# Training Implicit Networks for Image Deblurring using Jacobian-Free Backpropagation

Linghai Liu [†], Shuaicheng Tong [‡], and Lisa Zhao [§]

*Project advisor: Samy Wu Fung* [¶]

**Abstract.** Recent efforts in applying implicit networks to solve inverse problems in imaging have achieved competitive or even superior results when compared to feedforward networks. These implicit networks require only constant memory during backpropagation, regardless of the number of layers. However, they are not necessarily easy to train because they require backpropagating through a fixed point, which leads to computationally expensive gradient calculations. In particular, this process requires solving a large linear system whose size is determined by the number of features in the fixed point iteration. To circumvent such calculations, Jacobian-free Backpropagation (JFB) was recently proposed. This paper explores its application in image deblurring problems. Our results show that JFB is comparable to the classical TV method, Plug-n-Play, and Deep Equilibrium at a reduced computational cost.

**1. Introduction.** Inverse problems consist of recovering a signal, such as an image or a parameter of a partial differential equation (PDE), from noisy measurements, where direct observation of the signal is not possible. As an effort to solve these problems, deep learning techniques have been utilized to acquire high-quality medical images like magnetic resonance imaging (MRI) and computed tomography (CT) [1, 2, 3].

Conventional deep learning approaches for solving inverse problems use deep unrolling [4, 5, 6, 7, 8, 9, 10], which utilizes a fixed number of iterations usually chosen heuristically. A deep network is "unfolded" into a wider and shallower network, where each layer is split into multiple sub-layers. While this method allows the network to learn complex patterns in the input data, it suffers from overfitting and the well-known vanishing gradient problem [11], not to mention the lack of flexibility compared to other network structures [12, 13]. Moreover, they are challenging to train due to memory constraints in backpropagation. Another line of work, feed-forward denoising convolutional neural networks (CNNs) [14, 15, 16, 17], use deep convolutional neural networks to learn the residuals between the ground truth images and noisy observations instead of directly reconstructing the clean underlying images. These end-to-end models are not trained for a particular forward model, so they may require large amounts of labeled data for training [12, 13].

Recently, deep equilibrium models (DEQs) were proposed [18, 19, 20, 21, 22, 23, 24]. DEQs use implicit networks with weight-tying, input-injected layers that propel the dynamics of latent space representation by sharing the input across layers. Training involves backpropagating through a fixed point of the layer using implicit differentiation, where the number of layers can be deemed infinite. This feature allows implicit networks to save memory significantly since there is no need to save any intermediate values on the backpropagation graph. Despite

[†]Department of Applied Mathematics, Brown University (linghai_liu@brown.edu)
[‡]Department of Mathematics, University of California, Los Angeles (allentong24@ucla.edu)
[§]Department of Statistics, University of California, Berkeley (lisazhao@berkeley.edu)
[¶]Department of Applied Mathematics and Statistics, Colorado School of Mines

yielding fixed memory costs and matching performances of other state-of-the-art (SOTA) models, DEQs are still very expensive to train because backpropagation requires solving a Jacobian-based linear system at every gradient evaluation [13]. To this end, a Jacobian-Free Backpropagation (JFB) approach was recently introduced to avoid solving the linear system during training [25].

The theory of JFB allows us to replace the Jacobian matrix with the identity under certain conditions. JFB not only maintains a fixed memory cost but also avoids the substantial computational cost of Jacobian-based methods while ensuring a descent direction [25]. It has performed well in image classification tasks [25], computational tomography [26, 27], traffic routing [28], and finding the shortest paths [29]. A variation of JFB, where the inverse Jacobian was approximated as a perturbed identity matrix, also proved to be successful in image classification [24]. When the norm between the approximation and the true inverse grows beyond a threshold, this method then falls back to JFB. In this paper, we investigate the effectiveness of JFB in training implicit networks for inverse problems arising from image deblurring [1].

## 2. Related Works.

**2.1. Deep Unrolling for Inverse Problems.** Deep unrolling methods [4, 5, 6, 7, 8, 9, 10] for inverse problems involve a fixed number of layers where each layer resembles the iteration of an optimization algorithm. These networks can thus be interpreted as an algorithm to solve an optimization problem with a fixed number of iterations and a regularizer parametrized to adaptively regularize the training process that minimizes the loss of the estimate at each iteration. Deep unrolling has achieved successful results in other inverse problems in imaging, such as low-dose CT [7], light-field photography [10], blind image deblurring [30], and emission tomography [31]. This method involves a finite number of iterations $K$ that is fixed. From prior numerical experiments [12, 23], $K$ is usually a relatively small number chosen as a result of fine-tuning and computational limitations in time and space for both training and inference.

