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 only require constant memory during backpropagation, regardless of the number of layers. However, they are not necessarily easy to train. Gradient calculations are computationally expensive because they require backpropagating through a fixed point. In particular, this process requires solving a large linear system whose size is determined by the number of features in the fixed point iteration. This paper explores a recently proposed method, Jacobian-free Backpropagation (JFB), a fast and easy-to-implement scheme for backpropagation that circumvents such calculation in the context of image deblurring problems. Our results show that JFB is comparable against fine-tuned optimization schemes, state-of-the-art (SOTA) feedforward networks, and existing implicit networks 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. Deep learning techniques, for instance, have been utilized to acquire high-quality medical images like magnetic resonance imaging (MRI) and computed tomography (CT) [45, 3, 44].

Conventional deep learning approaches for solving inverse problems use deep unrolling [2, 7, 33, 43, 29, 28, 8], 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 [23], not to mention the lack of flexibility compared to other network structures [14, 35]. Moreover, they are challenging to train due to memory constraints during backpropagation. Another line of work, feed-forward denoising convolutional neural networks [46, 47, 48, 39], uses deep convolutional neural networks (CNNs) to learn the residuals between the clean 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 [14, 35].

Recently, deep equilibrium models (DEQs) were proposed [5, 6, 42, 16, 26, 36, 37]. DEQs use implicit networks with weight-tying, input-injected layers that propel the dynamics of latent space representation. 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 costs significantly since there is no need to

[†]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

save any intermediate values on the backpropagation graph. Despite 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 the computation of a Jacobian-based linear system at every gradient evaluation [35, 13]. Further, as over-parametrized networks, running more iterations at test time cannot increase their performances. [36] Therefore, a Jacobian-Free Backpropagation (JFB) approach was recently introduced to avoid solving the linear system during training [10].

The theory of JFB allowed us to replace the Jacobian matrix with the identity under certain conditions. JFB not only maintained a fixed memory cost but also avoided a substantial computational cost while ensuring a descent direction. [10] It has performed well in image classification tasks [10], computational tomography [21, 22, 19], traffic routing [21], and finding the shortest paths [31]. A variation of JFB, where the inverse Jacobian was approximated as a perturbed identity matrix and falls back to JFB when the approximation yields a huge norm compared to the true inversion [37], also proved to be successful in image classification. In this paper, we investigate its effectiveness in training implicit networks for inverse problems arising in image deblurring [1].

2. Mathematical Background.

2.1. Problem Setup. We have N noisy blurred images $\{d_i\}_{i=1}^N \subseteq \mathbb{R}^n$ that we refer to as *measurements*. The underlying *original images*, denoted as $\{x_i\}_{i=1}^N \subseteq \mathbb{R}^n$, are hidden from the experimenter(s). We use the model:

$$d = \mathcal{A}x + \boldsymbol{\varepsilon} \tag{2.1}$$

where the forward operator $\mathcal{A}$ is a mapping from signal space of original images to measurement space and $\boldsymbol{\varepsilon} \in \mathbb{R}^m$ is a noise term that models measurement errors. In this work, we deal with image deblurring, where the forward process is a Gaussian blur. The value of each pixel in a measurement is a weighted average of its neighborhood under a Gaussian kernel with discretized weights of a 2-dimensional Gaussian density, plus noise generated from the hypothesized measurement process.

2.2. Traditional Optimization for Deblurring. A natural idea is to apply $\mathcal{A}^{-1}$ to Eq. 2.1 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). This can amplify the noise and result in really bad 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:

$$x^* = \operatorname*{arg\,min}_{x \in \mathbb{R}^n} \frac{1}{2}||\mathcal{A}x - d||_{L^2}^2 + \lambda R(x) \tag{2.2}$$

where $\lambda > 0$ is a tunable parameter, $R(x)$ is a regularizer chosen based on common practice [15] or potentially learned from given data [4, 18, 20, 1]. We can solve Eq. 2.2 by applying the

[1]Access the GitHub repository at https://github.com/lliu58b/Jacobian-free-Backprop-Implicit-Networks
SWF: make sure this is updated

gradient descent algorithm, with a common choice of $R(x) = \frac{1}{2}||x||^2_{L^2}$. However, we observe in our numerical experiments that results are not ideal (low SSIM values for reconstructed images). To this end, we explore the use of use of implicit networks in this work.

