Fast Training of Implicit Networks with Applications in Inverse Problems

Linghai Liu [1], Allen Tong [2], Lisa Zhao [3]
Mentor: Samy Wu Fung [4]

Emory University REU/RET Computational Mathematics for Data Science

June 30, 2022

[1] Brown University
[2] University of California, Los Angeles
[3] University of California, Berkeley
[4] Department of Applied Mathematics and Statistics, Colorado School of Mines

Acknowledgements

- We sincerely thank the guidance of our mentor, Dr. Samy Wu Fung, and other mentors at Emory University for the opportunity.
- Our work is supported by the US National Science Foundation awards DMS-2051019 and DMS-1751636.

What are Inverse Problems?

Inverse problems consist of recovering a signal x^* (e.g. an image, a parameter of a PDE, etc.) from indirect, noisy measurements d. This measurement process is usually modeled as an operator $\mathcal{A}$, satisfying the following:

$$d = \mathcal{A}x^* + \varepsilon,$$

What are Inverse Problems?

Inverse problems consist of recovering a signal x^* (e.g. an image, a parameter of a PDE, etc.) from indirect, noisy measurements d. This measurement process is usually modeled as an operator $\mathcal{A}$, satisfying the following:

$$d = \mathcal{A}x^* + \varepsilon,$$

Our task deals with image deblurring, i.e.,

- $d \in \mathbb{R}^{n \times n}$: blurred image with noise
- $x^* \in \mathbb{R}^{n \times n}$: original image
- $\varepsilon \in \mathbb{R}^{n \times n}$: random noise (unknown) in $\mathbb{R}^{n \times n}$

From a Classical Approach

Direct Inverse:

$$d = \mathcal{A}x^* + \varepsilon \Longrightarrow x^* = \mathcal{A}^{-1}d - \mathcal{A}^{-1}\varepsilon$$

From a Classical Approach

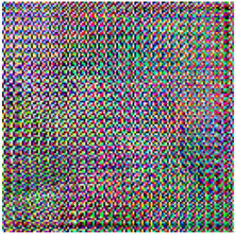
Direct Inverse:

$$d = \mathcal{A}x^* + \varepsilon \implies x^* = \mathcal{A}^{-1}d - \mathcal{A}^{-1}\varepsilon$$

Original Image

Blurred Noisy Image

Apply Inverse



Classical Approach Cont.

Optimization: Formulate an optimization problem as follows:

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

where $R(x)$ is chosen based on prior knowledge of your data, $\lambda > 0$ is a tunable parameter.

Classical Approach Cont.

Optimization: Formulate an optimization problem as follows:

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

where $R(x)$ is chosen based on prior knowledge of your data,
$\lambda > 0$ is a tunable parameter.
E.g. $R(x) = 0 \implies x^* = \mathcal{A}^{-1}d$ when $\mathcal{A}$ invertible

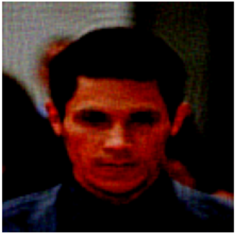
Classical Approach Cont.

Optimization: Formulate an optimization problem as follows:

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

where $R(x)$ is chosen based on prior knowledge of your data,
$\lambda > 0$ is a tunable parameter.
E.g. $R(x) = 0 \implies x^* = \mathcal{A}^{-1}d$ when $\mathcal{A}$ invertible
E.g. Use gradient descent where $R(x) = \frac{\lambda}{2}||x||_{L^2}^2$:

Original Image | Blurred Noisy Image | Apply Gradient Descent



Implicit Deep Learning

- Use dataset $\{(d_i, x_i^*)\}_{i=1}^m$ and physics (namely $\mathcal{A}$)

[5] Davis Gilton, Gregory Ongie, and Rebecca Willett. "Deep equilibrium architectures for inverse problems in imaging." IEEE Transactions on Computational Imaging 7 (2021): 1123-1133.

