Equilibrium Image Denoising With Implicit Differentiation

Qi Chen, Yifei Wang, Zhengyang Geng, Yisen Wang, Jiansheng Yang, and Zhouchen Lin, *Fellow, IEEE*

***Abstract*— Recent efforts on learning-based image denoising approaches use unrolled architectures with a fixed number of repeatedly stacked blocks. However, due to difficulties in training networks corresponding to deeper layers, simply stacking blocks may cause performance degradation, and the number of unrolled blocks needs to be manually tuned to find an appropriate value. To circumvent these problems, this paper describes an alternative approach with implicit models. To our best knowledge, our approach is the first attempt to model iterative image denoising through an implicit scheme. The model employs implicit differentiation to calculate gradients in the backward pass, thus avoiding the training difficulties of explicit models and elaborate selection of the iteration number. Our model is parameter-efficient and has only one implicit layer, which is a fixed-point equation that casts the desired noise feature as its solution. By simulating infinite iterations of the model, the final denoising result is given by the equilibrium that is achieved through accelerated black-box solvers. The implicit layer not only captures the non-local self-similarity prior for image denoising, but also facilitates training stability and thereby boosts the denoising performance. Extensive experiments show that our model leads to better performances than state-of-the-art explicit denoisers with enhanced qualitative and quantitative results.**

***Index Terms*— Image denoising, deep equilibrium models.**

I. Introduction

ALTHOUGH the number of digital images taken every day is rapidly increasing, noise corruption is inevitable irrespective of the acquisition method, making it challenging to acquire high quality images. Therefore, image denoising is required to recover a clean image from its noise-corrupted observation. As a classical and fundamental problem in low level vision, image denoising has been extensively studied for many years; however, it is still a research hotspot because denoising can be adopted by many other image restoration problems as a building block [1], [2], [3], [4], which further broadens its applicability.

Many well-known classical image denoising approaches [5], [6], [7] are implemented by iterative algorithms that decompose the task into successive subtasks. As the iteration progresses, we expect to have an image sequence, whose restoration qualities get better as the iterations progress. Instead of making a clean image estimate only based on the noisy observation, taking previous estimates into consideration can provide valuable information for recovering the high-quality clean image. Despite numerous architectures proposed [6], [7], [8], it remains an open task about how to effectively design an iterative denoiser. Traditional model-based denoisers perform noise removal with manually designed iteration maps and termination conditions. Although such approaches can yield desirable theoretical guarantees, they have limited effectiveness in recovering signals due to their inability to leverage large amounts of available data. As an alternative, one popular learning-based approach that involves augmenting standard iterative denoisers with learned deep networks, has achieved a significant boost in performance by learning from a training set of degraded and ground-truth image pairs [9]. The basic idea is to unroll a fixed number of neural network blocks, where each block corresponds to an iteration, and then the parameters are optimized end-to-end by gradient methods. However, training deep neural networks is a challenging task. First, backpropagation via the chain rule necessitates that the intermediate values of these layers be stored. As a result, the number of iterations are constrained by memory and computation consumption. Second, training deep models faces optimization problems such as gradient exploding and gradient saturation as the number of iteration steps increases. Consequently, after a few numbers of blocks, simply increasing the unrolling steps cannot improve the performance further, therefore they have to manually tune the iteration number to find an appropriate value. This is quite different from traditional variational denoisers which iterate until convergence to produce the final prediction. With only limited iterations, the dynamic of existing iterative approaches is non-convergent, which means that the output of each layer may vary a lot. As a result, when we increase (or decrease)

Manuscript received 14 April 2022; revised 16 December 2022; accepted 13 February 2023. Date of publication 14 March 2023; date of current version 21 March 2023. This work was supported in part by the National Key R&D Program of China under Grant 2022ZD0160302, in part by the Major Key Project of PCL, China, under Grant PCL2021A12, in part by the National Natural Science Foundation of China under Grant 62006153 and Grant 62276004, in part by the Open Research Projects of Zhejiang Laboratory under Grant 2022RC0AB05, in part by Huawei Technologies Inc., in part by Qualcomm, and in part by the PKU-Baidu Fund, China, under Project 2020BD006. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Zhenzhong Chen. *(Corresponding author: Zhouchen Lin.)*

Qi Chen, Yifei Wang, and Jiansheng Yang are with the School of Mathematical Sciences, Peking University, Beijing 100871, China.

Zhengyang Geng is with the School of Intelligence Science and Technology, Peking University, Beijing 100871, China.

Yisen Wang is with the National Key Laboratory of General AI, School of Intelligence Science and Technology, Peking University, Beijing 100871, China, and also with the Institute for Artificial Intelligence, Peking University, Beijing 100871, China.

Zhouchen Lin is with the National Key Laboratory of General AI, School of Intelligence Science and Technology, Peking University, Beijing 100871, China, also with the Institute for Artificial Intelligence, Peking University, Beijing 100871, China, and also with the Peng Cheng Laboratory, Shenzhen 518055, China (e-mail: zlin@pku.edu.cn).

Digital Object Identifier 10.1109/TIP.2023.3255104

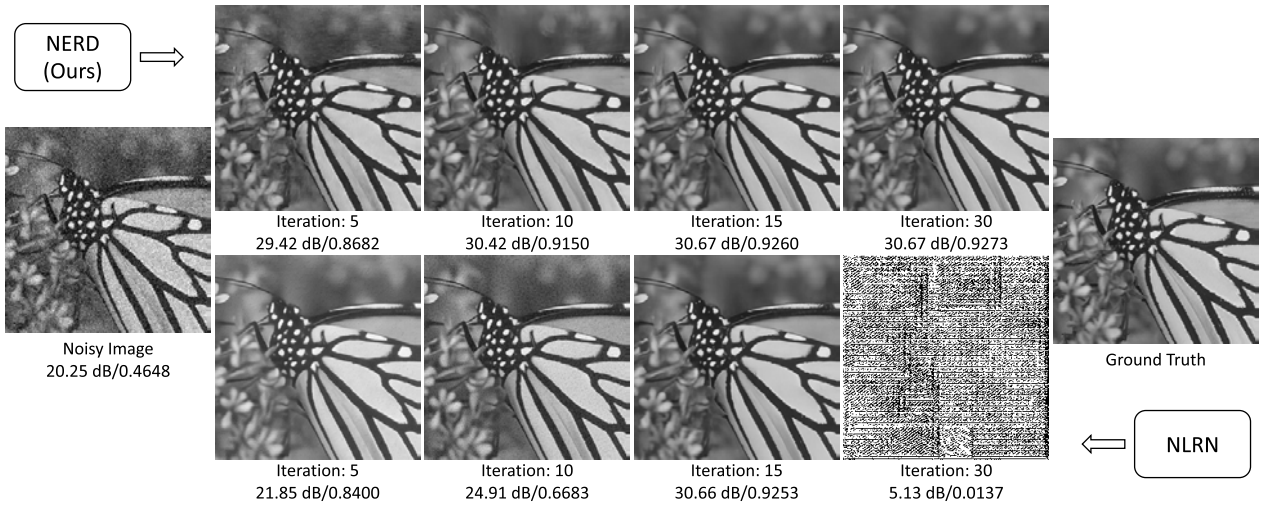

Fig. 1. An example showing that an inappropriate number of iterations in existing denoising methods, *e.g.,* NLRN [10], may result in inferior performance. As a comparison, our model gives reliable denoised results with a limited number of solver iterations, and at the same time we do not face a performance degradation as the iteration number grows. From left to right: i) Monarch, an image in the Set12 dataset corrupted with the noise level of 25, ii)-v) the denoised results with different number of iterations, vi) the ground truth clean image. The PSNR and SSIM values are given below the corresponding image.

the iteration number, the performance will degrade a lot, as the model will perform differently from how it was trained. In other words, these models have to go through the same computation graph in inference as they did in training, as Fig. 1 shows.

In this work, we take an implicit scheme for image denoising. Ideally, we hope to design a denoiser, which will not face performance degradation as the iteration grows, thus we can have a better performance with infinite iterations than existing iterative approaches with finite iterations. In order to achieve the goal, we develop a novel implicit denoising neural network. The core of the network is a fixed-point equation, whose solution (*i.e.,* "equilibrium") is the desired representation to restore the noise. We name the model **N**oise **E**quilibrium **R**eaching **D**enoiser (NERD), since we cast the noise feature as the equilibrium, and can reach the noise equilibrium when making denoised image predictions. Compared to previous iterative approaches, our method has the following advantages: (i) It largely mitigates the training problems of iterative scheme by employing implicit differentiation as a powerful tool to calculate gradients. (ii) It is totally learnable and does not have to manually select the iteration number by simulating models with "infinite" iterations. (iii) The model allows us to flexibly adopt different numbers of iterations for training and inference, thus giving us the ability to navigate a trade-off between computation time and accuracy during inference. As shown in Fig. 1, we have a reliable equilibrium approximate with very limited iterations, and at the same time we do not face a performance degradation as the iteration progresses.

To summarize, this paper has three key contributions:

- To our best knowledge, the proposed method is the first attempt to model iterative image denoising using an implicit scheme. Benefiting from the scheme, we do not have to manually tune the iteration number.
- We propose a novel model design for NERD, which not only takes advantage of the self-similarity property, but also can be explained by traditional variational denoisers. In addition, we introduce useful techniques that further help training of implicit models.
- Experiments show that our model yields a consistent improvement in performance above state-of-the-art approaches, both visually and quantitatively. Moreover, the parameter amount is significantly reduced, which indicates the high efficiency of our model.

We structure the paper as follows. First, we provide background materials on related iterative denoising approaches and implicit neural networks in Sec. II. Next, we introduce the overall proposed method in Sec. III, and describe the implicit denoising module in detail in Sec. IV. Then we introduce the model training in Sec. V and describe the relationships to prior works in Sec. VI. We show empirical understanding of the proposed method and experimentally compare it with state-of-the-art approaches in Sec. VII. Finally, we draw conclusions in Sec. VIII.

II. Related Work

A. Image Denoising

Problem Setup. The problem of image denoising aims to reconstruct the original image $\mathbf{x}$ from its noisy observation

$$\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{n}. \tag{1}$$

In this setting, $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{n} \in \mathbb{R}^{NC}$ are the vectorized versions of the latent clean image, the observed noisy image and the noise, respectively. Here N is the number of pixels, and C is the number of image channels.

