Non-smooth Bayesian learning for artificial neural networks

Mohamed Fakhfakh¹,² · Lotfi Chaari² · Bassem Bouaziz¹ · Faiez Gargouri¹

Received: 11 November 2021 / Accepted: 30 May 2022 / Published online: 25 June 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract
Artificial neural networks (ANNs) are being widely used in supervised machine learning to analyze signals or images for many applications. Using an annotated learning database, one of the main challenges is to optimize the network weights. A lot of work on solving optimization problems or improving optimization methods in machine learning has been proposed successively such as gradient-based method, Newton-type method, meta-heuristic method. For the sake of efficiency, regularization is generally used. When non-smooth regularizers are used especially to promote sparse networks, such as the $\ell_1$ norm, this optimization becomes challenging due to non-differentiability issues of the target criterion. In this paper, we propose an MCMC-based optimization scheme formulated in a Bayesian framework. The proposed scheme solves the above-mentioned sparse optimization problem using an efficient sampling scheme and Hamiltonian dynamics. The designed optimizer is conducted on four (4) datasets, and the results are verified by a comparative study with two CNNs. Promising results show the usefulness of the proposed method to allow ANNs, even with low complexity levels, reaching high accuracy rates of up to 94%. The proposed method is also faster and more robust concerning overfitting issues. More importantly, the training step of the proposed method is much faster than all competing algorithms.

Keywords · Artificial neural networks · Machine learning · Optimization · Hamiltonian dynamics

1 Introduction

Machine learning (ML) (Shakshuki et al. 2020) is a sub-field of artificial intelligence (AI). It has grown at a remarkable rate, attracting a great number of researchers which are intended to study how a system can perform a task through learning. In fact, an ML system does not follow instructions, but learns from experience, for example, make predictions or decisions by learning from data and keep improving the performance by examining more data. ML research achieved outstanding results on several complex cognitive tasks, including Computer Vision (Alsarhan et al. 2021), Medical diagnoses (Chaabene et al. 2021; Sree et al. 2021), Signal Processing (Jaini et al. 2021), etc.

During the last two decades, Deep Learning (DL) architectures (Devunooru et al. 2021; Goyal and Singh 2021) have demonstrated their ability to deal with more voluminous and complex data. Moreover, it has gradually become the most widely used computational approach in the field of ML, achieving outstanding results on several cognitive tasks, matching or even beating those reached by human performance. One of the benefits and difficulties at the same time of DL is the ability to learn from massive amounts of data.

In the same sense, Convolutional neural networks (CNN) (Drewek-Ossowicka et al. 2021; Sajja and Kalluri 2021; Fakhfakh et al. 2020a; Ostad-Ali-Askari et al. 2017; Ostad-Ali-Askari and Shayan 2021) are one of the state-of-art deep learning techniques. CNNs are designed to automatically and adaptively learn spatial hierarchies of features through backpropagation (Rumelhart et al. 1986) by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers. However, training a CNN is a challenging task, especially for deep architecture involving a high number of parameters (model weights) to be estimated. Sophisticated optimization algorithms need therefore

---

1 · MIRACL laboratory, University of Sfax, Sfax, Tunisia
2 · University of Toulouse, INP, IRIT, Toulouse, France
to be used. This is indeed the key step in order to fit a given architecture for learning data to minimize the error between ground truth and estimates.

In this sense, many optimization algorithms have been proposed in recent years. The performance of the algorithms strongly depends on the convexity and differentiability of the target loss function. Hence, choosing an optimization strategy that seeks to find the global optima in the learning stage is generally challenging. A non-appropriate optimization technique may for instance lead the network to lie at a local minimum during the training phase. Speeding up the optimization process is also a challenging issue for large databases.

All learning models, particularly Deep Neural Networks (DNN), are well known for being overparameterized, in fact, relatively few network weights are actually necessary to accurately learn data characteristics. They are also known to have many redundant parameters (Scardapane et al. 2017; Cheng et al. 2015). Moreover, the large number of weight parameters often leads to a heavy memory cost and computation resources. To address this computational issue, several strategies for reducing the number of network weights (weight sparsification) have been proposed, either on pre-trained models or during the training phase.

To promote sparsity of DNNs, three main categories of methods can be identified: pruning, dropout, and sparse optimization based techniques.