2.3. Implicit Networks and Challenges.

Implicit networks [9] are newly proposed models that also leverage the dataset $\{(d_i, x_i)\}_{i=1}^N$ and are capable of representing a wide range of feedforward models. The idea is to find a fixed point for their weight-tying layers, modeled as a non-linear function $T(\cdot)$, and map it to the inference space. We formulate our implicit network as follows:

$$\begin{aligned} &[\text{Equilibrium equation}] \quad T_\Theta(x) = x \\ &[\text{Prediction equation}] \quad x^*(d) = x \end{aligned} \tag{2.3}$$

The iteration to find the fixed point reads:

$$x_i^{k+1} = x_i^k - \eta \left(\nabla_x ||\mathcal{A}x_i^k - d_i||^2_{L^2} + S_\Theta(x_i^k) \right) := T_\Theta(x_i^k) \tag{2.4}$$

where $\eta > 0$ is step size, K is the number of iterations (layers) in our neural network $\mathcal{N}_\Theta(\cdot)$, and $S_\Theta(\cdot)$ is a trainable network containing all the weights of $\mathcal{N}_\Theta(\cdot)$. Note that the [Equilibrium equation] is satisfied as $K \to \infty$ when $T_\Theta(\cdot)$ is a contraction. The output of the network is $\mathcal{N}_\Theta(d) := x^*(d)$, given by the [Prediction equation]. This iterative scheme is called `DE-GRAD` [14].

To train the weights Θ, we perform implicit differentiation on the equilibrium equation in (2.3)

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

and substitute $\frac{dx^*}{d\Theta}$ into the gradient descent scheme after setting up a loss function ℓ:

$$\Theta \leftarrow \Theta - \alpha \frac{d\ell}{dx^*} \left(I - \frac{dT_\Theta(x^*)}{dx^*} \right)^{-1} \frac{\partial T_\Theta(x^*)}{\partial \Theta} \tag{2.5}$$

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

3. Related Works.

3.1. Deep Unrolling for Inverse Problems.

Deep Unrolling [2, 7, 33, 43, 29, 28, 8] is used to unpack deep neural networks, whose black-box nature hinders their training and interpretation. Intuitively, each iteration is “unfolded” into smaller layers and then concatenated to form a deep network. Therefore, these neural networks can be interpreted as an optimization problem as in Eq. 2.2, with a fixed number of K iterations upon initialization, where the regularizer $R(x)$ can be parametrized to adaptably regularize the training process to minimize the loss of each estimate $\hat{x}^{(k)}$. [14] Deep Unrolling has achieved successful results in other inverse problems in imaging, such as low-dose CT [43], light-field photography [8], blind image deblurring [27], and emission tomography [32]. In our setup, this method corresponds to Eq. 2.4, with $K < +\infty$ being fixed. From prior numerical experiments [14, 36], K is eventually a small, handcrafted number as a result of fine-tuning and limitations in time and space for both training and inference.

3.2. Implicit Networks. Deep Equilibrium Models (DEQs), a type of implicit networks, were proposed [5, 6, 42, 16, 26]. The advantage of DEQ lies in that it requires less memory because it uses a weight-tied, input-injected design, where it only has one layer of actual weights and the original input is fed into each of the identical layers. It solves the fix-point 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.

