SHINE: SHaring the INverse Estimate from the forward pass for bi-level optimization and implicit models

Zaccharie Ramzi, Florian Mannel, Shaojie Bai, Jean-Luc Starck, Philippe Ciuciu, Thomas Moreau

SHINE: SHaring the INverse Estimate from the forward pass for bi-level optimization and implicit models

Zaccharie Ramzi[*]
CEA (Neurospin and Cosmostat), Inria (Parietal)
Gif-sur-Yvette, France
zaccharie.ramzi@inria.fr

Florian Mannel
University of Graz
Graz, Austria

Shaojie Bai
Carnegie Mellon University
Pittsburgh, USA

Jean-Luc Starck
AIM, CEA, CNRS
Université Paris-Saclay
Université Paris Diderot
Sorbonne Paris Cité

Philippe Ciuciu
CEA (Neurospin), Inria (Parietal)
Gif-sur-Yvette, France

Thomas Moreau
Inria (Parietal)
Gif-sur-Yvette, France

Abstract

In recent years, implicit deep learning has emerged as a method to increase the depth of deep neural networks. While their training is memory-efficient, they are still significantly slower to train than their explicit counterparts. In Deep Equilibrium Models (DEQs), the training is performed as a bi-level problem, and its computational complexity is partially driven by the iterative inversion of a huge Jacobian matrix. In this paper, we propose a novel strategy to tackle this computational bottleneck from which many bi-level problems suffer. The main idea is to use the quasi-Newton matrices from the forward pass to efficiently approximate the inverse Jacobian matrix in the direction needed for the gradient computation. We provide a theorem that motivates using our method with the original forward algorithms. In addition, by modifying these forward algorithms, we further provide theoretical guarantees that our method asymptotically estimates the true implicit gradient. We empirically study this approach in many settings, ranging from hyperparameter optimization to large Multiscale DEQs applied to CIFAR and ImageNet. We show that it reduces the computational cost of the backward pass by up to two orders of magnitude. All this is achieved while retaining the excellent performance of the original models in hyperparameter optimization and on CIFAR, and giving encouraging and competitive results on ImageNet.

1 Introduction

Implicit deep learning models such as Neural ODEs [10], OptNets [2] or Deep Equilibrium models (DEQs) [3, 4] have recently emerged as a way to train infinitely deep models without the associated memory cost. Indeed, while it has been observed that the performance of deep learning models increases with their depth [43], an increase in depth also translates into an increase in the memory footprint required for training, which is hardware-constrained. While other works such as invertible

[*]https://www.cosmostat.org/people/zaccharie-ramzi

Preprint. Under review.

neural networks [19, 40] or gradient checkpointing [11] also tackle this issue, implicit models do so using an $\mathcal{O}(1)$ memory cost and with constraints on the architecture that are usually not detrimental to the model performance [3].

In general, the formulation of DEQs can be cast as a bi-level problem of the following form:

$$\arg\min_{\theta} \mathcal{L}(z^\star) \quad \text{subject to} \quad g_\theta(z^\star) = 0 \tag{1}$$

This formulation allows us to consider both DEQs and other bi-level problems such as bi-level optimization under the same framework. We will refer to the root finding problem $g_\theta(z) = 0$ as the *inner problem*, and call its resolution the *forward pass*. On the other hand, we will refer to $\arg\min_\theta \mathcal{L}(z^\star)$ as the *outer problem*, and call the computation of the gradient of $\mathcal{L}(z^\star)$ w.r.t. θ the *backward pass*. The core idea for DEQs is that their output $z^\star$ is expressed as a fixed point of a parametric function f_θ from $\mathbb{R}^d$ to $\mathbb{R}^d$, i.e., $g_\theta(z^\star) = z^\star - f_\theta(z^\star) = 0$.[2] This model is said to have infinitely many weight-tied layers as $z^\star$ can be obtained by successively applying the layer f_θ infinitely many times, provided f_θ is contractive. To train this type of network efficiently and avoid high memory cost, one does not compute the gradient through back-propagation but relies on the implicit function theorem [26] which gives an analytical expression of the partial derivative $\frac{\partial z^\star}{\partial \theta}$. These models have been successfully applied to large-scale tasks such as language modeling [3], computer vision tasks [4] and inverse problems [18, 23].

While their training is memory efficient, it requires the computation of matrix-vector products involving the inverse of a large Jacobian matrix, which is computationally demanding. To make this computation tractable, one needs to rely on an iterative algorithm based on vector-Jacobian products, which renders the training particularly slow, as highlighted by the original authors [4] (see also the break down of the computational effort in Appendix E.3). With the increasing popularity of DEQs, a core question is how to reduce the computational cost of the training. This would make DEQs more accessible for practitioners and reduce the associated energy cost.

More generally, this computation issue is prevalent in many bi-level problems where the implicit function theorem is used. For instance, Pedregosa [38] proposed to use such formulation to perform hyperparameter optimization for logistic regression, where a similar operation involving the iterative inversion of a large scale matrix slows down the method. In this case, the inner optimization problem $\min_z r_\theta(z)$ is smooth and convex, which enables us to write them also in the form of Eq. (1) where $g_\theta(z) = \nabla_z r_\theta(z)$. In this work, while study DEQs as a high-dimensional problem instance in the deep learning context, we also show that our approach applies to the differentiation of these general bi-level problem formulations.

In particular, for DEQ models, the forward pass is usually computed with a quasi-Newton (qN) algorithm, such as Broyden's method [6], which approximates tefficiently he Jacobian matrix and its inverse for root-finding. More generally, bi-level optimization problems often rely on the LBFGS [31] algorithm to solve the inner problem while approximating the inverse of the Hessian in the direction of the steps. In both cases, the generated quasi-Newton (qN) matrices efficiently approximate the (inverse of the) Jacobian/Hessian. In this work, we propose to exploit these properties and design extra updates of the qN matrices which maintain the approximation property in the direction of the steps, and ensure that the inverse Jacobian is approximated in an additional direction. Specifically, this direction is selected such that the gradient approximation provably converges to the true gradient. In effect, we can compute the gradient using the inverse of the final qN matrix instead of an iterative algorithm to invert the Jacobian in the gradient's direction, while stressing that the inverse of a qN matrix, and thus the multiplication with it, can be computed very efficiently. The contributions of our paper are the following:

- We introduce a new method to greatly accelerate the backward pass of DEQs (and generally, the differentiation of bi-level problems) using qN matrices that are available as a by-product of the forward computations. We call this method **SHINE** (**SH**aring the **IN**verse **E**stimate).
- We enhance this method by incorporating knowledge from the outer problem into the inner problem resolution.
- We provide strong theoretical guarantees for this approach in various settings.
- We additionally showcase its use in hyperparameter optimization. Here, we demonstrate that it provides a gain in computation time compared to the state of the art.

[2] Here, we do not explicitly write the dependence of f_θ on the input x of the DEQ, usually referred to as the injection.

- We test it for DEQs for the classification task on two datasets, CIFAR and ImageNet. Here, we show that it decreases the training time while remaining competitive in terms of performance.
- We empirically show that accelerated backward methods perform well even when contractivity assumptions are not met for DEQs.

We emphasize that the goal of this paper is **neither** to improve the algorithms used to compute $z^\star$, **nor** is it to demonstrate how to perform the inversion of a matrix in a certain direction as a stand-alone task. Rather, we are describing an approach that combines the resolution of the inner problem with the computation of the gradient of the outer problem to accelerate the overall process.

The idea to use additional updates of the qN matrices to ensure additional approximation properties is not new, and it is also known that a full matrix inversion can be accomplished in this way. For instance, Gower and Richtárik [20] used sketching to design appropriate extra secant conditions in order to obtain guarantees of uniform convergence towards the inverse of the Jacobian. However, to the best of our knowledge, our work is the first in which the additional update is designed to yield the inverse in a single direction (which is substantially cheaper than computing the inverse). In particular, we have not seen this idea in machine learning.

A concurrent work by Fung et al. [16] is also concerned with the acceleration of DEQs' training, where the inverse Jacobian is approximated with the identity. Under strong contractivity and conditioning assumptions, it is proved that the resulting approximation is a descent direction and this is empirically tested with constrained networks. In this paper, we extend the Jacobian-Free method to large scale multiscale DEQs and show that it performs well in this setting, which was not done in the original paper [16]. We also show that the Jacobian-Free method is not suitable for more general bi-level problems.

2 Hypergradient Optimization with Approximate Jacobian Inverse

2.1 SHINE: Hypergradient Descent with Approximate Jacobian Inverse

Hypergradient Optimization Hypergradient optimization is a first-order method used to solve Problem (1). We recall that in the case of smooth convex optimization, $\frac{\partial g_\theta}{\partial z}$ is the Hessian of the inner optimization problem, while for deep equilibrium models, it is the Jacobian of the root equation. In the rest of this paper, with a slight abuse of notation, we will refer to both these matrices with J_{g_θ} whenever the results can be applied to both contexts. To enable Hypergradient Optimization, i.e. gradient descent on $\mathcal{L}$ with respect to θ, Bai et al. [3, Theorem 1] show the following theorem, which is based on implicit differentiation [26]:

Theorem 1 (Hypergradient [3, 26])**.** *Let $\theta \in \mathbb{R}^p$ be a set of parameters, let $\mathcal{L} : \mathbb{R}^d \to \mathbb{R}$ be a loss function and $g_\theta : \mathbb{R}^d \to \mathbb{R}^d$ be a root-defining function. Let $z^\star \in \mathbb{R}^d$ such that $g_\theta(z^\star) = 0$ and $J_{g_\theta}(z^\star)$ is invertible, then the gradient of the loss $\mathcal{L}$ wrt. θ, called Hypergradient, is given by*

$$\left.\frac{\partial \mathcal{L}}{\partial \theta}\right|_{z^\star} = \nabla_z \mathcal{L}(z^\star) J_{g_\theta}(z^\star)^{-1} \left.\frac{\partial g_\theta}{\partial \theta}\right|_{z^\star}. \quad (2)$$