Implicit Deep Learning

- Use dataset $\{(d_i, x_i^*)\}_{i=1}^m$ and physics (namely $\mathcal{A}$)
- Mimic gradient descent [5], but replace $\lambda \nabla_x R$ with a trainable network S_Θ: $\forall i$ and $0 \leq k \leq K-1$,

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

where:

- $\eta > 0$ is step size
- $T_\Theta(\cdot)$ is a layer of our neural network $\mathcal{N}_\Theta(\cdot)$
- K is the number of layers

[5] Davis Gilton, Gregory Ongie, and Rebecca Willett. "Deep equilibrium architectures for inverse problems in imaging." IEEE Transactions on Computational Imaging 7 (2021): 1123-1133.

Implicit Deep Learning

- Use dataset $\{(d_i, x_i^*)\}_{i=1}^m$ and physics (namely $\mathcal{A}$)
- Mimic gradient descent [5], but replace $\lambda \nabla_x R$ with a trainable network S_Θ: $\forall i$ and $0 \le k \le K-1$,

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

where:

  - $\eta > 0$ is step size
  - $T_\Theta(\cdot)$ is a layer of our neural network $\mathcal{N}_\Theta(\cdot)$
  - K is the number of layers

- Problems: memory, choice of K

[5] Davis Gilton, Gregory Ongie, and Rebecca Willett. "Deep equilibrium architectures for inverse problems in imaging." IEEE Transactions on Computational Imaging 7 (2021): 1123-1133.

Implicit Deep Learning

- Implicit Deep Learning: Send $K \to \infty$ until a fixed point of $T_\Theta(\cdot)$ is found, i.e. $x_i^* = T_\Theta(x_i^*)$

Implicit Deep Learning

- Implicit Deep Learning: Send $K \to \infty$ until a fixed point of $T_\Theta(\cdot)$ is found, i.e. $x_i^* = T_\Theta(x_i^*)$
- Output: given the image d_i, $\mathcal{N}_\Theta(d_i) := x_i^*$

Implicit Deep Learning

- Implicit Deep Learning: Send $K \to \infty$ until a fixed point of $T_\Theta(\cdot)$ is found, i.e. $x_i^* = T_\Theta(x_i^*)$
- Output: given the image d_i, $\mathcal{N}_\Theta(d_i) := x_i^*$
- Question: why convergence?

Implicit Deep Learning

- Implicit Deep Learning: Send $K \to \infty$ until a fixed point of $T_\Theta(\cdot)$ is found, i.e. $x_i^* = T_\Theta(x_i^*)$
- Output: given the image d_i, $\mathcal{N}_\Theta(d_i) := x_i^*$
- Question: why convergence?
- - Convergent if $T_\Theta(\cdot)$ is a contraction mapping with Lipschitz constant $\gamma \in [0, 1)$, i.e,
  $\forall y_1, y_2 \in \mathbb{R}^{n^2}, ||T_\Theta(y_1) - T_\Theta(y_2)||_{L^2} \leq \gamma ||y_1 - y_2||_{L^2}$

Implicit Deep Learning

- Implicit Deep Learning: Send $K \to \infty$ until a fixed point of $T_\Theta(\cdot)$ is found, i.e. $x_i^* = T_\Theta(x_i^*)$
- Output: given the image d_i, $\mathcal{N}_\Theta(d_i) := x_i^*$
- Question: why convergence?
- - Convergent if $T_\Theta(\cdot)$ is a contraction mapping with Lipschitz constant $\gamma \in [0, 1)$, i.e,
  $\forall y_1, y_2 \in \mathbb{R}^{n^2}, ||T_\Theta(y_1) - T_\Theta(y_2)||_{L^2} \leq \gamma ||y_1 - y_2||_{L^2}$
  Then, by *Banach fixed-point theorem*, there exists $y^* \in \mathbb{R}^{n^2}$ s.t. $T_\Theta(y^*) = y^*$

