SHINE: SHARING THE INVERSE ESTIMATE FROM THE FORWARD PASS FOR BI-LEVEL OPTIMIZATION AND IMPLICIT MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

In recent years, implicit deep learning has emerged as a method to increase the effective 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 (Chen et al., 2018), OptNets (Amos and Zico Kolter, 2017) or Deep Equilibrium models (DEQs) (Bai et al., 2019; 2020) have recently emerged as a way to train deep models with infinite effective depth without the associated memory cost. Indeed, while it has been observed that the performance of deep learning models increases with their depth (Telgarsky, 2016), 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 neural networks (Gomez et al., 2017; Sander et al., 2021) or gradient checkpointing (Chen et al., 2016) 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 (Bai et al., 2019). These models have been successfully applied to large-scale tasks such as language modeling (Bai et al., 2019), computer vision tasks (Bai et al., 2020) and inverse problems (Gilton et al., 2021; Heaton et al., 2021).

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

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

We will refer to the root finding problem $g_\theta(z^\star) = 0$ 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., $g_\theta(z^\star) = z^\star - f_\theta(z^\star) = 0$.[1] 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. In

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

practice, DEQs' forward pass is not computed by applying successively the function but usually relies on quasi-Newton (qN) algorithms, such as Broyden's method (Broyden, 1965), which approximates efficiently the Jacobian matrix $\frac{\partial g_\theta}{\partial z}$ and its inverse for root-finding.

To compute DEQs' gradient efficiently and avoid high memory cost, one does not rely on back-propagation but uses the implicit function theorem (Krantz and Parks, 2013) which gives an analytical expression of the Jacobian of $z^\star$ with respect to θ, $\frac{\partial z^\star}{\partial \theta}$. While this method 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 (Bai et al., 2020) (see also the break down of the computational effort in Section E.3).

Moreover, the formulation (1) allows us to also consider general bi-level problems such as hyper-parameter optimization under the same framework. For instance, hyperparameter optimization for Logistic Regression (LR) can be written as

$$\min_\theta \mathcal{L}_{\text{val}}(z^*) \quad \text{subject to} \quad z^* = \min_z r_\theta(z) \triangleq \mathcal{L}_{\text{train}}(z) + \theta \|z\|_2^2 \tag{2}$$

where $\mathcal{L}_{\text{train}}$ and $\mathcal{L}_{\text{val}}$ correspond to the training and validation losses from the LR problem (Pedregosa, 2016). Here, z corresponds to the weights of the LR model while θ is the regularisation parameter. As the training loss is smooth and convex, the inner problem can be written as in (1) with $g_\theta = \nabla_z r_\theta$ to fit (1). Similarly to DEQ, the inner problem is often solved using qN methods, which approximate the inverse of the Hessian in the direction of the steps, such as the LBFGS algorithm (Liu and Nocedal, 1989), and the gradient computation suffers from the same drawback as it is also obtained using the implicit function theorem. Lorraine et al. (2020) showed extensively how to tackle this problem using different methods to compute the hypergradient and for different tasks.

With the increasing popularity of DEQs and the ubiquity of bi-level problems in machine learning, a core question is how to reduce the computational cost of the resolution of (1). This would make these methods more accessible for practitioners and reduce the associated energy cost. In this work, we propose to exploit the estimates of the (inverse of the) Jacobian/Hessian produced by qN methods in the hypergradient computation. Moreover, we also propose 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. 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.

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. Our work is the first to consider modifying the inner problem resolution in order to account for the bi-level structure of the optimization 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 (2017) used sketching to design appropriate extra secant conditions in order to obtain guarantees of uniform convergence towards the inverse of the Jacobian. The novelty in our work is that we integrate additional update to yield the inverse in a specific direction, which is substantially cheaper than computing the inverse. A concurrent work by Fung et al. (2021) 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 proven that the resulting approximation is a descent direction and the authors show good empirical performances for small scale problems.

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. This allows us to 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 state-of-the-art methods.
- 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 extend the empirical evaluation of the Jacobian-Free method to large scale multiscale DEQs and show that it performs well in this setting. We also show that it is not suitable for more general bi-level problems.
- We propose and evaluate a natural refinement strategy for approximate Jacobian inversion methods (both SHINE and Jacobian-Free) that allows a trade-off between computational cost and performances.