Algorithm 1: qN method to solve $g_\theta(z^\star) = 0$

Result: Root $z^\star$, qN matrix B
$b =$ `true` if using Broyden's method,
$b =$ `false` if using BFGS
$n = 0$, $z_0 = 0$, $B_0 = I$
while *not converged* **do**
  $p_n = -B_n^{-1} g_\theta(z_n)$, $z_{n+1} = z_n + \alpha_n p_n$
  `// ` α_n `can be 1 or determined by line-search`
  $y_n = g_\theta(z_{n+1}) - g_\theta(z_n)$
  $s_n = z_{n+1} - z_n$
  if b **then**
    $B_{n+1} = \underset{X:\, Xs_n = y_n}{\arg\min} \|X - B_n\|_F$
  else
    $B_{n+1} = \underset{X:\, X = X^T \wedge Xs_n = y_n}{\arg\min} \|X^{-1} - B_n^{-1}\|$
    `// The norm used in BFGS is a weighted Frobenius norm`
  end
  $n \leftarrow n + 1$
end
$z^\star = z_n$, $B = B_n$

In practice, we use an algorithm to approximate $z^\star$, and Theorem 1 gives a plug-in formula for the backward pass. We highlight that this formula is independent of the choice of the algorithm. Moreover, as opposed to explicit networks, we do not need to store intermediate activations, resulting in the aforementioned training time memory gain for DEQs. Once $z^\star$ has been obtained, one of the major bottlenecks in the computation of the Hypergradient is the inversion of $J_{g_\theta}(z^\star)$ in the directions $\left.\frac{\partial g_\theta}{\partial \theta}\right|_{z^\star}$ or $\nabla_z \mathcal{L}(z^\star)$.

Quasi-Newton methods In practice, the forward pass is often carried out with qN methods. For instance, in the case of bi-level optimization for Logistic Regression, Pedregosa [38] used L-BFGS [31], while for Deep Equilibrium Models, Bai et al. [3] used Broyden's method [6], later adapted to the multi-scale case in a limited-memory version [4].
These quasi-Newton methods were first inspired by Newton's method, which finds the root of g_θ via the recurrent Jacobian-based updates $z_{n+1} = z_n - J_{g_\theta}(z_n)^{-1} g_\theta(z_n)$. Specifically, they replace the Jacobian $J_{g_\theta}(z_n)$ by an approximation B_n that is based on available values of the iterates z_n and g_θ rather than its derivative. These B_n, called qN matrices, are defined recursively via an optimization problem with constraints called secant conditions. Solving this problem leads to expressing B_n as a rank-one or rank-two update of B_{n-1}, so that B_n is the sum of the initial guess B_0 (in our settings, the identity) and n low-rank matrices (less than n in limited memory settings). This low rank structure allows efficient multiplication by B_n and B_n^{-1}. We now explain how the use of qN methods as inner solver can be exploited to resolve this computational bottleneck.

SHINE Roughly speaking, our proposition is to use $B^{-1} = \lim_{n\to\infty} B_n^{-1}$ as a replacement for $J_{g_\theta}(z^\star)^{-1}$ in Eq. (2), i.e. to share the inverse estimate between the forward and the backward passes. This gives the approximate Hypergradient

$$p_\theta = \nabla_z \mathcal{L}(z^\star) B^{-1} \frac{\partial g_\theta}{\partial \theta}\bigg|_{z^\star} \tag{3}$$

In practice we will consider the non-asymptotical direction $p_\theta^{(n)} = \nabla_z \mathcal{L}(z_n) B_n^{-1} \frac{\partial g_\theta}{\partial \theta}\Big|_{z_n}$. Thanks to the Sherman-Morrison formula [42], the inversion of B_n can be done very efficiently (using scalar products) compared to the iterative methods needed to invert the true Jacobian $J_{g_\theta}(z^\star)$. In turn, this significantly reduces the computational cost of the Hypergradient computation.

Relationship to the Jacobian-Free method Because $B_0 = I$ in our setting, we may regard B as an identity matrix perturbed by a few rank-one updates. In the directions that are used for updates, B is going to be different from the identity, and hopefully closer to the true Jacobian in those directions. However, in all orthogonal directions we fall exactly into the setting of the Jacobian-Free method introduced by Fung et al. [16]. In that work, $J_{g_\theta}(z^\star)^{-1}$ is approximated by I, and the authors highlight that this is equivalent to using a preconditioner on the gradient. Under strong assumptions on g_θ they show that this preconditioned gradient is still a descent direction. In their experiments, they force g_θ to respect such assumptions, thereby constraining the performance of the network.

2.2 Convergence to the true gradient

To further justify and formalize the idea of SHINE, we show that the direction $p_\theta^{(n)}$ converges to the Hypergradient $\frac{\partial \mathcal{L}}{\partial \theta}\Big|_{z^\star}$. We now collect the assumptions that will be used for this purpose.

Assumption 1 (Uniform Linear Independence (ULI) [30])**.** *There exist a positive constant $\rho > 0$ and natural numbers $n_0 \geq 0$ and $m \geq d$ with the following property: For any $n \geq n_0$ we can find indices $n \leq n_1 \leq \ldots \leq n_d \leq n + m$ such that, for p_n defined in Algorithm 1, the smallest singular value of the $d \times d$ matrix*

$$\left(\frac{p_{n_1}}{\|p_{n_1}\|}, \frac{p_{n_2}}{\|p_{n_2}\|}, \ldots, \frac{p_{n_d}}{\|p_{n_d}\|} \right)$$

is no smaller than ρ.

Assumption 2 (Smoothness and convergence to the fixed point)**.** *(i) $\sum_{n=0}^{\infty} \|z_n - z^\star\| < \infty$ for some $z^\star$ with $g_\theta(z^\star) = 0$; (ii) g_θ is C^1, J_{g_θ} is Lipschitz continuous near $z^\star$, and $J_{g_\theta}(z^\star)$ is invertible; (iii) $\nabla_z \mathcal{L}$ is continuous, and $\forall \theta$, $\frac{\partial g_\theta}{\partial \theta}$ is continuous.*

Remark. *The Assumption 2 (i) implies $\lim_{n\to\infty} z_n = z^\star$. The existence of the Jacobian and its inverse are assumptions that are already made in the regular DEQ setting just to train the model.*

Theorem 2 (Convergence of SHINE to the Hypergradient using ULI)**.** *Let us denote $p_\theta^{(n)}$, the SHINE direction for iterate n in Algorithm 1 with $b =$ `true`. Under Assumptions 1 and 2, for a given parameter θ, (z_n) converges q-superlinearly to $z^\star$ and*

$$\lim_{n\to\infty} p_\theta^{(n)} = \frac{\partial \mathcal{L}}{\partial \theta}\bigg|_{z^\star}$$

Proof. From [35, Theorem 5.7] we obtain that $\lim_{n\to\infty} B_n = J_{g_\theta}(z^\star)$. We can then conclude using the continuity of the inversion operator on the space of invertible matrices and of the right and left matrix vector multiplications. A complete proof is given in Appendix B.1. □

Theorem 2 establishes convergence of the SHINE direction to the true Hypergradient, but relies on Assumption 1 (ULI). While ULI is often used to prove convergence results for qN matrices, e.g. in [12, 30, 36], it is a strong assumption whose satisfaction in practice is debatable, cf., e.g., [15]. For Broyden's method, ULI is violated in all numerical experiments in [32–34], and those works also prove that ULI is necessarily violated in certain settings (but the setting of this work is not covered). In the following we therefore derive results that do not involve ULI.

2.3 Outer Problem Awareness

The ULI assumption guarantees convergence of B_n^{-1} to $J_{g_\theta}(z^\star)^{-1}$. However, Eq. (2) only requires the multiplication of $J_{g_\theta}(z^\star)^{-1}$ with $\frac{\partial g_\theta}{\partial \theta}|_{z^\star}$ from the right and $\nabla_z \mathcal{L}(z^\star)$ from the left.

BFGS with OPA In order to strengthen Theorem 2, let us consider the setting of bi-level optimization with a single regularizing hyperparameter θ. There, the partial derivative $\frac{\partial g_\theta}{\partial \theta}|_{z^\star}$ is a d-dimensional vector and it is possible to compute its approximation $\frac{\partial g_\theta}{\partial \theta}|_{z_n}$ at a reasonable cost. We propose to incorporate additional updates of the quasi-Newton matrix B_n into Algorithm 1 that improve the approximation quality of B_n^{-1} in the direction $\frac{\partial g_\theta}{\partial \theta}|_{z_n}$ (thus asymptotically in the direction $\frac{\partial g_\theta}{\partial \theta}|_{z^\star}$). Given a current iterate pair (z_n, B_n), these additional updates only change B_n, but not z_n. We will demonstrate that a suitable update direction $e_n \in \mathbb{R}^d$ is given by

$$e_n = t_n B_n^{-1} \frac{\partial g_\theta}{\partial \theta}\Big|_{z_n}, \tag{4}$$

where $(t_n) \subset [0, \infty)$ satisfies $\sum_n t_n < \infty$. This update direction will be used to create an extra secant condition $X^{-1}(g_\theta(z_n + e_n) - g_\theta(z_n)) = e_n$ for the additional update of B_n. Since this extra update is based on the outer problem, we refer to this technique as Outer-Problem Awareness (OPA). The complete pseudo code of the OPA method in the LBFGS algorithm [31] is given in Appendix A. We now prove that if extra updates are applied at a fixed frequency, then fast (q-superlinear) convergence of (z_n) to $z^\star$ is retained, while convergence of the SHINE direction to the true Hypergradient is also ensured. To show this, we use the following assumption.