(i) Pruning removes weight parameters that are insensitive to the performance of established dense networks. The main drawback is linked to the pruning criteria which requires manual setups of layer sensitivity. Heuristic assumptions are also necessary throughout the pruning process (Han et al. 2015; Anwar et al. 2017).

(ii) Dropout reduces the size of networks during training by randomly dropping units along with their connections from DNNs. This method can reduce overfitting efficiently and improve the performance. Nonetheless, training a Dropout network, usually takes more time than training a standard neural network (Srivastava et al. 2014).

(iii) Optimization-based methods promote sparsity in networks by introducing a regularization term into the cost function (loss) that promotes estimation with a large number of zero weights. Sparse neural networks are being widely investigated for applications (Fan et al. 2020; Han et al. 2017) and could even achieve better performance than their original networks.

Although optimization-based sparsification is the most promising class, introducing sparse regularizers generally leads to non-differentiable cost functions. Using gradient-based techniques is therefore sub-optimal. Moreover, non-convex regularizers such as the $\ell_0$ one, are more likely to produce unbiased model with sparser solutions. However, to the best of our knowledge, there is no previous work proposing a flexible optimization-based technique allowing to handle convex and non-convex regularizers, with non-differentiable cost functions. The originality of the present work lies in:

- The Bayesian formulation of the optimization problem for DNNs. It is a powerful strategy for finding the extrema of objective functions that are expensive to evaluate.
- The proposed flexible and efficient optimization procedure which solves the non-differentiability problem and allows handling different regularizers.
- The proposed method ensures convergence to the global minimum of the formulated cost-function in contrast to other techniques (for example gradient based).

In this context, the use of Bayesian techniques has made huge strides in a variety of disciplines over the decades, and there are many practical advantages. The core concept is to use a probabilistic formulation to integrate all uncertainties throughout the model. Resorting to Bayesian techniques in our case is motivated by the ability of these methods to incorporate flexible prior models translating these uncertainties, while reducing the user input as all the model parameters and hyperparameters can be estimated from the data. Moreover, Bayesian inference using Markov Chain Monte Carlo (MCMC) methods (Chaari et al. 2014, 2016) guarantees insensitivity to local minima issues. The target space can be fully explored once sufficient mixing properties are enjoyed by the sampled chains. Such methods can also be used as an alternative to variational methods for non-convex problems, where standard optimization techniques still suffer from computational and convergence limitations.

As mentioned above, the goal of this paper is to develop a Bayesian model to minimize the target non-linear cost function. The main contribution lies in the adaptation of non-smooth Hamiltonian methods to fit sparse ANNs as weights are subject to sparsity constraints. Under the Bayesian formulation, this can be modeled using a Laplace prior distribution.

On the other hand, despite the above-mentioned advantages of Bayesian formulations and MCMC-based inference schemes, these techniques remain time-consuming. Moreover, non-smooth priors such as the Laplace one make complicated the use of standard sampling methods. This also holds for gradient-based techniques when $\ell_1$ regularizations are used. It is worth noting that $\ell_1$ regularization is typically used for signal and image recovery problems where the target data enjoys high sparsity levels, either in the original or
a transform space (Loris et al. 2007). The $\ell_1$ problem is also used in the artificial neural networks literature under both constrained (Gen et al. 2020) and Lagrangian (Ashwini and Shital 2019) formulations.

In this sense, non-smooth Hamiltonian methods allow designing of fast and efficient sampling schemes while handling non-differentiable energy functions. Our contribution is therefore focused on a new Bayesian optimization scheme adapted to train sparse neural networks without user configuration.

The rest of this paper is organized as follows. After an introduction covering the context of this research, Sect. 2 is dedicated to the state of the art. Then, the addressed problem is formulated in Sect. 3. The proposed efficient Bayesian optimization scheme is developed in Sect. 4 and validated in Sect. 5. Therefore, The discussion will be presented in Sect. 6. Finally, The conclusion and future work are drawn in Sect. 7.

## 2 Related work

The performance of a DL algorithm mainly depends on the optimization procedure used during the learning process. The essence of most architectures is to build an optimization model and learn the parameters from the available learning data. Although there is not a single solution to find the optimal set of parameters (i.e. weights) for a neural network with reasonable complexity, several researches have focused on the improvement of optimization algorithms in order to enhance the efficiency of deep learning architectures in terms of accuracy, robustness and convergence time.

