$. The third problem (3.14) can be directly substituted into known values. As a result, the unsupervised training strategy of our proposed drop-DIP algorithm is summarized in Algorithm 1.

**Algorithm 1** Drop-DIP

**Input:** a single noisy image $\mathbf{y}$; the network parameter $\theta_0$; with probability$p_1, p_2$ for Bernoulli sampling $g_1, g_2$; the network input $\mathbf{z}$; hyper-parameter $\lambda, \varepsilon, K, M, N$.
**Output:** denoised image $\mathbb{E}_{g_1,g_2} g_2(f_\theta(g_1(\mathbf{z})))$

```
for k = 0 to K do
  for i = 1 to N and j = 1 to M do
    Bernoulli sample the input g_{1i}(z) and images g_{2j}(y)
    for the network f_{θ_k}, derive the denoised image f_{θ_k}(g_{1i}(z))
  end for i, j
  update θ_k via (3.12)by Radam optimizer
  update t_k according to (3.13)
  update u_k according to (3.14)
  if E_{g1,g2} ||f_{θ_k}(z) − f_{θ_k}(g_1(z))||_{g2} < ε, then
    break
    output E_{g1,g2} g_2(f_θ(g_1(z))) = (1/N)(1/M) Σ_{i=1}^{N} Σ_{j=1}^{M} g_{2j}(f_θ(g_{1i}(z)))
  end if
end for k
```

## 4. Experiments

In this section, we present the results of some numerical experiments to verify the performance of the proposed drop-DIP method compared to the traditional denoising method BM3D [21] and the unsupervised learning denoising methods DIP [17], Self2Self [9] and DIP-SURE [24] methods. Various noise models of (1.1) are considered, such as Gaussian noise, Pulse noise (salt and pepper noise), and Poisson noise. The network structure we selected in the experiment is shown in Figure 2, which is a formal improvement based on the U-Net network structure. In practical experiments, for the sampling probability $p$, too small $p$ (close to 0) will drop out lots of image information so that the network cannot fit the image correctly, and too large $p$ (close to 1) will make drop-DIP approach DIP. $p_1$ and $p_2$ from $g_1, g_2$ are usually set to 0.8 and 0.9 respectively to avoid losing too much image information. In the following subsections, we conduct image denoising experiments both on gray image dataset Set12 and color image dataset CSet9 [25]. Set12 and CSet9 are classic datasets in image denoising.

In the subsequent network training process, we choose the Radam optimizer for parameter update, and the learning rate is $1e-3$. To evaluate the performance of image denoising, we choose PSNR, SSIM, and LPIPS. PSNR values are high for over-smoothed images, but image details are lacking. On this basis, the LPIPS [28] indicator can pay attention to image details closer to human intuition. The author directly calculates this indicator based on the relevant procedures of AlexNet's public pre-trained weights.

### 4.1. The influence of $\lambda$.

The drop-DIP algorithm can be divided into two parts. One part uses the network as demonstrated in Figure 2 to update the parameters of (3.12), and the other part uses the ADMM algorithm to iteratively update (3.13) and (3.14). We perform five ADMM [18] iterations for each epoch during the training and freezing the network parameters. The algorithm is to minimize the loss function of (3.12), so it is essential to choose a suitable Lagrange

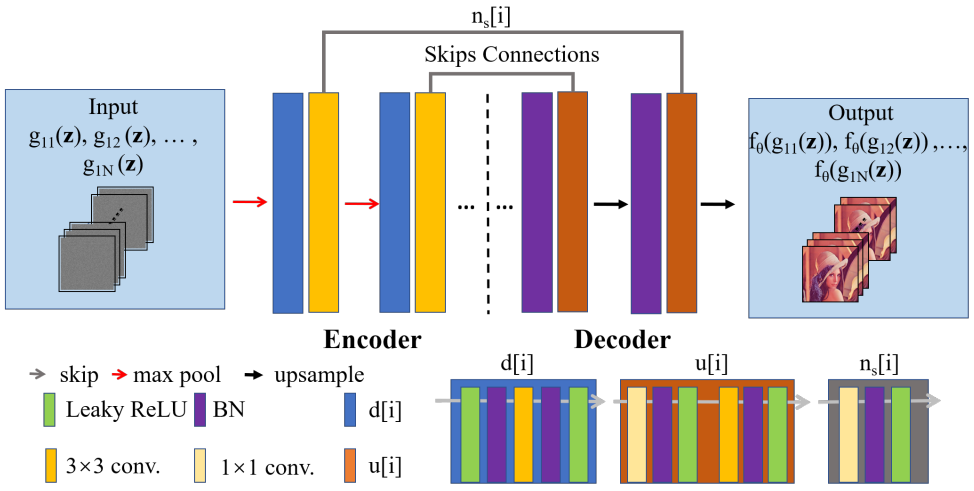



FIGURE 2. The network architecture used in drop-DIP.

multiplier $\lambda$. $\lambda$ is used to control the strength of the regularization term. Figure 3 shows the performance of drop-DIP under different $\lambda$ values on the Set12.

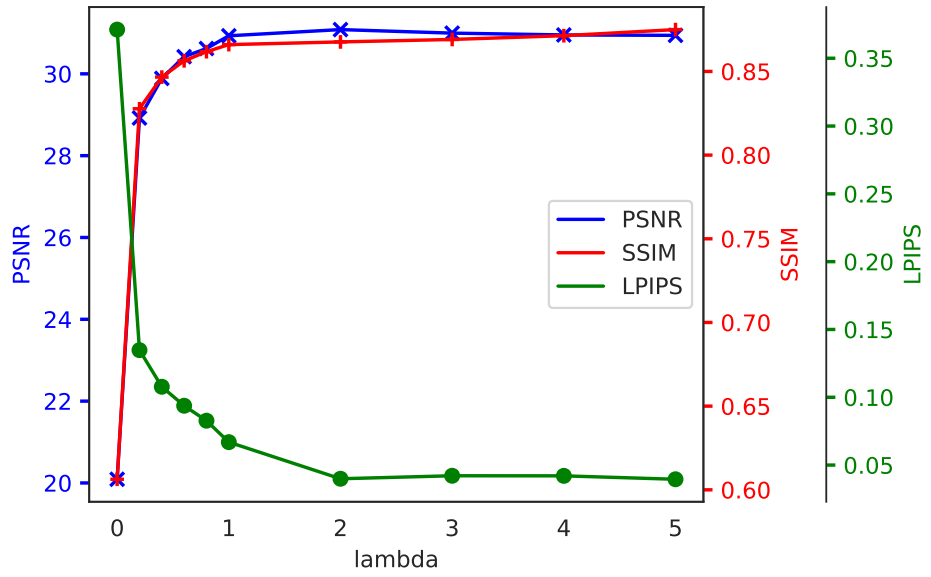



FIGURE 3. The effect of lagrange multiplier $\lambda$ on drop-DIP

When $\lambda = 0$, the regularization term is not included in the network training. At this time, the network is trained directly, and the network is easy to fit to obtain the Bernoulli sampling part of the image. Therefore, it is necessary to set a regularization term to improve the training effect of the network after sampling. It can be seen from the figure that when $\lambda = 2$, PSNR is better than other values, SSIM and LPIPS tend to be stable when $\lambda > 2$, so we choose $\lambda = 2$ for the network training in the following experiments.

