Journal of Intelligent Information Systems
https://doi.org/10.1007/s10844-025-00969-x

RESEARCH

An optimizer derived from Halpern's method for enhanced neural network convergence and reduced carbon emissions

Vittorio Colao[1] · Katherine Rossella Foglia[1] · Andrea Giordano[2] · Ettore Ritacco[3] · William Spataro[1]

Received: 17 November 2024 / Revised: 16 July 2025 / Accepted: 18 July 2025

Abstract
This work examines the Halpern's iterative method as a means to create new neural network optimizers that could surpass many current existing approaches. We introduce HalpernSGD, an innovative network optimizer that leverages Halpern's technique to enhance the rate of convergence of the Stochastic Gradient Descent (SGD), leading to reduced carbon emissions in neural network training processes. The combination of Halpern's iterative method and Gradient Descent (GD) has led to an algorithm with a quadratic rate of convergence compared to the simple GD. Experimental comparisons between their stochastic versions show that HalpernSGD achieves greater efficiency than SGD by requiring fewer training epochs, thus reducing energy consumption and carbon footprint while maintaining model accuracy. We also compare HalpernSGD with ADAM, identifying potential improvements to ADAM's approach in terms of stability and convergence, and suggesting a future direction for the development of combined optimizers. The implementation for HalpernSGD can be accessed at: https://github.com/EttoreRitacco/HalpernSGD.git

Keywords Halpern's iterative method · Optimizers · Gradient descent · Green AI · Environmental footprint · Carbon emission · Rate of convergence · Metric fixed point theory · Non-expansive mapping

1 Introduction

Artificial Intelligence (AI) is undergoing an unprecedented period of growth and evolution. AI advancement is driving the creation and deployment of larger and larger architectures. These models, primarily based on Deep Neural Networks (DNNs), show improved capabilities across various domains including computer vision (encompassing Segmentation (Sultana et al., 2020; Jain et al., 2023), Object Detection (Kaur & Singh, 2023; Chen et al., 2023), and Scene Recognition Xie et al., 2020; Chen et al., 2021), natural language processing (utilizing Transformers (Vaswani et al., 2017) and Large Language Models Brown et al., 2020), and the generation of realistic data (through Variational Autoencoders (Kingma & Welling, 2014), GANs (Goodfellow et al., 2014), Diffusion Models (Ho et al., 2020), and

Extended author information available on the last page of the article

synthetic data generation Manco et al., 2022; Liguori et al., 2021). However, the increasing complexity of these architectures leads to a significant demand for computational resources, which in turn gives rise to greater ecological concerns. Indeed, the environmental impact of AI, especially due to the increased computational intensity, leads to significant carbon dioxide emissions (Strubell et al., 2020; Xu et al., 2023; Dhar, 2020). It is worth noting that many studies (Delanoë et al., 2023; Das & J, 2023) emphasize the net reduction in greenhouse gas emissions through AI application, especially in industrial contexts. Nevertheless, the energy consumption of complex neural architectures necessitates addressing AI development sustainability. Researchers have proposed various approaches to decrease carbon emissions during DNN training and inference, including: (*i*) neural architecture size reduction, even through transfer learning, accepting quality loss, (*ii*) hardware optimization using GPUs or TPUs, (*iii*) utilization of energy-efficient cloud services in specialized data centers, (*iv*) strategic selection of computation unit locations based on energy costs and environmental factors, (*v*) regularization of the loss function, and (*vi*) data processing optimization to minimize information volume. Our research presents an alternative strategy to address carbon emission reduction. We focus on enhancing the optimization algorithms used in neural network training. While Gradient Descent (GD) remains popular, alternative methods exist in literature offering superior convergence rates, potentially reducing training steps and carbon emissions without compromising model quality. We introduce a new optimizer based on Halpern's iterative method (Halpern, 1967), which extends GD (Bottou, 1999) with quadratic convergence improvement, and implement its stochastic version, namely HalpernSGD. Our experimentation on established benchmark datasets compares HalpernSGD with SGD, demonstrating significant reductions in training epochs and carbon emissions with statistical relevance. Furthermore, we conducted a comparative analysis between HalpernSGD and ADAM (Kingma & Ba, 2015), a widely-adopted optimizer, exploring the possibility of developing similar generalization for ADAM to improve its convergence. The paper's structure is as follows: Section 2 reviews current literature on carbon emission reduction strategies; Section 3 explores the theoretical foundations of iterative methods, specifically GD, Halpern's algorithm and ADAM; Section 4 and 5 detail our experimental methodologies and provides analysis of results; and Section 6 presents concluding thoughts.

2 Related work

Training state-of-the-art models such as GPT-3 (with 175 billion parameters) consumed approximately 1287 MWh of electricity and emitted approximately 552 tonnes of CO_2, based on the average US energy mix (Patterson et al., 2021). Similarly, an analysis of global data center energy consumption reveals that, while consumption in 2010 was approximately 194 TWh, by 2018 it had risen to 205 TWh, representing approximately 1% of global electricity consumption and an increase of 6% on 2010. However, over the same period, the number of computing instances grew by 550%, suggesting a significant improvement in the energy efficiency of data centers, with the increase in workloads far outstripping the increase in energy consumption (Masanet et al., 2020). Nevertheless, in a worst-case scenario, the Information and Communication Technology (ICT) sector is projected to consume up to 51% of the world's electricity and contribute approximately 23% of global greenhouse gas emissions by 2030, unless significant improvements in efficiency and

renewable energy integration are achieved (Andrae & Edler, 2015). Another study from the University of Massachusetts, Amherst, estimates that training large AI models can emit over 626,000 pounds of CO_2, nearly five times the emissions of an average American car. The observed increase in emissions indicates that energy costs are proportional to the size of the model, with particularly high costs for tuning activities. This underscores the necessity to integrate energy efficiency into future AI research (Hao, 2019; Altman, 2025). Also studies such as Xu et al. (2023), Kuo and Madni (2023), and Xu et al. (2021) emphasize the critical need to incorporate energy efficiency and reduced carbon emissions into machine learning research and development, aiming for more environmentally sustainable technological solutions. These papers provide comprehensive insights into the challenges and prospects of sustainable AI development. In this context, the field of energy-efficient training of deep neural networks is attracting increasing scientific attention. There are several strategies to address these challenges, with certain approaches being more extensively developed than others. Research by Xu et al. (2023) explores the emergence of the Green AI movement, which promotes sustainable choices in software, hardware and data center operations to address the environmental impact of AI. In particular, this movement highlights the importance of strategic site selection and hardware choices for model training. Green learning, as described in Kuo and Madni (2023), presents an environmentally conscious alternative to traditional deep learning, emphasizing lightweight model architectures to minimize energy consumption in both cloud data centers and edge devices. The study (Xu et al., 2021) identifies carbon emissions as a key metric in evaluating green deep learning. It outlines various strategies to improve energy efficiency, including optimized data utilization, architectural refinement, and improved training and inference methods. These include techniques such as transfer learning for intelligent initialization, batch normalization, progressive training approaches and hyper-parameter optimization, all aimed at achieving an optimal balance between model accuracy and environmental impact. In addition, in support of more environmentally sustainable approaches, Hinton et al. (2015) and Gou et al. (2021) detail knowledge distillation processes, allowing information to be transferred from large and complex models to more compact and efficient ones, thereby reducing energy and hardware resource requirements. While these studies demonstrate significant potential for reducing the environmental footprint of AI, many unexplored approaches remain to be investigated. Our contribution is characterized by an innovative technique for optimizing network parameters. Indeed, we uniquely focus on optimizer improvement rather than loss function modification or regularization, unlike common literature approaches. The proposed optimizer is characterized by its convergence properties, which allow a significant reduction in the number of convergence iterations required. This approach results in notable energy savings and reduced carbon emissions, demonstrating its potential for advancing energy efficiency and environmental sustainability in DNN training.