Implicit Backpropagation

Suppose we find a fixed point x^* for the previous update, i.e.,

$$x^* = T_\Theta(x^*)$$

With implicit differentiation,

Implicit Backpropagation

Suppose we find a fixed point x^* for the previous update, i.e.,

$$x^* = T_\Theta(x^*)$$

With implicit differentiation,

$$\frac{dx^*}{d\Theta} = \frac{dT_\Theta(x^*)}{dx^*}\frac{dx^*}{d\Theta} + \frac{\partial T_\Theta(x^*)}{\partial\Theta}$$

$$\implies \left(I - \frac{dT_\Theta(x^*)}{dx^*}\right)\frac{dx^*}{d\Theta} = \frac{\partial T_\Theta(x^*)}{\partial\Theta} \tag{1}$$

So the update rule of trainable parameters becomes:

Implicit Backpropagation

Suppose we find a fixed point x^* for the previous update, i.e.,

$$x^* = T_\Theta(x^*)$$

With implicit differentiation,

$$\frac{dx^*}{d\Theta} = \frac{dT_\Theta(x^*)}{dx^*}\frac{dx^*}{d\Theta} + \frac{\partial T_\Theta(x^*)}{\partial\Theta}$$

$$\implies \left(I - \frac{dT_\Theta(x^*)}{dx^*}\right)\frac{dx^*}{d\Theta} = \frac{\partial T_\Theta(x^*)}{\partial\Theta} \tag{1}$$

So the update rule of trainable parameters becomes:

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

where $\alpha > 0$ is the learning rate.

Implicit Backpropagation

Suppose we find a fixed point x^* for the previous update, i.e.,

$$x^* = T_\Theta(x^*)$$

With implicit differentiation,

$$\frac{dx^*}{d\Theta} = \frac{dT_\Theta(x^*)}{dx^*}\frac{dx^*}{d\Theta} + \frac{\partial T_\Theta(x^*)}{\partial\Theta}$$

$$\Longrightarrow \left(I - \frac{dT_\Theta(x^*)}{dx^*}\right)\frac{dx^*}{d\Theta} = \frac{\partial T_\Theta(x^*)}{\partial\Theta} \tag{1}$$

So the update rule of trainable parameters becomes:

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

where $\alpha > 0$ is the learning rate.

Potential problem: solving (1) is highly nontrivial

Jacobian-Free Backpropagation (JFB)

- Goal: alleviate memory requirement and avoid high computational cost.

[6] S. W. Fung, H. Heaton, Q. Li, D. McKenzie, S. Osher, and W. Yin (2021) Jfb: Jacobian-free back-propagation for implicit networks.

Jacobian-Free Backpropagation (JFB)

- Goal: alleviate memory requirement and avoid high computational cost.
- Key idea: replace the Jacobian $\left(I - \frac{dT_\Theta(x^*)}{dx^*}\right)$ with I

[6] S. W. Fung, H. Heaton, Q. Li, D. McKenzie, S. Osher, and W. Yin (2021) Jfb: Jacobian-free back-propagation for implicit networks.

Jacobian-Free Backpropagation (JFB)

- Goal: alleviate memory requirement and avoid high computational cost.
- Key idea: replace the Jacobian $\left(I - \frac{dT_\Theta(x^*)}{dx^*}\right)$ with I
- Implicit Networks calculate the true gradient:
$\nabla_\Theta \ell = \frac{d\ell}{dx^*}\left(I - \frac{dT_\Theta(x^*)}{dx^*}\right)^{-1}\frac{\partial T_\Theta(x^*)}{\partial \Theta}$

[6] S. W. Fung, H. Heaton, Q. Li, D. McKenzie, S. Osher, and W. Yin (2021) Jfb: Jacobian-free back-propagation for implicit networks.