4. Proposed Methodology.

The convergence criterion of $T_\Theta(\cdot)$ in Eq. 2.4 (`DE-GRAD`) is discussed in more detail in [14], where $S_\Theta - I$ needs to be $\epsilon-$Lipschitz to ensure that $T_\Theta(\cdot)$ is contraction with parameter $\gamma \in [0, 1)$. We propose using Jacobian-free Backpropagation (JFB) [10], a recently-introduced algorithm that updates Θ at a lower computational cost. The idea is to circumvent the Jacobian calculation in (2.5) by replacing $\mathcal{J}_\Theta$ 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 descending direction for the loss function ℓ with more constraints on T_Θ [10]. Note the difference here compared to implicit networks is that we are inverting the identity matrix rather than the Jacobian $\mathcal{J}_\Theta$. As in other works, we used Anderson acceleration [41] to facilitate the process of finding fixed points for the mapping $T_\Theta(\cdot)$ while keeping `torch.no_grad()`. After finding the root x^*, we resume the gradient tape and output $T_\Theta(x^*) = x^*$ as an input of loss ℓ. Then, PyTorch would calculate $\frac{d\ell}{dx^*}\frac{\partial T_\Theta(x^*)}{\partial\Theta} = \frac{d\ell}{dx^*}\frac{\partial x^*}{\partial\Theta}$, which is $\mathcal{O}(n)$ because we fix the dimension of parameters once training starts. Even though JFB is not always performing the steepest descent, its gradient is much less costly to calculate, which makes its overall cost lower while ensuring descending. As in other works, we used Anderson acceleration [41] to facilitate the process of finding fixed points for the mapping $T_\Theta(\cdot)$ while keeping `torch.no_grad()`. After finding the root x^*, we resume the gradient tape and output $T_\Theta(x^*) = x^*$ as an input of loss ℓ. Then, PyTorch would calculate $\frac{d\ell}{dx^*}\frac{\partial T_\Theta(x^*)}{\partial\Theta} = \frac{d\ell}{dx^*}\frac{\partial x^*}{\partial\Theta}$, which is $\mathcal{O}(n)$ because we fix the dimension of parameters once training starts.

JFB is not a gradient computation, but rather a descent direction.

5. Experimental results.

We mimic the format in [14] while implementing our JFB approach. That is, we perform our experiments on the same dataset and use the same quality measures for image reconstruction.

- **Data**: We use a subset of size 10,000 of the CelebA dataset [30], 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 validating and testing purposes.
- **Preprocessing**: Each image is resized into 128×128 pixels with 3 channels (RGB) and normalized into range $[0, 1]$ with mean $\frac{1}{2}$ for each channel. The blurred images

Algorithm 4.1 Learning with JFB

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

are generated using Gaussian blurring kernels of size 5×5 with variance 5. 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 is followed by batch normalization and element-wise ReLU activation function. We also applied spectral normalization to each CNN layer to make sure that the mapping is Lipschitz continuous with Lipschitz constant no greater than 1.
- **Training**: With the aforementioned data and preprocessing specifics, the training strictly follows Algorithm 1, where $T_\Theta(\cdot)$ is the same as defined in Eq. 2.4. As part of T_Θ, the trainable network $S_\Theta(\cdot)$ is pretrained as also practiced in [14] to observe an improvement in reconstruction.
- **Visualization**: We first visualize the average training and validation loss per image over the number of epochs in Fig. 1, running on a sample of 2000 images, with an 80-20 training-validation split.

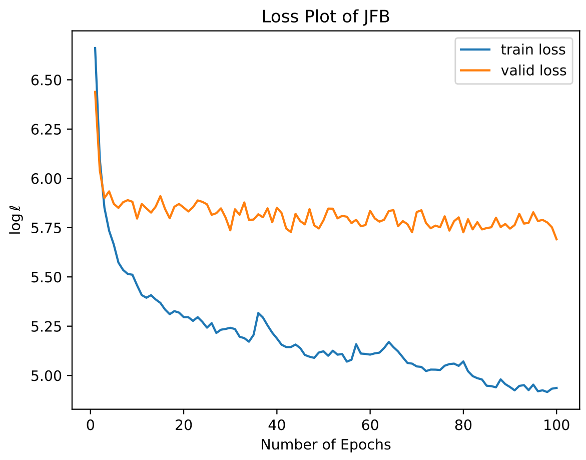


Figure 1. *Training and Validation losses of the DE-GRAD model with JFB*

We see that as the number of epochs increases, the training loss decreases, while the validation 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 validation images. However, the validation loss still shows a decreasing trend. 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. Nevertheless, the comparable PSNR and SSIM values reported in Table 2 were calculated with a JFB model that achieves an MSE of about 100 per image ($\log 100 \approx 4.605$).