3 Mathematical framework

A challenging aspect in the training of deep neural networks lies in the strategies for minimizing the loss function, which involves solving an unconstrained minimization problem of the form:

$$\min_{\theta \in \mathbb{R}^d} \mathcal{L}(\theta),$$

where θ represents the model parameters in a d-dimensional space. Given the high number of parameters, or the dimensionality of the space, the search for the optimum of this minimization problem requires a method that is straightforward to implement, with modest resource consumption and that is not subject to the 'curse of dimensionality', i.e. with a rate of convergence independent on the dimensionality of the space.

For the above reported reasons, the most frequently used approach is gradient descent or its variants; that is beginning from an initial point $\theta^{(0)} \in \mathbb{R}^d$ and the update rule is given by

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla \mathcal{L}(\theta^{(t)}). \tag{1}$$

Moreover, gradient-based methods are particularly popular due to the simplicity with which the gradient can be computed using backpropagation, which can efficiently calculate gradients in neural networks (see Rumelhart et al., 1986), thus providing a tool for managing an high number of layers.

On the other hand, the complexity of the loss landscape and its non-convexity prevent the straightforward application of gradient descent, as it may halt at suboptimal local minima. The classical adopted strategy is thus to carefully select the learning rate η so that it can adapt to potential "irregularities" in the landscape.

In this line of research, several celebrated works have introduced adaptive gradient-based methods that address the challenges posed by the complex loss landscape in deep neural networks. Foundational methods include SGD (Robbins & Monro, 1951), which introduces stochasticity; RMSprop (Tieleman & Hinton, 2012), which adjusts learning rates by averaging squared gradients to better cope with non-stationary objectives; AdaDelta (Zeiler, 2012), which refines AdaGrad's approach (Duchi et al., 2011) to avoid diminishing learning rates; and ADAM (Kingma & Ba, 2015), which combines momentum with adaptive learning rates for faster and more stable convergence. These methods have become central to deep learning, in particular, ADAM is among the most widely used optimizers. Nevertheless, the original proof presents several inaccuracies, and its convergence to a local optimum can be disproven in various cases, as documented in Reddi et al. (2018) and Zou et al. (2019), depicting situations where it fails to achieve convergence. Furthermore, ADAM can exhibit oscillatory behavior: research presented in Bock and Weiß (2019) demonstrates the existence of 2-limit cycles for this optimizer. Such cycles emerge when considering the scalar quadratic objective function (see subsection 4.3 for details) This phenomenon describes a scenario in which, after two iterations of the ADAM optimizer, the system reverts to its previous state. Consequently, the optimizer alternates between two points in the parameter space rather than converging to a single point that minimizes f.

In addition to this, we aim to address an imprecision in the original paper, specifically in Lemma 10.3 (Kingma & Ba, 2015):

Lemma 10.3 *Let $g_t = \nabla f_t(\theta_t)$ and $g_{t,i}$ as its i-th element. We define $g_{1:t,i} \in \mathbb{R}^t$ as a vector that contains the i-th dimension of the gradients over all iterations up to t, i.e. $g_{1:t,i} = (g_{1,i}, g_{2,i}, \ldots, g_{t,i})$. Assuming that $\|g_t\|_2 \leq G$ and $\|g_t\|_\infty \leq G_\infty$ then it holds that:*

$$\sum_{t=1}^{T} \sqrt{\frac{g_{t,i}^2}{t}} \leq 2G_\infty \|g_{1:T,i}\|_2 .$$

We observed that this result is not true in general, in fact setting $g_t = \frac{1}{t}$ and $G_\infty = 1$ we have:

$$\sum_{t=1}^{T} \frac{1}{t^{3/2}} \leq 2 \left(\sum_{t=1}^{T} \frac{1}{t^2} \right)^{1/2}$$

that implies

$$2.61 \leq 2.56$$

which of course is false and it directly represents a counter example for this lemma. It is significant to observe that Lemma 10.3 plays a crucial role in analyzing convergence for others optimizers derived from the ADAM algorithm (e.g. in Dubey et al., 2019).

A detailed analysis of gradient descent algorithms and their generalizations can be found within the broader class of iterative sequences for approximating fixed points of non-expansive mappings and their generalizations. These concepts have been extensively studied over recent decades within the field of Metric Fixed Point Theory (Chidume, 2009; Bauschke & Combettes, 2011; Berinde & Takens, 2007), which provides a comprehensive introduction to this framework.

In particular, these algorithms are advantageous due to their applicability in infinite-dimensional spaces, where dimensionality does not impact convergence. Furthermore, they are particularly adaptable in order to well-suit to the general context of normed spaces.

For a deeper understanding of the algorithm (1), it is useful to conduct analysis in a more general setting. In particular, the Metric Fixed Point Theory provides a natural generalization of the Gradient Descent method, more specifically in relation to the convergence of iterative mappings that satisfy specific metric-type conditions. This connection has its foundation in a now classical result originally presented in Baillon and Haddad (1977) (see also Bauschke and Combettes, 2010).

To explicitly state it, we recall that a map $T : Dom(T) \subset \mathbb{R}^d \to \mathbb{R}^d$ is said to be non-expansive if it does not alter the distance between points, i.e. if $\|T\theta - T\bar{\theta}\| \leq \|\theta - \bar{\theta}\|$, for any $\theta, \bar{\theta} \in Dom(T)$. We also recall that a fixed point θ^* for T is a point which remains fixed under the action of the map, that is $T\theta^* = \theta^*$.

Theorem 1 *(Baillon-Haddad) Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex and differentiable map, whose differential ∇f is L-Lipschitz and fix $\beta \in (0, 2/L)$. Then the "forward" operator $F : H \to H$ defined by $F = (I - \beta \nabla f)$ is averaged non-expansive, i.e.*

$$F = \lambda I + (1 - \lambda) T$$

for some constant $\lambda \in (0, 1)$ and $T : H \to H$ non-expansive. Moreover

$$\theta^* \in \arg\min_x f(x) \iff \theta^* = F\theta^* \iff \theta^* = T\theta^*,$$

that is, the set of fixed points for T coincides with $\arg\min f$.

In view of this Theorem, the iterative procedure (1) can be equivalently reformulated as

$$\begin{cases} \theta_0 \in \mathbb{R}^d \\ \theta_{m+1} = \lambda\theta_m + (1-\lambda)T\theta_m \end{cases} \tag{2}$$

where λ is a real parameter in [0, 1) and T is a non-expansive map. The convergence analysis thus becomes equivalent to demonstrating that the sequence θ_m converges to a fixed point θ^* of T. It can be proven that algorithm (2) converges in norm to a fixed point of the operator T, which translates to $\nabla f(\theta^*) = 0$, recalling the above statement (Chidume, 2009). The iterative process defined in (2) is known as the Krasnoselskii-Mann Algorithm and both its convergence properties and convergence rate have been extensively studied. For a detailed introduction to the topic, we recommend consulting the comprehensive works Berinde and Takens (2007) and Chidume (2009). Due to the fair general nature of the setting, the convergence speed of the previous algorithm is currently stated in terms of rate of asymptotic regularity, that is by evaluating the operator residual norm $\varepsilon_m := \|\theta_m - T\theta_m\|$. Finding an upper bound for this rate in the Kransoselskii-Mann iteration had been a persistent challenge in Fixed Point Theory, lastly solved by Baillon and Bruck (1996) (see also Cominetti et al. (2014) for a more general case) by proving that the estimate $\varepsilon_m \leq \frac{\|\theta_0-\theta^*\|}{\pi(\lambda(1-\lambda))^{1/2}}\frac{1}{m^{1/2}}$ holds for the general setting of Banach spaces and for any non-expansive map T. We highlight that, under this setting, the convergence rate (i.e., the residual norm) is $\varepsilon_m = O(1/\sqrt{m})$. When the above mentioned evaluation of the convergence rate is applied to the map $T = I - \beta\nabla f$, the well-known result concerning the convergence rate of the Gradient Descent algorithm is recovered.