Numerous successful iterative algorithms have been developed for image denoising over the years. We can broadly divide the methods into two categories: model-based approaches and learning-based approaches. Since the image denoising task is inherently ill-posed, traditional model-based approaches focus on incorporating manually designed priors, such as the low-rankness [5] and the non-local self-similarity [11]. The variational approach is one of the most popular

model-based strategies. It casts image denoising as a minimization problem of an objective function, called "energy function", where the minimizer corresponds to the recovered latent image. Since we usually cannot derive a closed-form minimizer of the energy function, iterative methods are adopted to solve the problem. For example, Vogel et al. [12] adopt total-variation [13] as a regularizer and propose a fixed point algorithm to minimize the energy function, and WNNM [7] uses weighted nuclear norm as a regularizer and proposes a new iterative method to solve the problem under different weighting conditions. Though robust and simple, these model-based methods are hand-crafted and cannot be optimized end-to-end by discriminant training. Consequently, they are limited in capacity and expressiveness, and unable to characterize complex image textures.

To circumvent this problem, subsequent learning-based neural network algorithms use training data to learn a model for image denoising and have achieved remarkable performance. Given an iterative model-based denoiser, a general idea is to first unfold the iterations into a layer-based structure, and then untie the model parameters across layers to obtain a neural network architecture that can easily be trained via gradient-based methods. A notable work is TNRD [14] that proposes a trainable diffusion model with learnable filters and truncated iterations, then its parameters are learned from training data in a supervised way. Later, NLNet [15], UNLNet [16] and Deep K-SVD [17] all utilize the proximal gradient method as an unrolled iteration to build model architectures. Also, DeamNet [18] propose an adaptive consistency prior to construct a learnable optimization objective for image denoising, and then unroll the iterative optimization steps. These models are all truncated after less than 10 iterations, despite the fact that each iteration corresponds to a gradient descent step to minimize an objective function, which usually takes much more iterations to converge.

Many works are also built by unfolding layers, although they may not state the relation to the iterative scheme. N^3Net [19] proposes a neural network with 3 stacked blocks, where each block is a continuous relaxation of the k-nearest neighbors (KNN). NLRN [10] stacks weight-tied non-local blocks to enable parameter efficiency. GCDN [20] and DAGL [21] stack graph convolutional blocks for image denoising. They only consider non-local correlations between several neighbors. GCDN construct the long-range correlations based on pixels, while DAGL focus on patches.

There are deeper models like RED [22] with up to 30 layers, and MemNet [23] with up to 80 layers. However, deep models trained using backpropagation have much computation and memory consumption. Moreover, simply stacking layers cannot improve the performance further after a few stages and even brings performance degradation due to the well-known training issues of deep models [24].

B. Implicit Deep Learning

Implicit models are emerging architectures in deep learning, where the layer defines an analytical condition for its output to satisfy [25]. One example is Neural ODE [26] that employs black-box ODE solvers to model recursive residual blocks implicitly. The other example is Deep Equilibrium Models (DEQs) [27] that reduce the forward pass to a root-solving problem, and directly differentiate through the final equilibrium by the implicit function theorem [28]. They do not need to build explicit computation graphs for forward propagation, and are able to simulate models with "infinite" iterations within a constant memory footprint. To guarantee the existence of a unique equilibrium point, subsequent works develop a new class of implicit model based on the theory of monotone operators [29] or Lipschitz boundedness [30]. However, these structural solutions rely extensively on specific layer parameterization, thus more inflexible to apply. As a result, most equilibrium models [27], [31], [32], [33] do not enforce the well-posedness condition. Instead, they stabilize the training process in a soft way. For example, Bai et al. [27], [31] employ a pretraining stage in which the model is unrolled as a shallow recurrent network for initialization. Bai et al. [34] explicitly regularize the Jacobian of the fixed-point update equations. Although effective, the pretraining and regularization bring extra hyperparameters, which are cumbersome to tune. Unlike previous works that calculate exact gradients, Geng et al. [35] propose an unrolling-based gradient estimate strategy called phantom gradient, which stably provides an update direction to the model training. Different from unrolling-based explicit models, their unrolling process is not a part of the model and only exists in the backward pass for gradient estimate. Compared to the exact gradient, phantom gradient demonstrates better robustness and higher tolerance to numerical errors.

Applications of implicit neural networks are still in its early stages. Recently, the DEQ-Transformer model proposed by Bai et al. [27] demonstrates its performance on language modeling [27]. The successive works include MDEQ [31], a convolutional model that applies DEQs to image classification and semantic segmentation tasks, and i-FPN [32], another convolutional model that applies DEQs to object detection. Only very recently, Gilton et al. [33] apply implicit models for image restoration tasks, including MRI reconstruction, image deblurring and compressed sensing. The models construct the fixed-point equation from classical optimization-based reconstruction methods, such as the plug-and-play (PNP) framework [1] and the Regularization by Denoising (RED) framework [3], which demand a fundamental off-the-shelf denoiser as the prior. Therefore, these models cannot be applied to image denoising.

III. Proposed NERD: An Overview

We propose a model called **N**oise **E**quilibrium **R**eaching **D**enoiser (NERD), since it models the noise feature as the equilibrium of a fixed-point equation, and is able to reach the equilibrium in the forward pass. As shown in Fig. 2, the core of our model is an implicit denoising module, which takes the feature of the input image to construct an input-dependent fixed-point equation, and gives the equilibrium as the layer output. An input injection module and a feature decoding module are used to transform between the feature

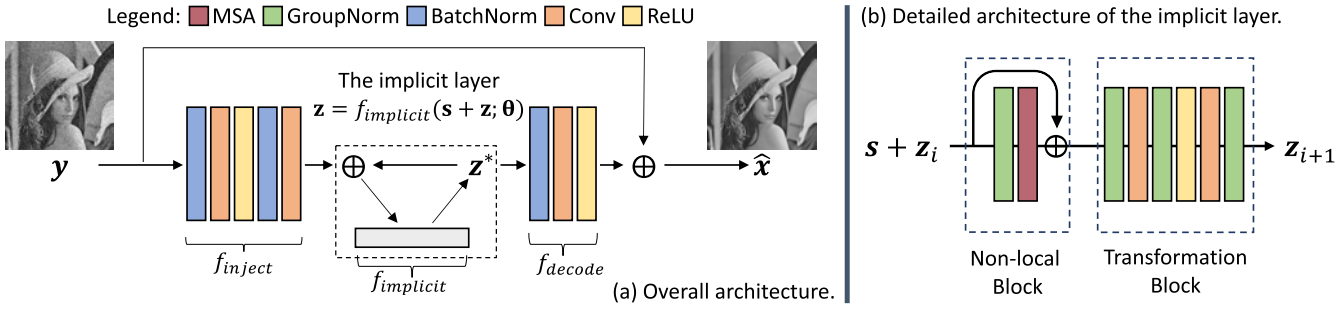

Fig. 2. (a) Overall architecture of our NERD. At first, the noisy image $\boldsymbol{y}$ is passed through f_{inject} to get a feature representation, then it is taken by the implicit layer (Eq. (11)) to solve the equilibrium z^*, which corresponds to the noise feature. Finally, we decode z^* in the image space by f_{decode} to produce the final image prediction $\hat{\boldsymbol{x}}$. (b) Detailed structure of the implicit layer. It is composed of two sub-layers: a non-local block to learn global correspondences and a transformation block to capture local patterns.

space and the image space. In general, given a noisy image $\boldsymbol{y}$, our model predicts the latent clean image by passing it through the following modules successively:

1) An *input injection module* that transforms the noisy input $\boldsymbol{y}$ into the feature space to obtain a feature $\boldsymbol{s} = f_{\text{inject}}(\boldsymbol{y};\boldsymbol{\phi})$, which represents the input information;
2) An *implicit denoising module* that parameterizes the fixed-point equation $\boldsymbol{z} = f_{\text{implicit}}(\boldsymbol{z},\boldsymbol{s};\boldsymbol{\theta})$ by the injection $\boldsymbol{s}$ for denoising, and obtains the corresponding equilibrium $\boldsymbol{z}^*$ as the noise estimate;
3) A *feature decoding module* that decodes the equilibrium $\boldsymbol{z}^*$ in the image space to produce the final image prediction by $\hat{\boldsymbol{x}} = \boldsymbol{y} + f_{\text{decode}}(\boldsymbol{z}^*;\boldsymbol{\psi})$.

Here, we use $\boldsymbol{\phi}$, $\boldsymbol{\theta}$ and $\boldsymbol{\psi}$ to denote the learnable parameters of f_{inject}, f_{implicit} and f_{decode} respectively. In the following, we explain the design of each component in further details.

A. Input Injection Module

The input injection module is a shallow block that contains two convolutional layers with an intermediate ReLU activation. We denote it as follows:

$$\boldsymbol{s} = f_{\text{inject}}(\boldsymbol{y};\boldsymbol{\phi}) = \text{Conv}(\text{BN}(\text{ReLU}(\text{Conv}(\text{BN}(\boldsymbol{y}))))), \quad (2)$$

where we employ batch normalization (BN) [36] before each convolutional layer. This module aims to project the noisy input to the feature space. The acquired representation is then injected into the implicit layer to facilitate the noise estimate.

B. Implicit Denoising Module

As the core of our model, the implicit denoising module is formulated as a fixed-point equation [27], which is parameterized by the input injection $\boldsymbol{s}$ as:

$$\boldsymbol{z} = f_{\text{implicit}}(\boldsymbol{z},\boldsymbol{s};\boldsymbol{\theta}). \quad (3)$$

The fixed-point iteration f_{implicit} takes the latent noise estimation $z^{[i]}$ as input and obtains an updated variable $z^{[i+1]}$ as the output. As shown in Fig. 3, different from feed-forward denoising neural networks, we model the output as the equilibrium z^* of the implicit layer (Eq. (3)), where $\mathbf{z}$ appears at both left and right sides and cannot be obtained by simple feed-forward computing. Instead, we could adopt black-box root-finding algorithms, *e.g.,* Broyden's method [37] or Anderson acceleration [38], to get a root approximate at a desired precision. Meanwhile, the model performance depends on the design of the implicit layer (Eq. (3)), which is central to our proposed denoiser. More details can be found in Sec. IV.

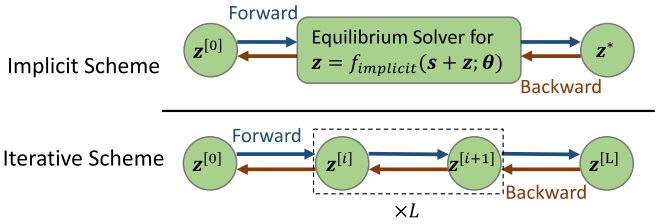

Fig. 3. Comparison between the iterative scheme and the implicit scheme. The iterative scheme loops for finite iterations and stores all the intermediate variables for backpropagation, while the implicit scheme calculates the equilibrium through accelerated black-box solvers without the need to store any intermediate variable.