4.2. **The early-stopping of training process.** It is found in Section 3.2 that as the parameters update iteratively, the noise fitting will cause error, so early stopping is needed to reduce the error. The conventional DIP method is implemented by setting the maximum number of iteration steps, which is difficult to achieve in practical applications. DIP-SURE uses the zero-cross stopping method, and the added SURE regularization term will obtain a negative value when the network fits the noise part of the image, so the network training can be stopped when the regularization term is less than zero. In drop-DIP, we can stop training by observing the deviation of $f_\theta(g_1(\mathbf{z}))$ from $f_\theta(\mathbf{z})$. Figure 4 demonstrates the changes in various metrics during training.

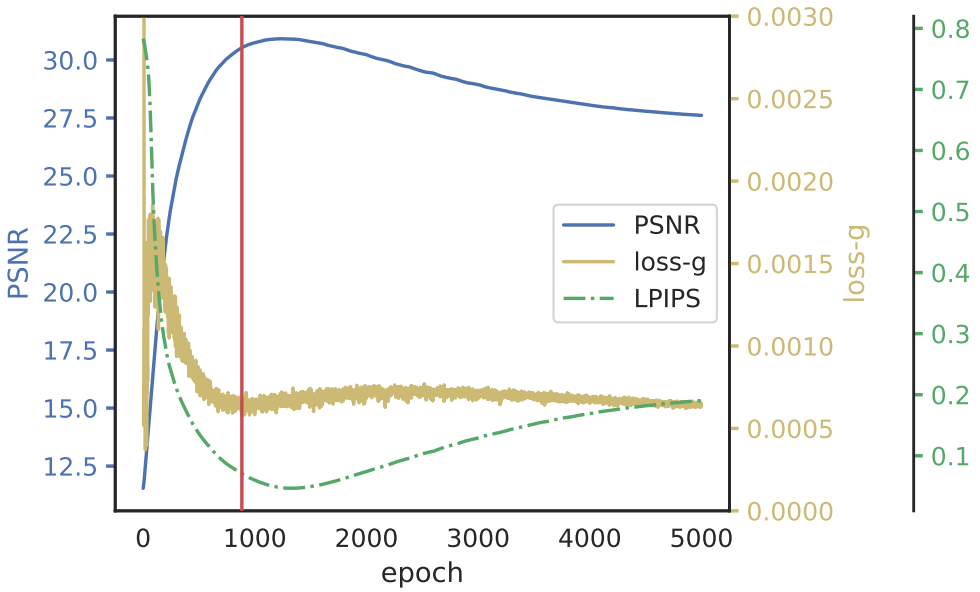
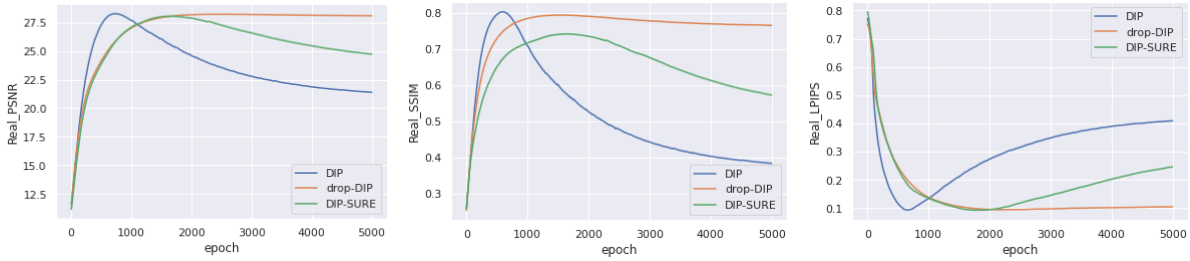


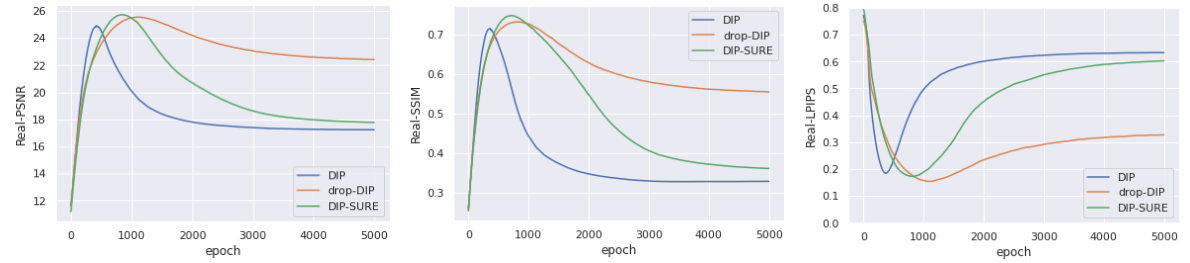

FIGURE 4. $loss_g$, PSNR and LPIPS in drop-DIP network training

Observing the difference between the network output $f_\theta(\mathbf{z})$ and $f_\theta(g_1(\mathbf{z}))$, we define the variable $loss_g = \mathbb{E}_{g_1,g_2} \|f_\theta(\mathbf{z}) - g_2(f_\theta(g_1(\mathbf{z})))\|_2^2$. At the beginning of the network fitting image, this variable is at a large value. As the network iterates, as shown in Figure 4, we found that this value decreases quickly and the increases slowly after the network fits the noisy part. Near the epoch corresponding to the minimum value of $loss_g$ (the red vertical line in Figure 4), the corresponding PSNR and LPIPS of $loss_g$ during the network training process. Therefore, we can stop training by judging whether $loss_g$ reaches a minimum value or stabilizes at a small value.