Among the techniques currently used for enhancing the asymptotic regularity rate, we cite the Nesterov process adopted in Boţ and Nguyen (2023), where it is proved that Krasnoselskii-Mann iteration with momentum reaches an asymptotic regularity rate of $O(\frac{1}{m^2})$.

Within the framework of Metric Fixed Point Theory, another approach has demonstrated significant effectiveness: the Halpern iterative procedure, given by

$$\begin{cases} \theta_0 \in \mathbb{R}^d \\ \theta_{m+1} = \lambda_m u + (1-\lambda_m)T\theta_m \end{cases}$$

where $u \in \mathbb{R}^d$ is a fixed element, termed anchor, and $\{\lambda_m\}$ is a sequence of parameters in (0, 1), whose properties will be later examined.

The Halpern iterative method, originating from Halpern (1967), represents a traditional approach to approximating fixed points of non-expansive mappings. This methodology, in recent years, has garnered significant attention (see e.g. Diakonikolas, 2020; Lee & Kim, 2021; Yoon & Ryu, 2021) due to the higher rate of asymptotic regularity $O(1/m)$, which is quadratic with respect to both the Krasnoselskii-Mann iteration and, by extension, the Gradient Descent method. This fact had been explicitly noted in Sabach and Shtern (2017) for the finite dimensional setting and by F. Lieder in Lieder (2021) for the infinite dimensional case. While a full investigation of lower bounds for Halpern iterations' asymptotic regular-

ity rate remains incomplete, initial progress has been made in establishing a preliminary lower bound for the error sequence $|\theta_m - \theta^*|$ in Colao and Marino (2021).

Regarding the sequence of step-dependent coefficients λ_m, in Xu (2002) it has been proved the convergence of the entire sequence θ_m to the solution θ^* under the following conditions:

$$(i)\ \lim_{m\to\infty} \lambda_m = 0, (ii)\ \sum_{m=1}^{\infty} \lambda_m = \infty,\ \text{and}\ (iii)\ \lim_{m\to\infty} \frac{\lambda_m - \lambda_{m-1}}{\lambda_m} = 0.$$

We observe that these conditions are mild enough to include the choice $\lambda_m = 1/m$, which seems to guarantee the best rate of asymptotic regularity.

4 Optimization comparison

To gain deeper insights into the intrinsic behavior of the HalpernSGD optimizer independent of neural network training, we conducted additional experiments using two widely recognized benchmark functions in optimization research: the Rastrigin and Rosenbrock functions. These functions are known for their complex, non-convex landscapes, making them ideal for assessing the convergence properties and robustness of optimization algorithms.

In this analysis, we compare HalpernSGD against standard optimizers such as SGD and ADAM. The results, presented both quantitatively and visually, illustrate the distinct optimization trajectories and oscillatory dynamics of each method as they navigate the search space toward the global minimum. In particular, HalpernSGD demonstrates smoother convergence patterns and improved stability compared to the baseline optimizers.

This new evaluation complements our prior experiments on real datasets by providing a controlled environment to study the optimizer's fundamental characteristics and validate its effectiveness beyond standard neural network training scenarios. We exploited the *PyTorch-optimizer* package.[1]

4.1 The Rastrigin function

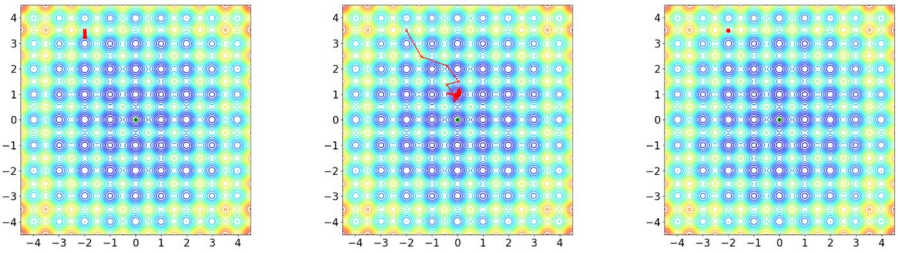
The Rastrigin function is a well-known benchmark for optimization algorithms due to its non-convex nature, containing a number of local minima that often can trick the optimizers to converge to a suboptimal value (Rastrigin, 1974). It is defined as:

$$f(x, y) = 20 + \left(x^2 + y^2 - 10\cos(2\pi x) - 10\cos(2\pi y)\right).$$

Among the local minima, its global minimum is located at $(x, y) = (0, 0)$, where the function reaches the lowest value $\bar{f} = 0$.

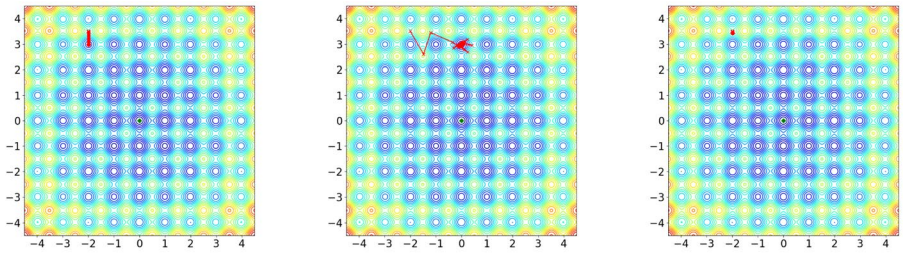
From Figs. 1, 2, 3, and 4 we observe the trajectory landscapes of the three optimizers, in all cases starting from the point $(-2, 3.5)$ and across different learning rates. It is important to note that all the three optimizers fail to reach the global minimum, since all the algorithms are trapped in local minima, thus failing to suitably explore the search space. However, the

[1] https://github.com/jettify/pytorch-optimizer



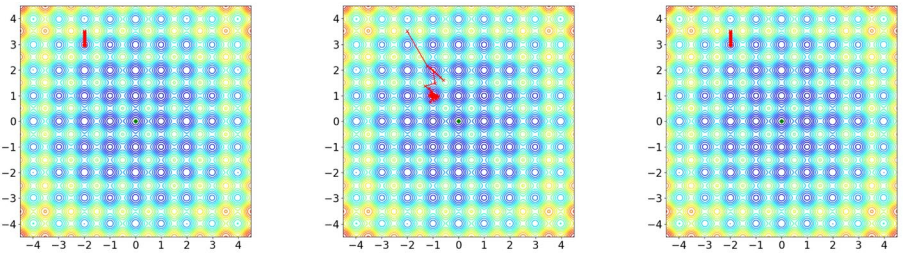
(a) ADAM (b) HalpernSGD (c) SGD

Fig. 1 Optimizers trends comparison for the Rastrigin function with 5e-05 learning rate



(a) ADAM (b) HalpernSGD (c) SGD

Fig. 2 Optimizers trends comparison for the Rastrigin function with 0.0005 learning rate



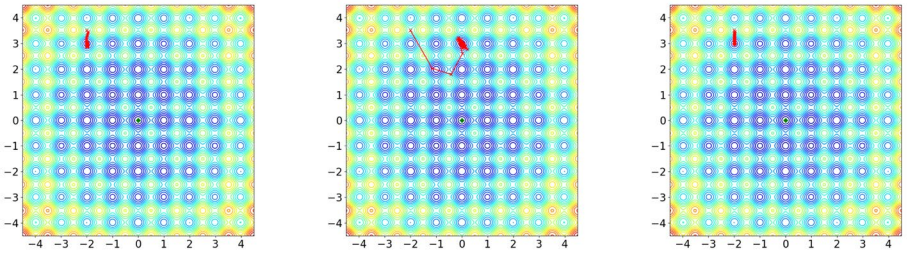
(a) ADAM (b) HalpernSGD (c) SGD

Fig. 3 Optimizers trends comparison for the Rastrigin function with 0.005 learning rate

position of the anchor, chosen randomly in our setting, significantly influences the path chosen by HalpernSGD, thus permitting to avoid certain local minima where both ADAM and SGD are instead attracted.