**2.2. Implicit Networks.** Deep equilibrium models (DEQs), a type of implicit networks, were proposed [12, 18, 19, 20, 21, 22, 32]. DEQ requires less memory because it is designed to have only one layer of actual weights (weight-tied) and the original input is fed into each of the identical layers (input-injected). It solves the fixed-point problem and uses implicit differentiation to calculate the gradient for backpropagation. On the other hand, SOTA deep feed-forward networks such as deep unrolling have memory issues since they store intermediate values while iterating through each network layer.

## 3. Mathematical Background.

**3.1. Problem Setup.** We have $N$ noisy, blurred images $\{d_i\}_{i=1}^N \subseteq \mathbb{R}^n$ referred to as *measurements*. The corresponding original images are denoted as $\{x_i\}_{i=1}^N \subseteq \mathbb{R}^n$. We use the model:

$$d = \mathcal{A}x + \boldsymbol{\varepsilon}, \tag{3.1}$$

[1] Access the GitHub repository at https://github.com/lliu58b/Jacobian-free-Backprop-Implicit-Networks

where $x, d \in \mathbb{R}^n$, the forward operator $\mathcal{A} : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a mapping from signal space of original images to measurement space, and $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ is a noise term that models measurement errors.

**3.2. Traditional Optimization for Deblurring.** A natural idea is to apply $\mathcal{A}^{-1}$ to (3.1) and obtain $x^* = \mathcal{A}^{-1}(d - \varepsilon)$ when $\mathcal{A}$ is invertible, which is the case in denoising ($\mathcal{A} = I$) and deblurring ($\mathcal{A}$ is the Toeplitz matrix of a convolution operator). However, doing this can amplify the noise and result in really poor reconstructions when $\mathcal{A}$ is ill-conditioned. Therefore, we estimate the true image $x^*$ by formulating a regularized optimization problem that minimizes the difference between the reconstructed image and the observed image:

$$x^* = \underset{x \in \mathbb{R}^n}{\arg\min} \ \frac{1}{2}||\mathcal{A}x - d||_2^2 + \lambda R(x), \tag{3.2}$$

where $\lambda > 0$ is a tunable parameter and $R(x)$ is a regularizer chosen based on common practice, such as Total Variation [33], or potentially learned from given data [34, 35, 36, 37].

We can solve (3.2) by applying the gradient descent algorithm, and the iteration is

$$x^{k+1} = x^k - \eta \left( \nabla_x ||\mathcal{A}x^k - d||_2^2 + \lambda \nabla_x R(x^k) \right), \quad k = 0, 1, \ldots, K-1 \tag{3.3}$$

where $K$ is the number of steps of gradient descent, $\eta > 0$ is the step size, and $x^0$ is an initial guess.

**3.3. Anderson Acceleration.** To find $x$ such that $x = f(x)$ for a known function $f$, one method is to use fixed-point iteration, where we perform $x^{k+1} = f(x^k)$ for $k \in \mathbb{N}$ starting from an initial guess $x^0$. However, fixed-point iteration tends to converge slowly or not at all. On the other hand, Anderson acceleration [38, 39] serves as an alternative method to find a fixed point. We first translate the problem into finding the root of $g(x) = f(x) - x$. The idea is to assign a linear combination of the past to the current iteration. Given an initial guess $x^0$ and a parameter $m$ for the maximum length of the past considered, Andersen acceleration [38, 39] does the following:

**Algorithm 3.1** Anderson acceleration

**for** $k = 1, 2, \cdots$ **do**

Set $m_k = \min(m, k)$

Set $G_k = [g(x^{k-m_k}) \ \cdots \ g(x^k)]$

Find $\alpha^k \in \mathbb{R}^{m_k+1}$ that solves

$$\min_{\alpha \in \mathbb{R}^{m_k+1}} ||G_k \alpha||_2^2 \quad \text{s.t. } \alpha_0 + \ldots + \alpha_{m_k} = 1$$