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 (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. (2019, Theorem 1) show the following theorem, which is based on implicit differentiation (Krantz and Parks, 2013):

Theorem 1 (Hypergradient (Bai et al., 2019; Krantz and Parks, 2013))**.** *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*

$$\left.\frac{\partial \mathcal{L}}{\partial \theta}\right|_{z^\star} = \nabla_z \mathcal{L}(z^\star)^\top J_{g_\theta}(z^\star)^{-1} \left.\frac{\partial g_\theta}{\partial \theta}\right|_{z^\star}. \tag{3}$$

In practice, we use an algorithm to approximate $z^\star$, and Theorem 1 gives a plug-in formula for the backward pass. Note that this formula is independent of the algorithm chosen to compute $z^\star$. 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 $\nabla_z \mathcal{L}(z^\star)$.

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**
  $p_n = -B_n^{-1} g_\theta(z_n)$, $z_{n+1} = z_n + \alpha_n p_n$
  `// $\alpha_n$ 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
$z^\star = z_n$, $B = B_n$

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 (2016) used L-BFGS (Liu and Nocedal, 1989), while for Deep Equilibrium Models, Bai et al. (2019) used Broyden's method (Broyden, 1965), later adapted to the multi-scale case in a limited-memory version (Bai et al., 2020).

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 (3), i.e. to share the inverse estimate between the forward and the backward passes. This gives the approximate Hypergradient

$$p_\theta = \nabla_z \mathcal{L}(z^\star) B^{-1} \frac{\partial g_\theta}{\partial \theta}\Big|_{z^\star} \tag{4}$$

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 (Sherman and Morrison, 1950), 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. (2021). 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.

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 (Bai et al., 2019; 2020; Pedregosa, 2016).

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) (Li et al., 1998))**.** *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 $b = \texttt{true}$. 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. From More and Trangenstein (1976, 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 Section 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 (Conn et al., 1991; Li et al., 1998; Nocedal and Wright, 2006), it is a strong assumption whose satisfaction in practice is debatable, cf., e.g., (Fayez Khalfan et al., 1993). For Broyden's method, ULI is violated in all numerical experiments in (Mannel, 2020; 2021a;b), 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, (3) 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

$$e_n = t_n B_n^{-1} \frac{\partial g_\theta}{\partial \theta}\Big|_{z_n}, \tag{5}$$

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 (Liu and Nocedal, 1989) 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 Section 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 (5). 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*

$$\lim_{n\to\infty} p_\theta^{(n)} = \frac{\partial \mathcal{L}}{\partial \theta}\Big|_{z^\star} \tag{6}$$

Proof. It follows from known results that the extra updates do not destroy the q-superlinear convergence of (z_n). The proof of (6) 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 (6) follows. A full proof is provided in Section 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 (3). An appropriate secant condition is given by

$$v_n^T B_{n+1} = v_n^T J_{g_\theta}(z_{n+1}), \tag{7}$$

where

$$v_n^T = \nabla_z \mathcal{L}(z_n) B_n^{-1}. \tag{8}$$

To incorporate the secant condition (7), we use the Adjoint Broyden's method (Schlenkrich et al., 2010), 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. Broyden et al. (1973, Theorem 3.2) for Broyden's method and the BFGS method, respectively, Schlenkrich et al. (2010, 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 by Li and Fukushima (2000). 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 (7) and extra update in the direction v_n defined in (8). 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 Schlenkrich et al. (2010, 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 Section 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 (Fung et al., 2021). We draw the reader's attention to the fact that although the Jacobian-Free method (Fung et al., 2021) is used outside the assumptions needed to have theoretical guarantees[2] 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 (Pedregosa, 2016)[3], which is based on the Python scientific ecosystem (Harris et al., 2020; Pedregosa et al., 2011; Virtanen et al., 2020). Deep Equilibrium experiments were done using the PyTorch (Paszke et al., 2019) code for Multiscale DEQ (Bai et al., 2020)[4], which was distributed under

[2] See the results on contractivity in Section E.2.