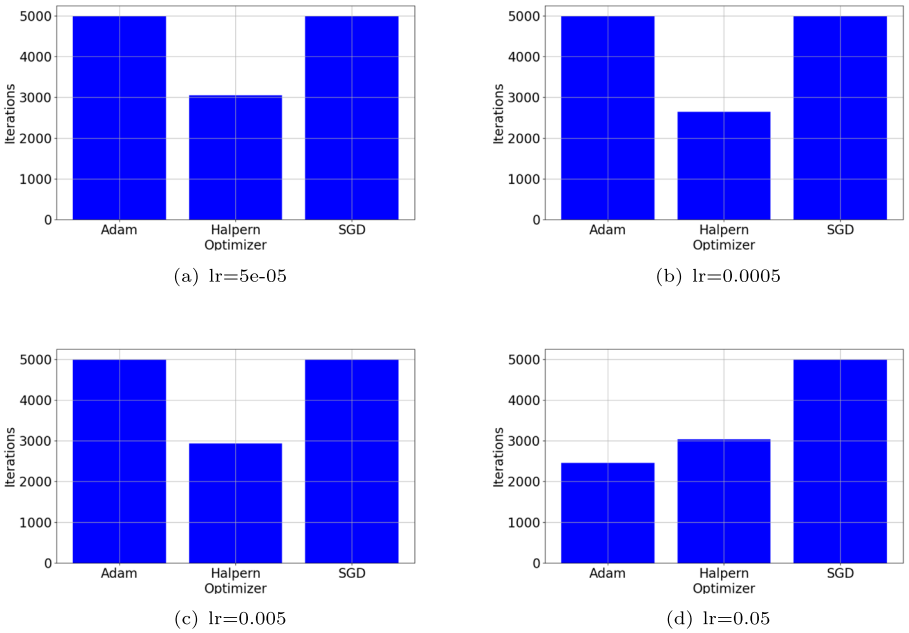
Figures 1 to 4 clearly show how HalpernSGD results in a better exploration of the landscape due to the randomness of the anchor given that the iteration forces the convergence to a minimum point in proximity to the anchor.

Analyzing the iterations to convergence, see Fig. 5, it can be pointed out that the HalpernSGD outperforms the competitors in all the considered learning rates except for the highest one. Eventually, Fig. 6 show a faster rate of convergence for the HalpernSGD iteration despite large oscillations in the initial steps.



(a) ADAM (b) HalpernSGD (c) SGD

Fig. 4 Optimizers trends comparison for the Rastrigin function with 0.05 learning rate



(a) lr=5e-05 (b) lr=0.0005

(c) lr=0.005 (d) lr=0.05

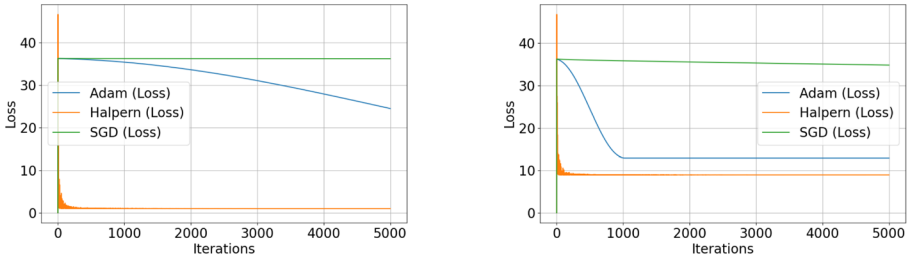
Fig. 5 Number of iterations to converge comparison for the Rastrigin function with different learning rates

4.2 The Rosenbrock function

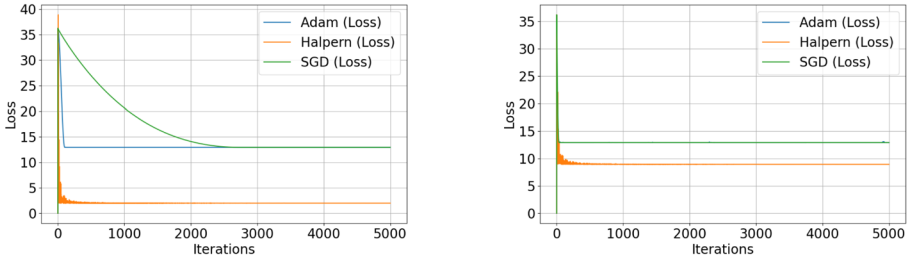
The Rosenbrock function is characterized by the region hosting the unique minimum value to be a quasi-plateau with a nearly zero gradient (Rosenbrock, 1960). It is defined as:

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2.$$

The global minimum is located at $(x, y) = (1, 1)$, where the function reaches the lowest value $\bar{f} = 0$. From Figs. 7, 8, 9, and 10 we observe the trajectory landscapes of the three optimizers, in all cases starting from the point (-2.0, 2.0).

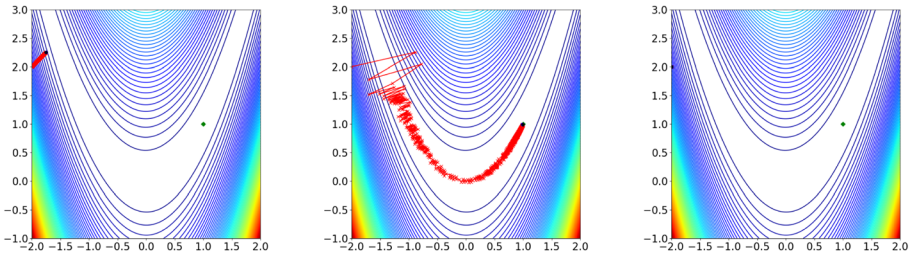


(a) lr=5e-05 (b) lr=0.0005



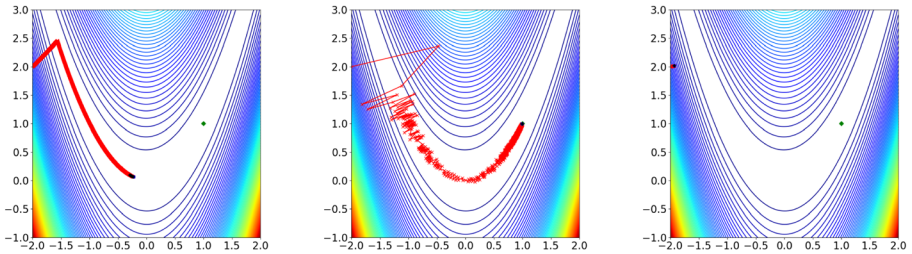
(c) lr=0.005 (d) lr=0.05

Fig. 6 Loss trends comparison for the Rastrigin function with different learning rates



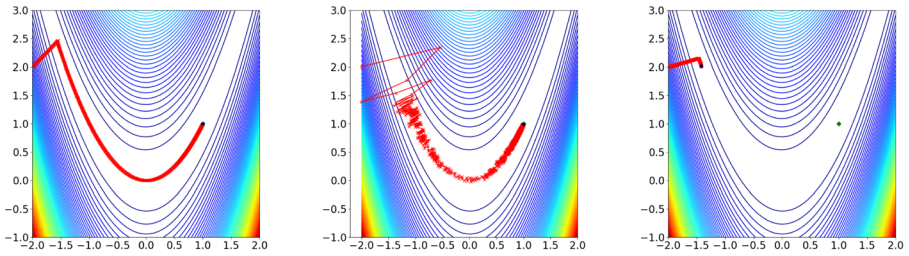
(a) ADAM (b) HalpernSGD (c) SGD

Fig. 7 Optimizers trends comparison for the Rosenbrock function with 5e-05 learning rate