Assumption 3 (Assumptions for BFGS)**.** *Let $g_\theta(z) = \nabla_z r_\theta(z)$ for some C^2 function $r_\theta : \mathbb{R}^d \to \mathbb{R}$. Consider Algorithm 1 with $b =$ `false`. We assume some regularity on r and that an appropriate line search is used. An extended version of this assumption is given in Appendix B.2 (Assumption 5).*

Theorem 3 (Convergence of SHINE to the Hypergradient for BFGS with OPA)**.** *Let us consider $p_\theta^{(n)}$, the SHINE direction for iterate n in Algorithm 1 that is enriched by extra updates in the direction e_n defined in (4). Under Assumptions 2 (ii-iii) and 3, for a given parameter θ, we have the following: Algorithm 1, for any symmetric and positive definite matrix B_0, generates a sequence (z_n) that converges q-superlinearly to $z^\star$, and there holds*

$$\lim_{n\to\infty} p_\theta^{(n)} = \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{z^\star} \tag{5}$$

Proof. It follows from known results that the extra updates do not destroy the q-superlinear convergence of (z_n). The proof of (5) relies firstly on the fact that by continuity of the derivative of g_θ, we have $\lim_{n\to\infty} \frac{\partial g_\theta}{\partial \theta}|_{z_n} = \frac{\partial g_\theta}{\partial \theta}|_{z^\star}$. Due to the extra updates we can show convergence of the qN matrices to the true Hessian in the direction of the extra steps e_n, from which (5) follows. A full proof is provided in Appendix B.2. □

Remark. *Theorem 3 also holds without line searches (i.e., $\alpha_n = 1$ for all n) and any C^2 function r_θ (such that $g_\theta(z) = \nabla_z r_\theta(z)$) with locally Lipschitz continuous Hessian if z_0 is close enough to some $z^\star$ with $\nabla_z r_\theta(z^\star) = 0$ and $\nabla^2_{zz} r_\theta(z^\star)$ positive definite.*

We note that Theorem 3 guarantees fast convergence of the iterates (z_n) and that z_0 does not have to be close to $z^\star$ for that guarantee. Also, there is no restriction on B_0 other than being symmetric and positive definite (which is satisfied for our choice $B_0 = I$). Finally, Theorem 3 does not rely on ULI. From a practical standpoint we thus regard Theorem 3 as a much stronger result than Theorem 2.

Adjoint Broyden with OPA It is not practical to use the partial derivative $\frac{\partial g_\theta}{\partial \theta}$ in the DEQ setting because it is a huge Jacobian that we do not have access to in practice. In order to still leverage the core idea of OPA, we propose to use extra updates that ensure that B_n^{-1} approximates $J_{g_\theta}(z^\star)^{-1}$ in the direction $\nabla_z \mathcal{L}(z^\star)$ applied by left-multiplication, as required by (2). An appropriate secant condition is given by

$$v_n^T B_{n+1} = v_n^T J_{g_\theta}(z_{n+1}), \tag{6}$$

where

$$v_n^T = \nabla_z \mathcal{L}(z_n) B_n^{-1}. \tag{7}$$

To incorporate the secant condition (6), we use the Adjoint Broyden's method [41], a qN method relying on the efficient vector-Jacobian multiplication by J_{g_θ} using auto-differentiation tools. To prove convergence of the SHINE direction for this method, we need the following assumption.

Assumption 4 (Uniform boundedness of the inverse qN matrices)**.** *The sequence (B_n) generated by Algorithm 1 satisfies*

$$\sup_{n \in \mathbb{N}} \|B_n^{-1}\| < \infty.$$

Remark. *Convergence results for quasi-Newton methods usually include showing that Assumption 4 holds, cf. [7, Theorem 3.2] for Broyden's method and the BFGS method, respectively, [41, Theorem 1] for the Adjoint Broyden's method. It can also be proved that Assumption 4 holds for globalized variants of these methods, e.g., for the line-search globalizations of Broyden's method proposed in [29]. We point out that Assumption 1 entails $\lim B_n = J_{g_\theta}(z^\star)$ and thus $\lim B_n^{-1} = J_{g_\theta}(z^\star)^{-1}$, so it is clearly stronger than Assumption 4.*

Theorem 4 (Convergence of SHINE to the Hypergradient for Adjoint Broyden with OPA)**.** *Let us consider $p_\theta^{(n)}$, the SHINE direction for iterate n in Algorithm 1 with the Adjoint Broyden secant condition (6) and extra update in the direction v_n defined in (7). Under Assumptions 2 and 4, for a given parameter θ, we have q-superlinear convergence of (z_n) to $z^\star$ and*

$$\lim_{n \to \infty} p_\theta^{(n)} = \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{z^\star}$$

Proof. The q-superlinear convergence of (z_n) follows from [41, Theorem 2]. To establish convergence of the SHINE direction, we proceed in three steps. First, it is shown that for $\nabla_z \mathcal{L}(z^\star) = 0$ the claim holds due to continuity and Assumption 4. Then $\nabla_z \mathcal{L}(z^\star) \neq 0$ is considered and it is proved that the desired convergence holds on the subsequence that corresponds to the additional updates. Lastly, this result is transferred to the entire sequence by involving the fixed frequency of the additional updates. The complete proof is provided in Appendix B.3. □

Using the Adjoint Broyden's method comes at a computational cost. Indeed, because we now rely on J_{g_θ}, we have to store the activations of $g_\theta(z)$ (which has a computational cost in addition to a memory cost), but also perform the vector-Jacobian product in addition to the function evaluation.

3 Results

We test our method in 3 different setups and compare it to the original iterative inversion and its closest competitor, the Jacobian-Free method [16]. We draw the reader's attention to the fact that although the Jacobian-Free method [16] is used outside the assumptions needed to have theoretical guarantees[3] of descent, it still performs relatively well in the Deep Equilibrium setting. The same is true for SHINE: While the ULI assumption is not met (and we are in practice far from the fixed point convergence), it performs well in practice.

Implementations. All the bi-level optimization experiments were done using the HOAG code [38][4], which is based on the Python scientific ecosystem [21, 39, 44]. All the Deep Equilibrium experiments were done using the PyTorch [37] code for Multiscale DEQ [4][5], which was distributed under the MIT license. Plots were done using Matplotlib [24], with Science Plots style [17]. DEQ trainings were done in a publicly funded HPC, using nodes with 4 V100 GPUs.

In practice, we never reach convergence of (z_n), hence the approximate gradient might be far from the true gradient. To improve the approximation quality, we now propose two variants of our method.

[3] See the results on contractivity in Appendix E.2.

[4] https://github.com/fabianp/hoag

[5] https://github.com/locuslab/mdeq

Transition to the exact Jacobian Inverse. The approximate gradient $p_\theta^{(n)}$ can also be used as the initialization of an iterative algorithm for inverting $J_{g_\theta}(z^\star)$ in the direction $\nabla_z \mathcal{L}(z^\star)$. With a good initialization, faster convergence can be expected. Moreover, if the iterative algorithm is also a qN method, which is the case in practice in the Multiscale DEQ implementation, we can use the qN matrix B from the forward pass to initialize the qN matrix of this algorithm. We refer to this strategy as the *refine strategy*. Because the refine strategy is essentially a smart initialization scheme, it recovers all the theoretical guarantees of the original method [3, 4, 38].

Fallback in the case of wrong inversion. Empirically, we noticed that using B can sometimes produce bad approximations, although with very low probability. We propose to detect this with by monitoring a telltale sign based on the norm of the approximation, as we verified on several examples that cases with a huge norm compared to the correct inversion also had a very bad correlation with the correct inversion. In these cases, we can simply fallback onto another inversion method. For the Deep Equilibrium experiments, when the norm of the inversion using SHINE is 1.3 times above the norm of the inversion using the Jacobian-Free method (which is available at no extra computational cost), we use the Jacobian-Free inversion. We refer to this strategy as the *fallback strategy*.

3.1 Bi-level optimization – Hyperparameter optimization in Logistic Regression

We first test SHINE in the simple setting of bi-level optimization for $L2$-regularized logistic regression, using the code from Pedregosa [38] and the same datasets. The code for these experiments is available here: https://github.com/zaccharieramzi/hoag/tree/shine. Convergence on unseen data is illustrated in Figure 2.[6] An acceptable level of performance is reached twice faster for the SHINE method compared to any other competitor, except Jacobian-Free on the real-sim dataset. However, its behaviour is unstable as one can notice with the 20news dataset. Another finding is that the refine strategy does not provide a definitive improvement over the vanilla version of SHINE. To make sure that SHINE performance does not results from poor hyperparameter choice, we verified that one cannot simply truncate the inversion iterations to a small number (see Appendix E.1).

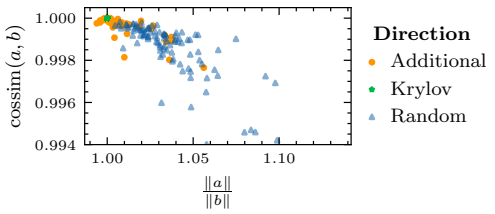

Figure 1: **Quality of the inversion using OPA :** Ratio of the inverse approximation $b = B_n^{-1} v$ over the exact inverse $a = J_{g_\theta}(z^\star)^{-1} v$ function of the cosine similarity between a and b for 3 different directions: the prescribed direction, the Krylov direction and a random direction.