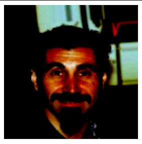
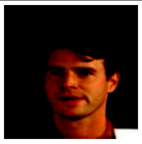
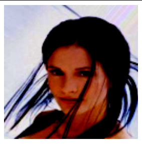
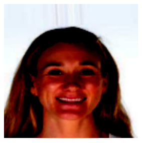
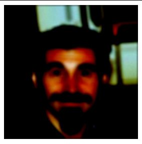
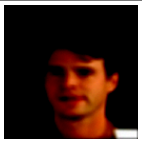
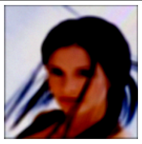
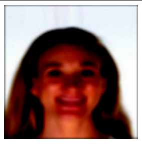
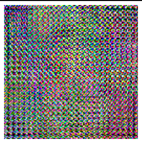
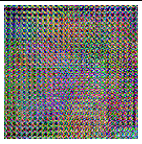
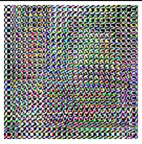
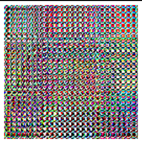
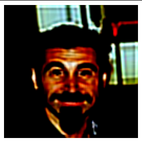
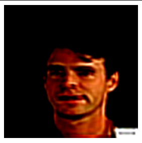
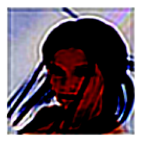
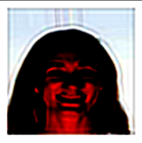
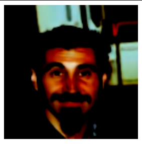
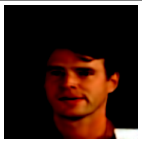
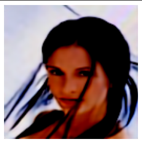
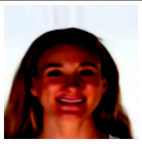
Ground Truth				
Noisy Blurred Image				
Direct Inverse				
Gradient Descent				
JFB				

Table 1
Visualization of JFB

We then visualize examples from the `DE-GRAD` model trained using JFB in Table 1, using the 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 Eq. 2.1. Gradient Descent images are obtained by applying gradient descent using mean squared error (MSE) as the loss function. We enforce early stopping so that the results are not corrupted. JFB images are obtained using Noisy Blurred images as inputs with JFB-trained weights for the `DE-GRAD` model.

- **Comparison of Quality**: We compare our results obtained from JFB with other state-of-the-art methods using total variation (TV), standard deep neural networks, and DEQ. The metrics we use to assess the quality of reconstructed images are peak-signal-to-noise ratio (PSNR, a positive number, best at $+\infty$) and structural similarity index measure (SSIM, also positive, best at 1) [24]. The data in Table 2 are calculated on the testing dataset of size 2000. Although we have not yet achieved results better than DEQs, which finds the true gradient in a complicated manner in Eq. 2.5, we currently observe results that are competitive with other techniques.