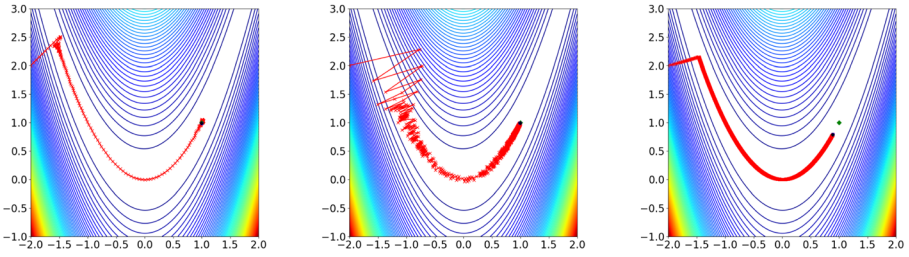
(a) ADAM (b) HalpernSGD (c) SGD

Fig. 8 Optimizers trends comparison for the Rosenbrock function with 0.0005 learning rate



(a) ADAM (b) HalpernSGD (c) SGD

Fig. 9 Optimizers trends comparison for the Rosenbrock function with 0.005 learning rate



(a) ADAM (b) HalpernSGD (c) SGD

Fig. 10 Optimizers trends comparison for the Rosenbrock function with 0.05 learning rate

The Figures clearly show how HalpernSGD outperforms ADAM and standard SGD for all the considered learning rates. In particular, HalpernSGD finds the global minimum in all cases. On the contrary, due to the low values for the learning rates SGD is never capable of finding the global minimum, while ADAM is able to reach it only in the last 2 cases (see Figs. 9 and 4). As for the Rastrigin case, also in this case the advantage of HalpernSGD is partially counterbalanced by a more oscillating trend in searching the space.

Finally, Fig. 11 shows an even more faster rate of convergence for the HalpernSGD iteration for the case of the Rosenbrock function.

4.3 Comparison between ADAM and Halpern GD

In this section we will take a closer look at the comparison between Adam and Halpern based optimization algorithms. In particular, we will discuss the differences in terms of convergence guarantees. Indeed, in Bock and Weiß (2019) it had been shown that ADAM might fail to converge to a minimum of a strictly convex quadratic function for a broader class of parameters. While, for Halpern based algorithms and as we already mentioned, the convergence to a minimum is ensured under the assumption of the function being convex and Lipschitz continuous.

Specifically, following Bock and Weiß (2019), the sequence $\{w_t\}$ generated by ADAM can exhibit stable limit cycles of period two, despite the objective being of the form

$$f(w) = \frac{1}{2} w^\top C w,$$

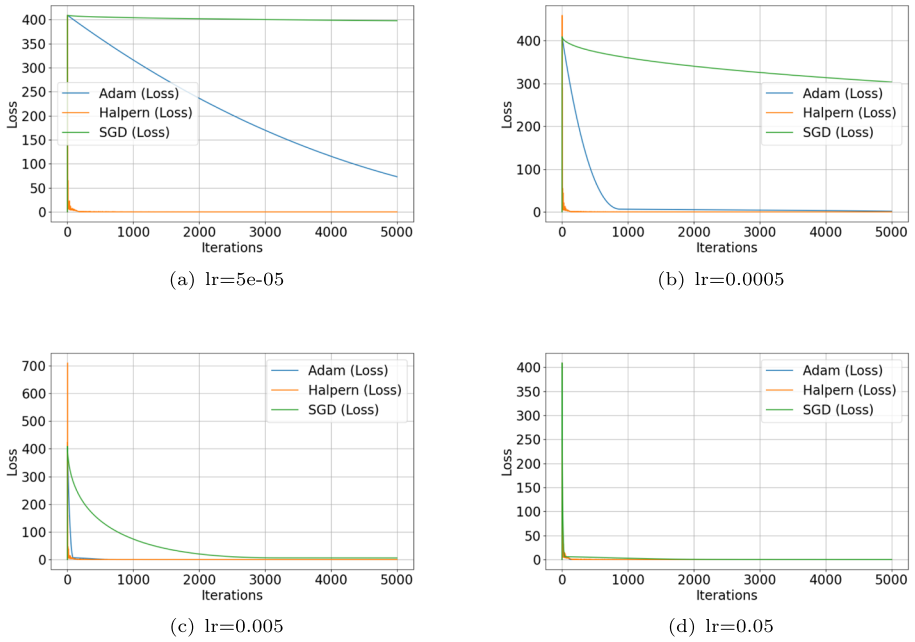


Fig. 11 Loss trends comparison for the Rosenbrock function with different learning rates

with C symmetric and positive definite. These phenomena are not limited to pathological hyperparameter choices but can persist even for standard settings such as $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Formally, ADAM in batch mode can be viewed as a non-autonomous dynamical system of the form

$$w_{t+1} = T(w_t),$$

where the iteration map T evolves according to:

$$\begin{aligned} m_{t+1} &= \beta_1 m_t + (1-\beta_1)\nabla f(w_t), \\ v_{t+1} &= \beta_2 v_t + (1-\beta_2)\nabla f(w_t) \odot \nabla f(w_t), \\ w_{t+1} &= w_t - \alpha \cdot \frac{m_{t+1}}{\sqrt{v_{t+1}} \oplus \varepsilon}. \end{aligned}$$

The non-convergence of ADAM can be then proved by showing the existence of the aforesaid 2-limit-cycles, i.e. periodical points w such that $T^2(w) = w$, even with bias correction.

In contrast, the Halpern iteration defined as:

$$\begin{aligned} T(w) &= w - \beta\nabla f(w), \\ w_{t+1} &= \lambda_t u + (1-\lambda_t)T(w_t), \end{aligned}$$

converges in norm to a fixed point w^* of the operator T, which translates to $\nabla f(w^*) = 0$ even in the more general case of convex and Lipschitz continuous functions f in infinite dimensional spaces (see e.g. Qi & Xu, 2021 for details).

Additionally, Halpern iterations exhibit a linear rate of asymptotic regularity, that is the rate of convergence of the quantities $\|w_{t+1} - w_t\|$ or, equivalently, $\|w_t - T(w_t)\|$ in a mild setting (see Körnlein, 2015; Sabach & Shtern, 2017; Cheval & Leuştean, 2025 for details).

5 Experimental evaluation

We performed a comparative study of Halpern's iterative method and Gradient Descent by examining their stochastic versions, considering the necessity of mini-batch usage in neural network training. This led to the comparison of two algorithms: HalpernSGD and the widely recognized SGD. Given that both algorithms leverage variable learning rates (refer to Section 3), we opted for the learning rate sequence $\eta_m = \eta_0/\sqrt{m}$, where m denotes the update step (distinct from epochs), and η_0 is the starting learning rate. Through empirical experimentation to find optimal rates for both optimizers, we chose an initial rate of 1.15. Additionally, in the HalpernSGD, we set $\lambda_m = 0.02/m$. Alongside HalpernSGD and SGD, we included the ADAM optimizer for baseline comparison, which is an adaptive learning rate optimization algorithm that builds on SGD. Our experiments were conducted on five renowned benchmark datasets:

- **Iris**[2] consists of 150 samples from 3 iris flower species, with each sample containing 4 descriptors: sepal length, sepal width, petal length, and petal width.
- **Optical Recognition of Handwritten Digits (Orhd)**[3] involves the task of recognizing handwritten digits (0-9) from 1, 797 images. The original 32×32 images were partitioned into 4×4 blocks, counting the on pixels in each block, resulting in an 8×8 input matrix where each value is within [0, 16].
- **MNIST**[4] comprises 70, 000 gray-scale images (28×28) of handwritten digits (0-9) dedicated to digit recognition, with each pixel serving as an input feature.
- **FashionMNIST (FM)**[5] is an alternative to MNIST, containing 70,000 grayscale 28×28 images of fashion items categorized into 10 classes (e.g., shirts, shoes, bags). In our experiments, we randomly subsample 33% of the full dataset to reduce computational cost.
- **CORA**[6] is a citation network dataset comprising 2,708 machine learning publications classified into 7 categories. Each paper is represented by a sparse binary word vector of dimension 1,433, indicating the presence of specific terms. The dataset includes 5,429 citation links and is commonly used for node classification in graph-based learning.