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

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

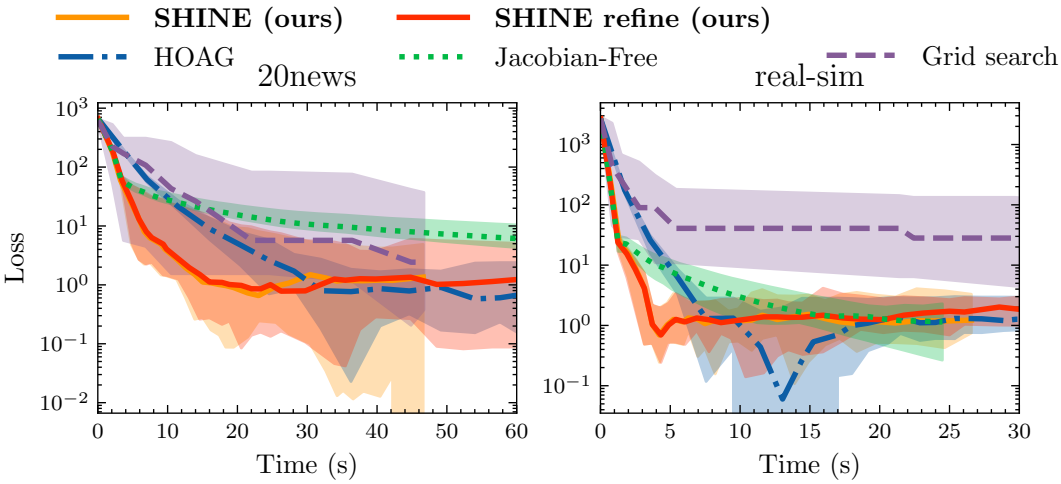

Figure 1: **Bi-level optimization:** Convergence of held-out test loss for different hyperparameter optimization methods on the ℓ_2-regularized logistic regression problem for the 2 datasets (20news (Lang, 1995) and real-sim (lib)) SHINE achieves the best performances for both problems while the Jacobian-Freee method is much slower, in particular on 20news. Note that the kink for HOAG on real-sim does not mean it is better as the optimization stops once the validation loss has converged and not the test one. An extended figure with more methods is provided in Section E.1.

the MIT license. Plots were done using Matplotlib (Hunter, 2007), with Science Plots style (Garrett and Peng, 2021). 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 a variant of our method.

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 ℓ_2-regularized LR, using the code from Pedregosa (2016) and the same datasets. Convergence on unseen data is illustrated in Figure 1.[5] An acceptable level of performance is reached twice faster for the SHINE method compared to any other competitor. Another finding is that the refine strategy does not provide a definitive improvement over the vanilla version of SHINE. In order to verify that the performance gain of SHINE is not simply driven by truncated inversion, we also run HOAG with limited number of inversion iteration and showed that this degrades its performances (see Section E.1).

We also tested our implementation of OPA on the 20news dataset and present the results in Figure 2. In order to get a fair comparison, we implemented both SHINE, SHINE-OPA and HOAG using the same full Python code instead of relying to the orignal code which relied on the Fortran implementation of L-BFGS from (Virtanen et al., 2020). While SHINE with OPA does not outperform the vanilla SHINE, it reaches similar performances, outperforming HOAG, and comes with strong theoretical grounding.

[5]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 ; the kink present in the case of the real-sim dataset is an example of that.

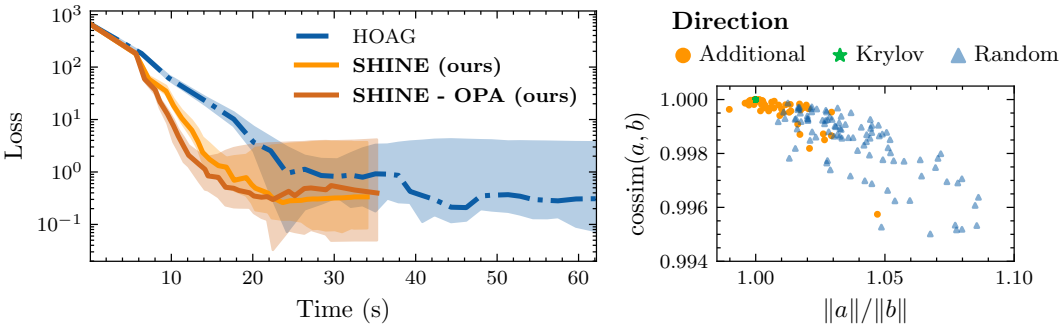

Figure 2: **Bi-level optimization with OPA:** (*left*) Convergence of different hyperparameter optimization methods on the ℓ_2-regularized LR problem for the 20news dataset (Lang, 1995) on held-out test data. SHINE with OPA achieves similar performance as SHINE wihtout OPA but with better convergence guarantees. (*right*) Evaluation of the inversion quality in direction v using OPA $b = B_n^{-1}v$ compared to the exact inverse $a = J_{g_\theta}(z^\star)^{-1}v$ for 3 different directions: the prescribed direction, the Krylov direction and a random direction. The points represent the cosine similarity between a and b function of the ratio of their norm and the closer to $(1, 1)$ the better. The inverse in the prescribed direction is better than in random direction.

We also showed on a smaller dataset, the breast cancer dataset (Dua and Graff, 2017), 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 $\left.\frac{\partial g_\theta}{\partial z}\right|_{z^\star}(z_n - z_{n-1})$ and a random direction not used in the qN algorithm. The results for 100 runs with different random seeds are depicted in Figure 2, where we can observe that OPA indeed ensures a better inversion in the prescribed direction compared to a random direction. We also notice that a poor direction for the inversion seems 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 same model configuration as in the original Multiscale DEQ paper (Bai et al., 2020) and did not fine tune any hyperparameter. For the different DEQ training methods, models for a given seed share the same unrolled-pretraining steps. We do not include OPA in the DEQ results because while the gradients are well correlated with the true ones (see Figure E.2), we observe a sharp initial performance drop that reduces its performance on Imagenet. We provide partial results in Section E.4.