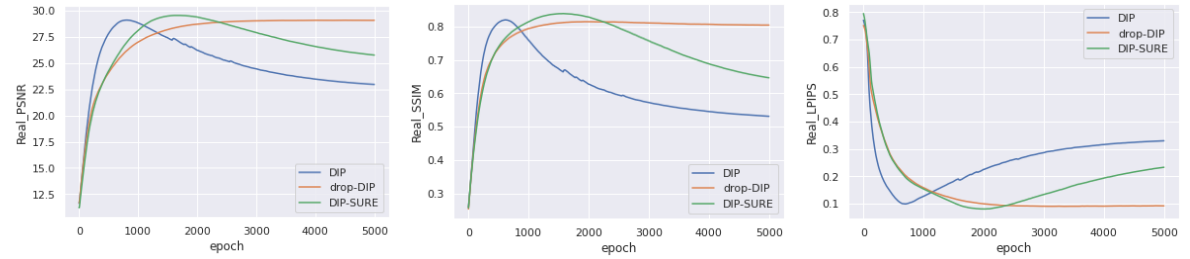
4.3. **The training process of DIP-based methods.** Below we compare the denoising methods for the specific example of the Camara image in Set12. The methods chosen for comparison are DIP [17], DIP-SURE [24], and our drop-DIP. Figure 5 is the changes of PSNR, SSIM, and LPIPS three indicators under different epochs of each method with Gaussian noise, Pulse noise, and Poisson noise.




(A) PSNR in training with Gaussian noise (B) SSIM in training with Gaussian noise (C) LPIPS in training with Gaussian noise




(D) PSNR in training with Pulse noise (E) SSIM in training with Pulse noise (F) LPIPS in training with Pulse noise




(G) PSNR in training with Poisson noise (H) SSIM in training with Poisson noise (I) LPIPS in training with Poisson noise

FIGURE 5. Compare PSNR, SSIM, LPIPS metrics in the training process of different DIP-based methods

Observing Figure 5, we can find that, under various noise types, the DIP and DIP-SURE performance decreases rapidly with the number of iteration steps. Nevertheless, drop-DIP performance is very stable under Gaussian and Poisson noise, and decreases slowly under Pulse noise, but it is still significantly better than the other two methods. For several image denoising goals with different noises, our drop-DIP algorithm seems to be the most robust, and its corresponding PSNR, SSIM, and LPIPS score curves do not have apparent oscillations, and it is more stable than other methods. At the same time, the optimal value corresponding to the training process is very close to other algorithms. The experiment also verifies that in Section 3.2, the fitting error of drop-DIP is smaller than the theory of the DIP method. In summary, a relatively stable solution can be obtained using the drop-DIP method to denoise a single noisy image.

4.4. **The denoising results of various single image denoising methods.** To verify the effectiveness of drop-DIP, we compared it with the current methods. Table 1 demonstrates the comparison results with other single image denoising methods on the grayscale dataset Set12 and the color dataset Cset9. The comparison methods include BM3D [21], Self2Self(S2S) [9], DIP [17], and DIP-SURE [24]. Except for BM3D, the rest of the methods are based on CNN and complete denoising through unsupervised learning. There are three types of noise: Gaussian noise, Pulse noise, and Poisson noise. Two damage levels are set for each noise. For example, the low-level Gaussian noise is $\sigma = 15$, and the high-level Gaussian noise is $\sigma = 50$.

TABLE 1. **The comparison of single-image denoising algorithms.** (↑) denotes that higher is better and (↓) denotes the lower is better. Best performance is in bold and the second best is underlined.

| Dataset | Noise type | PSNR(↑) | | | | |
|---|---|---|---|---|---|---|
| | | BM3D | DIP | DIP-SURE | S2S | drop-DIP |
| Set12 | low-level Gaussian | **31.7** | 30.79 | 30.19 | 30.24 | 31.11 |
| | high-level Gaussian | **25.52** | 23.12 | 23.98 | 23.52 | 24.40 |
| | low-level Pulse | 21.12 | 25.82 | 25.72 | 26.54 | **27.18** |
| | high-level Pulse | 15.68 | 22.00 | 22.65 | 22.71 | **23.29** |
| | low-level Poisson | **30.62** | 28.97 | 28.92 | 28.41 | 29.88 |
| | high-level Poisson | **26.77** | 24.89 | 25.20 | 25.48 | 26.23 |
| Cset9 | low-level Gaussian | 33.9 | 34.42 | 34.21 | **34.48** | 34.40 |
| | high-level Gaussian | 27.94 | 28.02 | 28.48 | 28.59 | **28.66** |
| | low-level Pulse | 20.89 | 27.35 | 29.61 | 27.75 | **30.13** |
| | high-level Pulse | 17.59 | 23.30 | 24.68 | 22.78 | **24.76** |
| | low-level Poisson | 32.26 | 32.33 | 32.70 | **32.88** | 32.79 |
| | high-level Poisson | 28.17 | 28.27 | 28.44 | **28.9** | 28.73 |
| Dataset | Noise type | SSIM(↑) | | | | |
| | | BM3D | DIP | DIP-SURE | S2S | drop-DIP |
| Set12 | low-level Gaussian | **0.89** | 0.83 | 0.85 | 0.76 | 0.87 |
| | high-level Gaussian | **0.74** | 0.61 | 0.68 | 0.60 | 0.69 |
| | low-level Pulse | 0.57 | 0.76 | 0.76 | 0.78 | **0.79** |
| | high-level Pulse | 0.26 | 0.58 | 0.65 | 0.60 | **0.66** |
| | low-level Poisson | **0.86** | 0.79 | 0.82 | 0.82 | 0.84 |
| | high-level Poisson | 0.68 | 0.69 | 0.72 | 0.66 | **0.75** |
| Cset9 | low-level Gaussian | 0.90 | 0.90 | 0.91 | 0.91 | **0.91** |
| | high-level Gaussian | 0.82 | 0.80 | 0.83 | 0.79 | **0.84** |
| | low-level Pulse | 0.53 | 0.76 | 0.84 | 0.76 | **0.85** |
| | high-level Pulse | 0.25 | 0.67 | 0.76 | 0.58 | **0.77** |
| | low-level Poisson | 0.87 | 0.87 | 0.88 | 0.87 | **0.89** |
| | high-level Poisson | 0.82 | 0.80 | 0.84 | 0.79 | **0.84** |
| Dataset | Noise type | LPIPS(↓) | | | | |
| | | BM3D | DIP | DIP-SURE | S2S | drop-DIP |
| Set12 | low-level Gaussian | 0.07 | 0.06 | 0.11 | 0.11 | **0.05** |
| | high-level Gaussian | **0.13** | 0.24 | 0.24 | 0.33 | 0.18 |
| | low-level Pulse | 0.43 | 0.19 | 0.21 | 0.20 | **0.16** |
| | high-level Pulse | 0.72 | 0.34 | 0.27 | 0.38 | **0.22** |
| | low-level Poisson | 0.07 | 0.13 | 0.13 | 0.13 | **0.07** |
| | high-level Poisson | 0.16 | 0.20 | 0.23 | 0.24 | **0.16** |
| Cset9 | low-level Gaussian | 0.09 | 0.05 | 0.08 | **0.05** | 0.07 |
| | high-level Gaussian | 0.17 | 0.19 | 0.16 | 0.17 | **0.13** |
| | low-level Pulse | 0.57 | 0.32 | 0.19 | 0.29 | **0.15** |
| | high-level Pulse | 0.68 | 0.35 | 0.26 | 0.45 | **0.20** |
| | low-level Poisson | 0.12 | 0.12 | 0.10 | 0.09 | **0.09** |
| | high-level Poisson | 0.17 | 0.20 | 0.17 | 0.18 | **0.14** |