Indeed, optimization methods can be divided into three categories (Sun et al. 2019; Zaheer and Shaziya 2019): (i) first-order optimization methods such as stochastic gradient; (ii) high-order optimization methods, mainly Newton’s algorithm; and (iii) heuristic/meta-heuristic derivative-free optimization methods.

First-order optimization algorithms (Xie and Zhang 2021), minimize an objective function parameterized by a model’s weight by updating the weights in the opposite direction of gradients of the objective function. When non-convex or non-differentiable functions are used, these methods may suffer from slow convergence and local minima issues. The Stochastic Gradient Descent (SGD) method (Robbins and Monro 1951; Sutskever et al. 2013) is one of the core techniques behind the success of deep neural networks since it alleviates the above-mentioned difficulties. Therefore, the limitation is using equal-sized steps for all parameters, irrespective of gradient behavior.

Adaptive moment estimation (Adam) (Kingma and Ba 2014) is one of the recent and popular variants. It allows computing adaptive learning rates (Bruno et al. 2021) for each parameter. Other variants have been widely used and have demonstrated their efficiency like AdaDelta (Sutskever et al. 2013) and Adamax (Kingma and Ba 2014).

High-order optimization (Shanno 1970; Pajarrinen et al. 2019) attracts widespread attention, but face more challenges. These methods are particularly useful where the objective function is highly non-linear and poorly conditioned. Newton-type methods use curvature information in the form of the Hessian matrix, in addition to the gradient. They are mainly introduced to extend high-order methods to large-scale data (Bollapragada et al. 2019). However, this family of methods has not been widely used in DL because of high per-iteration costs to store the inverse Hessian matrix. Other Newton-based methods have also been developed in the optimization literature (Byrd et al. 2016).

When the derivative of the objective function is not easy to calculate, gradient-free techniques can be used (Berahas et al. 2019). In this sense, heuristic and metaheuristic techniques have been widely used. The recent literature involves numerous works using such techniques for ANN training such as Particle Swarm Optimization (PSO) (Shi 2004), Genetic Algorithm (GA) (Whitley et al. 1990), Improved whale trainer (IWT) (Khishe and Mosavi 2019), Chimp Optimization Algorithm (ChOA) (Jia et al. 2021), salp swarm algorithm (SSA) (Khishe and Mohammadi 2019), Adaptive Best-Mass Gravitational Search Algorithm (ABGSA) (Mosavi et al. 2019), Dragonfly Algorithm (DA) (Khishe and Safari 2019), The Arithmetic Optimization Algorithm (AOA) (Abualigah et al. 2021). However, the implementation strategy of heuristic and metaheuristic algorithms in large-scale deep learning problems is still rarely investigated (Berahas et al. 2019). Indeed, although metaheuristic techniques may provide satisfactory solutions in a reasonable time, their main limitation lies in the difficulty to handle high-dimensional and complex optimization problems, as well as convergence guarantees and stability.

For specific cases, federated optimization has also been investigated in the recent literature developing Federated Learning (FL) techniques (Konečnỳ et al. 2016; Li et al. 2020; Yurochkin et al. 2019). Individual nodes hold a portion of the data, and the goal is to create a single common model that fits the entire distribution. A small batch gradient descent is generally used for weights optimization. FL is mainly useful when data portions can/must be kept locally on collaborating nodes. Specific variants such as fuzzy consensus have also been proposed (Polap 2021).