Update: $x^{k+1} = \sum_{i=0}^{m_k} \alpha_i^k f(x^{k-m_k+i})$

**end for**

**3.4. Implicit Networks and Challenges.** Implicit networks [40] are newly proposed models that are capable of representing a wide range of feedforward models. The idea is to find a

fixed point for their weight-tied layers, modeled as a non-linear function $T(\cdot)$, and map it to the inference space. We formulate our implicit network $\mathcal{N}_\Theta(\cdot)$ as follows:

$$\begin{aligned} &\text{[Equilibrium equation]} \quad T_\Theta(x) = x, \\ &\text{[Prediction equation]} \quad \mathcal{N}_\Theta(d) = x^*, \end{aligned} \tag{3.4}$$

where $\Theta$ denotes the trainable parameters, and

$$T_\Theta(x) = x - \eta\left(\nabla_x ||Ax - d||_2^2 + S_\Theta(x)\right) \tag{3.5}$$

with $S_\Theta(\cdot)$ being a trainable network containing all the weights of $\mathcal{N}_\Theta(\cdot)$. We then perform iterations to find the fixed point $x^*(d)$ of $T_\Theta(\cdot)$ in the [Equilibrium equation] in (3.4):

$$x^{k+1} = T_\Theta(x^k) = x^k - \eta\left(\nabla_x ||\mathcal{A}x^k - d||_2^2 + S_\Theta(x^k)\right), \quad k = 0, 1, \ldots, K-1 \tag{3.6}$$

where $\eta > 0$ is the step size, $x^0$ is an initial "guess" that depends on the input $d$, and $K$ is the number of iterations (layers) in our neural network $\mathcal{N}_\Theta(\cdot)$. Note that the [Equilibrium equation] is satisfied as $K \to \infty$ when $T_\Theta(\cdot)$ is a contraction. The output of the network is $\mathcal{N}_\Theta(d) := x^*$ given by the [Prediction equation], which depends on $d$ since $x^*$ is the fixed point for the [Equilibrium equation], where $T_\Theta(x)$ involves $d$, as in Eq. 3.6. This iterative scheme is called `DE-GRAD` in [12]. We can observe that (3.6) and (3.3) have the same form, only differing in that we replace the gradient of a chosen regularizer $\lambda\nabla_x R$ with a trainable network $S_\Theta$. Convergence of the iterative scheme (3.6) is guaranteed under certain assumptions proven in [12, Theorem 1]. For completeness, we restate it here.

Theorem 3.1 (Convergence of `DE-GRAD`). *Assume that $S_\Theta - I$ is $\epsilon-$Lipschitz, and let $L = \lambda_{\max}(\mathcal{A}^T\mathcal{A})$ and $\mu = \lambda_{\min}(\mathcal{A}^T\mathcal{A})$, where $\lambda_{\max}(\cdot)$ and $\lambda_{\min}(\cdot)$ denote the maximum and minimum eigenvalues, respectively. If the step-size parameter $\eta > 0$ is such that $\eta < 1/(L+1)$, then the `DE-GRAD` iterative map $T_\Theta(x)$ in 3.5 satisfies*

$$||T_\Theta(x; d) - T_\Theta(x'; d)|| \leq \underbrace{(1 - \eta(1+\mu) + \eta\epsilon)}_{:=\gamma} ||x - x'|| \tag{3.7}$$

*for all $x, x' \in \mathbb{R}^n$. If $\epsilon < 1 + \mu$, then $\gamma < 1$ and the iterates of `DE-GRAD` converge.*

To update the trainable parameters $\Theta$, we set up our objective as

$$\min_\Theta \mathbb{E}_{(x,d)\sim\mathcal{D}}\left[\ell(x, \mathcal{N}_\Theta(d))\right] \tag{3.8}$$

and then use the gradient descent scheme after setting up a loss function $\ell$ (in our numerical experiments, this is chosen to be mean squared error $\frac{1}{n}||x - \mathcal{N}_\Theta(d)||^2$). When backpropagating our implicit network, the hard part is to calculate $\frac{dx^*}{d\Theta}$, which can be derived by performing implicit differentiation on the [Equilibrium equation] in (3.4):

$$\frac{dx^*}{d\Theta} = \frac{\partial T_\Theta(x^*)}{\partial x^*}\frac{dx^*}{d\Theta} + \frac{\partial T_\Theta(x^*)}{\partial \Theta} \overset{\text{rearrange terms}}{\Longrightarrow} \left(I - \frac{\partial T_\Theta(x^*)}{\partial x^*}\right)\frac{dx^*}{d\Theta} = \frac{\partial T_\Theta(x^*)}{\partial \Theta} \tag{3.9}$$

Then, we can substitute this into the update rule of gradient descent:

$$\Theta \leftarrow \Theta - \alpha \frac{d\ell}{dx^*} \mathcal{J}^{-1} \frac{\partial T_\Theta(x^*)}{\partial \Theta}, \tag{3.10}$$

where $\alpha > 0$ is the learning rate and $\mathcal{J} = (I - \frac{dT_\Theta(x^*)}{dx^*})$ is the Jacobian matrix. This update rule is costly due to the need to invert $\mathcal{J}$, which motivates the search for other ways to speed up the backpropagation process.