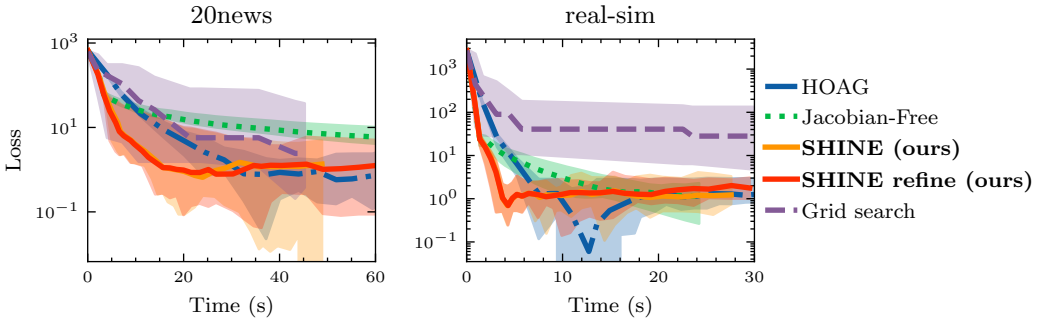

Figure 2: **Bi-level optimization:** Convergence of different hyperparameter optimization methods on the $L2$-regularized logistic regression problem for the 2 datasets (20news [28] and real-sim [1]) on held-out test data. An extended version of this figure with more methods is provided in Appendix E.1.

We also tested our implementation of OPA on the 20news dataset. We did not compare it to the Fortran L-BFGS implementation because we use a pure Python implementation for OPA. In the original method the inversion uses optimized code. In contrast, in our implementation of OPA the

[6] To facilitate the reader's understanding of the figures, we plot the empirical suboptimality, but we do remind them that there is no guarantee of convergence on held-out test data, as is better seen on the real-sim dataset.

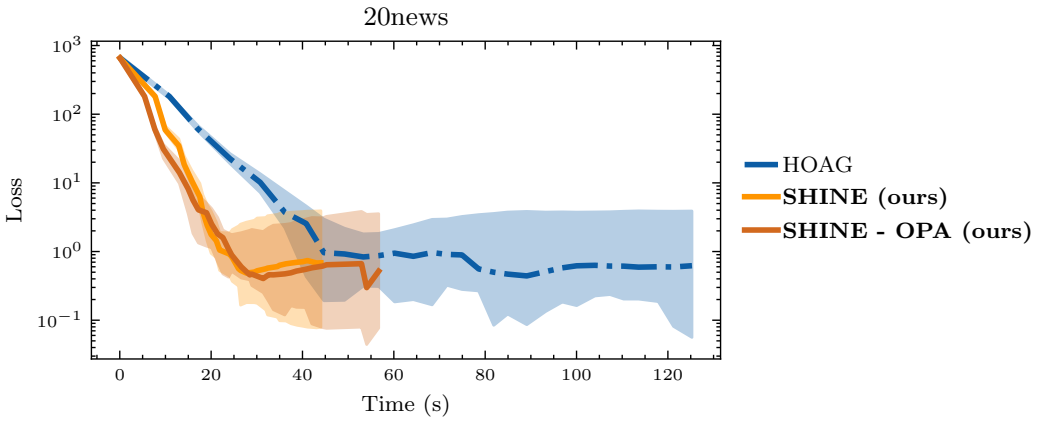

Figure 3: **Bi-level optimization with OPA:** Convergence of different hyperparameter optimization methods on the $L2$-regularized logistic regression problem for the 20news dataset [28] on held-out test data. The methods are implemented with a pure Python implementation of L-BFGS in order to allow the introduction of OPA.

inner solver is slow in comparison to the computation of hypergradients. However, the improvement of SHINE over the original method is still clear, cf. the results in Figure 3. We underline, though, that SHINE with OPA has a strong theoretical foundation.

We also showed on a smaller dataset, the breast cancer dataset [14], that OPA indeed ensures a better approximation of the inverse in the prescribed direction. For a given split of the data, we compared the quality of the approximation of the inversion in three different directions: a direction chosen randomly but used for the OPA update, the Krylov direction $\frac{\partial g_\theta}{\partial z}\Big|_{z^\star}(z_n - z_{n-1})$ and a random direction not used in the qN algorithm. The results for 100 runs with different random seeds are depicted in Figure 1, where we can observe that OPA indeed ensures a better inversion compared to a random direction. We also notice that a poor direction for the inversion is correlated with a small magnitude.

3.2 Deep Equilibrium Models

Next, we tested SHINE on the more challenging DEQ setup. Two experiments illustrate the performance of SHINE on the image classification task on two datasets. For both datasets, we used the exact same configuration of models as that used in the original Multiscale DEQ paper [4] and did not fine tune any hyperparameter. For the different DEQs training methods, models for a given seed share the same unrolled-pretraining steps. We do not include OPA in the DEQ results because we have not managed to optimize its implementation and fix all the bugs, but provide partial results in Appendix E.4. The code for these experiments is available here: `https://github.com/zaccharieramzi/shine`.

CIFAR-10. The first dataset is CIFAR-10 [27] which features 60,000 32×32 images representing 10 classes. For this dataset, the size of the multi-scale fixed point is $d = 50$k. We train the models for five different random seeds.

The results in Figure 4 show that for the vanilla version, SHINE slightly outperforms the Jacobian-Free method [16]. Additionally, our results suggest that SHINE (in its vanilla version) is able to reduce the time taken for the backward pass almost 10-fold compared to the original method while retaining a competitive performance (on par with Res-Net-18 [22] at 92.9%). Finally, we do highlight that the Jacobian-Free method [16] is able to perform well outside the scope of its theoretical assumptions, albeit with slightly worse performance than SHINE.

ImageNet. The second dataset is the ImageNet dataset [13] which features 1.2 million images cropped to 224×224, representing 1000 classes. This dataset is recognized as a large-scale computer vision problem and the dimension of the fixed point to find is $d = 190$k.

For this challenging task, we noticed that the vanilla version of SHINE was suffering a big drop just after the transition from unrolled pre-training to actual equilibrium training. To remedy partly this problem, we introduced the fallback to Jacobian-Free inversion. The results for a single random seed

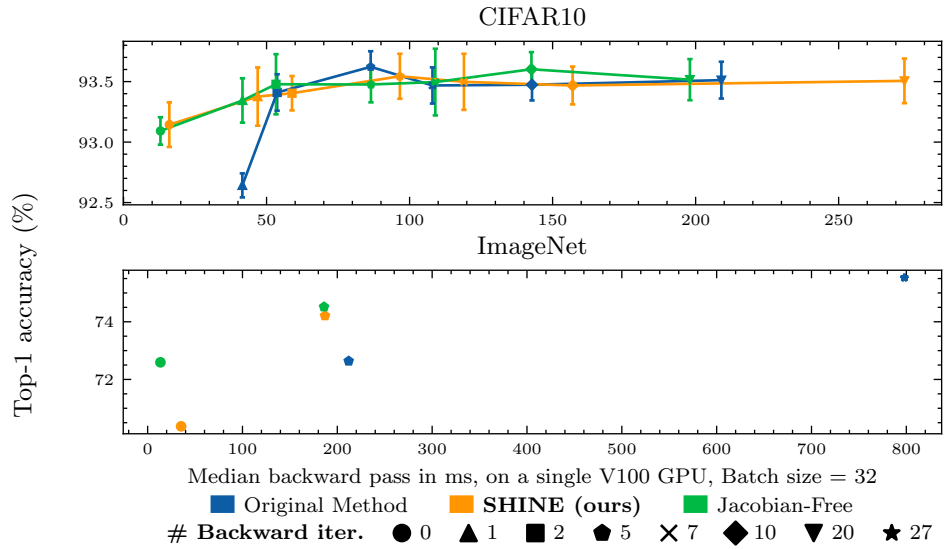

Figure 4: **DEQ:** Top-1 accuracy function of backward pass computational cost for the different methods considered to train DEQs, on CIFAR [27] and ImageNet [13].

presented in Figure 4 for the ImageNet dataset are given for SHINE with fallback. We verified that the fallback is barely used, by logging the proportion of samples that end up using it.

Despite the drop suffered at the beginning of the equilibrium training, SHINE in its refined version is able to perform on par with the Jacobian-Free method [16]. We also confirm the importance of choosing the right initialization to perform accelerated backpropagation, by showing that with a limited iterative inversion, the performance of the original method deteriorates. Finally, while the drop in performance for the accelerated methods is significant when applied in their vanilla version, we remind the reader that no fine-tuning was performed on the training hyperparameters, making those results encouraging (on par with architectures like ResNet-18 [22]).

4 Conclusion and Discussion

We introduced SHINE, a method that leverages the qN matrices from the forward pass to obtain an approximation of the gradient of the loss function, thereby reducing the time needed to compute this gradient. We showed that this method can be used on a wide range of applications going from bi-level optimization to small and large scale computer vision tasks. We found that both SHINE and the Jacobian-Free method reduce the required amount of time for the backward pass of implicit models, potentially lowering the barriers for training implicit models.

As those methods still suffer from a small performance drop, there is room for further improvement. In particular, a potential experimentation avenue would be to understand how to balance the efforts of the Adjoint Broyden method in order to come closer to guaranteeing the asymptotical correctness of the approximate inversion. On the theoretical side, this may involve the rate of convergence of the approximated gradient. It also seems desirable to develop a version of Theorem 4 in which convergence of (z_n) to $z^\star$ is not an assumption but rather follows from the assumptions, as achieved in Theorem 3. We have no doubt that the contraction assumption used for the Jacobian-Free method would allow to prove such a result, but expect that a significantly weaker assumption will suffice.

Broader Impacts

By reducing the computational burden needed to train DEQs, SHINE contributes to lowering the overall energy consumption required to train such models. It can also enable the training of such models on edge devices which was cited as an unclear development for these models by Bai et al. [4].

Another potential use of this technique is to be able to train more models and therefore make the research on DEQs more accessible or more thorough. This of course will favor both the potentially harmful or positive implementations of such models that were listed in the original Multiscale DEQ paper [4].

Acknowledgments

This work was granted access to the HPC resources of IDRIS under allocations 2021-AD011011172R1 and 2021-AD011011153 made by GENCI. We thank Pierre Ablin and Alexandre Gramfort for providing us the base Python code for L-BFGS.

References

[1] Libsvm datasets. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. Accessed: 2021-05-06.