FIGURE 6. Example comparison of single image denoising. The metrics below the picture are PSNR/SSIM/LPIPS

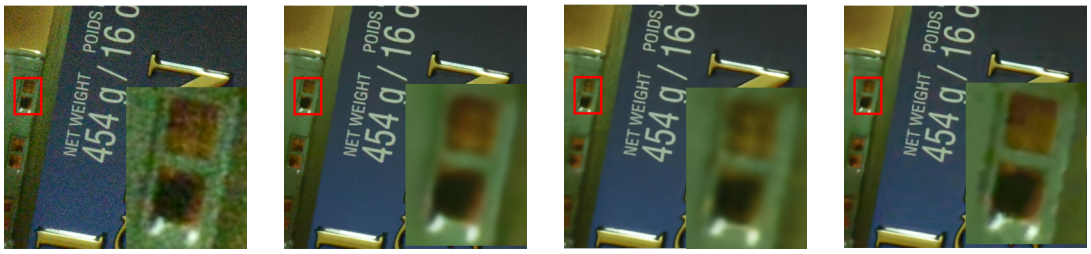
As seen from Table 1, for Pulse noise, our drop-DIP method can achieve the best performance on PSNR, SSIM, and LPIPS. For Poisson noise, although drop-DIP can only achieve the second-best effect on the PSNR score, it performs better on the LPIPS score. Because Self2Self takes the average of multiple network outputs to obtain the target image. This operation will significantly reduce the variance and improve the PSNR score, but it will lose the high-frequency details in the image, so the performance of the LPIPS score will deteriorate. Comparing the noise reduction effect of the grayscale image in Set12 with that of the color image in Cset9, image denoising is better in Cset9. It can be found that the more data contained in a single noisy image, the more various indicators of CNN are improved. Experiments show that the effect of neural network fitting image is affected by the information of the image itself. The more information, the better the fitting effect.

For impulse noise, Figure 6 shows the comparison of denoising effects of several methods. In Figure 6, it can be seen that for Pulse noise, the traditional BM3D method cannot effectively denoise the noise. The DIP method can achieve denoising but over-smoothing. DIP-SURE and our drop-DIP can remove the noise while retaining the details of the image.

4.5. **The comparison of real image denoising with unsupervised methods.** Unsupervised denoising methods have broad application prospects in real image denoising tasks. Zheng et al. [29] used neural network training to map real images to the hypothesis space where Gaussian noise was established, and then used traditional denoising methods, such as NN+BM3D. We compare the NN+BM3D method, the traditional DIP method, and our improved drop-DIP method on the real image dataset CC (Nam et al. [30]). The denoising results are listed in Table 2.

TABLE 2. **The comparison of real image denoising with unsupervised methods.** (↑) denotes that higher is better and (↓) denotes the lower is better. Best performance is in bold and the second best is underlined.

| Method | NN+BM3D | DIP | drop-DIP |
|---|---|---|---|
| PSNR(↑) | 36.88 | 36.79 | **37.81** |
| SSIM(↑) | 0.95 | 0.94 | **0.95** |
| LPIPS(↓) | 0.04 | 0.08 | **0.04** |



29.87/0.78/0.11 (A) Real

33.22/0.91/0.07 (B) NN-BM3D

33.02/0.90/0.11 (C) DIP

33.78/0.91/0.06 (D) drop-DIP

FIGURE 7. Example comparison of real image denoising. The metrics below the picture are PSNR/SSIM/LPIPS

Comparing with NN+BM3D and DIP, the PSNR, SSIM, and LPIPS of drop-DIP in the dataset CC are all better than other methods. Drop-DIP, an unsupervised denoising method, also performs superior for real image denoising. Figure 7 demonstrates a real image denoising result of the three methods. Furthermore, drop-DIP can preserve more image details than NN-BM3D.

## 5. CONCLUSIONS

We extended the DIP denoising by using the image drop-out technique to perform data augmentation by randomly sampling the input and output of the network. At the same time, for the training of the constrained network, the display regularization term is set for the network sampling, and the ADMM algorithm is used to realize the training of the network. The early-stopping criterion for training is set for the difference in the network output under different inputs. The denoising performance of the DIP method is improved without requiring manual early stopping. Our experimental verification shows that the performance of the drop-DIP method on PSNR, SSIM, and LPIPS indicators is significantly improved under the grayscale, color and real-world image datasets.

**Acknowledgements**

This work was supported by National Key Research and Development Program (2020YFA0713504) and National Natural Science Foundation of China (61977065).

## REFERENCES

[1] O. Scherzer, M. Grasmair, H. Grossauer, M. Haltmeier, F. Lenzen, Image and Noise Models. In: Variational Methods in Imaging. Applied Mathematical Sciences, vol 167. Springer, New York, 2009.