**4. Proposed Methodology.** We propose using JFB [25], a recently-introduced algorithm that updates the trainable parameters $\Theta$ at a lower computational cost. The idea is to circumvent the Jacobian calculation in (3.10) by replacing $\mathcal{J}$ with the identity $I$, leading to an approximation of the true gradient:

$$p_\Theta = \frac{d\ell}{dx^*} \frac{\partial T_\Theta(x^*)}{\partial \Theta} \tag{4.1}$$

which is still a descent direction for the loss function $\ell$ with more constraints on $T_\Theta$ [25]. Approximating $\mathcal{J}$ with $I$ is equivalent to taking the first term of the Neumann series

$$\left( I - \frac{\partial T_\Theta(x^*)}{\partial x^*} \right)^{-1} = \sum_{k=0}^{\infty} \left( \frac{\partial T_\Theta(x^*)}{\partial x^*} \right)^k. \tag{4.2}$$

Some researchers adopt the first several terms for finer approximations while training Jacobian-based implicit networks [32]. Note the difference here compared to implicit networks is that we are inverting the identity matrix rather than the Jacobian $\mathcal{J}$ in (3.10).

Algorithm 4.1 describes our proposed algorithm, DE-GRAD with JFB. We start by initializing a neural network $S_\Theta$ whose details are specified in Network Architecture in Section 5. Hence, we can build $T_\Theta$ and $\mathcal{N}_\Theta$ according to (3.5) and [Prediction equation] in (3.4), respectively. Given a pair of measurement and truth $(x, d)$, we use Anderson acceleration [38, 39] to facilitate the process of finding fixed points for the mapping $T_\Theta(\cdot)$ while keeping `torch.no_grad()`, like in other works [12, 21, 23, 25]. After finding the root $x^*$, we resume the gradient tape and output $\mathcal{N}_\Theta(d) = T_\Theta(x^*) = x^*$ as an input of loss $\ell$, which is implemented as the mean squared error in this paper. Then, we use PyTorch to calculate $\frac{d\ell}{dx^*} \frac{\partial T_\Theta(x^*)}{\partial \Theta} = \frac{d\ell}{dx^*} \frac{\partial x^*}{\partial \Theta}$ and update the trainable parameters $\Theta$.

The last step above is $\mathcal{O}(n^2)$ because we fixed the dimension of parameters once training starts. Even though JFB is not always performing the steepest descent, the JFB step is much less costly to calculate, which lowers its overall cost while ensuring a descent direction. In contrast, implementing other models with implicit differentiation requires multiplying by $\frac{d\ell}{dx^*}$ and $\frac{\partial T_\Theta(x^*)}{\partial \Theta}$, which is also $\mathcal{O}(n^2)$, to the left and right of the inverse $\mathcal{J}^{-1}$ [32]. The complexity of these update rules is augmented mainly by inverting the Jacobian, which is hard to build explicitly and requires solving a large linear system.

**5. Experimental results.** We mimic the format in [12] while implementing our JFB approach. That is, we perform our experiments on the same dataset and use the same quality measures for image reconstruction.

**Algorithm 4.1** DE-GRAD with JFB

**Require:** Implicit network $N_\Theta(\cdot)$ with weight-tying layers $T_\Theta(\cdot)$ as in (3.5).
Set learning rate $\alpha > 0$.
**for** measurement-truth pair $(d, x)$ in training set **do**
Find fixed point of $T_\Theta$ with Anderson acceleration while `torch.no_grad()`.
Output: $\mathcal{N}_\Theta(d) = T_\Theta(x^*)$.
Calculate loss: $\ell(x^*, x) = ||x^* - x||_2^2$.
Update: $\Theta \leftarrow \Theta - \alpha \frac{d\ell}{dx^*} \frac{\partial T_\Theta(x^*)}{\partial \Theta}$.
**end for**