[2] B. Amos and J. Zico Kolter. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *ICML*, 2017.

[3] S. Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. In *NeurIPS*, 2019.

[4] S. Bai, V. Koltun, and J. Z. Kolter. Multiscale deep equilibrium models. In *NeurIPS*, 2020.

[5] J. Bergstra and Y. Bengio. Random Search for Hyper-Parameter Optimization Yoshua Bengio. *Journal of Machine Learning Research*, 13:281–305, 2012.

[6] C. G. Broyden. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Mathematics of Computation*, 19(92):577–593, 1965.

[7] C. G. Broyden, J. E. jun. Dennis, and J. J. More. On the local and superlinear convergence of quasi-Newton methods. *Journal of the Institute of Mathematics and its Applications*, 12: 223–245, 1973.

[8] R. H. Byrd and J. Nocedal. A tool for the analysis of quasi-Newton methods with application to unconstrained minimization. *SIAM Journal on Numerical Analysis*, 26(3):727–739, 1989.

[9] R. H. Byrd, R. B. Schnabel, and G. A. Shultz. Parallel quasi-Newton methods for unconstrained optimization. *Mathematical Programming. Series A. Series B*, 42(2 (B)):273–306, 1988.

[10] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary differential equations. In *NeurIPS*, number NeurIPS, 2018.

[11] T. Chen, B. Xu, C. Zhang, and C. Guestrin. Training Deep Nets with Sublinear Memory Cost. Technical report, 2016.

[12] A. R. Conn, N. I. Gould, and P. L. Toint. Convergence of quasi-Newton matrices generated by the symmetric rank one update. *Mathematical Programming*, 50(1-3):177–195, 1991. ISSN 00255610. doi: 10.1007/BF01594934.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*. Institute of Electrical and Electronics Engineers (IEEE), 2009.

[14] D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[15] H. Fayez Khalfan, R. H. Byrd, and R. B. Schnabel. A theoretical and experimental study of the symmetric rank-one update. *SIAM Journal on Optimization*, 3(1):1–24, 1993.

[16] S. W. Fung, H. Heaton, Q. Li, D. Mckenzie, S. Osher, and W. Yin. Fixed Point Networks: Implicit Depth Models with Jacobian-Free Backprop. Technical report, 2021.

[17] J. D. Garrett and H.-H. Peng. garrettj403/SciencePlots, Feb. 2021. URL http://doi.org/10.5281/zenodo.4106649.

[18] D. Gilton, G. Ongie, and R. Willett. Deep Equilibrium Architectures for Inverse Problems in Imaging. Technical report, 2021.

[19] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse. The Reversible Residual Network: Backpropagation Without Storing Activations. In *NIPS*, 2017.

[20] R. M. Gower and P. Richtárik. Randomized quasi-Newton updates are linearly convergent matrix inversion algorithms. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1380–1409, 2017.

[21] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 9 2020.

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[23] H. Heaton, S. W. Fung, A. Gibali, and W. Yin. Feasibility-based Fixed Point Networks. Technical report, 2021.

[24] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science and Engineering*, 9(3):90–95, 2007.

[25] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *ICLR*. International Conference on Learning Representations, ICLR, 12 2015.

[26] S. G. Krantz and H. R. Parks. *The Implicit Function Theorem: History, Theory, and Applications*. Springer New York, 1 2013.

[27] A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, 2009.

[28] K. Lang. NewsWeeder: Learning to Filter Netnews. In *ICML*, pages 331–339. Elsevier, 1995.

[29] D. Li and M. Fukushima. A derivative-free line search and global convergence of Broyden-like method for nonlinear equations. *Optimization Methods & Software*, 13(3):181–201, 2000.

[30] D. Li, J. Zeng, and S. Zhou. Convergence of Broyden-Like Matrix. *Applied Mathematics Letter*, 11(5):35–37, 1998.

[31] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming, Series B*, 45(3):503–528, 1989.

[32] F. Mannel. On the convergence of the Broyden-like matrices. 2020.

[33] F. Mannel. Convergence properties of the Broyden-like method for mixed linear–nonlinear systems of equations. *Numerical Algorithms*, pages 1–29, 2021.

[34] F. Mannel. On the convergence of Broyden's method and some accelerated schemes for singular problems. 2021.

[35] J. More and J. Trangenstein. On the global convergence of Broyden's method. *Mathematics of Computation*, 30:523–540, 1976.

[36] J. Nocedal and S. Wright. Quasi-Newton Methods. In *Numerical Optimization*, pages 135–163. 2006.

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 12 2019.

[38] F. Pedregosa. Hyperparameter optimization with approximate gradient. *33rd International Conference on Machine Learning, ICML 2016*, 2:1150–1159, 2016.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[40] M. E. Sander, P. Ablin, M. Blondel, and G. Peyré. Momentum Residual Neural Networks. Technical report, 2021.

[41] S. Schlenkrich, A. Griewank, and A. Walther. On the local convergence of adjoint Broyden methods. *Mathematical Programming*, 121(2):221–247, 2010. ISSN 14364646. doi: 10.1007/s10107-008-0232-y.

[42] J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

[43] M. Telgarsky. Benefits of depth in neural networks. *Journal of Machine Learning Research*, 49 (June):1517–1539, 2 2016.

[44] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, I. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, A. Vijaykumar, A. P. Bardelli, A. Rothberg, A. Hilboll, A. Kloeckner, A. Scopatz, A. Lee, A. Rokem, C. N. Woods, C. Fulton, C. Masson, C. Häggström, C. Fitzgerald, D. A. Nicholson, D. R. Hagen, D. V. Pasechnik, E. Olivetti, E. Martin, E. Wieser, F. Silva, F. Lenders, F. Wilhelm, G. Young, G. A. Price, G. L. Ingold, G. E. Allen, G. R. Lee, H. Audren, I. Probst, J. P. Dietrich, J. Silterra, J. T. Webber, J. Slavič, J. Nothman, J. Buchner, J. Kulick, J. L. Schönberger, J. V. de Miranda Cardoso, J. Reimer, J. Harrington, J. L. C. Rodríguez, J. Nunez-Iglesias, J. Kuczynski, K. Tritz, M. Thoma, M. Newville, M. Kümmerer, M. Bolingbroke, M. Tartre, M. Pak, N. J. Smith, N. Nowaczyk, N. Shebanov, O. Pavlyk, P. A. Brodtkorb, P. Lee, R. T. McGibbon, R. Feldbauer, S. Lewis, S. Tygier, S. Sievert, S. Vigna, S. Peterson, S. More, T. Pudlik, T. Oshima, T. J. Pingel, T. P. Robitaille, T. Spura, T. R. Jones, T. Cera, T. Leslie, T. Zito, T. Krauss, U. Upadhyay, Y. O. Halchenko, and Y. Vázquez-Baeza. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, 3 2020.

A OPA algorithm

Algorithm LBFGS: (Limited memory) BFGS method with OPA

Input: initial guess (z_0, B_0^{-1}), where B_0^{-1} is symmetric and positive definite, tolerance $\epsilon > 0$, frequency of additional updates $M \in \mathbb{N}$, memory limit $L \in \mathbb{N} \cup \{\infty\}$, (t_n) a null sequence of positive numbers with $\sum_n t_n < \infty$

Let $F := \nabla_z g_\theta$

for $n = 0, 1, 2, \ldots$ **do**

  if $\|F(z_n)\| \leq \epsilon$ **then** let $z^\star := z_n$ and let $B := B_n$; STOP

  if $(n \bmod M) = 0$ **then**

    let $e_n := t_n B_n^{-1} \frac{\partial g_\theta}{\partial \theta}\Big|_{z_n}$, $\hat{y}_n := F(z_n + e_n) - F(z_n)$ and $\hat{r}_n := (e_n)^T \hat{y}_n$

    if $\hat{r}_n > 0$ **then**

      let $\hat{a}_n := e_n - B_n^{-1}\hat{y}_n$ and let

$$\hat{B}_n^{-1} := B_n^{-1} + \frac{\hat{a}_n (e_n)^T + e_n (\hat{a}_n)^T}{\hat{r}_n} - \frac{(\hat{a}_n)^T \hat{y}_n}{(\hat{r}_n)^2} e_n (e_n)^T$$

    else let $\hat{B}_n^{-1} := B_n^{-1}$

  Let $B_n^{-1} := \hat{B}_n^{-1}$

  if $n \geq L$ **then** remove update $n - L$ from B_n^{-1}

  Let $p_n := -B_n^{-1} F(z_n)$

  Obtain α_n via line-search and let $s_n := \alpha_n p_n$

  Let $z_{n+1} := z_n + s_n$, $y_n := F(z_{n+1}) - F(z_n)$ and $r_n := (s_n)^T y_n$

  if $r_n > 0$ **then**

    let $a_n := s_n - B_n^{-1} y_n$ and let

$$B_{n+1}^{-1} := B_n^{-1} + \frac{a_n (s_n)^T + s_n (a_n)^T}{r_n} - \frac{(a_n)^T y_n}{(r_n)^2} s_n (s_n)^T$$

  else let $B_{n+1}^{-1} := B_n^{-1}$

  if $n \geq L$ **then** remove update $n - L$ from B_{n+1}^{-1}

Output: $z^\star$, B

Remark. *A possible choice for (t_n) is to use an arbitrary $t_0 > 0$ and $t_n := \|s_{n-1}\|$ for $n \geq 1$.*

B Proofs of SHINE convergence

To facilitate reading, we restate the results before proving them.