[2] https://archive.ics.uci.edu/dataset/53/

[3] https://archive.ics.uci.edu/dataset/80/

[4] http://yann.lecun.com/exdb/mnist/

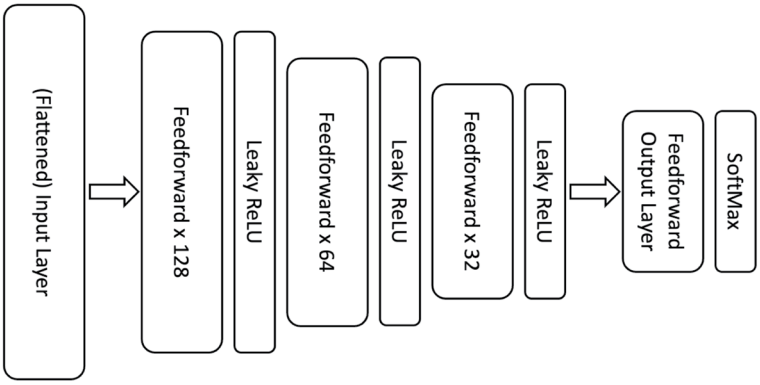
[5] https://github.com/zalandoresearch/fashion-mnist

[6] https://huggingface.co/datasets/gcaillaut/cora/

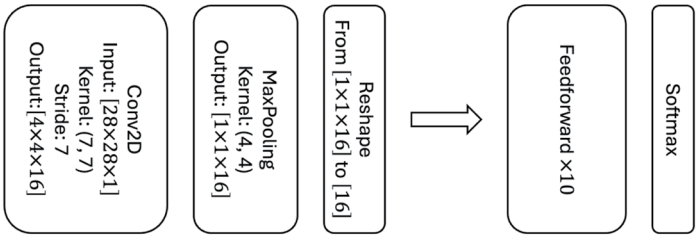
The optimal learning rates identified for ADAM were 0.001, 0.001, 0.01, 0.001, and 0.001, respectively, for the datasets.

The selected datasets are employed to evaluate optimizer performance across three different neural network architectures. The Iris, Orhd, and MNIST datasets are used to train a feedforward neural network (Fig. 12(a)) composed of an input layer and an output layer with dataset-specific dimensions ($4 \rightarrow 3$ for Iris, $64 \rightarrow 10$ for Orhd, and $784 \rightarrow 10$ for MNIST), along with three hidden linear layers of sizes 128, 64, and 32. Leaky ReLU activations are applied in the hidden layers, and the output layer uses a softmax activation to produce class probabilities.

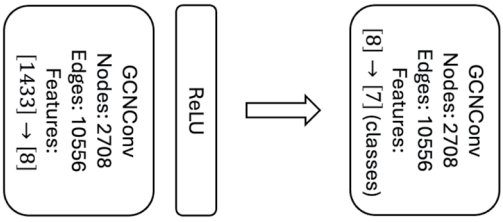
The FashionMNIST dataset is used to train a convolutional neural network (Fig. 12(b)) designed for image classification, consisting of a single convolutional layer (16 channels, kernel size 7, stride 7), followed by a max pooling layer (kernel size 4, stride 4), a flattening



(a) Feedforward Neural Network.



(b) Convolutional Neural Network.



(c) Graph Convolutional Network.

Fig. 12 The design of the neural networks utilized in the experiments

Table 1 Comparative analysis for HalpernSGD (H-SGD), SGD and ADAM

Dataset	Optimizer	Loss	Epochs	Accuracy	Emission (μg)	Time (s)
Iris	SGD	0.60 ± 0.05	279 ± 114	0.95 ± 0.05	0.48 ± 0.18	0.44 ± 0.18
	H-SGD	0.64 ± 0.12	226 ± 117	0.92 ± 0.12	0.43 ± 0.19	0.39 ± 0.19
	ADAM	0.58 ± 0.01	209 ± 155	0.97 ± 0.01	0.50 ± 0.33	0.46 ± 0.32
Orhd	SGD	1.57 ± 0.06	598 ± 411	0.89 ± 0.06	3.91 ± 2.60	2.88 ± 1.99
	H-SGD	1.53 ± 0.03	115 ± 27	0.94 ± 0.03	0.92 ± 0.27	0.58 ± 0.13
	ADAM	**1.50 ± 0.01**	218 ± 99	**0.96 ± 0.01**	1.80 ± 0.83	1.22 ± 0.55
MNIST	SGD	1.61 ± 0.00	2385 ± 370	0.85 ± 0.00	354.11 ± 54.18	367.08 ± 56.20
	H-SGD	1.58 ± 0.04	1107 ± 282	0.88 ± 0.04	179.42 ± 44.96	166.77 ± 42.36
	ADAM	1.50 ± 0.01	**195 ± 45**	**0.96 ± 0.00**	**31.93 ± 7.42**	**32.03 ± 7.41**
FM	SGD	1.99 ± 0.03	10000+	0.51 ± 0.03	177627+	1846+
	H-SGD	1.98 ± 0.03	9255 ± 885	0.51 ± 0.03	144666.67 ± 30141.34	1535.66 ± 278.68
	ADAM	**1.84 ± 0.04**	**1649 ± 238**	**0.62 ± 0.04**	**20121.32 ± 2945.64**	**214.13 ± 30.10**
CORA	SGD	0.98 ± 0.29	**86639 ± 29031**	0.69 ± 0.13	**604.26 ± 202.58**	**600.75 ± 200.17**
	H-SGD	0.87 ± 0.04	94265 ± 5082	0.75 ± 0.02	656.45 ± 34.68	652.54 ± 34.01
	ADAM	**0.44 ± 0.02**	**2138 ± 218**	**0.86 ± 0.01**	**15.19 ± 1.53**	**17.16 ± 2.58**

Entries in **bold** represent outcomes that statistically exceed HalpernSGD, while those in gray show inferior performance. Normal text is used when there's no significant statistical difference from HalpernSGD

operation, and a final fully connected layer projecting to 10 output classes. ReLU is applied after the convolution, and softmax is used at the output.

Finally, the CORA dataset is used to train a two-layer Graph Convolutional Network (GCN), where the first GCN layer maps 1433-dimensional node features to an 8-dimensional hidden representation, followed by ReLU activation, and the second GCN layer maps the hidden features to a 7-dimensional output corresponding to the target classes. The model is shown in Fig. 12(c).

To ensure a fair and consistent comparative analysis of the optimizers, all experiments were conducted using identical neural network architectures, identical initializations, and the same loss function, namely CrossEntropy loss.

To assess whether a statistically significant difference exists among the optimizers analyzed, we employed the 5×2 cross-validated paired t-test as proposed by Dietterich (1998), adopting a significance level of 95%. Within the cross-validated paired t-test framework, each training dataset is divided into a primary training subset and a validation subset, with a ratio of 20 : 80. The validation set was used to determine the stopping criterion during training: training was terminated if the validation loss did not improve by more than 10^{-4} for at least 30 consecutive epochs, or if the maximum number of epochs was reached: 10,000 for all datasets except CORA, for which the limit was set to 100,000. Optimization algorithms are evaluated based on their best loss value on the validation subset, their comprehensive accuracy across a balanced test dataset to reflect predictive capabilities, the number of epochs required to achieve convergence, their CO2 emissions measured in micrograms to assess environmental impact, and the elapsed training time, measured in seconds. This latter metric was quantified using the CodeCarbon tool[7].The experimental work was performed on a MacBook Pro M1 equipped with 32 GB of RAM, with no reliance on GPU acceleration.

The implementation of HalpernSGD is available at: https://github.com/EttoreRitacco/HalpernSGD.git.

Table 1 provides a summary of the outcomes of our experiments, presenting the average and standard deviation for each metric.

[7] https://codecarbon.io/

Excluding the CORA dataset, it is evident that HalpernSGD markedly surpasses SGD in terms of the number of epochs, reduced carbon emissions and elapsed training time across all experiments while maintaining predictive accuracy as indicated by the insignificant differences in validation loss and overall test accuracy, as stated in Section 3. In particular, for the FashionMNIST dataset, the SGD approach exceeded the 10,000-epoch limit without converging to a stable fixed point. Training was therefore halted upon reaching the maximum number of epochs.