[2] S. Gu, R. Timofte, A Brief Review of Image Denoising Algorithms and Beyond. In: S. Escalera, S. Ayache, J. Wan, M. Madadi, U. Güçlü, X. Baró, (eds) Inpainting and Denoising Challenges. The Springer Series on Challenges in Machine Learning. Springer, Cham, 2019.
[3] K. Zhang, W. Zuo, Y. Chen, D. Meng, L. Zhang, Beyond a gaussian denoiser: residual learning of deep CNN for image denoising, IEEE Trans. Image Process. 26 (2017), 3142-3155.
[4] K. Zhang, W. Zuo, L. Zhang, FFDNet: Toward a fast and flexible solution for CNN-based image denoising, IEEE Trans. Image Process. 27 (2018), 4608-4622.
[5] S. Anwar, N. Barnes, Real image denoising with feature attention, In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 3155-3164, 2019.
[6] C. Tian, Y. Xu, W. Zuo, Image denoising using deep CNN with batch renormalization, Neural Netw. 121 (2020), 461-473.
[7] S. Laine, T. Karras, J. Lehtinen, T. Aila, High-quality self-supervised deep image denoising, Adv. Neural Info. Process. Sys. 32 (2019), NeurIPS 2019.
[8] T. Pang, H. Zheng, Y. Quan, H. Ji, Recorrupted-to-Recorrupted: Unsupervised deep learning for image denoising, In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2043-2052, 2021.
[9] Y. Quan, M. Chen, T. Pang, H. Ji, Self2self with drop-out: Learning self-supervised denoising from single image, In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1890-1898, 2020.
[10] A. Krull, T.O. Buchholz, F. Jug, Noise2void-learning denoising from single noisy images, In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2129-2137, 2019.
[11] J. Batson, L. Royer, Noise2self: Blind denoising by self-supervision, In International Conference on Machine Learning PMLR, pp. 524-533, 2019.
[12] T. Huang, S. Li, X. Jia, H. Lu, J. Liu, Neighbor2neighbor: Self-supervised denoising from single noisy images, In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14781-14790, 2021.
[13] J. Lehtinen, J. Munkberg, J. Hasselgren, S. Laine, T. Karras, M. Aittala, T. Aila, Noise2Noise: Learning image restoration without clean data, arXiv preprint arXiv:1803.04189, 2018.
[14] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, G. Wetzstein, Implicit neural representations with periodic activation functions, Adv. Neural Info. Process. Syst. 33 (2020), 7462-7473.
[15] R. Heckel, P. Hand, Deep decoder: Concise image representations from untrained non-convolutional networks, arXiv preprint arXiv:1810.03982, 2018.
[16] H.J. Xue, X. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems, IJCAI International Joint Conference on Artificial Intelligence, pp. 3203-3209, 2017.
[17] D. Ulyanov, A. Vedaldi, V. Lempitsky, Deep image prior, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9446-9454, 2018.
[18] Y. Wang, W. Yin, J. Zeng, Global convergence of ADMM in nonconvex nonsmooth optimization, J. Sci. Comput. 78 (2019), 29-63.
[19] M.C. Motwani, M.C. Gadiya, R.C. Motwani, F.C., Harris, Survey of image denoising techniques, In Proceedings of GSPX, 27 (2004), 27-30.
[20] A. Buades, B. Coll, J.M.. Morel, A non-local algorithm for image denoising, In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) 2 (2005), 60-65.
[21] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, Image denoising by sparse 3-D transform-domain collaborative filtering, IEEE Trans. Image Process. 16 (2007), 2080-2095.
[22] V. Jain, S. Seung, Natural image denoising with convolutional networks, Advances in Neural Information Processing Systems 21 - Proceedings of the 2008 Conference, pp. 769 - 776, 2009.
[23] S. Gu, R. Timofte, L. Van Gool, Multi-bin trainable linear unit for fast image restoration networks, arXiv preprint arXiv:1807.11389, 2018.
[24] S. Soltanayev, S.Y. Chun, Training deep learning based denoisers without ground truth data, Advances in Neural Information Processing Systems, pp. 3257-3267, 2018.
[25] Y. Jo, S.Y. Chun, J. Choi, Rethinking deep image prior for denoising, In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5087-5096, 2021.

[26] R. Heckel, M. Soltanolkotabi, Denoising and regularization via exploiting the structural bias of convolutional generators, arXiv preprint arXiv:1910.14634, 2019.
[27] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, J. Han, On the variance of the adaptive learning rate and beyond, arXiv preprint arXiv:1908.03265, 2019.
[28] R. Zhang, P. Isola, A.A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, In Proceedings of the IEEE Cconference on Computer Vision and Pattern Recognition, pp. 586-595, 2018.
[29] D. Zheng, S.H. Tan, X. Zhang, Z. Shi, K. Ma, C. Bao, An unsupervised deep learning approach for real-world image denoising, In International Conference on Learning Representations, 2021,
[30] S. Nam, Y. Hwang, Y. Matsushita, S.J. Kim, A holistic approach to cross-channel image noise modeling and its application to image denoising, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1683-1691, 2016.
[31] H. Wang, T. Li, Z. Zhuang, T. Chen, H. Liang, J. Sun, Early stopping for deep image prior, arXiv preprint arXiv:2112.06074, 2021.

## APPENDIX A. PROOF OF PROPOSITION 3.1

**Proposition A.1.** *Assume* $\mathbf{y} = \mathbf{x} + \delta$ *with noise* $\delta$. *The implicit fidelity term* $\mathscr{L}_{imp}(\theta)$ *satisfies*

$$\begin{aligned}\mathscr{L}_{imp}(\theta) &= \mathbb{E}_{g_1,g_2} \|f_\theta(g_1(\mathbf{z})) - g_2(\mathbf{y})\|_{g_2}^2 \\ &= \mathbb{E}_{g_1,g_2} \|f_\theta(g_1(\mathbf{z})) - \mathbf{x}\|_{g_2}^2 + \mathbb{E}_{g_1,g_2} \|\delta\|_{g_2}^2 - 2\,\mathrm{tr}(\mu^T \mathbf{r}),\end{aligned} \tag{A.1}$$

*where* $\mu = \mathbb{E}_{g_1,g_2}(g_2 \odot \delta), \mathbf{r} = \mathbb{E}_{g_1,g_2}(g_2 \odot (f_\theta(g_1(\mathbf{z})) - \mathbf{x}))$.