Several approaches are investigated in the literature to solve the optimization issue for Machine Learning field. We summarize some of the mentioned optimization methods in terms of years of publications, purpose, advantages and disadvantages in Table 1. As described, the optimization approaches could be belong to some classes such as high-order methods, and metaheuristic methods.
| Categories | Method | Years | Purpose | Advantages | Disadvantages |
|-----------|--------|-------|---------|------------|---------------|
| First-Order | SGD Robbins and Monro (1951) and Sutskever et al. (2013) | 2013 | The update parameters are calculated using a randomly sampled mini-batch. The method converges at a sublinear rate | The computational time for each update does not depend on the total number of training samples | Difficult setting of an appropriate learning rate. The solution may be trapped at the saddle point in some cases |
| First-Order | Adam Kingma and Ba (2014) | 2014 | To dynamically adjust the learning rate of each parameter, use gradient first and second order moment estimations | Stable gradient descent process. Suitable for most non-convex optimization problems with large data sets and high dimensional space | The method may not converge in some cases |
| First-Order | Adadelta Zeiler (2012) | 2012 | Change the way of total gradient accumulation to exponential moving average | Improves the ineffective learning problem in the late stage of AdaGrad. Suitable for optimizing non-stationary and non-convex problems | In the late training stage, the update process may be repeated around the local minimum |
| First-Order | Adamax Kingma and Ba (2014) | 2017 | Generalization of Adam. It is based on adaptive lower-order moment estimation | Infinite-order norm makes the algorithm stable | The penalty parameter is related to both the original and dual residuals whose value is difficult to determine |
| High-order | Newton’s method Avriel (2003) | 2003 | Calculates the inverse of the Hessian matrix to obtain faster convergence than with first-order approaches | Faster convergence than the first-order gradient method. Quadratic convergence under certain conditions | Long computing time and large storage space at each iteration |
| High-order | Quasi-Newton method Nocedal and Wright (2006) | 2006 | Uses a Hessian matrix approximation or its inverse | There is no need to calculate the Hessian matrix’s inverse matrix, which reduces the computing time. It is possible to obtain superlinear convergence in most cases | Large storage space not suitable for large-scale problems |
| High-order | Hessian free (HF) method Martens (2010) | 2010 | Sub-optimization with the conjugate gradient: avoids the computation of inverse Hessian matrix | The second-order gradient information can be used. There’s no need to calculate Hessian matrices directly. Suitable for high dimensional optimization | Computation cost to calculate the matrix–vector product increases linearly with the training data. Not appropriate for large-scale issues |
| High-order | Sochastic Quasi-Newton method Bottou et al. (2018) | 2018 | Employs techniques of stochastic optimization: e.g. online-LBFGS Schraudolph et al. (2007) and SQN Byrd et al. (2016) | Can handle large-scale issues | More complex that the stochastic gradient method |
| High-order | IWT Khishe and Mosavi (2019) | 2019 | Using a suitable spiral shape inspired by a humpback whale to improve the exploitation phase of the standard whale optimization algorithm | Stronger global search ability. It can be used to effectively solve complex constrained optimization problems | Slow convergence and easy to fall into local optimum |
| Categories                          | Method                        | Years | Purpose                                                                 | Advantages                                                                 | Disadvantages                                                                 |
|------------------------------------|-------------------------------|-------|-------------------------------------------------------------------------|----------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| Derivative-free “Meta-heuristic”   | SSA Khishe and Mohammadi (2019) | 2019  | Is a bio-inspired optimization algorithm based on swarming mechanism of salps to enhance accuracy and reliability of the solution | Faster to execute because of its lower complexity. Improved capability in avoiding local minima | It may get stuck in the local area, which results in the failure to obtain the global optimal solution |
|                                    | DA Khishe and Safari (2019)   | 2019  | Inspired by the dynamic and static swarming behaviors of dragonflies to resolve local optima stagnation when solving challenging problems | Simple and easy to implement. Having few parameters for tuning             | It does not have an internal memory that can lead to premature convergence to the local optimum |
|                                    | ABGSA Mosavi et al. (2019)    | 2019  | Used to solve the problem of irrelevant classification accuracy, and to block local minima as well as low convergence speed for Multi-Layer Perceptron Neural Network | Reduced complexity and processing time                                      | Unaffordable sampling rate. Difficult because of randomness. Greatly influenced by initial solution |
| Federated optimization             | Fuzzy consensus Polap (2021)   | 2021  | The goal of FL tasks is to learn a single global model that minimizes the empirical risk function over the entire training dataset. The authors extend FL using a fuzzy consensus method to improve large-scale group decision-making (LSGDM). | In practical use, it has a great advantage due to the possibility of quick implementation and classification of the sample even during the training process. Capable of providing the most effective solution to complex issues | Adapting centralized training workflows such as hyperparameter tuning, and interpretability tasks to the federated learning setting present roadblocks to the widespread adoption of FL in practical settings |
Moreover, the main advantages are mainly summarized in term of convergence speed, adaptation to high-dimensional optimization, precision, and prevention of local minima.

Despite the existence of some works that may be suitable for non-convex problems, they should be improved especially for large-scale problems. Indeed, one can easily notice the slower convergence and the possibility of falling into a local optimum quite far from the global optimum, most of the optimization algorithms.