C. Feature Decoding Module

After obtaining the equilibrium feature $\mathbf{z}^*$, we decode it in the image space through the feature decoding module, which is also a shallow block composed of a batch normalization, a ReLU activation, and a convolutional layer successively:

$$\hat{\boldsymbol{n}} = f_{\text{decode}}(\boldsymbol{z}^*;\boldsymbol{\psi}) = \text{Conv}(\text{ReLU}(\text{BN}(\boldsymbol{z}^*))), \quad (4)$$

where $\hat{\boldsymbol{n}}$ denotes the final noise estimate. Accordingly, we get the image estimate by subtracting $\hat{\boldsymbol{n}}$ from the corrupted input $\boldsymbol{y}$:

$$\hat{\boldsymbol{x}} = \boldsymbol{y} + \hat{\boldsymbol{n}}. \quad (5)$$

D. Learning Objective

At last, we adopt the standard MSE loss to learn our implicit denoiser NERD by minimizing the estimation error of the ground-truth noise $\boldsymbol{n} = \boldsymbol{x} - \boldsymbol{y}$,

$$L = \ell(\mathbf{x},\boldsymbol{y};\boldsymbol{\xi}) = \|\boldsymbol{n} - \hat{\boldsymbol{n}}\|^2 = \|(\boldsymbol{x}-\boldsymbol{y}) - \hat{\boldsymbol{n}}\|^2, \quad (6)$$

where $\boldsymbol{\xi} = [\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{\psi}]$ denotes the collection of all learnable parameters.

IV. Proposed Implicit Denoising Module

Unlike previous deep neural network denoisers that stack repetitive layers, the implicit function only has one layer, which is composed of two sub-layers. The first is a non-local block to employ self-similarities, and the second is a transformation block to capture local patterns. We show the detailed architecture of the implicit function in Fig. 2.

A. Building Blocks

1) Non-Local Block: A non-local block can be described as a composition of a group normalization (GN), a multi-head self-attention mechanism (MSA) and a skip connection.

An attention layer is a soft selection procedure that uses scores to choose which pixels to focus on. It can be described as mapping a query and a set of key-value pairs to an output. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key [39]. Let $\boldsymbol{Y} \in \mathbb{R}^{n\times d}$ be an input matrix whose rows are the input vectors $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n \in \mathbb{R}^d$, then the attention layer computes the query, key, value as $\boldsymbol{Y}W_{\boldsymbol{\theta}}, \boldsymbol{Y}W_{\psi} \in \mathbb{R}^{n\times d_k}$ and $\boldsymbol{Y}W_g \in \mathbb{R}^{n\times d_v}$, respectively. Finally, the overall self-attention module (SA) returns a matrix whose columns are weighted averages of the inputs:

$$\hat{\boldsymbol{Y}} = \mathrm{SA}(\boldsymbol{Y}) = \mathrm{softmax}(\boldsymbol{Y}W_{\boldsymbol{\theta}}(\boldsymbol{Y}W_{\psi})^T/\sqrt{d_k})\boldsymbol{Y}W_g. \quad (7)$$

Instead of performing a single attention function with d_{model}-dimensional keys, queries and values, the multi-head attention [39] allows the model to jointly attend to information from different representation subspaces at different positions. We project the queries, keys and values H times with different linear projections, respectively. And then we perform the SA in parallel on each of these projections, yielding H output values. These values are concatenated and once again projected, resulting in the final values as follows:

$$\mathrm{MSA}(\boldsymbol{Y}) = (\mathrm{SA}_1(\boldsymbol{Y})\| \ldots \|\mathrm{SA}_H(\boldsymbol{Y}))W_O, \quad (8a)$$

$$\mathrm{SA}_h(\boldsymbol{Y}) = \mathrm{softmax}(\boldsymbol{Y}W^h_{\boldsymbol{\theta}}(\boldsymbol{Y}W^h_{\psi})^T/\sqrt{d_k})\boldsymbol{Y}W^h_g, \quad (8b)$$

for $h = 1, 2, \ldots, H$, where the $\|$ operator denotes concatenation, and the projections are parameter matrices $W^h_{\boldsymbol{\theta}}, W^h_{\psi} \in \mathbb{R}^{d_{\text{model}}\times d_k}$, $W^h_g \in \mathbb{R}^{d_{\text{model}}\times d_v}$, and $W_O \in \mathbb{R}^{Hd_v\times d_{\text{model}}}$.

Finally, given the input $\tilde{\boldsymbol{s}}$, we have the output of the non-local block as

$$\bar{\boldsymbol{s}} = \mathrm{NL}(\tilde{\boldsymbol{s}}) = \mathrm{MSA}(\mathrm{GN}(\tilde{\boldsymbol{s}})) + \tilde{\boldsymbol{s}}. \quad (9)$$

We adopt a skip-connection in the block, since it has been shown to help gradient backpropagation during training [40] and is broadly used to improve training convergence in denoising tasks [41].

2) Transformation Block: In addition to the non-local block, the implicit denoising module contains a transformation block, which is applied to extract local patterns. It consists of two convolutional layers with kernel size 3, as well as a ReLU activation in between and 3 group normalization layers as follows:

$$\mathrm{Trans}(\bar{\boldsymbol{s}}) = \mathrm{GN}(\mathrm{Conv}(\mathrm{ReLU}(\mathrm{GN}(\mathrm{Conv}(\mathrm{GN}(\bar{\boldsymbol{s}})))))). \quad (10)$$

Following previous explicit works [41], we use normalization layers in the transformation block to speed up the training process as well as boost the denoising performance. In addition, previous implicit works [27] also point out that normalization layers that constrain the output ranges help to make the implicit training process more stable.

Especially, we apply a normalization layer before the output to further improve training stability. We refer to this as *post-normalization* in the rest of the paper. Intuitively, it helps to limit the feasible set of equilibrium to a bounded area, which makes the equilibrium easier to be found. It will be explained in more detail in Sec. VI, where an analogy is drawn between the post-normalization and a proximal operator [42] that helps to stabilize the level of the estimated noise.

When it comes to the choice of normalization layers in the implicit layer, we follow previous works [27], [31] and use GN rather than BN, since the former one estimates population statistics only based on layers, thus the Jacobian of our implicit function would not scale badly as BN to make the equilibrium significantly harder to solve.

B. Designing the Implicit Layer

In this section, we introduce how we design the implicit layer. We expect that its equilibrium not only has an informative representation for image denoising, but also can be found in a reliable and efficient manner. We will first introduce the basic formulation of our implicit layer from the perspective of denoising task, and then explain the design from the perspective of model training.

1) Layer Formulation: When designing an implicit layer which is a fixed-point equation, a basic question is: what is its input? Assuming the *well-posedness* of the implicit layer holds, *i.e.,* its solution exists and is unique, the equilibrium representation is totally decided by the implicit layer itself, irrespective of the initial point. Consequently, we consider to "inject" the input information into the implicit layer by constructing an input-dependent equation. To be concrete, we take the output of the input injection module $\boldsymbol{s}$ as the parameter, and formulate our implicit layer as follows:

$$\boldsymbol{z} = f_{\text{implicit}}(\boldsymbol{s} + \boldsymbol{z}; \boldsymbol{\theta}) = \mathrm{Trans}(\mathrm{NL}(\boldsymbol{s} + \boldsymbol{z})), \quad (11)$$

where NL and Trans are the two sub-layers defined in (Eq. 9) and (Eq. 10), respectively.

To be specific, we model the equilibrium as the noise estimate. At the i-th solver iteration of f_{implicit}, we take an image estimate by subtracting the noise estimate $\boldsymbol{z}^{[i]}$ from the image representation $\boldsymbol{s}$, which is $\boldsymbol{s} + \boldsymbol{z}$, and then make another noise estimate $\boldsymbol{z}^{[i+1]}$. This is consistent with the feature decoding module we introduced in (Eq. 4), where the equilibrium $\boldsymbol{z}^*$ is used to restore the noise instead of the image.

2) Noise as the Equilibrium: In the iterative scheme, the image feature and the noise feature both reach steady states when the iterations converge. In the implicit scheme, we prefer the noise feature to the image feature as the equilibrium. To explain the benefits, we first introduce the relationship between the training stability and the Lipschitz constant of f_{implicit}, and based on which, we explain how casting the noise as the equilibrium facilitates model training.

The well-posedness of the implicit layer is closely related to the Lipschitz constant of f_{implicit}, which is defined as follows:

Definition 1: Given two metric spaces (X, d_X) and (Y, d_Y), where d_X denotes the metric on the set X and d_Y is the metric on set Y, a function $f : X \to Y$ is called Lipschitz continuous if there exists a real constant $K \geq 0$ such that, for all z_1 and z_2 in X, $d_Y(f(z_1) - f(z_2)) \leq K d_X(z_1 - z_2)$. If $0 \leq K < 1$ and f maps a metric space to itself, the function is called a contraction.

If f_{implicit} has a Lipschitz constant that is strictly less than 1, the unique solution is guaranteed according to the Banach fixed-point theorem [43]. In our specific layer formulation (Eq. (11)), the input injection $\boldsymbol{s}$ does not affect the Lipschitz constant, consequently it does not affect the well-posedness. The Lipschitz constant is also closely related to the efficiency when we seek the equilibrium. By Lyapunov linearization theorem, even the simplest fixed-point iteration solver could converge uniquely and enjoy global asymptotic stability. However, as previous works [34], [35] point out, strong contractivity on the dynamical system may significantly limit the representational capacity of the model in practice. Therefore, to constrain the Lipschitz constant of f_{implicit} in a soft way, we urge the parameters of f_{implicit} to be small by design.

To achieve this, we model the noise as the equilibrium. The point is that we expect the output of f_{implicit} (the noise) to have a smaller norm than its input (the image), which is reasonable and sustains in most real cases. As a result, during the training process, f_{implicit} will be optimized to have relatively smaller parameters $\boldsymbol{\theta}$, which reveals a smaller Lipschitz constant. Moreover, the setting makes $\mathbf{0}$ a reasonable initialization for the equilibrium, not only because smaller values generally suffice fixed-point equations, but also because it is nearer to z than $\boldsymbol{s} + z$.

V. Model Training

Given the above implicit layer (Eq. (11)), we now answer two questions: (i) In the forward pass, given an observation $\boldsymbol{y}$ and network weights, how do we compute a fixed point of (Eq. 11) efficiently? (ii) In the backward pass, given a collection of training samples, how do we directly differentiate through the equilibrium state?