*Proof.* Rewrite the implicit fidelity term as follows

$$\begin{aligned}&\mathbb{E}_{g_1,g_2} \|f_\theta(g_1(\mathbf{z})) - g_2(\mathbf{y})\|_{g_2}^2 = \mathbb{E}_{g_1,g_2} \|f_\theta(g_1(\mathbf{z})) - \mathbf{y}\|_{g_2}^2 \\ &= \mathbb{E}_{g_1,g_2} \|f_\theta(g_1(\mathbf{z})) - \mathbf{x}\|_{g_2}^2 + \mathbb{E}_{g_1,g_2} \|\delta\|_{g_2}^2 - 2\,\mathrm{tr}\left(\mathbb{E}_{g_1,g_2}(\delta)_{g_2}^T (f_\theta(g_1(\mathbf{z})) - \mathbf{x})_{g_2}\right),\end{aligned} \tag{A.2}$$

where $(\cdot)_{g_2} = (g_2 \odot \cdot)$. Correlate the expectation about $\delta$ with the second term in (A.2). For the last item in (A.2), we have

$$\begin{aligned}&-2\,\mathrm{tr}\left(\mathbb{E}_{g_1,g_2}(\delta)_{g_2}^T (f_\theta(g_1(\mathbf{z})) - \mathbf{x})_{g_2}\right) \\ &= -2tr(\mathbb{E}_{g_1,g_2}(\delta)_{g_2}^T \mathbb{E}_{g_1,g_2}(f_\theta(g_1(\mathbf{z})) - \mathbf{x})_{g_2}) - 2tr(\mathbb{COV}(\mu, \mathbf{r})) \\ &= -2tr(\mu^T \mathbf{r}).\end{aligned} \tag{A.3}$$

We believe that equation (A.3) holds when the noise term and $r$ are independent of each other. Combining (A.2) and (A.3) yields

$$\mathbb{E}_{g_1,g_2} \|f_\theta(g_1(\mathbf{z})) - \mathbf{x}\|_{g_2}^2 + \mathbb{E}_{g_1,g_2} \|\delta\|_{g_2}^2 - 2\,\mathrm{tr}(\mu^T \mathbf{r}), \tag{A.4}$$

where $\mu = \mathbb{E}_{g_1,g_2}(g_2 \odot \delta)$, and $\mathbf{r} = \mathbb{E}_{g_1,g_2}(g_2 \odot (f_\theta(g_1(\mathbf{z})) - \mathbf{x}))$. □

## APPENDIX B. PROOF OF THEOREM 3.1

We first restate our Theorem 3.1 in Theorem 2 in Heckel and Soltanolkotabi [26] and Theorem A.2 in Wang [31].

**Theorem B.1.** *Let* $\mathbf{x} \in \mathbb{R}^n$ *be a signal in the span of the first p trigonometric basis functions, and consider a noisy observation* $\mathbf{y} = \mathbf{x} + \delta$, *where the noise* $\delta \sim \mathcal{N}\left(\mu, \sigma^2 \cdot I\right)$ *is Gaussian noise. To denoise this signal, we fit a two-layer generator network* $f_\theta(g_1(\mathbf{z})) = ReLU(\mathbf{U} g_1(\mathbf{z})\theta)\mathbf{v}$, *where* $\mathbf{v} = [1, \ldots, 1, -1, \ldots, -1]/\sqrt{k}$, *and* $\mathbf{z} \sim_{iid} \mathcal{N}(0,1)$, *and* $\mathbf{U}$ *is an upsampling operator that implements circular convolution with a given kernel* $\mathbf{u}$. *Denote* $\sigma \doteq \|\mathbf{u}\|_2 \mid Fh(\mathbf{u} \circledast \mathbf{u}/\|\mathbf{u}\|_2^2) \mid^{1/2}$, *where* $h(t) = \left(1 - \cos^{-1}(t)/\pi\right) t$ *and* $\circledast$ *denotes the circular convolution. Fix any* $\xi \in (0, \sigma_p/\sigma_1]$, *and suppose* $k \geq C_u n/\varepsilon^8$, *where* $C_u > 0$ *is a constant only depending on u. Consider gradient descent with step size* $\eta \leq \|F\mathbf{u}\|_\infty^{-2}$ *(*$F\mathbf{u}$ *is the Fourier transform of* $\mathbf{u}$*) starting from* $\theta_0 \sim_{iid} \mathcal{N}\left(0, \omega^2\right)$, *entries,* $\omega \propto \frac{\|\mathbf{y}\|_2}{\sqrt{n}}$. *Then, for all iterates k obeying* $t \leq \frac{100}{\eta \sigma_p^2}$, *the mse of*

*our* $f_{\theta_k}(g_1(\mathbf{z}))$ *and the real signal* $\mathbf{x}$ *obeys*

$$\begin{aligned}\left\|f_{\theta_k}(g_1(\mathbf{z}))-\mathbf{x}\right\|_2 \leq & \underbrace{\|\left(1-\eta\sigma_p^2\right)^k \bar{g_2}(\mathbf{x})+\left(1-\eta(\lambda+1)\sigma_p^2\right)^k g_2(\mathbf{x})\|_2}_{\text{error in fitting signal}} \\ & +\underbrace{\left(\sum_{i=1}^n\left(\left(1-\eta\sigma_i^2\right)^k-1\right)^2\langle\mathbf{w}_i,\bar{g_2}(\delta)\rangle^2+\sum_{i=1}^n\left(\left(1-\eta(\lambda+1)\sigma_i^2\right)^k-1\right)^2\langle\mathbf{w}_i,g_2(\delta)\rangle^2\right)^{1/2}}_{\text{noise fitting}} \\ & +\xi\|\mathbf{y}\|_2\end{aligned} \tag{B.1}$$

*with high probability at least* $1-\exp\left(-k^2\right)-n^{-2}$, *where* $\bar{g_2}=\boldsymbol{I}-g_2, \xi\in(0,\sigma_p/\sigma_1]$.

*Proof.* First, for the nonlinear least squares problem on convolutional networks of the form:

$$\mathscr{L}(\theta)=\|f_\theta(g_1(\mathbf{z}))-\mathbf{y}\|_2^2+\lambda\left\|g_2(f_\theta(g_1(\mathbf{z})))-g_2(\mathbf{y})-(f_\theta(g_1(\mathbf{z}))-f_\theta(\mathbf{z}))\right\|_2^2. \tag{B.2}$$

Here, the network $f_\theta$ can be regarded as a nonlinear model with respect to the parameters $\theta$. To solve this problem, we run gradient descent with a fixed step size $\eta$, starting from an initial point $\theta_0$, updated by

$$\theta_{k+1}=\theta_k-\eta\nabla\mathscr{L}(\theta_k), \tag{B.3}$$

$$\begin{aligned}\text{where } \nabla\mathscr{L}(\theta)= & \mathscr{J}^T(\theta)(g_2((f_\theta(g_1(\mathbf{z}))-\mathbf{y}))) \\ & +g_2(\mathscr{J}(\theta))^T(g_2(f_\theta(g_1(\mathbf{z})))-g_2(\mathbf{y})-(f_\theta(g_1(\mathbf{z}))-f_\theta(\mathbf{z}))).\end{aligned}$$