On the other hand, the Bayesian framework has demonstrated its ability to provide reliable optimization models enjoying solid convergence guarantees and high stability level. Moreover, the flexibility of this framework allows introducing sophisticated constraints, such as those related to networks sparsification. As regards inference, MCMC-based techniques may also be adapted to large data problems (Quiroz et al. 2016).

A Bayesian framework assumes that all parameters are realizations of random variables. Likelihood and prior distributions are formulated to model the available information on the target parameters. An estimator for these parameters is generally derived using a maximum a posteriori (MAP) framework. However, the main difficulty is to derive analytical closed-form expressions of the estimators because of the posterior distribution form which can be complex if sophisticated priors are used, such as those promoting sparsity. In this case, MCMC techniques are generally used to sample coefficients from the target posterior (Fakhfakh et al. 2016b). The main limitation of such techniques lies in the high complexity level, especially when multidimensional data are handled. In such cases, efficient sampling methods have been proposed in the literature such as the random walk Metropolis Hastings (MH) algorithm (Lee et al. 2012) or the Metropolis-adjusted Langevin algorithm (MALA) (Roberts and Tweedie 1996). Recently, sampling using Hamiltonian dynamics (Hanson 2001) has been investigated developing the so called Hamiltonian Monte Carlo (HMC) sampling. A more sophisticated algorithm has been proposed in Chaari et al. (2016) called non-smooth Hamiltonian Monte Carlo (ns-HMC) sampling. This method solves the problem of HMC schemes that cannot be used in the case of exponential distributions with non-differentiable energy function.

The optimization methods in ANNs still face many challenges and open problems. There are mainly two major challenges with respect to data and model. The first one is insufficient training data, while the second is a non-convex objective function in DL architectures. In general, training a deep network requires large datasets to achieve good training. However, the lack of datasets to estimate the parameters in the learning models may lead to high variance (Chang et al. 2017) and overfitting (Hawkins 2004) problems. Regularization and Dropout are the most used techniques to alleviate the above-mentioned problems.

In this paper, we investigate the use of ns-HMC for the learning process of ANNs. Specifically, we propose a Bayesian optimization method to minimize the target cost function and derive the optimal weights vector. The proposed method targets regularization schemes promoting sparse networks (Mocanu et al. 2018). Indeed, gradient-based optimization methods in this case are not very efficient due to differentiability and convergence issues. Learning performances can therefore be altered. We demonstrate that using the proposed method leads to high accuracy results with different CNN architectures, which cannot be reached using competing optimizers.

### 3 Problem formulation

It is well known that weights optimization is one of the key steps to design an efficient artificial neural network. For instance, if we consider a classification problem, the ANN weight vector $W$ is updated during the learning phase by minimizing an error between the ground truth and the labels estimated using the network. An iterative procedure is generally performed, and gradient-based optimization procedures are used. For the sake of efficiency, regularization can also be performed in order to have a more accurate weights configuration.

The sparse optimization method can be used for various tasks to produce sparse solutions. The $\ell_1$ penalty added to the classification cost can be interpreted as a convexification of the $\ell_0$ penalty. In Han et al. (2015), weights with the smallest amplitude in pretrained networks are removed. Model sensitivity to weights can also be used Tartaglione et al. (2018); Gomez et al. (2019), where weights with a weak influence on the network output are pruned. The $\ell_0$ norm, which counts the number of non-zero elements, is the most intuitive form of sparse regularizers and can promote the sparsest solution. However, minimizing $\ell_0$ problem is usually NP-hard (Natarajan 1995). The $\ell_1$ norm is the most commonly used surrogate, which can be solved easily.

When applied in DNNs, sparse regularizer is supposed to zero redundant weights and thus remove unnecessary connections. However, if one aims at promoting sparse networks, sparse regularizations should be used, which makes the use of gradient-based algorithms inefficient since the error to be minimized in this case is no longer differentiable.