	Total Variation	Plug-n-Play [40]	Deep Equilibrium [14]	JFB (Ours)
PSNR	26.79	29.77	32.43	26.88
SSIM	0.86	0.88	0.94	0.91

Table 2
Comparison across Models

- **Comparison of Time and Complexity**: The most important part of our comparison is the speed of computation. While maintaining comparable image reconstruction quality measured by PSNR and SSIM, the JFB algorithm is faster and easier to implement with auto-differentiation libraries such as Tensorflow or PyTorch. Following the update rule as Eq. 2.5, we can use PyTorch to obtain the Jacobian matrix of each entry of $T_\Theta(x^*)$ with respect to x^* and then calculate the inverse of $I - \frac{dT_\Theta(x^*)}{dx^*}$, which is known to be of complexity $\mathcal{O}(n^3)$, where n is the dimension of the Euclidean space that the true underlying images (and measurements) live in. Using the CelebA dataset, after preprocessing, $n = 128 \times 128 \times 3 = 49152$.
  SWF: JFB implementation could be discussed in proposed methodology in Sec. 4.
  When implementing other models with implicit differentiation, in addition to calculating the inverse $(I - \frac{dT_\Theta(x^*)}{dx^*})^{-1}$, we have to multiply it by $\frac{d\ell}{dx^*}$ and $\frac{\partial T_\Theta(x^*)}{\partial \Theta}$ to its left and right, respectively, which is also $\mathcal{O}(n)$ SWF: double check FLOPs for matrix vector multiply. Compared to the calculation for JFB, the only significant time difference is the computation of this inverse matrix. To estimate this difference, we initialize a network and then estimate this computation time as a proof of concept. Naïvely taking the inverse by applying `numpy.linalg.inv()` depletes all possible RAM (more than 32 Gigabytes). Hence, we resort to inverting a 4000 by 4000 matrix, which takes about 2.27 seconds, not to mention the additional work spent by constructing the Jacobian using `torch.autograd.grad()`, which takes in a scalar output and calculates its gradient with respect to another tensor on the gradient tape. SWF: we should discuss this section.

6. Conclusions.

In this paper, we explored Jacobian-free Backpropagation for implicit networks with applications in image deblurring. Our approach recovers images rather effectively across the testing dataset. 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 its computational complexity and easiness for implementation in practice. Future work involves application to other inverse problems like denoising [34, 38], geophysical imaging [17, 11, 12, 25], and more.