A. Forward Pass

In the forward pass, we aim to solve the equilibrium of the implicit layer (Eq. (11)). Conventional iterative denoisers, if they converge to an equilibrium, can be considered as a form of fixed-point iterations. While in our work, we can exploit any black-box root-finding algorithm to solve for the equilibrium point. We formulate the model solving as the following root-finding problem [27]:

$$g(z, \boldsymbol{s}; \boldsymbol{\theta}) = f_{\text{implicit}}(\boldsymbol{s} + z; \boldsymbol{\theta}) - z,$$
$$z^* = \text{Rootfind}(g, z).$$

Newton's method is an ideal solver for the problem due to its efficiency. It has the following formulation:

$$z^{[i+1]} = z^{[i]} - \left(J_g^{-1}|_{z^{[i]}}\right) g(z^{[i]}, \boldsymbol{s}; \boldsymbol{\theta}),$$

where $J_g^{-1}|_{z^{[i]}}$ is the inverse Jacobian of g evaluated at $z^{[i]}$. However, accurately computing $J_g^{-1}|_{z^{[i]}}$ can be numerically unstable when it is ill-conditioned. To avoid calculating $J_g^{-1}|_{z^{[i]}}$ directly, Broyden's method [37] is proposed to use a low-rank matrix $B^{[i]}$ to approximate it:

$$z^{[i+1]} = z^{[i]} - \alpha B^{[i]} g(z^{[i]}, \boldsymbol{s}; \boldsymbol{\theta}),$$
$$B^{[i+1]} = B^{[0]} + \sum_{k=1}^{i} \boldsymbol{u}^{[k]} \boldsymbol{v}^{[k]T},$$

where $\boldsymbol{u}$ and $\boldsymbol{v}$ come from the Sherman-Morrison formula [44]. We initialize the internal state as $z^{[0]} = \mathbf{0}$, and leverage a limited-memory variant of Broyden's method [31], where we only keep the latest several low-rank updates at any step and discard the earlier ones. The Broyden iterations stop when either the relative residual $\left\| g(z^{[i]}, \boldsymbol{s}; \boldsymbol{\theta}) - g(z^{[i-1]}, \boldsymbol{s}; \boldsymbol{\theta}) \right\| / \left\| g(z^{[i-1]}, \boldsymbol{s}; \boldsymbol{\theta}) \right\|$ falls below a tolerance or when the maximum number of iterations is reached.

B. Backward Pass

In the backward pass, given a loss as (Eq. 6), the gradient can be calculated by the chain rule:

$$\frac{\partial \ell}{\partial (\cdot)} = \frac{\partial \ell}{\partial z^*} \frac{\partial z^*}{\partial (\cdot)}. \tag{12}$$

To calculate $\partial z^* / \partial (\cdot)$, we can implicitly differentiate two sides of (Eq. 3) with respect to $(\cdot)$:

$$\begin{aligned} \frac{\partial z^*}{\partial (\cdot)} &= \frac{\partial f_{\text{implicit}}(z^*, \boldsymbol{s}; \boldsymbol{\theta})}{\partial (\cdot)} \\ &= \frac{\mathrm{d} f_{\text{implicit}}(z^*, \boldsymbol{s}; \boldsymbol{\theta})}{\mathrm{d} (\cdot)} + \frac{\partial f_{\text{implicit}}(z^*, \boldsymbol{s}; \boldsymbol{\theta})}{\partial z^*} \frac{\partial z^*}{\partial (\cdot)}. \end{aligned} \tag{13}$$

Thus, differentiation through the implicit layer can be taken as solving the above linear Jacobian-based fixed-point equation about $\partial z^* / \partial (\cdot)$. We can derive its closed-form solution as follows:

$$\frac{\partial z^*}{\partial (\cdot)} = -\left(J_g^{-1}|_{z^*}\right) \frac{\mathrm{d} f_{\text{implicit}}(z^*, \boldsymbol{s}; \boldsymbol{\theta})}{\mathrm{d} (\cdot)}. \tag{14}$$

Plugging (Eq. 14) into (Eq. 12) gives

$$\frac{\partial \ell}{\partial (\cdot)} = -\frac{\partial \ell}{\partial z^*} \left(J_g^{-1}|_{z^*}\right) \frac{\mathrm{d} f_{\text{implicit}}(z^*, \boldsymbol{s}; \boldsymbol{\theta})}{\mathrm{d} (\cdot)}. \tag{15}$$

Previous works [27], [31] use Broyden's method [37] or Anderson Acceleration [38] to compute the exact gradient as in the forward pass, while we adopt a gradient estimate method called phantom gradient [35], which induces moderate gradient noise as regularization. Its core idea is to approximate J_g^{-1} with something easier to calculate. Specifically, we consider a damped variant of the fixed-point iteration without altering its equilibrium at the equilibrium z^*:

$$z_i = \lambda f_{\text{implicit}}(z_{i-1}, \boldsymbol{s}; \boldsymbol{\theta}) + (1 - \lambda) z_{i-1}, i = 1, 2, \ldots, T, \tag{16}$$

where λ, the damping factor, is a hyperparameter that contributes to maintain a small condition number of the Jacobian matrix. We use the subscript instead of the superscript to differ

it from the forward iteration when seeking the equilibrium. Note that the iterations start from the equilibrium z^*, instead of $\mathbf{0}$. By performing backpropagation through the unrolled steps of (Eq. 16), we have a gradient $\partial z_T/\partial \boldsymbol{\theta}$, as the estimate of the exact gradient $\partial z^*/\partial \boldsymbol{\theta}$. Under mild conditions, the former gives a descent direction of the loss landscape, and converges to the latter as T goes to infinity. In case the root-solver converges poorly, the gradient estimate resembles a backpropagation through time (BPTT) algorithm and still gives a descent direction of the loss landscape. Then, as training progresses, the solver becomes more stable and converges to the equilibrium better. As illustrated in Fig. 3, compared with unrolling-based explicit models, we use unrolling only in the backward pass for gradient estimate. In the forward pass, we do not need to store any intermediate variable.

VI. Relationship to Prior Works

Since we use a non-local block to build our model, in this section, we analyze its relationship with previous approaches that also incorporate the non-local self-similarity prior for denoising, showing that our model can be considered as an extension of the previous non-local module. We also show that our implicit layer can be considered as a learnable proximal operator [42] of an indicator function used in a variational approach, where the post-normalization helps to stabilize the level of the estimated noise.

A. Relations to Non-Local Operations

The seminal work of non-local means [11] triggers the wide study of non-local self-similarity (NSS) based methods for image denoising. NSS refers to the fact that there are many repeated local patterns across a natural image, and those non-local similar patches to a given patch can help much the reconstruction of it. Liu et al. [10] proposed a unified framework of non-local operations as follows:

$$\hat{\boldsymbol{Y}} = \text{diag}\{\mathcal{F}(\boldsymbol{Y})\}^{-1}\Phi(\boldsymbol{Y})G(\boldsymbol{Y}), \tag{17}$$

where $\boldsymbol{Y} \in \mathbb{R}^{N\times d_{\text{model}}}$ and $\hat{\boldsymbol{Y}} \in \mathbb{R}^{N\times d_{\text{model}}}$ denote the input and output of the non-local operations, respectively. $\Phi(\boldsymbol{Y}) \in \mathbb{R}^{N\times N}$ refers to the non-local correlation matrix. Each element $\Phi(\boldsymbol{Y})_{ij}$ represents the correspondence relationship between $\boldsymbol{Y}_i$ and $\boldsymbol{Y}_j$, where $\boldsymbol{Y}_i$ and $\boldsymbol{Y}_j$ refer to the image patch centered at i and j, respectively. $G(\boldsymbol{Y}) \in \mathbb{R}^{N\times d_{\text{model}}}$ is the embedding of $\boldsymbol{Y}$, and the diagonal matrix $diag\{\mathcal{F}(\boldsymbol{Y})\} \in \mathbb{R}^{N\times N}$ normalizes the output.

This framework works with various model-based non-local methods [6], [7], [11] and learning-based non-local methods [19], [20]. They differ most from each other on how to model the correlation matrix $\Phi(\boldsymbol{Y})$. Traditional methods such as BM3D [6] and early-stage neural network models such as BM3D-Net [45], NLNet [15] and UNLNet [16] use block matching to exploit non-local image structures. In these works, $\Phi(\boldsymbol{Y})$ is a hard 0-1 mask, where patches except the most similar ones to the referenced patch are ignored. It is non-differentiable and hard to optimize, thus restricting the denoising performance. Subsequent neural network models [10], [19] adopt soft block matching to learn deep feature representations. Our non-local module also belongs to this category. In addition, we extend the previous single-head framework (Eq. (17)) to a multi-head setting as follows:

$$\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{Y}}_1 \| \dots \| \hat{\boldsymbol{Y}}_H)W_O, \tag{18a}$$

$$\hat{\boldsymbol{Y}}_h = \text{diag}\{\mathcal{F}_h(\boldsymbol{Y})\}^{-1}\Phi_h(\boldsymbol{Y})G_h(\boldsymbol{Y}), h = 1, 2 \dots H, \tag{18b}$$

where $\hat{\boldsymbol{Y}}_h \in \mathbb{R}^{N\times d_v}$ and $W_O \in \mathbb{R}^{Hd_v\times d_{\text{model}}}$. In this framework, letting $\Phi_h(\boldsymbol{Y}) = \exp(\boldsymbol{Y}W^h_{\boldsymbol{\theta}}(\boldsymbol{Y}W^h_{\psi})^T)$, $G_h(\boldsymbol{Y}) = \boldsymbol{Y}W^h_g$, and $\mathcal{F}_h(\boldsymbol{Y})$ be a row-sum operator, we can recover the MSA used in our model (Eq. (8)). Moreover, our non-local module includes the non-local module in NLRN [10] as a degenerated case when H equals 1, where the model can only learn one kind of correspondence relationship and cannot characterize different image patterns jointly.