B.1 Convergence using ULI

Theorem 2 (Convergence of SHINE to the Hypergradient using ULI)**.** *Let us denote $p_\theta^{(n)}$, the SHINE direction for iterate n in Algorithm 1 with $b =$ `true`. Under Assumptions 1 and 2, for a given parameter θ, (z_n) converges q-superlinearly to $z^\star$ and*

$$\lim_{n \to \infty} p_\theta^{(n)} = \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{z^\star}$$

Proof. Under Assumptions 1 and 2, [35, Theorem 5.7] shows that B_n satisfies

$$\lim_{n \to \infty} B_n = J_{g_\theta}(z^\star)$$

The inversion operator is continuous in the space of invertible matrices, so we have:

$$\lim_{n \to \infty} B_n^{-1} = J_{g_\theta}(z^\star)^{-1}$$

Because $\nabla_z \mathcal{L}$ and $\frac{\partial g_\theta}{\partial \theta}$ are continuous at $z^\star$ by Assumption 2 (iii), we also have thanks to Assumption 2 (i):

$$\lim_{n\to\infty} \nabla_z \mathcal{L}(z_n) = \nabla_z \mathcal{L}(z^\star) \qquad \text{and} \qquad \lim_{n\to\infty} \frac{\partial g_\theta}{\partial \theta}\Big|_{z_n} = \frac{\partial g_\theta}{\partial \theta}\Big|_{z^\star}$$

By continuity we then deduce that, as claimed,

$$\lim_{n\to\infty} p_\theta^{(n)} = \lim_{n\to\infty} \nabla_z \mathcal{L}(z_n) B_n^{-1} \frac{\partial g_\theta}{\partial \theta}(z_n) = \nabla_z \mathcal{L}(z^\star) J_{g_\theta}(z^\star)^{-1} \frac{\partial g_\theta}{\partial \theta}\Big|_{z^\star} = \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{z^\star} \qquad \square$$

B.2 Convergence for BFGS with OPA

Assumption 5 (Extended Assumptions for BFGS)**.** *Let $g_\theta(z) = \nabla_z r_\theta(z)$ for some C^2 function $r_\theta : \mathbb{R}^d \to \mathbb{R}$. Consider Algorithm 1 with $b =$ `false` and suppose that*

1. *the set $\Omega := \{z \in \mathbb{R}^d : r_\theta(z) \leq r_\theta(z_0)\}$ is convex;*
2. *r_θ is strongly convex in an open superset of Ω (this implies that r_θ has a unique global minimizer $z^\star$) and has a Lipschitz continuous Hessian near $z^\star$;*
3. *there are positive constants η_1, η_2 such that the line search used in the algorithm ensures that for each $n \geq 0$ either*

$$r_\theta(z_{n+1}) \leq r_\theta(z_n) - \eta_1 \left[\frac{\nabla r_\theta(z_n)^T p_n}{\|p_n\|}\right]^2 \quad \text{or} \quad r_\theta(z_{n+1}) \leq r_\theta(z_n) + \eta_2 \nabla r_\theta(z_n)^T p_n$$

   is satisfied;

4. *the line search has the property that $\alpha_n = 1$ will be used if both*

$$\frac{\|(B_n - J_{g_\theta}(z_n))s_n\|}{\|s_n\|} \qquad \text{and} \qquad \|z_n - z^\star\|$$

   are sufficiently small.

Remark. *The requirements 3. and 4. on the line search are, for instance, satisfied under the well-known Wolfe conditions, see [9, section 3] for further comments.*

Theorem 3 (Convergence of SHINE to the Hypergradient for BFGS with OPA)**.** *Let us consider $p_\theta^{(n)}$, the SHINE direction for iterate n in Algorithm 1 that is enriched by extra updates in the direction e_n defined in (4). Under Assumptions 2 (ii-iii) and 3, for a given parameter θ, we have the following: Algorithm 1, for any symmetric and positive definite matrix B_0, generates a sequence (z_n) that converges q-superlinearly to $z^\star$, and there holds*

$$\lim_{n\to\infty} p_\theta^{(n)} = \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{z^\star} \tag{5}$$

Proof. The proof is divided into four steps. The first step is to establish the q-superlinear convergence of (z_n) to $z^\star$. Denoting by $N_e \subset \{0, M, 2M, \ldots\}$ the set of indices of extra updates that are actually applied, the second step consists of showing

$$\lim_{N_e \ni n\to\infty} (B_n - J_{g_\theta}(z^\star)) \frac{e_n}{\|e_n\|} = 0, \tag{8}$$

where, in this proof, B_n always represents the matrix from Algorithm LBFGS *before* the update in the direction e_n is applied, i.e., the matrix whose inverse appears in the definition of e_n. The third step is to prove that (8) implies the desired convergence (5) of the SHINE direction if the limit $n \to \infty$ is taken on the subsequence with indices $n \in N_e$. The fourth step is then to argue that this convergence holds not only on this subsequence, but on the entire sequence.
It is easy to check that instead of updating B_n^{-1} we can also obtain the sequence (B_n) by directly updating B_n according to

$$B_{n+1} = B_n + \frac{y_n y_n^T}{y_n^T s_n} - \frac{B_n s_n (B_n s_n)^T}{s_n^T B_n s_n}$$

for the usual update (skipping the update if $y_n^T s_n \leq 0$), respectively,

$$\hat{B}_n = B_n + \frac{\hat{y}_n \hat{y}_n^T}{\hat{y}_n^T e_n} - \frac{B_n e_n (B_n e_n)^T}{e_n^T B_n e_n}$$

for the extra update (skipping the update if $\hat{y}_n^T e_n \leq 0$). Here, the quantities y_n, $\hat{y}_n$ and e_n are defined as in Algorithm LBFGS. We can now argue essentially as in the proof of [9, Theorem 3.1] to show that (z_n) converges q-superlinearly to $z^\star$. As part of that proof we obtain that

$$\lim_{n\to\infty} \left(B_n - J_{g_\theta}(z^\star)\right) \frac{s_n}{\|s_n\|} = 0 \tag{9}$$

and that a fixed fraction of the extra updates is actually applied, i.e., $\hat{B}_n \neq B_n$ for at least $\lceil 0.5Q \rceil$ of the indices $n = 0, M, 2M, \ldots, QM$ for any $Q \in \mathbb{N}$ (namely for all $n \in N_e$ satisfying $n \leq QM$). This allows to apply [8, Theorem 3.2], which yields

$$\lim_{N_e \ni n\to\infty} \left(B_n - J_{g_\theta}(z^\star)\right) \frac{e_n}{\|e_n\|} = 0.$$

For the third step, we abbreviate $v_n := \frac{\partial g_\theta}{\partial \theta}|_{z_n}$. From the definition of e_n and (8) we infer that

$$0 = \lim_{N_e \ni n\to\infty} \left(B_n - J_{g_\theta}(z^\star)\right) \frac{e_n}{\|e_n\|} = \lim_{N_e \ni n\to\infty} \left(I - J_{g_\theta}(z^\star) B_n^{-1}\right) \frac{v_n}{\|B_n^{-1} v_n\|}.$$

After multiplication with $J_{g_\theta}(z^\star)^{-1}$ this entails

$$\lim_{N_e \ni n\to\infty} \left(J_{g_\theta}(z^\star)^{-1} - B_n^{-1}\right) \frac{v_n}{\|B_n^{-1} v_n\|} = 0,$$

which shows that

$$\lim_{N_e \ni n\to\infty} B_n^{-1} v_n = \lim_{N_e \ni n\to\infty} J_{g_\theta}(z^\star)^{-1} v_n = J_{g_\theta}(z^\star)^{-1} \frac{\partial g_\theta}{\partial \theta}|_{z^\star}$$

by Assumption 2 (iii). Using Assumption 2 (iii) again it follows that

$$\lim_{N_e \ni n\to\infty} p_\theta^{(n)} = \lim_{N_e \ni n\to\infty} \nabla_z \mathcal{L}(z_n) B_n^{-1} \frac{\partial g_\theta}{\partial \theta}\Big|_{z_n} = \nabla_z \mathcal{L}(z^\star) J_{g_\theta}(z^\star)^{-1} \frac{\partial g_\theta}{\partial \theta}\Big|_{z^\star} = \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{z^\star},$$

concluding the third step. To infer that (5) holds, it suffices to argue that $\lim_{N_e \ni n\to\infty} \|B_{j_n} - B_n\| = 0$ for any sequence $(j_n)_{n\in N_e}$ such that $\sup_{n\in N_e} |n - j_n| < \infty$ and such that $\{j_n, j_n + 1, \ldots, n\} \cap N_e = \{n\}$ for all $n \in N_e$. Indeed, since for $C := \max\{\sup_n \|B_n\|, \sup_n \|B_n^{-1}\|\}$, which is finite by [8, Theorem 3.2], there holds

$$(B_n) \subset \left\{ A \in \mathbb{R}^{d\times d} : \ A^{-1} \text{ exists }, \ \|A\| \leq C, \ \|A^{-1}\| \leq C \right\}$$

and the set on the right-hand side of the inclusion is compact by the Banach lemma, inversion is a *uniformly* continuous operation on this set, hence $\lim_{N_e \ni n\to\infty} \|B_{j_n}^{-1} - B_n^{-1}\| = 0$, so continuity yields

$$\lim_{N_e \ni n\to\infty} \|p_\theta^{(n)} - p_\theta^{(j_n)}\| = 0,$$

and therefore

$$\lim_{N_e \ni n\to\infty} p_\theta^{(j_n)} = \lim_{N_e \ni n\to\infty} p_\theta^{(n)} = \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{z^\star}$$

by the third step, establishing the claim.