Regarding the CORA dataset, both SGD and HalpernSGD reach the 100,000-epoch limit in several runs of the statistical test. Consequently, this experiment can be interpreted as a fixed-budget comparison: given the same number of epochs, how do the two optimizers behave? It is clearly observable that HalpernSGD achieves significantly lower loss values and higher overall accuracy compared to SGD. This observation suggests two possible interpretations. One is that HalpernSGD's trajectory comes closer to a convergence point shared with SGD, which aligns with the findings discussed in Section 4.2. The other is that HalpernSGD has identified a path leading to a better local minimum, similar to the behavior observed in Section 4.1. Consequently, HalpernSGD exhibits less favorable results in terms of required epochs for convergence, training time and carbon emissions.

In examining the biggest datasets (MNIST, FashionMNIST and CORA), ADAM emerges as the primary competitor. For the Iris dataset, despite the absence of significant statistical disparities between HalpernSGD and ADAM, and for the Orhd dataset, HalpernSGD tends to yield lower CO2 emissions and elapsed training times. This is potentially attributable to fewer iterations required for HalpernSGD to achieve convergence or ADAM's higher memory consumption. Specifically, ADAM necessitates consideration of previously used learning rates at each iteration. Additionally, it sometimes faces challenges with oscillatory behaviors and convergence; thus, it could be beneficial to integrate ADAM with the Halpern method, known for its good convergence properties and stability.

6 Conclusion and future work

This paper represented an extension of the investigations carried out in Foglia et al. (2024). In particular, we investigated iterative methods for approximating fixed point of non-expansive mappings and analyzed their application to neural network loss function minimization, leveraging the Baillon-Haddad theorem and convergence in the abstract infinite-dimensional setting. These methods enable the derivation of an optimization algorithm for neural network training with a fast convergence rate.

We developed the HalpernSGD optimizer, based on Halpern iteration, and compared its performance against SGD and ADAM on benchmark datasets. The results show that HalpernSGD and SGD have similar fitting performance, but HalpernSGD achieves convergence in fewer iterations, resulting in lower carbon footprint. However, it is important to acknowledge that it does not outperform established methods such as ADAM in terms of accuracy or generalization, even if ADAM is affected by convergence issues and oscillatory behavior, which can also be observed in simple loss landscapes.

This study should therefore be considered a proof of concept leading for further exploration and refinement of environmentally conscious optimization algorithms. Our future work will focus on integrating the robust convergence properties of Halpern's method with the

adaptive efficiency of optimizers such as ADAM. We hope that this initial effort will inspire the research community to further advance this promising direction, contributing to the development of more sustainable and energy-efficient machine learning techniques.

Acknowledgements This publication was partially funded by the PhD program in Mathematics and Computer Science at University of Calabria, Cycle XXXVIII with the support of a scholarship financed by DM 351/2022 (CUP H23C22000440007), based on the NRPP funded by the European Union.

Author Contributions All authors equally contributed to this work, hence they should be considered all first authors.

Funding Open access funding provided by Università della Calabria within the CRUI-CARE Agreement.

Data Availability The data that support the findings of this study are freely available on the respective links indicated in the experimental section.

Declarations

Competing interests The authors declare no competing interests.

References

Altman, S. (2025). The gentle singularity. https://blog.samaltman.com/the-gentle-singularity. Accessed 16 July 2025.

Andrae, A. S., & Edler, T. (2015). On global electricity usage of communication technology: Trends to 2030. *Challenges, 6*(1), 117–157. https://doi.org/10.3390/challe6010117

Baillon, J., & Bruck, R. E. (1996). The rate of asymptotic regularity is [CDATA[O(1/\sqrt{n})]]$O(1/\sqrt{n})$. *Lecture Notes in Pure and Applied Mathematics,* 51–82.

Baillon, J. B., & Haddad, G. (1977). Quelques propriétés des opérateurs angle-bornés et n-cycliquement monotones. *Israel Journal of Mathematics, 26*, 137–150. https://doi.org/10.1007/BF03007664

Bauschke, H. H., & Combettes, P. L. (2010). The Baillon-Haddad theorem revisited. *Journal of Convex Analysis, 17*(3 &4), 781–787. https://doi.org/10.48550/arXiv.0906.0807

Bauschke, H. H., & Combettes, P. L. (2011). Convex analysis and monotone operator theory in Hilbert spaces. Springer. https://doi.org/10.1007/978-1-4419-9467-7

Berinde, V., & Takens, F. (2007). Iterative approximation of fixed points (Vol. 1912). Springer. https://doi.org/10.1007/978-3-540-72234-2

Bock, S., & Weiß, M. (2019). Non-convergence and limit cycles in the adam optimizer. In *Numerical analysis and optimization*. Springer. https://doi.org/10.1007/978-3-030-30484-3_20

Boţ, R. I., & Nguyen, D. K. (2023). Fast Krasnoselskii-Mann algorithm with a convergence rate of the fixed point iteration of $O(1/k)$[CDATA[O(1/k)]]. *SIAM Journal on Numerical Analysis, 61*(6), 2813–2843. https://doi.org/10.1137/22M1504305

Bottou, L. (1999). On-line learning and stochastic approximations (pp. 9–42). Cambridge University Press. https://doi.org/10.1017/CBO9780511569920

Brown, T., Mann, B., Ryder, N., et al. (2020). Language models are few-shot learners. In *Proceedings of the 34th international conference on neural information processing systems*. Curran Associates Inc. https://doi.org/10.5555/3495724.3495883

Chen, L., Lin, S., Lu, X., Cao, D., et al. (2021). Deep neural network based vehicle and pedestrian detection for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems, 22*(6), 3234–3246. https://doi.org/10.1109/TITS.2020.2993926

Chen, S., Sun, P., Song, Y., & Luo, P. (2023). Diffusiondet: Diffusion model for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 19830–19843). https://doi.org/10.1109/ICCV51070.2023.01816

Cheval, H., & Leuştean, L. (2025). Linear rates of asymptotic regularity for halpern-type iterations. *Mathematics of Computation, 94*(353), 1323–1333. https://doi.org/10.1090/mcom/3991

Chidume, C. (2009). Geometric properties of Banach spaces and nonlinear iterations. Springer. https://doi.org/10.1007/978-1-84882-190-3

Colao, V., & Marino, G. (2021). On the rate of convergence of Halpern iterations. *Journal of Nonlinear Convex Analysis, 22*(12), 2639–2646.

Cominetti, R., Soto, J. A., & Vaisman, J. (2014). On the rate of convergence of Krasnosel'skiĭ-Mann iterations and their connection with sums of Bernoullis. *Israel Journal of Mathematics, 199*(2), 757–772. https://doi.org/10.1007/s11856-013-0045-4

Das, K. P., & J, C. (2023). A survey on artificial intelligence for reducing the climate footprint in healthcare. *Energy Nexus, 9*, 100167. https://doi.org/10.1016/j.nexus.2022.100167

Delanoë, P., Tchuente, D., & Colin, G. (2023). Method and evaluations of the effective gain of artificial intelligence models for reducing CO2 emissions. *Journal of Environmental Management, 331*, 117261. https://doi.org/10.1016/j.jenvman.2023.117261

Dhar, P. (2020). The carbon impact of artificial intelligence. *Nature Machine Intelligence, 2*, 423–425. https://doi.org/10.1038/s42256-020-0219-9

Diakonikolas, J. (2020). Halpern iteration for near-optimal and parameter-free monotone inclusion and strong solutions to variational inequalities. In *Conference on learning theory* (pp. 1428–1451).