In this paper, we propose a method to allow weights optimization under non-smooth regularizations. Let us denote by $x$ an input to be presented to the ANN. The estimated label will be denoted by $\hat{y}(x, W)$ as a non-linear function of the input $x$ and the weights vector $W \in \mathbb{R}^N$, while the ground truth label will be denoted by $y$. 
Using a quadratic error with an $\ell_1$ regularization with $M$ input data for the learning step, the weights vector can be estimated as
\[
\hat{W} = \arg \min_W \mathcal{L}(W) = \arg \min_W \sum_{m=1}^{M} ||\tilde{y}(x^m; W) - y^m||_2^2 + \lambda ||W||_1,
\] (1)

where $\lambda$ is a regularization parameter balancing the solution between the data fidelity and regularization terms, and $M$ is the number of learning data. It is worth noting that other regularization terms can be used in Eq. (1) ($\ell_0$, $\ell_p$, ...). The $\ell_1$ norm is used to promote weights sparsity.

Since the optimization problem in Eq. (1) is not differentiable, the use of gradient-based algorithms with back-propagation is not possible and the learning process is costly and very complicated.

In Sect. 4 we present a method to efficiently estimate the weights vector without an increase of learning complexity. The optimization problem in Eq. (1) is formulated and solved in a Bayesian framework. This formulation has two main advantages. The first one is related to the flexibility of such models to handle a large panel of regularization terms through an exponential formulation to mimic the variational form in Eq. (1). The second advantage is related to the ability to design fully automatic schemes without user intervention/configuration. Indeed, this is very important, especially for complex problems where parameters fitting are complicated.

4 Bayesian optimization

As stated above, the weights optimization problem is formulated in a Bayesian framework. In this sense, the problem parameters and hyperparameters are assumed to follow probability distributions. More specifically, a likelihood distribution is defined to model the link between the target weights vector and the data, while a prior distribution is defined to model the prior knowledge about the target weights.

4.1 Hierarchical Bayesian model

According to the principle of minimizing the error between the reference label $y$ and the estimated one $\tilde{y}$, and assuming a quadratic error (first term in (1)), we define the likelihood distribution as
\[
\text{lik}(\hat{y}; W, \sigma) \propto \prod_{m=1}^{M} \exp \left( -\frac{1}{2\sigma^2} ||\tilde{y}(x^m; W) - y^m||_2^2 \right),
\] (2)

where $\sigma^2$ is a positive parameter to be set.

As regards prior information about the target weights, and to promote sparsity of the estimated vector (and hence the sparsity of the deep network), a common choice is to resort to an $\ell_1$ penalization. Under a Bayesian framework, the Laplace distribution can be used.

\[
f(W; \lambda) \propto \prod_{k=1}^{N} \exp \left( -\frac{||W[k]||_1}{\lambda} \right),
\] (3)

where $\lambda$ is a hyperparameter to be set. This prior allows us to introduce exactly the same prior information as the $\ell_1$ norm in Eq. (1).

By adopting a MAP approach, we first need to express the posterior distribution. Based on the defined likelihood and prior, this posterior writes:
\[
f(W; y, \sigma, \lambda) \propto f(y; W, \sigma) f(W; \lambda)
\propto \prod_{m=1}^{M} \exp \left( -\frac{1}{2\sigma^2} ||\tilde{y}(x^m; W) - y^m||_2^2 \right)
\times \prod_{k=1}^{N} \exp \left( -\frac{||W[k]||_1}{\lambda} \right).\] (4)

It is clear that this posterior is not straightforward to handle in order to derive a closed-form expression of the estimate $\hat{W}$. For this reason, we resort to a stochastic sampling approach in order to numerically approximate the posterior, and hence to calculate an estimator for $\hat{W}$. The following Section details the adopted sampling procedure.

4.2 Hamiltonian Monte Carlo

Hamiltonian Monte Carlo (Li et al. 2015) is a class of sampling algorithms inspired by the Hamiltonian dynamics. It is a reformulation of the theory of classical mechanics which is intended to describe the motion of objects, and therefore to model the dynamic physical systems (Alder and Wainwright 1959). A dynamic particle of mass $m$ can be characterized essentially by its positions $W$ and momentum $q = mv$, where $v = \frac{\partial W}{\partial t}$ which represent the velocity of the particle. The Hamiltonian system models the total Kinetic of this particle, respectively, the potential energy $E(W)$ and the Kinetic energy $K(v) = \frac{1}{2}mv^2$, which can likewise be expressed as a function of momentum by $K(q) = \frac{1}{2}q^2$. Thus, the Hamiltonian $H(W, q)$ can be expressed as:
\[
H(W, q) = E(W) + K(q).
\] (5)

and the dynamics of the particle can be specified by a set of coupled differential equations (Neal 2011).
\[ \frac{dq}{dt} = \frac{\partial H}{\partial W} \]  \hspace{1cm} (6)

\[ \frac{dW}{dt} = -\frac{\partial H}{\partial q} \]  \hspace{1cm} (7)

From any time interval of duration \( s \), these equations define a mapping \( T_s \), from the state at any time \( t \) to the state at time \( t + s \).