Jacobian-Free Backpropagation (JFB)

- Goal: alleviate memory requirement and avoid high computational cost.
- Key idea: replace the Jacobian $\left(I - \frac{dT_\Theta(x^*)}{dx^*}\right)$ with I
- Implicit Networks calculate the true gradient:
$\nabla_\Theta \ell = \frac{d\ell}{dx^*}\left(I - \frac{dT_\Theta(x^*)}{dx^*}\right)^{-1}\frac{\partial T_\Theta(x^*)}{\partial\Theta}$
- JFB approximates the gradient: $p_\Theta = \frac{d\ell}{dx^*}\frac{\partial T_\Theta(x^*)}{\partial\Theta}$ which is a **descent direction** for ℓ if the following conditions hold (next slide) [6]:

[6] S. W. Fung, H. Heaton, Q. Li, D. McKenzie, S. Osher, and W. Yin (2021) Jfb: Jacobian-free back-propagation for implicit networks.

JFB Conditions

If:

i. T_Θ is contraction mapping with Lipschitz constant γ

ii. T_Θ is continuously differentiable w.r.t. Θ

iii. $M := \frac{\partial T_\Theta}{\partial \Theta}$ has full column rank

iv. M is well-conditioned, i.e., $\kappa(M^T M) < \frac{1}{\gamma}$

Then

$$p_\Theta = \frac{d\ell}{dx^*} \frac{\partial T_\Theta}{\partial \Theta}$$

is a **descent direction** for loss function ℓ.

Numerical Experiments

- Dataset: CelebA [7] (annotated celebrity faces)

[7] Liu, Ziwei, et al. "Deep learning face attributes in the wild." Proceedings of the IEEE international conference on computer vision. 2015.

Numerical Experiments

- Generate blurred noisy images:

original image

blurred noisy image

PSNR = 21.57, SSIM=0.80

Numerical Experiments

- Generate blurred noisy images:

original image 

blurred noisy image 

PSNR = 21.57, SSIM=0.80

- Train with JFB

Numerical Experiments

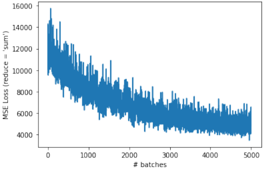
- Generate blurred noisy images:

original image | blurred noisy image

PSNR = 21.57, SSIM=0.80

- Train with JFB
- Preliminary results:

Loss v.s. number of SGD iterations | Reconstructed image



PSNR = 25.69, SSIM=0.86

Future Work

- Train models using different learned optimization algorithms, e.g. *Proximal Gradient Descent* and *Alternating Directions Method of Multipliers (ADMM)*
- Experiment with fastMRI data [8]
- Compare training speeds and accuracy with Jacobian-based algorithms

[8] Zbontar, Jure, et al. "fastMRI: An open dataset and benchmarks for accelerated MRI." arXiv preprint arXiv:1811.08839 (2018).

Future Work

- Train models using different learned optimization algorithms, e.g. *Proximal Gradient Descent* and *Alternating Directions Method of Multipliers (ADMM)*
- Experiment with fastMRI data [8]
- Compare training speeds and accuracy with Jacobian-based algorithms

Thank you!

[8] Zbontar, Jure, et al. "fastMRI: An open dataset and benchmarks for accelerated MRI." arXiv preprint arXiv:1811.08839 (2018).

References

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. (2011)
Distributed optimization and statistical learning via the alternating direction method of multipliers
Foundations and Trends in Machine Learning 3:1–122, 01 2011.

S. Boyd and L. Vandenberghe (2004)
Convex Optimization
Cambridge University Press

Z. Liu, P. Luo, X. Wang, and X. Tang (2015)
Deep learning face attributes in the wild
Proceedings of the IEEE international conference on computer vision 3730–3738, 2015

C. Vogel (2002)
Computational Methods for Inverse Problems
Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics

References Cont.