CIFAR-10. The first dataset is CIFAR-10 (Krizhevsky, 2009) 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 3 show that for the vanilla version, SHINE slightly outperforms the Jacobian-Free method (Fung et al., 2021). 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 (He et al., 2016) at 92.9%). Finally, we do highlight that the Jacobian-Free method (Fung et al., 2021) is able to perform well outside the scope of its theoretical assumptions, albeit with slightly worse performance than SHINE. We speculate that the batched stochastic gradient descent helps accelerated methods by averaging out the errors made in the approximation.

ImageNet. The second dataset is the ImageNet dataset (Deng et al., 2009) 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

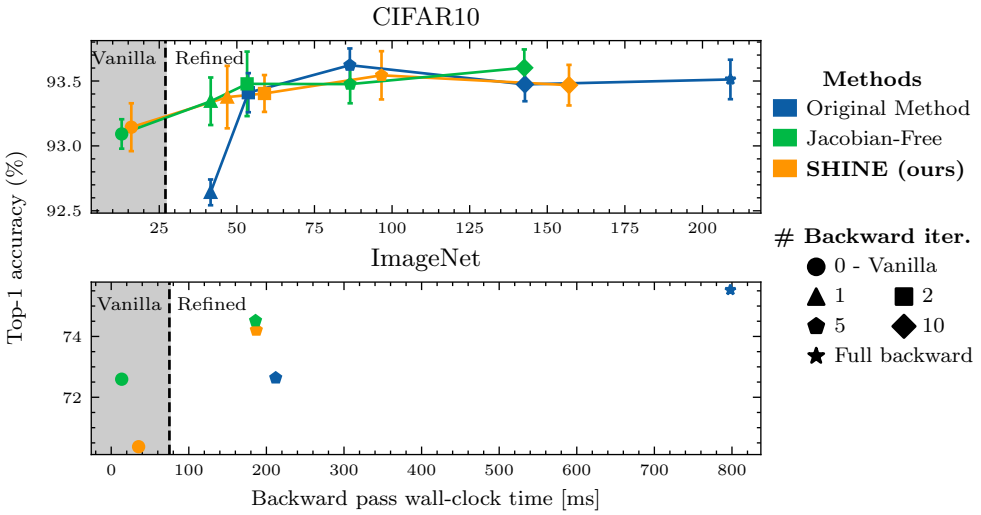

Figure 3: **DEQ:** Top-1 accuracy function of backward pass runtime for the different methods considered to train DEQs, on CIFAR (Krizhevsky, 2009) and ImageNet (Deng et al., 2009). The original DEQ training method corresponds to the Full backward pass points and the vanilla SHINE and Jacobian-Free methods correspond to direct use of the inverse approximation without further refinement. The other points correspond to further refinements of the different methods with different number of iterations used to inverse $J_{g_\theta}(z^\star)$ in the direction of $\nabla_z \mathcal{L}(z^\star)$. This highlights the tradeoff between computations and performances driving the refinement choice.

presented in Figure 3 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 (Fung et al., 2021). 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 (He et al., 2016)).

The key take-away from Figure 3 is that both SHINE and Jacobian-Free approximation methods allow to accelerate the DEQ's backward pass at a relatively low accuracy cost.[6] Moreover, using the proposed refined versions of these methods, the performance drop can be reduced by reducing the acceleration.

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.

[6] More on the overall computational effort can be found in Table E.2

REPRODUCIBILITY STATEMENT

We provide with the submission of this paper the full code necessary to reproduce the figures and the other quantitative results of the paper, from the training, to the evaluation and the actual figure drawing. We made sure to use seeds and verified that the seeding was indeed allowing reproducible results. We also provide time estimates for the reproduction of the figures. We made sure to provide the full proofs for our theorems in the supplementary material of this submission. The core concepts used in the proofs, and their sketches are also laid out in the main text.

REFERENCES

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

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

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

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

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

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

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.

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.

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.

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

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

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.

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.

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

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.

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.

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

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

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

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.

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.

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

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

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

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

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

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

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

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.

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

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.

J. Lorraine, P. Vicol, and D. Duvenaud. Optimizing Millions of Hyperparameters by Implicit Differentiation. In *AISTATS*, 2020.

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

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

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

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

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

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.

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

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.

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

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.

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.

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

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.