B. Relations to Variational Denoisers

We show in this section that our implicit function can be considered as a learnable proximal operator [42] of an indicator function used in a variational approach. To verify this, we return to the inverse problem presented in (Eq. 1). To mitigate the ill-posed nature of image denoising, variational approaches [16] attempt to cast it as a minimization problem as follows:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{u}} r(\boldsymbol{u}), s.t. \|\boldsymbol{y} - \boldsymbol{u}\|_2^2 \le \varepsilon. \tag{19}$$

The objective is to minimize $r(\boldsymbol{u})$ within a convex set $\mathcal{C} = \{\boldsymbol{u} | \|\boldsymbol{y} - \boldsymbol{u}\|_2^2 \le \varepsilon\}$, where $r(\boldsymbol{u})$ is a regularization term that encodes prior knowledge for image denoising, and ε is a parameter that measures the proximity of the solution $\boldsymbol{u}$ to the noisy observation $\boldsymbol{y}$. In other word, ε is directly associated with the noise level. To solve the constrained problem (Eq. 19), we refer to the following unconstrained minimization problem:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{u}} r(\boldsymbol{u}). \tag{20}$$

Suppose that $h(\cdot)$ is an update rule (*e.g.,* a gradient descent step) to solve (Eq. 20), we can solve (Eq. 19) by first applying $h(\cdot)$ and then projecting onto $\mathcal{C}$, which gives the updating scheme as follows [16]:

$$\boldsymbol{u}^{[i]} = \boldsymbol{y} + \varepsilon\frac{h(\boldsymbol{u}^{[i-1]}) - \boldsymbol{y}}{\max(\|h(\boldsymbol{u}^{[i-1]}) - \boldsymbol{y}\|_2^2, \varepsilon)}. \tag{21}$$

If we denote $\tilde{f}_{\text{implicit}}$ as a variant of f_{implicit} that drops the post-normalization, *i.e.,* $f_{\text{implicit}}(\cdot) = \text{GN}(\tilde{f}_{\text{implicit}}(\cdot))$, and denote $\boldsymbol{s}^{[i]} = \boldsymbol{s} + \boldsymbol{z}^{[i]}$ as the i-th image estimate, we can reformulate our implicit layer (Eq. (11)) as follows:

$$\boldsymbol{s}^{[i]} = \boldsymbol{s} + \text{GN}(\tilde{f}_{\text{implicit}}(\boldsymbol{s}^{[i-1]}; \boldsymbol{\theta})). \tag{22}$$

Then we have an obvious observation that (Eq. 21) coincides with (Eq. 22) in the sense that: (i) $\boldsymbol{s}^{[i]}$ corresponds to the feature of $\boldsymbol{u}^{[i]}$, and $\boldsymbol{s}$ corresponds to the feature of $\boldsymbol{y}$; (ii) $\tilde{f}_{\text{implicit}}(\boldsymbol{s}^{[i-1]}; \boldsymbol{\theta})$ corresponds to the feature of $h(\boldsymbol{u}^{[i-1]}) - \boldsymbol{y}$, which is an update rule to improve the accuracy of the noise estimate without constraints on the noise level; (iii) The post-normalization in our model corresponds to the projection operation in (Eq. 21), where ε is absorbed into $\boldsymbol{\theta}$ as a learnable

TABLE I
NATURAL IMAGE DENOISING RESULTS TRAINED ON BSD. METRICS ARE PNSR (DB) AND SSIM

Dataset	Noise	BM3D	WNNM	DnCNN	NLRN	N^3Net	GCDN	NERD
Set12	15	32.37/0.8952	32.70/0.8982	32.86 / 0.9031	33.16 / 0.9070	-/-	33.14/0.9072	**33.20/0.9118**
	25	29.97/0.8504	30.28/0.8557	30.44 / 0.8622	30.79 / 0.8689	30.50/0.8702	30.78/0.8687	**30.84/0.8736**
	50	26.72/0.7676	27.05/0.7775	27.18 / 0.7829	27.64 / 0.7980	27.41/0.8022	27.60/0.7957	**27.72/0.8075**
BSD68	15	31.07/0.8717	31.37/0.8766	31.73 / 0.8907	31.88 / 0.8932	-/-	31.83/0.8933	**31.91/0.8975**
	25	28.57/0.8012	28.83/0.8087	29.23 / 0.8278	29.41 / 0.8331	29.30/**0.8379**	29.35/0.8332	**29.43**/0.8370
	50	25.62/0.6864	25.87/0.6982	26.23 / 0.7189	26.47 / 0.7298	26.39/0.7378	26.38/**0.7389**	**26.49/0.7389**
Urban100	15	32.34/0.9220	32.97/0.9271	32.68 / 0.9255	33.42 / 0.9348	-/-	33.47/**0.9358**	**33.48**/0.9341
	25	29.70/0.8777	30.39/0.8885	29.97 / 0.8797	30.86 / 0.9003	30.19/0.8921	30.95/**0.9020**	**31.03**/0.9013
	50	25.95/0.7791	26.83/0.8047	26.28 / 0.7874	27.40 / 0.8244	26.83/0.8158	27.41/0.8160	**27.62/0.8292**

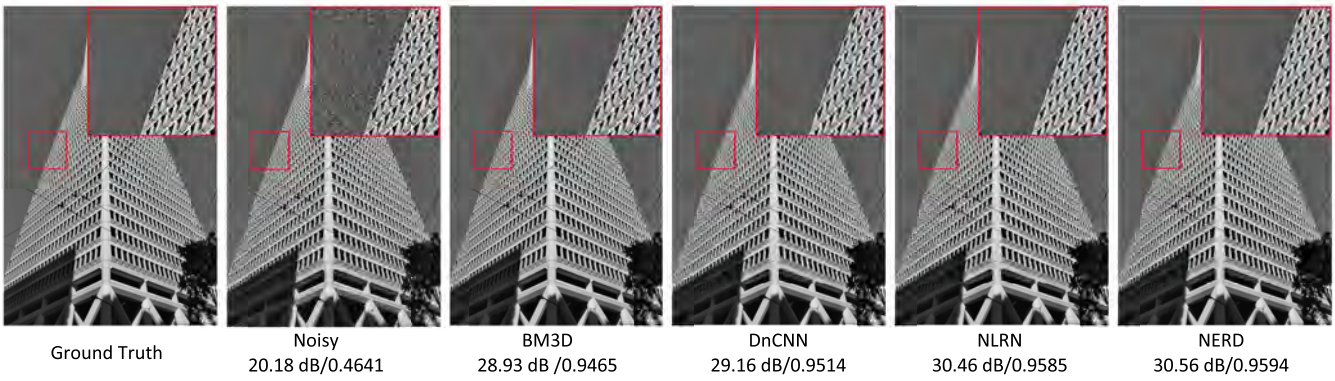

Fig. 4. Qualitative comparison of image denoising results with the noise level of 25. The zoom-in region in the red bounding box is shown in the upper right corner. From left to right: i) the ground truth image from Urban100 scene 48, ii) the corresponding noisy image, iii) the BM3D denoising result, iv) the DnCNN denoising result, v) the NLRN denoising result and vi) the NERD denoising result.

parameter. As a result, our implicit function can be considered as a learnable variant of the updating rule (Eq. (21)) used to solve the constrained minimization problem (Eq. (19)) in a variational approach, and the post-normalization corresponds to the projection operation which helps to stabilize the level of the estimated noise.

VII. EXPERIMENTS

In this section, we evaluate our NERD model on benchmark image denoising datasets, including natural images and depth maps. In addition, we perform through ablation studies to validate the design of the proposed architecture, the training strategy and the selection of the hyperparameters.

A. Training Details

1) Datasets: To make a fair comparison, we adopt two different training datasets. The first one is selected from the Berkeley Segmentation Dataset (BSD) [46], which is used by most previous works [10], [47]. We choose the combination of 200 images from the train set and 200 images from the test set in the BSD for training. Note that training images are strictly separated from the validation set of the BSD. The second one is DIV2K [48] dataset, which contains 800 high-quality images and is used by some recent works [18], [21]. All the images are converted to gray-scale in each experiment setup. Peak-Signal-to-Noise (PSNR) and Structured Similarity Index Measure (SSIM) [49] are adopted for measuring quantitative restoration performance.

2) Training: Unless otherwise stated, we adopt the following training protocol. For each epoch, we randomly crop 128 patches of size 64×64 from each training image. We use horizontal and vertical flipping as well as random rotations as further data augmentation. We add independent and identically distributed Gaussian noise with zero mean and variance σ^2 to the original image as the noisy input during training, where $\sigma \in \{15, 25, 50\}$ refers to the noise level. We train for 100 epochs in total with a batch size of 4, using the AdamW optimizer [50] with default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to minimize the mean squared error (MSE) between the denoised patch output $\hat{\boldsymbol{x}}$ by the network and the ground truth $\boldsymbol{x}$. For all experiments, we use the cosine learning rate schedule and initially set the learning rate to 10^{-3}. About the model architecture, we use GroupNorm with 4 groups, and keep learnable affine parameters. We use 4-head attention for the non-local module. The number of features used in all convolutional layers is 128, except for our transformation block, which has a $4\times$ expansion in its first layer. As we mentioned in Sec. V, to solve the equilibrium in the forward pass, we use a limited-memory version of Broyden's method [31] with maximum step 17, and to calculate the gradient in the backward pass, we use the phantom gradient strategy with $T = 8$ and $\lambda = 0.8$ to approximate the exact value. Our model also adopts some common techniques beneficial for explicit neural networks in our implicit layer, such as variational dropout [51] and weight normalization [52].

B. Comparison With State of the Art

1) Natural Image Denoising: We compare our method with state-of-the-art denoising methods for the Gaussian denoising task of natural grayscale images. The reference methods can be classified as model-based algorithms such as BM3D [6] and WNNM [7], and neural network models such as DnCNN [41], NLRN [10], N^{3}Net [19], GCDN [20], DAGL [21] and DeamNet [18]. In particular, the two traditional methods are both non-local iterative approaches. Among the neural network methods, all learnable models except DnCNN are non-local methods, and all the models are compared with baselines trained on the same dataset for a fair comparison, except DeamNet that adopts a larger dataset. Following them, we test on three different datasets: a set of twelve commonly used benchmark images (Set12), the 68 images subset [53] of the BSD validation set, and the Urban100 [54] dataset, which contains 100 images of urban scenes with repetitive patterns. Table I shows the denoising results trained on BSD with three different noise levels obtained from their papers, except for N^{3}Net at $\sigma = 15$ which is unavailable. It can be seen that the proposed method achieves state-of-the-art performance on all datasets, despite that they have different statistics. For example, the Urban100 dataset contains higher resolution images compared with the other two, and it is mainly composed of photos of buildings and other regular structures, where exploiting self-similarity is very important, while the Set12 dataset contains images with a jumble of irregular hair and carpet textures. We also notice that as the noise level increases, our model degrades less than other methods do. For example, on the Urban100 dataset with the noise level of 15, 25 and 50, our PSNR is 0.01, 0.08 and 0.21 dB higher than GCDN, and 0.06, 0.17 and 0.22 dB higher than NLRN, respectively. The phenomenon shows that our model is more robust to high-level noise. This can be explained by that iterative methods whose iteration steps are fixed have more difficulty in learning pixel relationships from the noisy image at higher noise levels, while our model can flexibly iterate until reaching a stable state of the correspondence relationship. In addition, our proposed method provides a better visual quality, which can be verified by Fig. 4 that shows denoising results for an image from the Urban100 dataset. We notice that BM3D and DnCNN have lots of patch-like artifacts in the sky, and NLRN also has an obvious gray point artifact. Compared with them, our NERD produces a denoising result with the least artifacts in uniform areas, while at the same time preserves the sharpest edges in structural areas.