Here $\mathscr{J}(\theta)$ is the Jacobian associated with the nonlinear map f with entries given by$[\mathscr{J}(\theta)]_{i,j}=\frac{\partial f_i(\theta)}{\partial\theta_j}$, and $\bar{g_2}=\mathbf{1}-g_2$. Since the network parameters are frozen when calculating $f_\theta(\mathbf{z})$, its corresponding Jacobian matrix is $\mathbf{0}$. In order to study the properties of the gradient descent iterates, we relate the non-linear least squares problem to a linearized one in a ball around the initialization $\theta_0$. The associated linearized least-squares problem is defined as

$$\begin{aligned}\mathscr{L}_{\text{lin}}(\theta)= & \left\|f_{\theta_0}(g_1(\mathbf{z}))-\mathbf{y}+\mathbf{J}(\theta-\theta_0)\right\|_2^2 \\ & +\lambda\left\|g_2(f_{\theta_0}(g_1(\mathbf{z})))-g_2(\mathbf{y})-(f_{\theta_0}(g_1(\mathbf{z}))-f_{\theta_0}(\mathbf{z}))+g_2(\mathbf{J})(\theta-\theta_0)\right\|_2^2.\end{aligned} \tag{B.4}$$

Here, $\mathbf{J}$ is called the reference Jacobian, a fixed matrix independent of $\theta$ that approximates the Jacobian map $\mathscr{J}(\theta_0)$ at initialization. Starting from the same initial point $\theta_0$. In order to better measure the deviation of the network estimate from the observed value $y$, we change the $g_2$ norm of the first term to the L2 norm.

To simplify the notation, we write $\hat{\mathbf{y}}=\mathbf{y}-f_\theta(\mathbf{z}), \hat{\mathbf{y}}_{g_1}=\mathbf{y}-f_\theta(g_1(\mathbf{z})), \mathbf{J}=\mathscr{J}(\theta_0)$, and $\mathbf{a}=\theta-\theta_0$. So the least-squares objective in (B.4) is equal to

$$\mathscr{L}_{\text{lin}}(\theta)=\left\|\hat{\mathbf{y}}_{g_1}-\mathbf{J}\mathbf{a}\right\|_2^2+\lambda\left\|\bar{g_2}(\hat{\mathbf{y}}_{g_1})-g_2(\mathbf{J})\mathbf{a}+\hat{\mathbf{y}}\right\|_2^2, \tag{B.5}$$

and the gradient update is

$$\mathbf{a}_k=\mathbf{a}_{k-1}-\eta\mathbf{J}^T(\mathbf{J}\mathbf{a}_{k-1}-\hat{\mathbf{y}}_{g_1})-\eta\lambda g_2(\mathbf{J})^T(g_2(\mathbf{J})\mathbf{c}_{k-1}-\bar{g_2}(\hat{\mathbf{y}}_{g_1})-\hat{\mathbf{y}}), \tag{B.6}$$

where $\mathbf{a}_0=0$. And the residual of gradient descent at iteration $k$ is

$$\mathbf{r}_{\text{lin}}(k)=\hat{\mathbf{y}}_{g_1}-\mathbf{J}\mathbf{a}_k \tag{B.7}$$

$$= \hat{\mathbf{y}}_{g_1} - \mathbf{J}(\mathbf{a}_{k-1} - \eta \mathbf{J}^T(\mathbf{J}\mathbf{a}_{k-1} - \hat{\mathbf{y}}_{g_1}) - \eta\lambda g_2(\mathbf{J})^T(g_2(\mathbf{J})\mathbf{a}_{k-1} - \bar{g_2}(\hat{\mathbf{y}}_{g_1}) - \hat{\mathbf{y}})) \tag{B.8}$$

$$\approx (\mathbf{I} - \eta\mathbf{J}\mathbf{J}^T)(\hat{\mathbf{y}}_{g_1} - \mathbf{J}\mathbf{a}_{k-1}) - \eta\lambda\mathbf{J}\mathbf{J}^T g_2(\hat{\mathbf{y}}_{g_1} - \mathbf{J}\mathbf{a}_{k-1}) \tag{B.9}$$

$$= (\mathbf{I} - \eta\mathbf{J}\mathbf{J}^T - \eta\lambda\mathbf{J}\mathbf{J}^T)g_2(\hat{\mathbf{y}}_{g_1} - \mathbf{J}\mathbf{a}_{k-1}) + (\mathbf{I} - \eta\mathbf{J}\mathbf{J}^T)\bar{g_2}(\hat{\mathbf{y}}_{g_1} - \mathbf{J}\mathbf{a}_{k-1}) \tag{B.10}$$

$$= (\mathbf{I} - \eta\mathbf{J}\mathbf{J}^T - \eta\lambda\mathbf{J}\mathbf{J}^T)^2 g_2(\hat{\mathbf{y}}_{g_1} - \mathbf{J}\mathbf{a}_{k-2}) + (\mathbf{I} - \eta\mathbf{J}\mathbf{J}^T)^2\bar{g_2}(\hat{\mathbf{y}}_{g_1} - \mathbf{J}\mathbf{a}_{k-2}) (\text{using} \quad g \odot \bar{g} = 0) \tag{B.11}$$

$$= \dots$$

$$= (\mathbf{I} - \eta\mathbf{J}\mathbf{J}^T - \eta\lambda\mathbf{J}\mathbf{J}^T)^k g_2(\hat{\mathbf{y}}_{g_1} - \mathbf{J}\mathbf{a}_0) + (\mathbf{I} - \eta\mathbf{J}\mathbf{J}^T)^k\bar{g_2}(\hat{\mathbf{y}}_{g_1} - \mathbf{J}\mathbf{a}_0) (\text{using} \quad \mathbf{a}_0 = 0) \tag{B.12}$$

$$= (\mathbf{I} - \eta\mathbf{J}\mathbf{J}^T - \eta\lambda\mathbf{J}\mathbf{J}^T)^k g_2(\hat{\mathbf{y}}_{g_1}) + (\mathbf{I} - \eta\mathbf{J}\mathbf{J}^T)^k\bar{g_2}(\hat{\mathbf{y}}_{g_1}) \tag{B.13}$$

$$= (\mathbf{I} - \eta\mathbf{W}\Sigma^2\mathbf{W}^T - \eta\lambda\mathbf{W}\Sigma^2\mathbf{W}^T)^k g_2(\hat{\mathbf{y}}_{g_1}) + (\mathbf{I} - \eta\mathbf{W}\Sigma^2\mathbf{W}^T)^k\bar{g_2}(\hat{\mathbf{y}}_{g_1}). \tag{B.14}$$