S. W. Fung, H. Heaton, Q. Li, D. McKenzie, S. Osher, and W. Yin (2021)
Jfb: Jacobian-free back-propagation for implicit networks.
arXiv preprint arXiv:2103.12803

D. Gilton, G. Ongie, and R. Willett. (2021)
Deep equilibrium architectures for inverse problems in imaging.
IEEE Transactions on Computational Imaging 7:1123–1133

G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett (2020)
Deep learning techniques for inverse problems in imaging
IEEE Journal on Selected Areas in Information Theory 1(1):39–56

S. Bai, J. Z. Kolter, and V. Koltun. D (2019)
Deep equilibrium models
Advances in Neural Information Processing Systems 32

J. Zbontar, F. Knoll, A. Sriram, T. Murrell, Z. Huang, M. J. Muckley, A. Defazio, R. Stern, P. Johnson, M. Bruno, et al. (2018)
fastmri: An open dataset and benchmarks for accelerated mri
arXiv preprint arXiv:1811.08839, 2018.

Banach Fixed-Point Theorem

We demonstrate it in $\mathbb{R}^d$:
Suppose $T : \mathbb{R}^d \mapsto \mathbb{R}^d$ is a contraction map with Lipschitz constant $\gamma \in [0, 1)$.
$\forall x_0 \in \mathbb{R}^d$, iterate as follows:

$$\begin{aligned} x_1 &= T(x_0) \\ x_2 &= T(x_1) \\ &\vdots \\ x_{i+1} &= T(x_i) \\ &\vdots \end{aligned}$$

Then we obtain a sequence $\{x_m\}_{m \in \mathbb{N}}$

Banach Fixed-Point Theorem

Observe that

$$
\begin{aligned}
||x_2 - x_1|| &= ||T(x_1) - T(x_0)|| \leq \gamma ||x_0 - x_1|| \\
||x_3 - x_2|| &= ||T(x_2) - T(x_1)|| \leq \gamma ||x_2 - x_1|| \leq \gamma^2 ||x_0 - x_1|| \\
&\vdots \\
||x_{i+1} - x_i|| &\leq \gamma^i ||x_0 - x_1|| \\
&\vdots
\end{aligned}
$$

So $\lim_{m\to\infty} ||x_{m+1} - x_m|| \leq \lim_{m\to\infty} \gamma^m ||x_0 - x_1|| = 0$
We also know that $0 \leq \lim_{m\to\infty} ||x_{m+1} - x_m||$.
$\implies \lim_{m\to\infty} ||x_{m+1} - x_m|| = 0$ by *Squeeze Theorem*

Banach Fixed-Point Theorem

By *Triangular Inequality*,

$$
\begin{aligned}
||x_{m+k} - x_m|| &\leq ||x_m - x_{m+1}|| + ||x_{m+1} - xm + k|| \\
&\leq ||x_m - x_{m+1}|| + (||x_{m+1} - x_{m+2}|| + ||x_{m+2} - x_{m+k}||) \\
&\vdots \\
&\leq ||x_m - x_{m+1}|| + ||x_{m+1} - x_{m+2}|| + \cdots \\
&+ ||x_{m+k-2} - x_{m+k-1}|| + ||x_{m+k-1} - x_{m+k}||
\end{aligned}
$$

By *Squeeze Theorem* again, $\{x_m\}_{m\in\mathbb{N}}$ is a Cauchy sequence, which is equivalent to $\lim\limits_{m\to\infty} x_m = x^*$ exists.
$\implies x^* = T(x^*)$ is a fixed point. $\square$

Proximal Gradient Descent

With a function $h(\cdot)$, we can define a proximal operator

$$\mathrm{prox}_h(x) = \arg\min_u \frac{1}{2}||u - x||^2_{L^2} + h(u)$$

Then the updating rule becomes:

$$x^{k+1} = \mathrm{prox}_{h,\eta}\left(x^k - \eta\nabla_x||\mathcal{A}x^k - d||^2_{L^2}\right)$$

We can replace this prox_h with a trainable network $R_\Theta(\cdot)$:

$$x^{k+1} = R_\Theta\left(x^k - \eta\nabla_x||\mathcal{A}x^k - d||^2_{L^2}\right)$$