Moreover, as we can see in Table II, when trained on the larger DIV2K dataset, our NERD performs significantly better on the largest dataset Urban100. To be specific, our NERD outperforms the other models by more than 0.18 dB with all the noise levels. The performance is even comparable to Transformer denoising models. For example, SwinIR [55] archives 33.70 dB, 31.30 dB and 27.98 dB on Urban100 with the noise level of 15, 25 and 50, respectively. And our results are 33.77 dB, 31.22 dB and 27.96 dB, respectively. Note that SwinIR adopts 8594 images from 4 different datasets to achieve the results, while our model only adopts 800 images from one DIV2K dataset. We think the superior performance of DeamNet on BSD68 may result from its training dataset, where BSD68 is drawn from the original BSD500 dataset. Also, we think the reason for DAGL's superior performance on Set12 might be that its patch-wise self-similarity assumption is well suited to Set12. Also, since Set12 only has 12 images, we think the performance difference is negligible.

TABLE II
NATURAL IMAGE DENOISING RESULTS TRAINED ON DIV2K. METRICS ARE PNSR (DB) AND SSIM

Training Dataset		DIV2K	DIV2K+BSD	DIV2K
Dataset	Noise	DAGL	DeamNet	NERD
Set12	15	**33.24/0.9137**	33.19/0.9097	33.20/0.9111
	25	**30.89/0.8773**	30.81/0.8717	30.83/0.8724
	50	**27.77/0.8116**	27.74/0.8057	27.64/0.8053
BSD68	15	31.82/0.8890	**31.91/0.8957**	31.88/**0.8957**
	25	29.41/0.8278	**29.44/0.8373**	29.38/0.8345
	50	**26.56**/0.7228	26.54/**0.7368**	26.47/0.7358
Urban100	15	33.28/0.9298	33.37/0.9372	**33.77/0.9389**
	25	31.01/0.8974	30.85/**0.9048**	**31.22**/0.9035
	50	27.78/0.8283	27.53/0.8373	**27.96/0.8384**

2) Depth Map Denoising: To further verify the effectiveness and robustness of our NERD when extended to datasets with piecewise smooth content, we compare with state-of-the-art methods for denoising of depth maps [56], *e.g.,* images generated by time-of-flight cameras. We follow GCDN and report results achieved on a standard set of depth maps.[1] Since each image has disparity maps from two views, we choose the left view for evaluation, and obtain the results by running the pretrained models provided by the authors. As shown in Table III, the proposed NERD outperforms other baselines significantly on PSNR results. Considering the piece-wise smoothness property of the depth map dataset as shown in Fig. 5, we think GCDN's higher SSIM might result from its non-local operation, which selects only a few most similar pixel neighbors for each query pixel. In a piece-wise smooth image, pixels are easy to find almost identical neighbors. Consequently, considering a few very similar pixels is enough for prediction, while considering all neighbors may bring interruption. Fig. 5 shows fragments of the image Bowling, where the original fragment and the noisy version, accompanied by the denoised results, are presented for comparison. We notice that our NERD is the only model that restores the shape of the bowling pin. Moreover, our model simultaneously smooths the flat areas, while GCDN and NLRN have many patch-wise artifacts on the surface of the ball. Also, note that GCDN has 18M parameters, while our model has only 1.4M parameters, which is less than 1/10 of GCDN.

C. Ablation Studies on Architecture Design

1) Noise as the Equilibrium: First, to verify that the equilibrium z^* in our model corresponds to the noise, and the injection feature s corresponds to the image, as we claimed in Sec. IV-B, we visualize them by using the decoding function f_{decode} as a visualization tool to transform them into the

[1] https://vision.middlebury.edu/stereo/data/scenes2006/HalfSize/zip-2views/

TABLE III
DEPTH MAP DENOISING RESULTS TRAINED ON BSD. METRICS ARE PNSR (DB) AND SSIM

σ	Method	Aloe	Bowling	Flowerpots	Lampshade	Midd	Monopoly	Plastic	Average
15	NLRN	40.57/0.9849	43.48/0.9906	42.82/0.9900	43.32/0.9906	42.64/0.9897	44.02/0.9918	45.54/0.9933	43.20/0.9900
	GCDN	40.36/**0.9870**	43.24/**0.9922**	42.67/**0.9938**	43.16/**0.9929**	42.38/**0.9921**	43.31/**0.9923**	45.10/**0.9949**	42.89/**0.9922**
	NERD	**40.93**/0.9865	**43.85**/0.9908	**43.13**/0.9867	**43.72**/0.9910	**43.29**/0.9899	**44.30**/0.9920	**46.17**/0.9936	**43.63**/0.9901
25	NLRN	37.25/0.9722	39.91/0.9823	38.98/0.9816	39.63/0.9819	39.06/0.9805	40.15/0.9835	42.12/0.9878	39.59/0.9814
	GCDN	37.11/**0.9769**	39.95/**0.9868**	38.95/**0.9874**	39.73/**0.9877**	38.95/**0.9857**	39.90/**0.9871**	41.95/**0.9919**	39.51/**0.9862**
	NERD	**37.50**/0.9736	**40.17**/0.9818	**39.13**/0.9805	**39.94**/0.9819	**39.36**/0.9805	**40.37**/0.9838	**42.72**/0.9866	**39.88**/0.9812
50	NLRN	33.34/0.9454	35.58/0.9666	34.13/0.9550	35.19/0.9617	34.26/0.9547	35.25/0.9669	36.95/0.9767	34.96/0.9610
	GCDN	33.37/**0.9538**	35.28/**0.9713**	33.91/**0.9692**	35.03/**0.9711**	34.14/**0.9680**	35.03/**0.9693**	36.75/**0.9811**	34.79/**0.9691**
	NERD	**33.81**/0.9494	**36.03**/0.9646	**34.40**/0.9584	**35.39**/0.9618	**34.65**/0.9603	**35.60**/0.9649	**37.43**/0.9711	**35.33**/0.9615

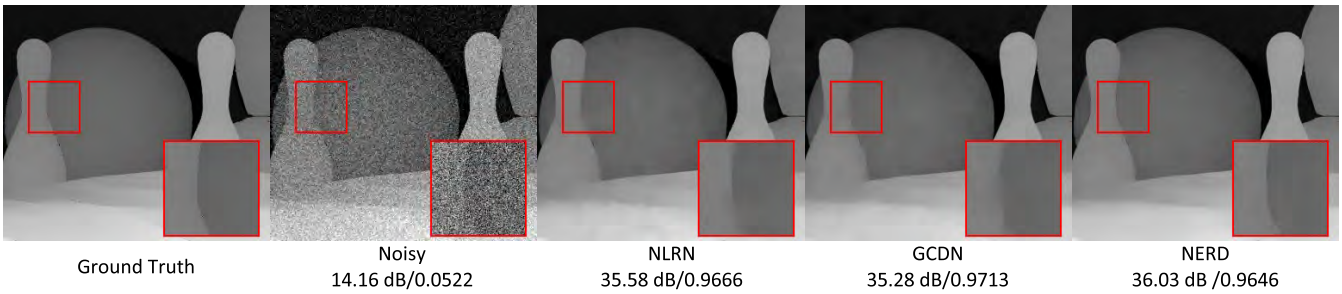

Fig. 5. Qualitative comparison of image denoising results with the noise level of 50. The zoom-in region in the red bounding box is shown in the lower right corner. From left to right: i) the ground truth image called Bowling from the depth map dataset, ii) the corresponding noisy image, iii) the NLRN denoising result, iv) the GCDN denoising result, v) the NERD denoising result.

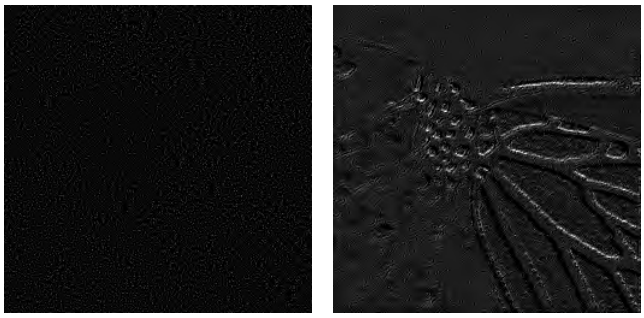

Fig. 6. Visualization of features by transforming them into the image space through f_{decode}. Left: the noise equilibrium when processing the image Monarch; and right: the corresponding input injection.

image space. Fig. 6 shows the visualization results of the learned equilibrium and injection feature when denoising the image Monarch, an image in Set12. Its ground truth and corrupted version are shown in Fig. 1. We can clearly see that the equilibrium feature z^* corresponds to a group of randomly distributed scatter points, which is compatible with the characteristic of the noise. On the contrary, the injection feature $\boldsymbol{s}$ contains patterns in the input image. We can clearly recognize the butterfly wings from its visualization. This substantiates our claim that the learned implicit layer is a noise estimator, the input injection module keeps the information of the corrupted image, and the subsequent updates refine the noise equilibrium and remove it from the injection, as shown in Fig. 1.

Second, we demonstrate the effectiveness of stabilizing the training process by taking the noise as the equilibrium.

TABLE IV
COMPARISON OF MODELS TAKING THE IMAGE VS. THE NOISE AS THE EQUILIBRIUM ON SET12 WITH DIFFERENT NOISE LEVELS

Noise Level	Equilibrium	Performance
15	Noise	**33.20/0.9118**
15	Image	31.92/0.8883
25	Noise	**30.84/0.8736**
25	Image	30.40/0.8663
50	Noise	**27.72/0.8075**
50	Image	26.55/0.7440

To construct a baseline where we take the image instead of the noise as the equilibrium, we decode the equilibrium as the image, *i.e.*, $\hat{\boldsymbol{x}} = f_{\text{decode}}(z^*)$ (compared with $\hat{\boldsymbol{x}} = \mathbf{y} + f_{\text{decode}}(z^*)$). We compare the two settings with three different noise levels: 15, 25, and 50. Table IV shows their best denoising results among the training epochs. As we can see, models casting the image as the equilibrium achieve much worse performances than models which cast the noise as the equilibrium. To visually assess the training process, we plot the learning curves with the noise level of 50 in Fig. 7. We observe that taking the image as the equilibrium leads to severe fluctuations in the early training stage and confines the final performance.