### 5.1. Experiment Setup.

- **Data**: We use a subset of size 10,000 of the CelebA dataset [41], which contains around 200,000 centered human faces with annotations. Among the subset of 10,000, 8,000 images are used for training and the rest are left for testing purposes.
- **Preprocessing**: Each image is resized to $128 \times 128$ pixels with 3 channels (RGB) and normalized to the range $[0, 1]$ with mean $\frac{1}{2}$ for each channel. The blurred images are generated using Gaussian blurring kernels of size $5 \times 5$ with variance 1, i.e., $\mathcal{A}$ is a Gaussian blur operator. The measurements are then crafted by adding white Gaussian noise with standard deviation $\sigma = 10^{-2}$ to the blurred images.
- **Network Architecture**: We use a convolutional neural network (CNN) structure with 17 layers. Except for the first and last CNN layer, each intermediate layer consists of $3 \times 3$ convolutional kernels that mapping from 64 channels to 64 channels, followed by batch normalization and element-wise ReLU activation function. For the first layer, it is a CNN layer with $3 \times 3$ kernels and raises the RGB image into 64 channels. The last layer is also a CNN, mapping its input from 64 channels to 3 channels. To ensure that we have a contractive mapping $T_\Theta$, we scale the weights in each layer with a constant $\gamma < 1$ using spectral normalization [42].
- **Training**: With the aforementioned data and preprocessing specifics, the training strictly follows Algorithm 4.1, where $T_\Theta(\cdot)$ is the same as defined in (3.6). As part of $T_\Theta$, the trainable network $S_\Theta(\cdot)$ is initialized as the above "Network Architecture" and is also pre-trained as practiced in [12] to observe an improvement in reconstruction. We let step size $\eta = 10^{-3}$. The stopping criterion is either the maximal change in the norm of the subsequent iteration is too small or we exceed the maximum number of iterations.
- **Visualization**: We first visualize the average training and testing loss per image over the number of epochs in Fig. 1, running on a sample of 2000 images, with an 80-20 training-testing split.

  We see that as the number of epochs increases, the training loss decreases, while the testing loss remains relatively high. For this proof of concept, we are using only a subset of our dataset, so it is hard for the model to generalize well to the testing images. Also, the JFB algorithm only promises a descent direction for the loss function, rather than a technique that learns the structure of the data distribution. We then visualize examples from the `DE-GRAD` model trained using JFB in Table 1, using the

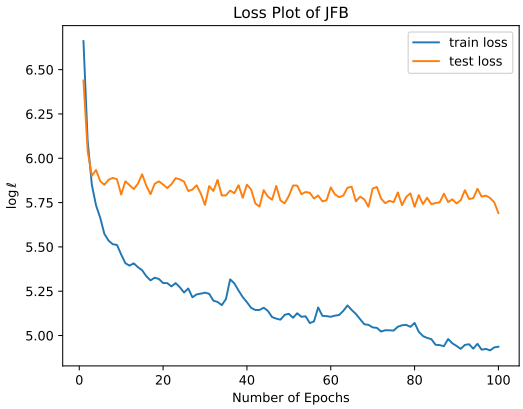



**Figure 1.** *Training and Testing Losses of the DE-GRAD Model with JFB. Training losses are calculated on the training set of size 8,000. Testing losses are calculated on the test set of size 2,000.*

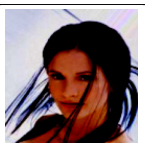
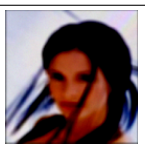
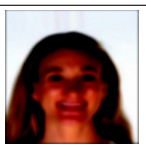
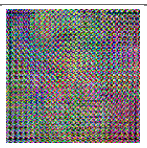
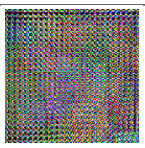
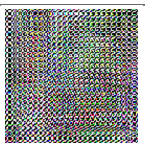
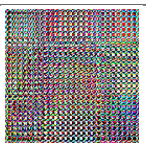
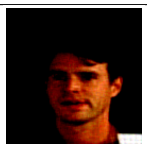
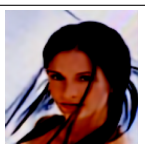
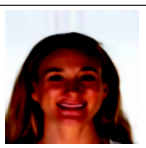
| | | | | |
|---|---|---|---|---|
| Ground Truth |  |  |  |  |
| Noisy Blurred Image |  |  |  |  |
| Direct Inverse |  |  |  |  |
| Gradient Descent |  |  |  |  |
| DE-GRAD with JFB |  |  |  |  |

**Table 1**
*Visualization of Several Images across Different Models*

same four images for each row. Ground truths are the images in CelebA dataset resized to 128 × 128 pixels. Noisy blurred images are generated by a 5 × 5 Gaussian kernel mentioned above. Direct inverse images are the results of applying the inverse of $\mathcal{A}$ on blurred images, which is defined in (3.1). Gradient descent images are obtained by

applying gradient descent using total variation. JFB images are obtained using Noisy blurred images as inputs with JFB-trained weights for `DE-GRAD`.