When estimating a random variable \( \theta \) with probability density function \( f(\theta) \) in the HMC method, we define an auxiliary momentum variable \( q \). The pdf of the Hamiltonian dynamic energy defined in (5) is given by

\[ f_q(W, q) \propto \exp(-H(W, q)) \]

\[ \propto f(W; \theta) \exp(-K(q)). \]  \hspace{1cm} (8)

HMC methods iteratively proceed with updating \( W \) and \( q \) by sampling according to the distribution (8). The sampling of the model is performed by two-step. The first sampling \( q \) according to the multivariate Gaussian distribution \( \mathcal{N}(0, I_N) \), where \( I_N \) is the \( N \times N \) identity matrix. The second step, updates both momentum \( q \) and position \( W \) by proposing two candidates \( W^* \) and \( q^* \). These two candidates are generated by simulating the Hamiltonian dynamics, which are discretized using some discretization techniques which is leapfrog method (Hanson 2001). The discretization can be performed using \( L_f \) steps of the leapfrog method with a step-size \( \epsilon > 0 \). \( L_f \) can either be manually fixed or automatically tuned (Wang et al. 2013).

### 4.3 ns-HMC

As the HMC is a successful approach for sampling from continuous densities, however, it has difficulty simulating Hamiltonian dynamics with non-smooth functions, leading to poor performance. A novel scheme called Non-smooth Hamiltonian Monte Carlo (ns-HMC) has been proposed in Chaari et al. (2016) to make feasible the use of Hamiltonian dynamics of sampling even for target distributions with non-smooth energy functions. The sampling technique relies on some interesting results from convex optimization and Hamiltonian Monte Carlo methods. The main idea of the ns-HMC scheme is to modify the leapfrog discretization scheme by introducing a step calculation the proximity operator of \( E_\theta \). The detailed ns-HMC scheme is given by algorithm 1, where \( L_f \) represent the number of leapfrog step and \( \epsilon \) is the stepsize.

Algorithm 1: ns-HMC algorithm Chaari et al. (2016)

- Initialize with some \( W^{(0,0)} \), set the iteration number \( r = 0 \), \( L_f \) and \( \epsilon \).

for \( r = 1 \ldots , S \) do

- Sample \( q^{(r,0)} \sim \mathcal{N}(0, I_N) \);

- Compute \( q^{(r,\frac{1}{2})} = q^{(r,0)} - \frac{\epsilon}{2} \left[ W^{(r-1,0)} - \text{prox}_{E_\theta}(W^{(r-1,0)}) \right] \);

- Compute \( W^{(r,\epsilon)} = W^{(r-1,0)} + \epsilon q^{(r,\frac{1}{2})} \);

for \( l_f = 1 \text{ to } L_f - 1 \) do

* Compute \( q^{(r,(l_f + \frac{1}{2})\epsilon)} = q^{(r,l_f\epsilon)} - \frac{\epsilon}{2} \left[ W^{(r,l_f\epsilon)} - \text{prox}_{E_\theta}(W^{(r,l_f\epsilon)}) \right] \);

* Compute \( W^{(r,(l_f + 1)\epsilon)} = W^{(r,l_f\epsilon)} + \epsilon q^{(r,(l_f + \frac{1}{2})\epsilon)} \);

end

- Compute \( q^{(r,(L_f + \frac{1}{2})\epsilon)} = q^{(r,L_f\epsilon)} - \frac{\epsilon}{2} \left[ W^{(r,L_f\epsilon)} - \text{prox}_{E_\theta}(W^{(r,L_f\epsilon)}) \right] \);

- Apply standard MH acceptance rule by taking \( q^* = q^{(r,L_f\epsilon)} \) and \( W^* = W^{(r,L_f\epsilon)} \);

end

However, analytic calculation of the proximity operator for a wide class of energy functions is not possible. This drawback prevents the use of the ns-HMC algorithm in the case of sparse target distributions where the proximity operator of the energy function is difficult to be calculated.