2) Post-Normalization in the Transformation Block: We also verify that the post-normalization in the transformation block, as we claimed in Sec. VI-B, not only stabilizes the training process, but also facilitates the learned noise level. First, to verify its effectiveness on stabilizing the training process, we construct a baseline by discarding the post-normalization

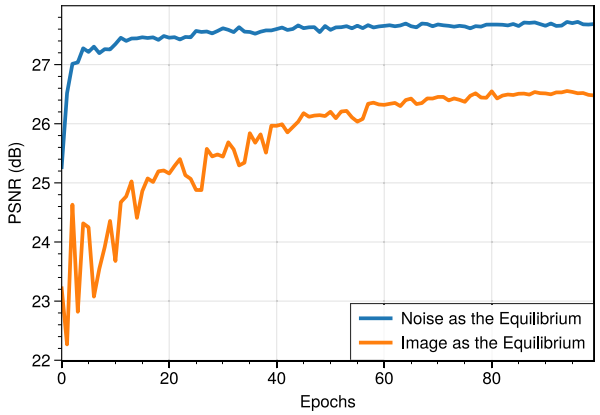

Fig. 7. The learning curves of models with noise as the equilibrium and image as the equilibrium. On the vertical axis is the average PSNR on Set12 with the noise level of 50.

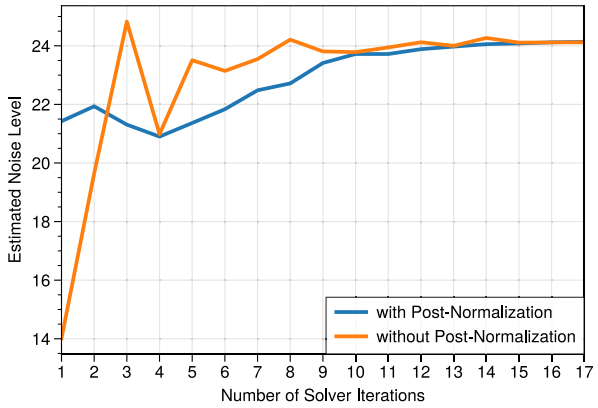

Fig. 8. Changing curves of the estimated noise level when denoising the image "Monarch" with the noise level of 25. On the vertical axis is the estimated noise level at each solver iteration when seeking the equilibrium, which is calculated by averaging the noise standard deviations among image patches.

TABLE V

COMPARISON OF MODELS W/ VS. W/O POST-NORMALIZATION WHEN TESTING ON SET12 WITH THE NOISE LEVEL OF 25

	w/ Post-Norm	w/o Post-Norm
Performance	**30.84/0.8736**	30.78/0.8735
Norm of Equilibrium	11173.89	36920.28
Var of Equilibrium	8.28	88.78

TABLE VI

QUANTITATIVE DENOISING RESULTS WITH DIFFERENT NUMBER OF ATTENTION HEADS ON SET12 CORRUPTED WITH THE NOISE LEVEL OF 25

Number of attention heads	1	4	8
PSNR	30.79	**30.84**	30.78
SSIM	**0.8736**	**0.8736**	0.8735

and leaving other parts unchanged. In Table V we present the quantitative denoising results, as well as the Frobenius norm and variance of the learned equilibrium when testing on the Set12 dataset. It can be seen that normalizing the equilibrium reduces the Frobenius norm by one third and the variance by more than 90%, indicating that the model is more stable. Meanwhile, it also improves the denoising performance from 30.78 dB to 30.84 dB.

Second, to verify that it stabilizes the level of the estimated noise, we still take the image Monarch as an example and print the value at each solver iteration from $i = 1$ to 17 for assessment. For each intermediate variable $\boldsymbol{z}^{[i]}$, we transform it into the image space by f_{decode} to get the corresponding noise estimate $\boldsymbol{n}^{[i]}$. Then we take its standard deviation as the estimated noise level and plot the changing curves in Fig. 8. We can clearly see that the post-normalization helps to stabilize the estimated noise level around the expected value. In contrast to that, without post-normalization, the estimated noise level changes drastically at the first few evaluations. Note that the final denoising result is 30.64 dB/0.9261 without the post-normalization, compared with 30.67 dB/0.9273 with the post-normalization.

3) Multi-Head Correlation Maps: In Sec. VI-A, we have claimed that our model benefits from learning multiple kinds of non-local correlation maps, thus is more powerful to model complex correspondence relationships. We experimentally verify this in Table VI, which investigates the impact of attention heads on denoising results. As we can see, although the SSIM metric only changes slightly as the number of the heads increases, the PSNR metric has a peak performance with 4 heads, 0.05 dB higher than the single head setting. Therefore, it is necessary to choose an appropriate number of attention heads for image denoising.

Besides the quantitative comparison, we also visualize the correlation maps computed by a 4-head NERD model. The correlation map, in this case, corresponds to the attention map of the MSA in our non-local block (see Sec. IV-A). We show a patch extracted from the Set12 dataset in Fig. 9, as well as two red points and their corresponding correlation maps. One point is in the white flat area (the first row) and the other is in the black edge area (the second row). It can be seen that our non-local module learns different kinds of correspondence relationships. For example, for the red point located at the white flat area, we can see that its first and third heads capture relatively local relationships within the white area, while its second and fourth heads capture relatively global correspondence over the whole patch. In addition, the second head captures relationships with the edge patterns, which complements the other heads that only consider flat areas, showing that our model can characterize different image patterns jointly. Moreover, the maps differ in their values, which reflects the degree the pixels are mixed with each other.

4) Architecture of the Implicit Denoising Module: As we described in Sec. IV-A, our implicit function contains a non-local block to employ self-similarities, and a transformation block to capture local patterns. To see how they contribute to the final performances, we compare the denoising results with only one individual block. As shown in Table VII, a single transformation block works better than a single non-local block, and neither of them works as well as when combined. This ablation study verifies the necessity of capturing both

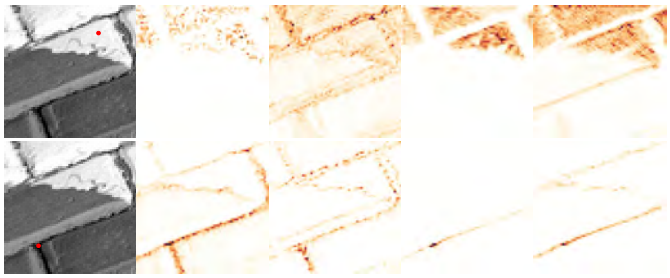

Fig. 9. Visualization of the multi-head correlation maps. From left to right: the neighborhood of the red pixel and its corresponding correlation map of the 4 heads.

TABLE VII

QUANTITATIVE DENOISING RESULTS WITH DIFFERENT IMPLICIT MODULES ON SET12 CORRUPTED WITH THE NOISE LEVEL OF 25

Non-local Block	Transformation Block	Performance PSNR/SSIM
✓		30.10/0.8558
	✓	30.62/0.8716
✓	✓	**30.84/0.8736**

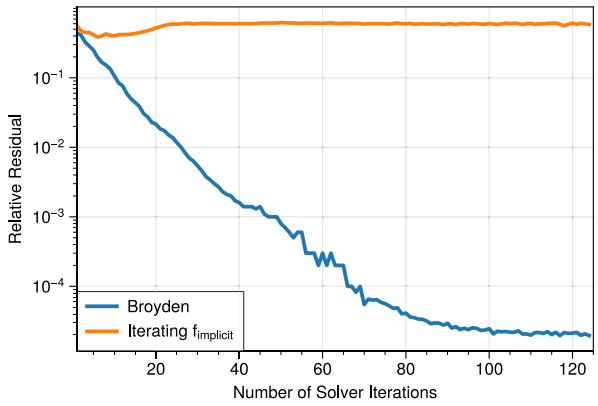

Fig. 10. NERD finds the equilibrium in a stable and efficient manner (whereas simply iterating f_{implicit} could oscillate around the fixed point).

non-local and local patterns, showing that they are both indispensable.

D. Empirical Studies on Model Training

1) Convergence Analysis: In Sec. V, we introduced that we adopt a limited-memory variant of Broyden's method [31] to calculate the equilibrium of (Eq. 11) in the forward pass. We verify in experiments that our model can converge to the equilibrium in practice. Fig. 10 displays the relative residual $\left\|\mathbf{z}^{[i+1]}-\mathbf{z}^{[i]}\right\| / \left\|\mathbf{z}^{[i]}\right\|$ as a function of the number of times we evaluate f_{implicit}. Note that i starts from 1 instead of 0, since $\mathbf{z}^{[0]}$ is initialized as $\mathbf{0}$. We can see that simply iterating f_{implicit} cannot reach the equilibrium efficiently, while our NERD exhibits stable convergence with the Broyden's method.

2) Memory Footprint and Runtime Consumption: We provide a memory and runtime analysis here to demonstrate the efficiency of our model. For conventional deep networks with L layers, the training memory complexity is $O(L)$ since all intermediate activations are stored for backpropagation. In comparison, our NERD have an $O(1)$ memory footprint

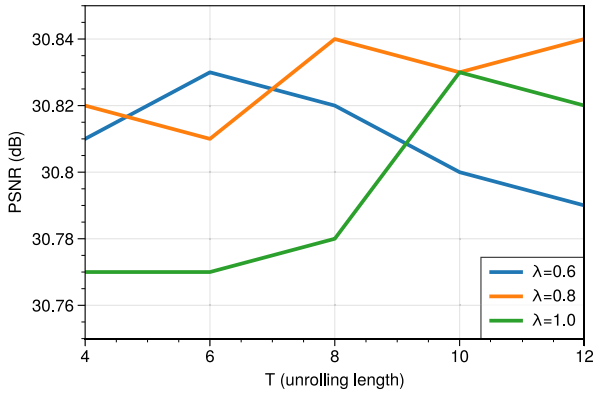

Fig. 11. The influence of the hyperparameters λ and T in the phantom gradient. On the vertical axis is the average denoising results on Set12 with the noise level of 25.