It remains to show the validity of $\lim_{N_e \ni n\to\infty} \|B_{j_n} - B_n\| = 0$ for any sequence $(j_n)_{n\in N_e}$ such that $\sup_{n\in N_e} |n - j_n| < \infty$ and $\{j_n, j_n + 1, \ldots, n\} \cap N_e = \{n\}$ for all $n \in N_e$. Since at least every second extra update is actually carried out, the condition on the empty intersection means that $n - j_n < 2M$ for all $n \in N_e$, which implies $\sup_{n\in N_e} |j_n - n| < \infty$. Now let $(j_n)_{n\in N_e}$ be any such sequence. Then the difference $B_n - B_{j_n}$ is a sum of at most $2M - 1$ BFGS updates in search directions, but contains no extra updates. Hence, the secant conditions $B_{n-l} s_{n-1-l} = y_{n-1-l}$, $l \in \{0, 1, \ldots, n - j_n\}$, are satisfied, allowing us to deduce

$$\begin{aligned} \|B_{n-l} - B_{n-l-1}\| &= \frac{\|(B_{n-l} - B_{n-l-1}) s_{n-1-l}\|}{\|s_{n-1-l}\|} \\ &\leq \frac{\|y_{n-1-l} - J_{g_\theta}(z^\star) s_{n-1-l}\|}{\|s_{n-1-l}\|} + \frac{\|(B_{n-1-l} - J_{g_\theta}(z^\star)) s_{n-1-l}\|}{\|s_{n-1-l}\|} \end{aligned}$$

for all $l \in \{0, 1, \ldots, n - j_n - 1\}$. Using (9) this implies $\lim_{N_e \ni n\to\infty} \|B_n - B_{j_n}\| = 0$, which finishes the fourth step and thus concludes the proof. □

B.3 Convergence for Adjoint Broyden with OPA

Theorem 4 (Convergence of SHINE to the Hypergradient for Adjoint Broyden with OPA)**.** *Let us consider $p_\theta^{(n)}$, the SHINE direction for iterate n in Algorithm 1 with the Adjoint Broyden secant condition (6) and extra update in the direction v_n defined in (7). Under Assumptions 2 and 4, for a given parameter θ, we have q-superlinear convergence of (z_n) to $z^\star$ and*

$$\lim_{n\to\infty} p_\theta^{(n)} = \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{z^\star}$$

Proof. Due to Assumption 2, the superlinear convergence of (z_n) follows from [41, Theorem 2]. The proof of the remaining claim is divided into two cases.

Case 1: Suppose that $\nabla_z \mathcal{L}(z^\star) = 0$. By continuity this implies $\lim_{n\to\infty} \nabla_z \mathcal{L}(z_n) = 0$. Since the sequence $(B_n^{-1} \frac{\partial g_\theta}{\partial \theta}|_{z_n})$ is bounded by Assumption 4, it follows that

$$\lim_{n\to\infty} p_\theta^{(n)} = \lim_{n\to\infty} \nabla_z \mathcal{L}(z_n) B_n^{-1} \frac{\partial g_\theta}{\partial \theta}\Big|_{z_n} = 0 = \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{z^\star},$$

as claimed.

Case 2: Suppose that $\nabla_z \mathcal{L}(z^\star) \neq 0$. By continuity this implies $\nabla_z \mathcal{L}(z_n) \neq 0$ for all sufficiently large $n \in \mathbb{N}$. Let us denote by $N_e \subset \mathbb{N}$ the set of indices of extra updates. We stress that this set is infinite since, by construction, every M-th update is an extra update. We have $v_n \neq 0$ for all sufficiently large $n \in N_e$, hence [41, Lemma 3] yields

$$\lim_{N_e \ni n\to\infty} \frac{\|\nabla_z \mathcal{L}(z_n)(I - B_n^{-1} J_{g_\theta}(z^\star))\|}{\|(\nabla_z \mathcal{L}(z_n) B_n^{-1})^T\|} = \lim_{N_e \ni n\to\infty} \frac{\|(v_n)^T (B_n - J_{g_\theta}(z^\star))\|}{\|v_n\|} = 0.$$

This implies

$$\lim_{N_e \ni n\to\infty} \frac{\|\nabla_z \mathcal{L}(z_n)(J_{g_\theta}(z^\star)^{-1} - B_n^{-1})\|}{\|\nabla_z \mathcal{L}(z_n) B_n^{-1}\|} = 0,$$

thus necessarily

$$\lim_{N_e \ni n\to\infty} \|\nabla_z \mathcal{L}(z_n)(J_{g_\theta}(z^\star)^{-1} - B_n^{-1})\| = 0.$$

Since $\lim_{N_e \ni n\to\infty} \nabla_z \mathcal{L}(z_n) J_{g_\theta}(z^\star)^{-1} = \nabla_z \mathcal{L}(z^\star) J_{g_\theta}(z^\star)^{-1}$ by continuity, we find

$$\lim_{N_e \ni n\to\infty} \nabla_z \mathcal{L}(z_n) B_n^{-1} = \nabla_z \mathcal{L}(z^\star) J_{g_\theta}(z^\star)^{-1},$$

whence

$$\lim_{N_e \ni n\to\infty} p_\theta^{(n)} = \lim_{N_e \ni n\to\infty} \nabla_z \mathcal{L}(z_n) B_n^{-1} \frac{\partial g_\theta}{\partial \theta}\Big|_{z_n} = \nabla_z \mathcal{L}(z^\star) J_{g_\theta}(z^\star)^{-1} \frac{\partial g_\theta}{\partial \theta}\Big|_{z^\star} = \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{z^\star}, \quad (10)$$

where we have used continuity again. To prove that these limits hold not only for $N_e \ni n \to \infty$ but in fact for all $\mathbb{N} \ni n \to \infty$, we establish, as intermediate claim, that for any fixed $m \in \mathbb{N}$ we have $\lim_{n\to\infty} \|B_{n+m} - B_n\| = 0$. Note that this claim is equivalent to $\lim_{n\to\infty} \|B_{n+1} - B_n\| = 0$. Denoting by $L \geq 0$ the Lipschitz constant of J_{g_θ} near $z^\star$, we find

$$\begin{aligned}\|B_{n+1} - B_n\| = \frac{\|v_n v_n^T [J_{g_\theta}(z_{n+1}) - B_n]\|}{\|v_n\|^2} &\leq \|J_{g_\theta}(z_{n+1}) - J_{g_\theta}(z^\star)\| + \frac{\|[J_{g_\theta}(z^\star) - B_n]^T v_n\|}{\|v_n\|} \\ &\leq L\|z_{n+1} - z^\star\| + \frac{\|E_n^T v_n\|}{\|v_n\|}.\end{aligned}$$

Both terms on the right-hand side go to zero as n goes to infinity: the first one due to $\lim_{n\to\infty} z_n = z^\star$ and the second one since $\lim_{n\to\infty} \frac{\|E_n^T v_n\|}{\|v_n\|} = 0$ by [41, Lemma 3]. This shows that $\lim_{n\to\infty} \|B_{n+1} - B_n\| = 0$, which concludes the proof of the intermediate claim.

From $\lim_{n\to\infty} \|B_{n+m} - B_n\| = 0$ for any fixed $m \in \mathbb{N}$ it follows that for any sequence $(j_n) \subset \mathbb{N}$ with $\sup_n |j_n - n| < \infty$ there holds $\lim_{n\to\infty} \|B_{j_n} - B_n\| = 0$. This implies for any such sequence (j_n) the limit $\lim_{n\to\infty} \|B_{j_n}^{-1} - B_n^{-1}\| = 0$. To establish this, note that for $C := \max\{\sup_n \|B_n\|, \sup_n \|B_n^{-1}\|\}$, which is finite by Assumption 4 and the combination of the bounded deterioration principle [41, Lemma 2] with Assumption 2 (i), the set

$$\left\{A \in \mathbb{R}^{d\times d} : \ A^{-1} \text{ exists }, \ \|A\| \leq C, \ \|A^{-1}\| \leq C\right\}$$

includes the sequence (B_n) and is compact by the Banach lemma, so inversion is a *uniformly* continuous operation on this set.
Now let us construct a sequence $(j_n) \subset N_e$ by defining, for every $n \in \mathbb{N}$, $j_n := \arg\min_{m \in N_e} |n-m|$. That is, for every n, j_n denotes the member of N_e with the smallest distance to n. It is clear that $|n - j_n| \leq M - 1$ for all n, hence $\lim_{n\to\infty} \|B_{j_n}^{-1} - B_n^{-1}\| = 0$. Using this and, again, continuity it is easy to see that

$$\lim_{n\to\infty} \|p_\theta^{(n)} - p_\theta^{(j_n)}\| = 0,$$

which implies by (10) that

$$\lim_{n\to\infty} p_\theta^{(n)} = \lim_{n\to\infty} p_\theta^{(j_n)} = \lim_{N_e \ni n\to\infty} p_\theta^{(n)} = \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{z^\star},$$

thereby establishing the claim. □

Remark. *An inspection of the proof reveals that if B_n is never updated in the direction z_n, but only updated in the direction v_n defined in (7), then Assumption 4 can be replaced by the significantly weaker assumption that the sequence $(B_n^{-1} \frac{\partial g_\theta}{\partial \theta}|_{z_n})$ is bounded. The price to pay is that the convergence rate of (z_n) to $z^\star$ will be slower (q-linear instead of q-superlinear) since the updates in the direction z_n are critical for ensuring fast convergence of (z_n) to $z^\star$.*

C Logistic Regression Hyperparameters

For both datasets we split the data randomly (with a different seed for each run) between training-validation-test, with the following proportions: 90%-5%-5%. The hyperparameters are the same as in the original HOAG work [38], except:

- We use a memory limitation of 30 updates (not grid-searched) for accelerated methods (Jacobian-Free and SHINE), compared to 10 for the original method. This is because the approximation should be better using more updates. We verified that using 30 updates for the original method does not improve the convergence speed. That number is 60 for OPA.
- We use a smaller exponential decrease of 0.78 (not grid-searched) for the accelerated methods, compared to 0.99 for the original method. This is because in the very long run, the approximation can cause oscillations.

We also use the same setting as Pedregosa [38] for the Grid and Random Search. Finally, we highlight that warm restart is used for both the inner problem and the Hessian inversion in the direction of the gradient.

OPA inversion experiments For the OPA experiments, we used a memory limitation of 60, and a tolerance of 10^{-6}. The OPA update is done every 5 regular updates.