**5.2. Comparison of Quality.** We compare our results obtained from JFB with total variation (TV) [33], Plug-n-Play [43], and Deep Equilibrium (DEQ) [12]. TV is a classical method for image denoising that aims to preserve the sharp edges of the image [33]. Plug-n-Play is a framework that uses denoising algorithms as priors for model-based image reconstruction [12]. The metrics we use to assess the quality of reconstructed images are peak-signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) [44]. The values reported in Table 2 are the average PSNR and SSIM values calculated on the test set.

To compare against TV, Plug-n-Play, and Deep Equilibrium, we followed the same set-up in [12]: blurry images are simulated using $9 \times 9$ pixel Gaussian blur kernel with variance 5 and additive white Gaussian noise with variance $\sigma = 0.01$. Although we have not achieved better results than DEQs, which find the true gradient in a complicated manner in (3.10), we currently observe results that are competitive with other techniques. TV

| | Total Variation | Plug-n-Play | Deep Equilibrium | JFB (Ours) |
|---|---|---|---|---|
| PSNR | 26.70 | 29.77 | 32.43 | 26.88 |
| SSIM | 0.90 | 0.88 | 0.94 | 0.91 |

**Table 2**

*Comparison of PSNR and SSIM values on the test set across models*

**5.3. Comparison of Time and Complexity.** We compare the complexity per gradient/step computation using JFB and Jacobian-based backpropagation on the CelebA dataset. Here, $n = 128 \times 128 \times 3 = 49152$ after preprocessing. Importantly, we note that JFB requires a Jacobian matrix-vector product per sample, leading to complexity $\mathcal{O}(n^2)$. Jacobian-based backpropagation, however, also requires the inverse of $\mathcal{J}$, i.e., solving a linear system as shown in (3.10).

Naïvely using PyTorch for gradient tracking and taking the inverse of the Jacobian during our experiments depletes all possible RAM (more than 32 Gigabytes). Hence, we use the conjugate gradient method to solve the linear system in (3.10). The idea is to let $w = \frac{d\ell}{dx^*}\mathcal{J}^{-1}$, and solve $w\mathcal{J}\mathcal{J}^T = \frac{d\ell}{dx^*}\mathcal{J}^T$ instead of $w\mathcal{J} = \frac{d\ell}{dx^*}$.

In our experiments, we fix a batch of 16 images in the CelebA dataset, run both the Jacobian-based update and JFB for 20 repetitions, and then record the average time needed in seconds for each parameter update. The dimensions of the images are square with 3 channels, so the lengths vary from 16 to 128 with an increment of 16.

Figure 2 shows that while maintaining comparable image reconstruction quality measured by PSNR and SSIM, JFB algorithm is faster and easier to implement with auto-differentiation libraries such as Tensorflow or PyTorch.

**6. Conclusions.** In this paper, we explored JFB for implicit networks with applications in image deblurring. Our approach recovers images rather effectively across the test set. Moreover, JFB is competitive with other state-of-the-art methods whose hand-crafted parameters have been fine tuned across all stages. We also demonstrated the advantage of JFB in terms of

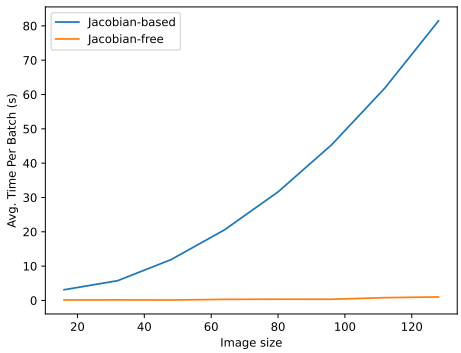



**Figure 2.** *Computation Time per Batch for Jacobian-based Method v.s. DE-GRAD with JFB*

its computational complexity and ease of implementation in practice. Future work involves application to other inverse problems like denoising [45, 33], geophysical imaging [46, 47, 48, 49], and more.

**7. Acknowledgements.** The authors sincerely thank the guidance of our mentor, Dr. Samy Wu Fung, and other mentors at Emory University for the opportunity. We also thank Emory University for providing the GPUs that made our numerical experiments possible. This work was partially funded by National Science Foundation awards DMS-2309810 and DMS-2051019.