TABLE VIII

THE INFLUENCE OF THE NEIGHBORHOOD SIZE ON THE DENOISING PERFORMANCE ON SET12 WITH THE NOISE LEVEL OF 25

Neighborhood Size	50	55	60	65	70
PSNR	30.79	30.81	30.82	**30.84**	30.83
SSIM	0.8737	**0.8745**	0.8739	0.8736	0.8735

for inference due to the root-finding formulation. In backward pass, although the phantom gradient requires $O(T)$ memory, where T corresponds to the unrolling steps we use to estimate the gradient, it avoids the pretraining stage that is used by the exact implicit differentiation for initialization. For example, the MDEQ model [31] employs a 10-layer unrolling for pretraining, which usually consumes more memory compared with the phantom gradient estimate. In a word, we trade off a little consumption for training to optimize the equilibrium more stably and efficiently.

Compared to the 15-layer NLRN model trained on the same dataset for the same epochs, our model with $T = 8$ costs $1.8\times$ runtime due to the root-finding process. However, we empirically observe that we can decouple the numbers of solver iterations used for training and inference flexibly to allow the model to be evaluated faster with only a small degradation in performance, as Fig. 1 shows.

E. Hyperparameter Analysis

We investigate the influence of varying hyperparameters T and λ in phantom gradient estimate. As shown in Fig. 11, given the trade-off between the restoration accuracy and inference time, we adopt $T = 8$ and $\lambda = 0.8$ for NERD in all other experiments.

Table VIII investigates the influence of the neighborhood size q in the non-local module on denoising results. The PSNR performance peaks at $q = 65$, while the SSIM performance peaks at $q = 55$, showing that limiting the neighborhood helps concentrate the correlation calculation on relevant features in the spatial vicinity and enhance correlation estimation. This is consistent with observations in preceding works [10]. Note that with our method, it is flexible to use different neighborhood

sizes for inference if one wants to balance between efficiency and effectiveness.

VIII. Conclusion

In this paper, we propose a novel implicit network for image denoising, which largely mitigates the training difficulties of deep neural networks. We construct an implicit layer as our denoising module, which is an input-dependent fixed-point equation. Each fixed-point iteration takes an image estimate as input, and makes a noise estimate by passing the feature through a non-local block to capture image self-similarities, and a successive transformation block to capture local patterns. We adopt a post-normalization mechanism to stabilize the level of the estimated noise and at the same time facilitate the model training. In the forward pass, the model makes prediction by achieving the noise equilibrium through accelerated black-box solvers. While in the backward pass, the model employs implicit differentiation, thereby avoids the training problems of explicit models and benefits from infinite iterations. Extensive experiments demonstrate that our model significantly improves both quantitative and qualitative performances of image denoising even with high noise levels, while being light-weighted.

References

[1] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *Proc. IEEE Global Conf. Signal Inf. Process.*, Dec. 2013, pp. 945–948.

[2] M. V. Afonso, J.-M. Bioucas-Dias, and M. A. T. Figueiredo, "Fast image recovery using variable splitting and constrained optimization," *IEEE Trans. Image Process.*, vol. 19, no. 9, pp. 2345–2356, Sep. 2010.

[3] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (RED)," *SIAM J. Imag. Sci.*, vol. 10, no. 4, pp. 1804–1844, 2017.

[4] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3929–3938.

[5] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.

[6] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3D transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[7] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted nuclear norm minimization with application to image denoising," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2862–2869.

[8] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. Image Process.*, vol. 15, no. 12, pp. 3736–3745, Dec. 2006.

[9] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18–44, Mar. 2021.

[10] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang, "Non-local recurrent network for image restoration," in *Proc. NeurIPS*, 2018, pp. 1673–1682.

[11] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2005, pp. 60–65.

[12] C. R. Vogel, *Computational Methods for Inverse Problems*, vol. 23. Philadelphia, PA, USA: SIAM, Jan. 2002.

[13] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D, Nonlinear Phenomena*, vol. 60, nos. 1–4, pp. 259–268, 1992.

[14] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1256–1272, Aug. 2017.

[15] S. Lefkimmiatis, "Non-local color image denoising with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3587–3596.

[16] S. Lefkimmiatis, "Universal denoising networks: A novel CNN architecture for image denoising," in *Proc. CVPR*, Jun. 2018, pp. 3204–3213.

[17] M. Scetbon, M. Elad, and P. Milanfar, "Deep K-SVD denoising," *IEEE Trans. Image Process.*, vol. 30, pp. 5944–5955, 2021.

[18] C. Ren, X. He, C. Wang, and Z. Zhao, "Adaptive consistency prior based deep network for image denoising," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8596–8606.

[19] T. Plötz and S. Roth, "Neural nearest neighbors networks," in *Proc. NeurIPS*, 2018, pp. 1087–1098.

[20] D. Valsesia, G. Fracastoro, and E. Magli, "Deep graph-convolutional image denoising," *IEEE Trans. Image Process.*, vol. 29, pp. 8226–8237, 2020.

[21] C. Mou, J. Zhang, and Z. Wu, "Dynamic attentive graph learning for image restoration," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 4328–4337.

[22] X. Mao, C. Shen, and Y.-B. Yang, "Image restoration using very deep convolutional encoder–decoder networks with symmetric skip connections," in *Proc. NeurIPS*, vol. 29, 2016, pp. 2802–2810.

[23] Y. Tai, J. Yang, X. Liu, and C. Xu, "MemNet: A persistent memory network for image restoration," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4539–4547.

[24] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proc. ICML*, 2013, pp. 1310–1318.

[25] L. El Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Tsai, "Implicit deep learning," *SIAM J. Math. Data Sci.*, vol. 3, no. 3, pp. 930–958, 2021.

[26] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Proc. NeurIPS*, vol. 31, 2018, pp. 1–13.

[27] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," in *Proc. NeurIPS*, vol. 32, 2019, pp. 688–699.

[28] S. G. Krantz and H. R. Parks, *The Implicit Function Theorem: History, Theory, and Applications*. New York, NY, USA: Springer, 2012.

[29] E. Winston and J. Z. Kolter, "Monotone operator equilibrium networks," in *Proc. NeurIPS*, vol. 33, 2020, pp. 10718–10728.

[30] M. Revay, R. Wang, and I. R. Manchester, "Lipschitz bounded equilibrium networks," 2020, *arXiv:2010.01732*.

[31] S. Bai, V. Koltun, and J. Z. Kolter, "Multiscale deep equilibrium models," in *Proc. NeurIPS*, vol. 33, 2020, pp. 5238–5250.

[32] T. Wang, X. Zhang, and J. Sun, "Implicit feature pyramid network for object detection," 2020, *arXiv:2012.13563*.

[33] D. Gilton, G. Ongie, and R. Willett, "Deep equilibrium architectures for inverse problems in imaging," *IEEE Trans. Comput. Imag.*, vol. 7, pp. 1123–1133, 2021.

[34] S. Bai, V. Koltun, and Z. Kolter, "Stabilizing equilibrium models by Jacobian regularization," in *Proc. ICML*, 2021, pp. 554–565.

[35] Z. Geng, X.-Y. Zhang, S. Bai, Y. Wang, and Z. Lin, "On training implicit models," in *Proc. NeurIPS*, vol. 34, 2021, pp. 24247–24260.

[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015, pp. 448–456.

[37] C. G. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Math. Comput.*, vol. 19, no. 92, pp. 577–593, 1965.

[38] H. F. Walker and P. Ni, "Anderson acceleration for fixed-point iterations," *SIAM J. Numer. Anal.*, vol. 49, no. 4, pp. 1715–1735, Jan. 2011.

[39] A. Vaswani et al., "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 5998–6008.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[41] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Jul. 2017.

[42] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. New York, NY, USA: Springer, 2011, pp. 185–212.

[43] D. Kinderlehrer and G. Stampacchia, *An Introduction to Variational Inequalities and Their Applications*. Philadelphia, PA, USA: SIAM, 2000.

[44] J. Sherman and W. J. Morrison, "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix," *Ann. Math. Statist.*, vol. 21, no. 1, pp. 124–127, Jan. 1950.

[45] D. Yang and J. Sun, "BM3D-Net: A convolutional neural network for transform-domain collaborative filtering," *IEEE Signal Process. Lett.*, vol. 25, no. 1, pp. 55–59, Jan. 2018.
[46] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 2, Jun. 2001, pp. 416–423.
[47] W. Bae, J. Yoo, and J. C. Ye, "Beyond deep residual learning for image restoration: Persistent homology-guided manifold simplification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 145–153.
[48] R. Timofte, E. Agustsson, L. Van Gool, M.-H. Yang, and L. Zhang, "NTIRE 2017 challenge on single image super-resolution: Methods and results," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 114–125.
[49] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[50] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. ICLR*, 2018. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7
[51] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Proc. NeurIPS*, vol. 29, 2016, pp. 1019–1027.
[52] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proc. NeurIPS*, vol. 29, 2016, pp. 901–909.
[53] S. Roth and M. J. Black, "Fields of experts," *Int. J. Comput. Vis.*, vol. 82, no. 2, p. 205, 2009.
[54] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5197–5206.
[55] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "SwinIR: Image restoration using swin transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 1833–1844.
[56] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.

Qi Chen received the B.S. degree from the School of Mathematical Sciences, Peking University, in 2017, where she is currently pursuing the Ph.D. degree with the School of Mathematical Science. Her main research interests include equilibrium models and their applications in graph neural networks and image processing.

Yifei Wang received the B.S. degree from the School of Mathematical Sciences, Peking University, in 2017, where he is currently pursuing the Ph.D. degree with the School of Mathematical Sciences. His research interests include unsupervised learning, robust learning, and graph learning.

Zhengyang Geng is currently pursuing the Ph.D. degree in computer science with Carnegie Mellon University. From 2018 to 2021, he was a Research Assistant with the School of Intelligence Science and Technology, Peking University. His research interests include machine learning, optimization, and dynamic systems.

Yisen Wang received the Ph.D. degree from Tsinghua University in 2018. He is currently an Assistant Professor with Peking University. His research interests include machine learning and deep learning, such as adversarial learning, graph learning, and weakly/self-supervised learning.

Jiansheng Yang received the B.S., M.S., and Ph.D. degrees from Peking University, Beijing, China, in 1988, 1991, and 1994, respectively. Since his graduation, he has been a Faculty Member with Peking University, where he is currently a Professor of mathematics. His research interests include wavelet analysis, image reconstruction, and computer algorithms.

Zhouchen Lin (Fellow, IEEE) received the Ph.D. degree in applied mathematics from Peking University in 2000. He is currently a Boya Special Professor with the National Key Laboratory of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University. He has published over 280 articles and 5 monographs, collecting more than 28,000 Google Scholar citations. His research interests include machine learning and numerical optimization. He is a fellow of IAPR and CSIG.