7. Acknowledgements.

We 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. run by Lars to

add grant information

REFERENCES

[1] B. M. Afkham, J. Chung, and M. Chung, *Learning regularization parameters of inverse problems via deep neural networks*, Inverse Problems, 37 (2021), p. 105017.
[2] H. K. Aggarwal, M. P. Mani, and M. Jacob, *Modl: Model-based deep learning architecture for inverse problems*, IEEE Transactions on medical imaging, 38 (2018), pp. 394–405.
[3] E. Ahishakiye, M. Bastiaan Van Gijzen, J. Tumwiine, R. Wario, and J. Obungoloch, *A survey on deep learning in medical image reconstruction*, Intelligent Medicine, 1 (2021), pp. 118–127.
[4] G. S. Alberti, E. De Vito, M. Lassas, L. Ratti, and M. Santacesaria, *Learning the optimal tikhonov regularizer for inverse problems*, Advances in Neural Information Processing Systems, 34 (2021), pp. 25205–25216.
[5] S. Bai, J. Z. Kolter, and V. Koltun, *Deep equilibrium models*, Advances in Neural Information Processing Systems, 32 (2019).
[6] S. Bai, V. Koltun, and J. Z. Kolter, *Multiscale deep equilibrium models*, Advances in Neural Information Processing Systems, 33 (2020), pp. 5238–5250.
[7] H. Chen, Y. Zhang, Y. Chen, J. Zhang, W. Zhang, H. Sun, Y. Lv, P. Liao, J. Zhou, and G. Wang, *Learn: Learned experts' assessment-based reconstruction network for sparse-data ct*, IEEE transactions on medical imaging, 37 (2018), pp. 1333–1347.
[8] I. Y. Chun, Z. Huang, H. Lim, and J. Fessler, *Momentum-net: Fast and convergent iterative neural network for inverse problems*, IEEE transactions on pattern analysis and machine intelligence, (2020).
[9] L. El Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Tsai, *Implicit deep learning*, SIAM Journal on Mathematics of Data Science, 3 (2021), pp. 930–958.
[10] S. W. Fung, H. Heaton, Q. Li, D. Mckenzie, S. Osher, and W. Yin, *Jfb: Jacobian-free backpropagation for implicit networks*, Proceedings of the AAAI Conference on Artificial Intelligence, 36 (2022), pp. 6648–6656.
[11] S. W. Fung and L. Ruthotto, *A multiscale method for model order reduction in pde parameter estimation*, Journal of Computational and Applied Mathematics, 350 (2019), pp. 19–34.
[12] S. W. Fung and L. Ruthotto, *An uncertainty-weighted asynchronous admm method for parallel pde parameter estimation*, SIAM Journal on Scientific Computing, 41 (2019), pp. S129–S148.
[13] Z. Geng, X.-Y. Zhang, S. Bai, Y. Wang, and Z. Lin, *On training implicit models*, Advances in Neural Information Processing Systems, 34 (2021), pp. 24247–24260.
[14] D. Gilton, G. Ongie, and R. Willett, *Deep equilibrium architectures for inverse problems in imaging*, IEEE Transactions on Computational Imaging, 7 (2021), pp. 1123–1133.
[15] G. H. Golub, P. C. Hansen, and D. P. O'Leary, *Tikhonov regularization and total least squares*, SIAM journal on matrix analysis and applications, 21 (1999), pp. 185–194.
[16] S. Gurumurthy, S. Bai, Z. Manchester, and J. Z. Kolter, *Joint inference and input optimization in equilibrium networks*, Advances in Neural Information Processing Systems, 34 (2021), pp. 16818–16832.
[17] E. Haber, *Computational methods in geophysical electromagnetics*, SIAM, 2014.
[18] E. Haber and L. Tenorio, *Learning regularization functionals—a supervised training approach*, Inverse Problems, 19 (2003), p. 611.
[19] H. Heaton and S. W. Fung, *Explainable ai via learning to optimize*, Scientific Reports, 13 (2023), p. 10103.
[20] H. Heaton, S. W. Fung, A. T. Lin, S. Osher, and W. Yin, *Wasserstein-based projections with applications to inverse problems*, SIAM Journal on Mathematics of Data Science, 4 (2022), pp. 581–603.
[21] H. Heaton, D. McKenzie, Q. Li, S. W. Fung, S. Osher, and W. Yin, *Learn to predict equilibria via fixed point networks*, arXiv preprint arXiv:2106.00906, (2021).
[22] H. Heaton, S. Wu Fung, A. Gibali, and W. Yin, *Feasibility-based fixed point networks*, Fixed Point Theory and Algorithms for Sciences and Engineering, 2021 (2021), pp. 1–19.
[23] S. Hochreiter, *The vanishing gradient problem during learning recurrent neural nets and problem so-*