## REFERENCES

[1] Gushan Zeng, Yi Guo, Jiaying Zhan, Zi Wang, Zongying Lai, Xiaofeng Du, Xiaobo Qu, and Di Guo. A review on deep learning MRI reconstruction without fully sampled k-space. *BMC Medical Imaging*, 21(1):195, 2021.

[2] Emmanuel Ahishakiye, Martin Bastiaan Van Gijzen, Julius Tumwiine, Ruth Wario, and Johnes Obungoloch. A survey on deep learning in medical image reconstruction. *Intelligent Medicine*, 1(03):118–127, 2021.

[3] Muhammad Yaqub, Feng Jinchao, Kaleem Arshid, Shahzad Ahmed, Wenqian Zhang, Muhammad Zubair Nawaz, and Tariq Mahmood. Deep learning-based image reconstruction for different medical imaging modalities. *Computational and Mathematical Methods in Medicine*, 2022, 2022.

[4] Hemant K Aggarwal, Merry P Mani, and Mathews Jacob. MoDL: Model-based deep learning architecture for inverse problems. *IEEE Transactions on medical imaging*, 38(2):394–405, 2018.

[5] Hu Chen, Yi Zhang, Yunjin Chen, Junfeng Zhang, Weihua Zhang, Huaiqiang Sun, Yang Lv, Peixi Liao, Jiliu Zhou, and Ge Wang. LEARN: Learned experts' assessment-based reconstruction network for sparse-data CT. *IEEE transactions on medical imaging*, 37(6):1333–1347, 2018.

[6] Vishal Monga, Yuelong Li, and Yonina C. Eldar. Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing. *IEEE Signal Processing Magazine*, 38:18–44, 2021.

[7] Dufan Wu, Kyungsang Kim, and Quanzheng Li. Computationally efficient deep neural network for computed tomography image reconstruction. *Medical physics*, 46(11):4763–4776, 2019.

[8] Risheng Liu, Shichao Cheng, Long Ma, Xin Fan, and Zhongxuan Luo. Deep proximal unrolling: Algorithmic framework, convergence analysis and applications. *IEEE Transactions on Image Processing*, 28(10):5013–5026, 2019.

[9] Dong Liang, Jing Cheng, Ziwen Ke, and Leslie Ying. Deep magnetic resonance image reconstruction:

Inverse problems meet neural networks. *IEEE Signal Processing Magazine*, 37(1):141–151, 2020.

[10] Il Yong Chun, Zhengyu Huang, Hongki Lim, and Jeff Fessler. Momentum-Net: Fast and convergent iterative neural network for inverse problems. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[11] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[12] Davis Gilton, Gregory Ongie, and Rebecca Willett. Deep equilibrium architectures for inverse problems in imaging. *IEEE Transactions on Computational Imaging*, 7:1123–1133, 2021.

[13] Gregory Ongie, Ajil Jalal, Christopher A Metzler, Richard G Baraniuk, Alexandros G Dimakis, and Rebecca Willett. Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56, 2020.

[14] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017.

[15] Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep CNN denoiser prior for image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3929–3938, 2017.

[16] Kai Zhang, Wangmeng Zuo, and Lei Zhang. FFDNet: Toward a fast and flexible solution for CNN-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[17] Chunwei Tian, Yong Xu, Lunke Fei, Junqian Wang, Jie Wen, and Nan Luo. Enhanced CNN for image denoising. *CAAI Transactions on Intelligence Technology*, 4(1):17–23, 2019.

[18] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019.

[19] Shaojie Bai, Vladlen Koltun, and J Zico Kolter. Multiscale deep equilibrium models. *Advances in Neural Information Processing Systems*, 33:5238–5250, 2020.

[20] Ezra Winston and J Zico Kolter. Monotone operator equilibrium networks. *Advances in neural information processing systems*, 33:10718–10728, 2020.

[21] Swaminathan Gurumurthy, Shaojie Bai, Zachary Manchester, and J Zico Kolter. Joint inference and input optimization in equilibrium networks. *Advances in Neural Information Processing Systems*, 34:16818–16832, 2021.

[22] Kenji Kawaguchi. On the Theory of Implicit Deep Learning: Global Convergence with Implicit Layers. In *International Conference on Learning Representations (ICLR)*, 2021.