D DEQ training details

The training details are the same as the original Multiscale DEQ paper [4]: all the hyperparameters are kept the same and not fine-tuned, and the data split is the same. We recall here some important aspects. For both datasets, the network is first trained in an unrolled weight-tied fashion for a few epochs in order to stabilize the training.
We also underline that the DEQ models, in addition to having a fixed-point-defining sub-network, also have a classification and a projection head.

D.1 CIFAR

Adam optimizer [25] is used with a 10^{-3} start learning rate, and a cosine annealing schedule.

D.2 ImageNet

The Stochastic Gradient Descent optimizer is used with a 5×10^{-2} start learning rate, and a cosine annealing schedule.
The images are downsampled 2 times before being fed to the fixed-point defining sub-network.

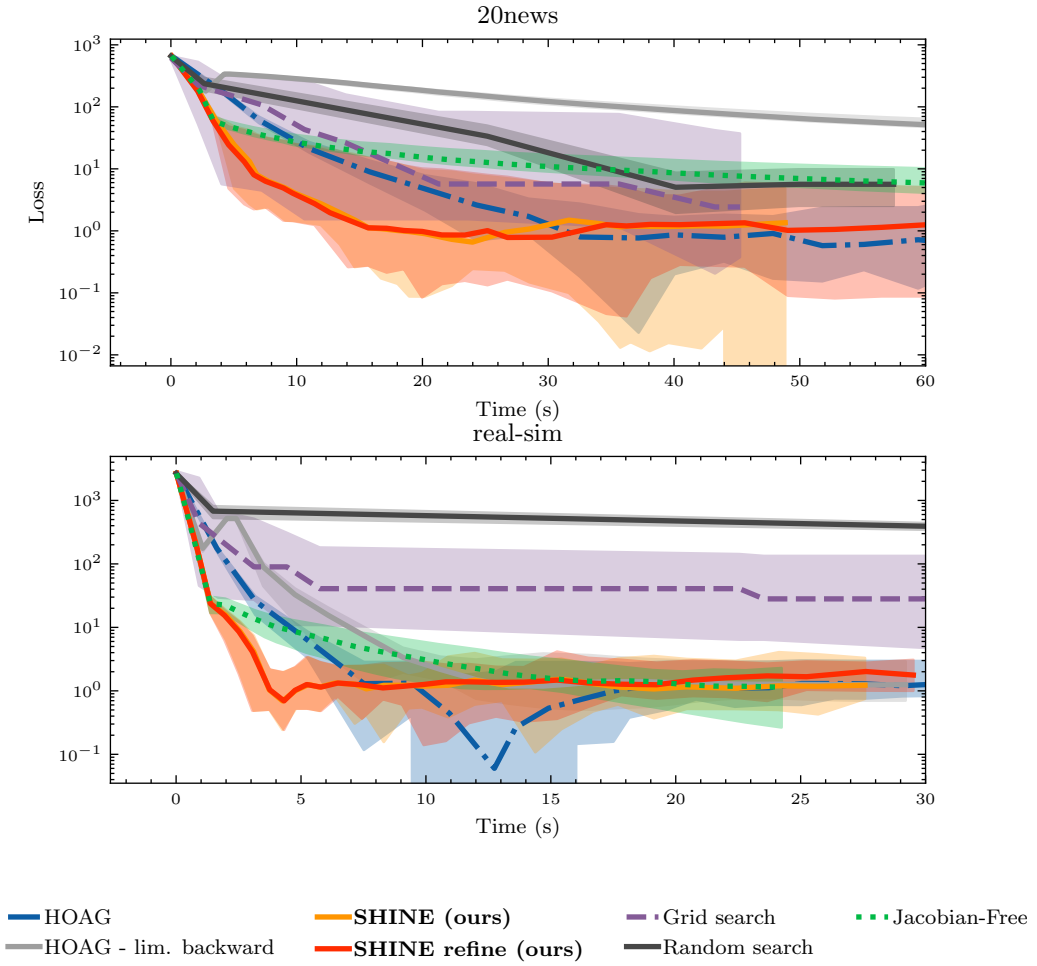

Figure E.1: **Bi-level optimization:** Convergence of different hyperparameter optimization methods on the $L2$-regularized logistic regression problem for two datasets (20news [28] and real-sim [1]) on held-out test data.

E Additional results

E.1 Bi-level optimization extended

In order to make sure that SHINE was indeed improving over HOAG [38], we also looked at the results obtained when performing an inversion with a precision lower than that prescribed by Pedregosa [38] originally (i.e. truncating the iterative inversion). These results, also complemented with Random Search [5], can be seen in Figure E.1. They confirm that the advantage provided by SHINE cannot be retrieved with a looser tolerance on the inversion.

E.2 Contractivity assumption

One of the main limiting assumptions in the original Jacobian-Free method work [16], is the contractivity assumption. We showed here that it was not important to enforce this in order to achieve excellent results, but one can wonder whether this assumption is not met in practice thanks to the unrolled pretraining of DEQs. We looked at the contractivity of the fixed-point defining sub-network empirically by using the power-method applied to a non-linear function, in the CIFAR setting. The results, summarized in Table E.1, show that the fixed-point defining sub-network is not contractive at all.

Table E.1: Non-linear spectral radius obtained by the power method for the fixed-point defining sub-network for the 3 different methods.

Method	**Non-linear spectral radius**
Original	230.5
Jacobian-Free	193.7
SHINE	234.2

Table E.2: The time required for each method on the different datasets during the equilibrium training. For the forward and backward passes, the time is measured offline, for a single batch of 32 samples, with a single GPU, using the median to avoid outliers. This time is given in milliseconds. For the epochs, the time is measured by taking an average of the 6 first epochs, and given in hours-minutes for Imagenet and minutes-seconds for CIFAR. The epoch time for SHINE without improvement on Imagenet is not given because it never reaches the 26 forward steps: the implicit depth is too short. Fallback is not used for CIFAR. Numbers in parenthesis indicate the number of inversion steps for the refined versions.

Dataset Name	**CIFAR [27]**			**ImageNet [13]**		
Method Name	**Forward**	**Backward**	**Epoch**	**Forward**	**Backward**	**Epoch**
Original [4]	256	210	4min40	644	798	3h38
Jacobian-Free [16]	249	12.9	3min10	621	13.5	2h02
SHINE Fallback (ours)	218	16.0	3min20	622	35.3	2h13
SHINE Fallback refine (5, ours)	272	96.6	3min50	622	212	2h44
Jacobian-Free refine (5)	260	86.5	3min40	620	186	2h43
Original limited backprop	281	86.4	3min50	653	187	2h40

E.3 Time gains

Because the total training time is not only driven by backward pass but also by the forward pass and the evaluation, we show for completeness in Table E.2 the time gains for the different acceleration methods for the overall epoch. We do not report in this table the time taken for pre-training which is equivalent across all methods, and is not something on which SHINE has an impact. It is clear in Table E.2 that accelerated methods can have a significant impact on the training of DEQs because we see that half the time of the total pass is spent on the backward pass (more on ImageNet [13]). We also notice that while SHINE has a slightly slower backward pass than the Jacobian-Free method [16], the difference is negligible when compared to the total pass computational cost.

E.4 DEQ OPA results

We can clearly see in Figure E.2 that in the case of DEQs, OPA also significantly improves the inversion over the other accelerated methods. We also see that the improvements of SHINE over the Jacobian-Free method without OPA are marginal.

Because the inversion is so good, we would expect that the performance of SHINE with OPA would be on par with the original method's. However, this is not what we see in the results presented in Table E.3. Indeed, OPA does improve on SHINE with only Adjoint Broyden, but it does not outperform SHINE done with Broyden.

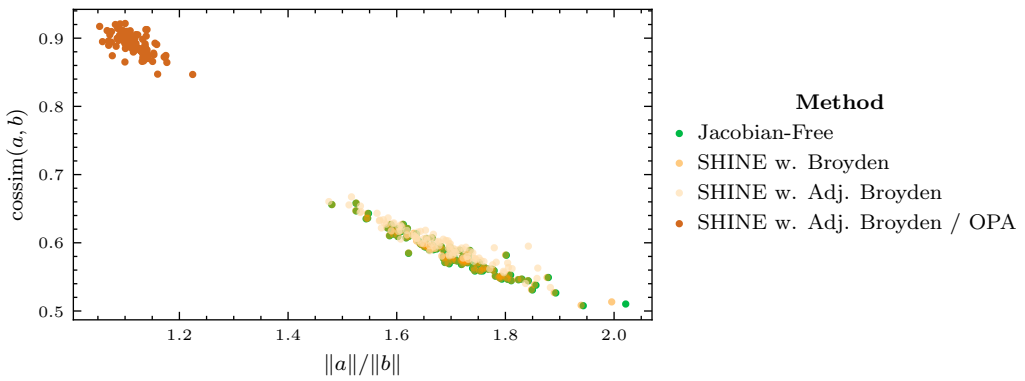

Figure E.2: **Quality of the inversion using OPA in DEQs :** Ratio of the inverse approximation over the exact inverse function of the cosine similarity between the inverse approximation $b = \nabla_z \mathcal{L}(z^\star) B_n^{-1}$ and the exact inverse $a = \nabla_z \mathcal{L}(z^\star) J_{g_\theta}(z^\star)^{-1}$ for different methods. For OPA, the extra update frequency is 5. 100 runs were performed with different batches.

Table E.3: **CIFAR DEQ OPA results :** Top-1 accuracy of different methods on the CIFAR dataset, and epoch mean time.

Methode name	**Top-1 Accuracy (%)**	**Epoch mean time**
Original	93.51	4min40
Jacobian-Free	93.09	3min10
SHINE (Broyden)	93.14	3min20
SHINE (Adj. Broyden)	92.89	4min
SHINE (Adj. Broyden/OPA)	93.04	4min40