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation, 10*(7), 1895–1923. https://doi.org/10.1162/089976698300017197

Dubey, S. R., Chakraborty, S., Roy, S. K., et al. (2019). diffGrad: an optimization method for convolutional neural networks. *IEEE transactions on neural networks and learning systems, 31*(11), 4500–4511. https://doi.org/10.1109/TNNLS.2019.2955777

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)* (pp. 257–269). https://doi.org/10.5555/1953048.2021068

Foglia, K. R., Colao, V., & Ritacco, E. (2024). HalpernSGD: A Halpern-inspired optimizer for accelerated neural network convergence and reduced carbon footprint. In *International symposium on methodologies for intelligent systems* (pp. 296–305). Springer. https://doi.org/10.1007/978-3-031-62700-2_26

Goodfellow, I., Pouget-Abadie, J., Mirza, M., et al. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (Vol. 27). Curran Associates, Inc. https://doi.org/10.5555/2969033.2969125

Gou, J., Yu, B., Maybank, S. J., et al. (2021). Knowledge distillation: a survey. *International Journal of Computer Vision, 129*, 1789–1819. https://doi.org/10.1007/s11263-021-01453-z

Halpern, B. (1967). Fixed points of nonexpanding maps. *Bulletin of the American Mathematical Society, 73*, 957–961. https://doi.org/10.1090/S0002-9904-1967-11864-0

Hao, K. (2019). Training a single ai model can emit as much carbon as five cars in their lifetimes. https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/. Accessed 16 July 2025.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv:1503.02531. https://doi.org/10.48550/arXiv.1503.02531

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. In *Proceedings of the 34th international conference on neural information processing systems*. Curran Associates Inc. https://doi.org/10.5555/3495724.3496298

Jain, J., Li, J., Chiu, M. T., Hassani, A., Orlov, N., & Shi, H. (2023). Oneformer: One transformer to rule universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2989–2998). https://doi.org/10.1109/CVPR52729.2023.00292

Kaur, R., & Singh, S. (2023). A comprehensive review of object detection with deep learning. *Digital Signal Processing, 132*, 103812. https://doi.org/10.1016/j.dsp.2022.103812

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International conference on learning representations, ICLR*. https://doi.org/10.48550/arXiv.1412.6980

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In *International conference on learning representations, ICLR*. https://doi.org/10.48550/arXiv.1312.6114

Körnlein, D. (2015). Quantitative results for Halpern iterations of nonexpansive mappings. *Journal of Mathematical Analysis and Applications, 428*(2), 1161–1172. https://doi.org/10.1016/j.jmaa.2015.03.020

Kuo, C. C. J., & Madni, A. M. (2023). Green learning: Introduction, examples and outlook. *Journal of Visual Communication and Image Representation, 90*, 103685. https://doi.org/10.1016/j.jvcir.2022.103685

Lee, S., & Kim, D. (2021). Fast extra gradient methods for smooth structured nonconvex-nonconcave minimax problems. *Advances in Neural Information Processing Systems, 34*, 22588–22600. https://doi.org/10.5555/3540261.3541991

Lieder, F. (2021). On the convergence rate of the Halpern-iteration. *Optimization Letters, 15*(2), 405–418. https://doi.org/10.1007/s11590-020-01617-9

Liguori, A., Manco, G., Pisani, F. S., & Ritacco, E. (2021). Adversarial regularized reconstruction for anomaly detection and generation. In *IEEE International Conference on Data Mining, ICDM* (pp. 1204–1209). IEEE. https://doi.org/10.1109/ICDM51629.2021.00145

Manco, G., Ritacco, E., Rullo, A., Saccà, D., & Serra, E. (2022). Machine learning methods for generating high dimensional discrete datasets. *WIREs Data Mining and Knowledge Discovery, 12*(2) (2022). https://doi.org/10.1002/widm.1450

Masanet, E., Shehabi, A., Lei, N., Smith, S., & Koomey, J. (2020). Recalibrating global data center energy-use estimates. *Science, 367*(6481), 984–986. https://doi.org/10.1126/science.aba3758

Patterson, D., Gonzalez, J., Le, Q., et al. (2021). Carbon emissions and large neural network training. arXiv:2104.10350. https://doi.org/10.48550/arXiv.2104.10350

Qi, H., & Xu, H. K. (2021). Convergence of Halpern's iteration method with applications in optimization. *Numerical Functional Analysis and Optimization, 42*(15), 1839–1854. https://doi.org/10.1080/01630563.2021.2001826

Rastrigin, L. A. (1974). Systems of extremal control. *Nauka*

Reddi, S. J., Kale, S., & Kumar, S. (2018). On the convergence of Adam and beyond. In *International Conference on Learning Representations (ICLR)*. https://doi.org/10.48550/arXiv.1904.09237

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics, 22*(3), 400–407. https://doi.org/10.1214/aoms/1177729586

Rosenbrock, H. H. (1960). An automatic method for finding the greatest or least value of a function. *The Computer Journal, 3*(3), 175–184. https://doi.org/10.1093/comjnl/3.3.175

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*, 533–536. https://doi.org/10.1038/323533a0

Sabach, S., & Shtern, S. (2017). A first order method for solving convex bilevel optimization problems. *SIAM Journal on Optimization, 27*(2), 640–660. https://doi.org/10.1137/16M105592X

Strubell, E., Ganesh, A., & McCallum, A. (2020). Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence, 34*(09), 13693–13696. https://doi.org/10.1609/AAAI.V34I09.7123

Sultana, F., Sufian, A., & Dutta, P. (2020). Evolution of image segmentation using deep convolutional neural network: A survey. *Knowledge-Based Systems, 201*, 106062. https://doi.org/10.1016/j.knosys.2020.106062

Tieleman, T., & Hinton, G. E. (2012). Lecture 6.5 — RMSProp: Divide the gradient by a running average of its recent magnitude. https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. Lecture slides.

Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*. https://doi.org/10.5555/3295222.3295349

Xie, L., Lee, F., Liu, L., Kotani, K., & Chen, Q. (2020). Scene recognition: A comprehensive survey. *Pattern Recognition, 102*, 107205. https://doi.org/10.1016/j.patcog.2020.107205

Xu, H. K. (2002). Another control condition in an iterative method for nonexpansive mappings. *Bulletin of the Australian Mathematical Society, 65*(1), 109–113. https://doi.org/10.1017/S0004972700020116

Xu, J., Zhou, W., Fu, Z., Zhou, H., & Li, L. (2021). A Survey on green deep learning. https://doi.org/10.48550/arXiv.2111.05193

Xu, Y., Martínez-Fernández, S., Martinez, M., & Franch, X. (2023). Energy efficiency of training neural network architectures: An empirical study. In *Hawaii international conference on system sciences*. https://doi.org/10.24251/HICSS.2023.098

Yoon, T., & Ryu, E. K. (2021). Accelerated algorithms for smooth convex-concave minimax problems with $O(1/k^2)$[CDATA[O(1/k^2)]] rate on squared gradient norm. In *International conference on machine learning* (pp. 12098–12109).

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. arXiv:1212.5701. https://doi.org/10.48550/arXiv.1212.5701

Zou, F., Shen, L., Jie, Z., Zhang, W., & Liu, W. (2019). A sufficient condition for convergences of Adam and RMSProp. https://doi.org/10.48550/arXiv.1811.09358

Authors and Affiliations

Vittorio Colao[1] · Katherine Rossella Foglia[1] · Andrea Giordano[2] · Ettore Ritacco[3] · William Spataro[1]

✉ Katherine Rossella Foglia
katherine.foglia@unical.it

Vittorio Colao
vittorio.colao@unical.it

Andrea Giordano
andrea.giordano@icar.cnr.it

Ettore Ritacco
ettore.ritacco@uniud.it

William Spataro
william.spataro@unical.it

[1] Department of Mathematics and Computer Science, University of Calabria, Via P. Bucci, 30B, Arcavacata di Rende 87036, (CS), Italy

[2] Institute for High Performance Computing and Networking, National Research Council of Italy, Via P. Bucci, 8/9C, Arcavacata di Rende 87036, (CS), Italy

[3] Department of Mathematics, Computer Science and Physics, University of Udine, Via delle Scienze, 206, Udine 33100, (UD), Italy