[23] Zaccharie Ramzi, Pierre Ablin, Gabriel Peyré, and Thomas Moreau. Test like you Train in Implicit Deep Learning, 2023.

[24] Zaccharie Ramzi, Florian Mannel, Shaojie Bai, Jean-Luc Starck, Philippe Ciuciu, and Thomas Moreau. SHINE: SHaring the INverse Estimate from the forward pass for bi-level optimization and implicit models, 2023.

[25] Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel Mckenzie, Stanley Osher, and Wotao Yin. JFB: Jacobian-Free Backpropagation for Implicit Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):6648–6656, Jun. 2022.

[26] Howard Heaton, Samy Wu Fung, Aviv Gibali, and Wotao Yin. Feasibility-based fixed point networks. *Fixed Point Theory and Algorithms for Sciences and Engineering*, 2021(1):1–19, 2021.

[27] Howard Heaton and Samy Wu Fung. Explainable AI via learning to optimize. *Scientific Reports*, 13(1):10103, 2023.

[28] D McKenzie, H Heaton, Q Li, S Wu Fung, S Osher, and W Yin. Three-Operator Splitting for Learning to Predict Equilibria in Convex Games. *SIAM Journal on Mathematics of Data Science*, 6(3):627–648, 2024.

[29] Daniel McKenzie, Samy Wu Fung, and Howard Heaton. Faster predict-and-optimize with three-operator splitting. *arXiv preprint arXiv:2301.13395*, 2023.

[30] Yuelong Li, Mohammad Tofighi, Junyi Geng, Vishal Monga, and Yonina C Eldar. Deep algorithm unrolling for blind image deblurring. *arXiv preprint arXiv:1902.03493*, 2019.

[31] Abolfazl Mehranian and Andrew J Reader. Model-based deep learning PET image reconstruction using forward–backward splitting expectation–maximization. *IEEE transactions on radiation and plasma*

*medical sciences*, 5(1):54–64, 2020.

[32] Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. *Advances in Neural Information Processing Systems*, 34:24247–24260, 2021.

[33] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[34] Giovanni S Alberti, Ernesto De Vito, Matti Lassas, Luca Ratti, and Matteo Santacesaria. Learning the optimal tikhonov regularizer for inverse problems. *Advances in Neural Information Processing Systems*, 34:25205–25216, 2021.

[35] Eldad Haber and Luis Tenorio. Learning regularization functionals—a supervised training approach. *Inverse Problems*, 19(3):611, 2003.

[36] Howard Heaton, Samy Wu Fung, Alex Tong Lin, Stanley Osher, and Wotao Yin. Wasserstein-based projections with applications to inverse problems. *SIAM Journal on Mathematics of Data Science*, 4(2):581–603, 2022.

[37] Babak Maboudi Afkham, Julianne Chung, and Matthias Chung. Learning regularization parameters of inverse problems via deep neural networks. *Inverse Problems*, 37(10):105017, 2021.

[38] Homer F Walker and Peng Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

[39] Zico Kolter, David Duvenaud, and Matt Johnson. Chapter 4: Deep Equilibrium Models. *NeurIPS 2020 tutorial: Deep Implicit Layers - Neural ODEs, Deep Equilibirum Models, and Beyond*, 2020.

[40] Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Tsai. Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3):930–958, 2021.

[41] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[42] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks, 2018.

[43] Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. Plug-and-play priors for model based reconstruction. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 945–948. IEEE, 2013.

[44] Alain Hore and Djemel Ziou. Image quality metrics: PSNR vs. SSIM. In *2010 20th international conference on pattern recognition*, pages 2366–2369. IEEE, 2010.

[45] Jennifer L Mueller and Samuli Siltanen. *Linear and nonlinear inverse problems with practical applications*. SIAM, 2012.

[46] Eldad Haber. *Computational methods in geophysical electromagnetics*. SIAM, 2014.

[47] Samy Wu Fung and Lars Ruthotto. A multiscale method for model order reduction in PDE parameter estimation. *Journal of Computational and Applied Mathematics*, 350:19–34, 2019.

[48] Samy Wu Fung and Lars Ruthotto. An uncertainty-weighted asynchronous ADMM method for parallel PDE parameter estimation. *SIAM Journal on Scientific Computing*, 41(5):S129–S148, 2019.

[49] Kelvin Kan, Samy Wu Fung, and Lars Ruthotto. PNKH-B: A Projected Newton–Krylov Method for Large-Scale Bound-Constrained Optimization. *SIAM Journal on Scientific Computing*, 43(5):S704–S726, 2021.