lutions, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6 (1998), pp. 107–116.
[24] A. Hore and D. Ziou, *Image quality metrics: Psnr vs. ssim*, in 2010 20th international conference on pattern recognition, IEEE, 2010, pp. 2366–2369.
[25] K. Kan, S. W. Fung, and L. Ruthotto, *Pnkh-b: A projected newton–krylov method for large-scale bound-constrained optimization*, SIAM Journal on Scientific Computing, 43 (2021), pp. S704–S726.
[26] K. Kawaguchi, *On the theory of implicit deep learning: Global convergence with implicit layers*, in International Conference on Learning Representations (ICLR), 2021.
[27] Y. Li, M. Tofighi, J. Geng, V. Monga, and Y. C. Eldar, *Deep algorithm unrolling for blind image deblurring*, arXiv preprint arXiv:1902.03493, (2019).
[28] D. Liang, J. Cheng, Z. Ke, and L. Ying, *Deep magnetic resonance image reconstruction: Inverse problems meet neural networks*, IEEE Signal Processing Magazine, 37 (2020), pp. 141–151.
[29] R. Liu, S. Cheng, L. Ma, X. Fan, and Z. Luo, *Deep proximal unrolling: Algorithmic framework, convergence analysis and applications*, IEEE Transactions on Image Processing, 28 (2019), pp. 5013–5026.
[30] Z. Liu, P. Luo, X. Wang, and X. Tang, *Deep learning face attributes in the wild*, in Proceedings of the IEEE international conference on computer vision, 2015, pp. 3730–3738.
[31] D. McKenzie, S. W. Fung, and H. Heaton, *Faster predict-and-optimize with three-operator splitting*, arXiv preprint arXiv:2301.13395, (2023).
[32] A. Mehranian and A. J. Reader, *Model-based deep learning pet image reconstruction using forward–backward splitting expectation–maximization*, IEEE transactions on radiation and plasma medical sciences, 5 (2020), pp. 54–64.
[33] V. Monga, Y. Li, and Y. C. Eldar, *Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing*, IEEE Signal Processing Magazine, 38 (2021), pp. 18–44.
[34] J. L. Mueller and S. Siltanen, *Linear and nonlinear inverse problems with practical applications*, SIAM, 2012.
[35] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett, *Deep learning techniques for inverse problems in imaging*, IEEE Journal on Selected Areas in Information Theory, 1 (2020), pp. 39–56.
[36] Z. Ramzi, P. Ablin, G. Peyré, and T. Moreau, *Test like you train in implicit deep learning*, 2023, https://arxiv.org/abs/2305.15042.
[37] Z. Ramzi, F. Mannel, S. Bai, J.-L. Starck, P. Ciuciu, and T. Moreau, *Shine: Sharing the inverse estimate from the forward pass for bi-level optimization and implicit models*, 2023, https://arxiv.org/abs/2106.00553.
[38] L. I. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D: nonlinear phenomena, 60 (1992), pp. 259–268.
[39] C. Tian, Y. Xu, L. Fei, J. Wang, J. Wen, and N. Luo, *Enhanced cnn for image denoising*, CAAI Transactions on Intelligence Technology, 4 (2019), pp. 17–23.
[40] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, *Plug-and-play priors for model based reconstruction*, in 2013 IEEE Global Conference on Signal and Information Processing, IEEE, 2013, pp. 945–948.
[41] H. F. Walker and P. Ni, *Anderson acceleration for fixed-point iterations*, SIAM Journal on Numerical Analysis, 49 (2011), pp. 1715–1735.
[42] E. Winston and J. Z. Kolter, *Monotone operator equilibrium networks*, Advances in neural information processing systems, 33 (2020), pp. 10718–10728.
[43] D. Wu, K. Kim, and Q. Li, *Computationally efficient deep neural network for computed tomography image reconstruction*, Medical physics, 46 (2019), pp. 4763–4776.
[44] M. Yaqub, F. Jinchao, K. Arshid, S. Ahmed, W. Zhang, M. Z. Nawaz, and T. Mahmood, *Deep learning-based image reconstruction for different medical imaging modalities*, Computational and Mathematical Methods in Medicine, 2022 (2022).
[45] G. Zeng, Y. Guo, J. Zhan, Z. Wang, Z. Lai, X. Du, X. Qu, and D. Guo, *A review on deep learning mri reconstruction without fully sampled k-space*, BMC Medical Imaging, 21 (2021), p. 195.
[46] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, *Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising*, IEEE transactions on image processing, 26 (2017), pp. 3142–3155.

[47] K. Zhang, W. Zuo, S. Gu, and L. Zhang, *Learning deep cnn denoiser prior for image restoration*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 3929–3938.

[48] K. Zhang, W. Zuo, and L. Zhang, *Ffdnet: Toward a fast and flexible solution for cnn-based image denoising*, IEEE Transactions on Image Processing, 27 (2018), pp. 4608–4622.