Assume the SVD of $\mathbf{J}$ as $\mathbf{J} = \mathbf{W}\Sigma\mathbf{V}^T$. Expanding $\hat{\mathbf{y}}_{g_1}$ in terms of the singular vectors $\mathbf{w}_i$ (i.e., the columns of $\mathbf{W}$), as $\hat{\mathbf{y}}_{g_1} = \sum_i \mathbf{w}_i \langle \mathbf{w}_i, \hat{\mathbf{y}}_{g_1} \rangle$, and noting that $(\mathbf{I} - \eta\mathbf{W}\Sigma^2\mathbf{W}^T)^k = \sum_i (1 - \eta\sigma_i^2)^k \mathbf{w}_i\mathbf{w}_i^T$. Then

$$\mathbf{r}_{\text{lin}}(k) = \sum_i (1 - \eta\sigma_i^2 - \eta\lambda\sigma_i^2)^k \mathbf{w}_i \langle \mathbf{w}_i, g_2(\hat{\mathbf{y}}_{g_1}) \rangle + (1 - \eta\sigma_i^2)^k \mathbf{w}_i \langle \mathbf{w}_i, \bar{g_2}(\hat{\mathbf{y}}_{g_1}) \rangle . \tag{B.15}$$

Similar to linear residuals, we define nonlinear residuals.

$$\text{nonlinear residual:} \quad \mathbf{r}(k) := y - f_{\theta_k}(g_1(\mathbf{z})) \quad (r(0) = \hat{\mathbf{y}}_{g_1}), \tag{B.16}$$

$$\text{linear residual:} \quad \mathbf{r}_{\text{lin}}(k) := (\mathbf{I} - \eta\mathbf{J}\mathbf{J}^T - \eta\lambda\mathbf{J}\mathbf{J}^T)^k(g_2 \odot \mathbf{r}(0)) + (\mathbf{I} - \eta\mathbf{J}\mathbf{J}^T)^k(\bar{g_2} \odot \mathbf{r}(0)). \tag{B.17}$$

The following is an analogy to the proof of Theorem 2 in Heckel and Soltanolkotabi [26].

$$\left\| f_{\theta_k}(g_1(\mathbf{z})) - \mathbf{x} \right\|_2 = \left\| f_{\theta_k}(g_1(\mathbf{z})) + \delta - \mathbf{y} \right\|_2 \tag{B.18}$$

$$= \|\mathbf{r}(k) - \delta\|_2 \tag{B.19}$$

$$= \|\mathbf{r}_{\text{lin}}(k) - \delta + \mathbf{r}(k) - \mathbf{r}_{\text{lin}}(k)\|_2 \tag{B.20}$$

$$\leq \|\mathbf{r}_{\text{lin}}(k) - \delta\|_2 + \|\mathbf{r}(k) - \mathbf{r}_{\text{lin}}(k)\|_2 \tag{B.21}$$

$$\leq \|\mathbf{r}_{\text{lin}}(k) - \delta\|_2 + \xi\|\mathbf{r}(0)\|_2 \tag{B.22}$$

$$= \left\| \mathbf{W}(\mathbf{I} - \eta\Sigma^2 - \eta\lambda\Sigma^2)^k\mathbf{W}^T(g_2 \odot \mathbf{r}(0)) + \mathbf{W}(\mathbf{I} - \eta\Sigma^2)^k\mathbf{W}^T(\bar{g_2} \odot r(0)) - \delta \right\|_2 + \xi\|\mathbf{r}(0)\|_2 \tag{B.23}$$

$$= \left\| \mathbf{W}(\mathbf{T}_0)^k\mathbf{W}^T(g_2 \odot (\mathbf{x} + \delta - f_{\theta_0}(g_1(\mathbf{z})))) + \mathbf{W}(\mathbf{T}_1)^k\mathbf{W}^T(\bar{g_2} \odot (\mathbf{x} + \delta - f_{\theta_0}(g_1(\mathbf{z})))) - \delta \right\|_2 + \xi\|\mathbf{r}(0)\|_2 \tag{B.24}$$

$$\leq \left\| (\mathbf{T}_0)^k\mathbf{W}^T(g_2 \odot \mathbf{x}) + (\mathbf{T}_1)^k\mathbf{W}^T(\bar{g_2} \odot \mathbf{x}) \right\|_2 + \left\| ((\mathbf{T}_0)^k - \mathbf{I})\mathbf{W}^T(g_2 \odot \delta) + ((\mathbf{T}_1)^k - \mathbf{I})\mathbf{W}^T(\bar{g_2} \odot \delta) \right\|_2 + \|f_{\theta_0}(g_1(\mathbf{z}))\|_2 + \frac{1}{2}\xi\|\mathbf{y}\|_2 \tag{B.25}$$

$$\leq \| \left(1 - \eta\sigma_p^2\right)^k \bar{g_2}(\mathbf{x}) + \left(1 - \eta(\lambda + 1)\sigma_p^2\right)^k g_2(\mathbf{x})\|_2$$

$$+\left(\sum_{i=1}^{n}\left(\left(1-\eta\sigma_i^2\right)^k-1\right)^2\langle \mathbf{w}_i,\bar{g_2}(\delta)\rangle^2+\sum_{i=1}^{n}\left(\left(1-\eta(\lambda+1)\sigma_i^2\right)^k-1\right)^2\langle \mathbf{w}_i,g_2(\delta)\rangle^2\right)^{1/2}$$
$$+\xi\|y\|_2. \tag{B.26}$$

Here, (B.21) is from the triangular inequality, (B.22) is from the Theorem 4 in Heckel[26]. This means that linear residual is very close to nonlinear residual, (B.24) is from $\mathbf{T}_0=\mathbf{I}-\eta\Sigma^2-\eta\lambda\Sigma^2,\mathbf{T}_1=\mathbf{I}-\eta\Sigma^2$, and $\mathbf{r}(0)=\mathbf{y}-f_{\theta_0}(g_1(\mathbf{z}))=\mathbf{x}+\delta-f_{\theta_0}(g_1(\mathbf{z}))$, and (B.25, B.26) from the fact that $\mathbf{x}\in span\{\mathbf{w}_1,\ldots,\mathbf{w}_p\}$ and $\|\mathbf{W}(\mathbf{T}_0)^k\mathbf{W}^T\|\leq 1$, $\|\mathbf{r}(0)\|_2\leq\frac{1}{2}\|\mathbf{y}\|_2$,$\|f_{\theta_0}(g_1(\mathbf{z}))\|_2\leq\frac{1}{2}\|\mathbf{y}\|_2$. This proves bound (B.1). □