In order to solve this problem, in Chaari et al. (2017), a modified ns-HMC sampling scheme, called general ns-HMC, have been proposed involving a Bayesian calculation of the proximity operator. Thus, instead of calculating the proximity operator at each step as shown in the Algorithm 1 that can be led to an increased computational cost, with the general ns-HMC scheme, the calculation of the proximity operator is only calculated at the initialization step. The calculated value is then used to update the proximity operator value at different points. Another advantage of the method is that it does not depend on the initial point where the proximity operator is calculated first.

### 4.4 Hamiltonian sampling

Let us denote \( a = \frac{\lambda}{\sigma^2}, \theta = \{\sigma^2, \lambda\} \). For a weight \( W^k \) we define the following energy function

\[ E_\theta^k(W^k) = \frac{\alpha}{2} \sum_{m=1}^{M} \| y(x^m; W) - y^{(m)} \|_2^2 + \| W^k \|_1. \]  \hspace{1cm} (9)

The posterior in (4) can be reformulated as
\( f(W|y, \theta) \propto \exp \left( - \sum_{k=1}^{N} E^{k}_{\theta}(W^{k}) \right) \). 

(10)

To sample according to this exponential posterior, and since direct sampling is not possible due to the form of the energy function \( E^{k}_{\theta} \), Hamiltonian sampling is adopted. Indeed, Hamiltonian dynamics (Hanson 2001) strategy has been widely used in the literature to sample high dimensional vectors. However, sampling using Hamiltonian dynamics requires computing the gradient of the energy function, which is not possible in our case due to the \( \ell^{1} \) term. To overcome this difficulty, we resort to a non-smooth Hamiltonian Monte Carlo (ns-HMC) strategy. Indeed, this strategy requires calculation of the proximity operator only at an initial point and uses the shift property (Moreau 1965) to deduce the proximity operator during the iterative procedure. As regards the proximity operator calculation, let us denote by \( G_{\ell}(W^{k}) \) the gradient of the quadratic term of the loss function \( L \) with respect to the weight \( W^{k} \). Let us also denote by \( \varphi(W^{k}) = \| W^{k} \|_{1} \). Following the proximity operator standard definition, we can write for any real \( z \)

\[
\text{prox}_{\varphi}(z) = p \iff z - p \in \partial \varphi(p).
\]

(11)

Straightforward calculations lead to the following expression of the proximity operator:

\[
\text{prox}_{\varphi}(z) = p \iff z - p \in \partial \varphi(p) + \frac{\alpha}{2} G_{\ell}(W^{k})
\]

\[
\iff z - p \in \partial \varphi(p) + \frac{\alpha}{2} G_{\ell}(W^{k})
\]

\[
\iff p = \text{prox}_{\varphi} \left( z - \frac{\alpha}{2} G_{\ell}(W^{k}) \right).
\]

(12)

Since \( \text{prox}_{\varphi} \) is nothing but the soft thresholding operator (Chaux et al. 2007), the proximity operator in (11) can be easily calculated once a single gradient step is applied (backpropagation) to calculate \( G_{\ell}(W^{k}) \).

We therefore propose the following leapfrog discretization scheme to be integrated in Algorithm 1, where \( P_{0} = \text{prox}_{\varphi}(z^{(0,0)}) \):

\[
q^{(r, L_{1} + \frac{1}{2})} = q^{(r, L_{1})} - \frac{\epsilon}{2} \left[ 2z^{(r, L_{1})} - z^{(0,0)} - P_{0} \right]
\]

\[
q^{(r, L_{1})} = q^{(r, L_{1})} + \epsilon \frac{q^{(r, L_{1} + \frac{1}{2})}}{2}
\]

\[
q^{(r, L_{1} + \frac{1}{2})} = q^{(r, L_{1})} - \frac{\epsilon}{2} \left[ 2z^{(r, L_{1})} - z^{(0,0)} - P_{0} \right].
\]

(13)

The Gibbs sampler resulting from the proposed leapfrog discretization scheme is summarized in Algorithm 2.

The proposed candidates are given by \( q^{*} = q^{(r, c_{L_{1}})} \) and \( z^{*} = z^{(r, c_{L_{1}})} \) after \( L_{f} \) leapfrog steps. These candidates are then accepted based on the standard MH rule, i.e., with the following probability

\[
\min \left\{ 1, \exp \left[ H(z^{(r)}, q^{(r)}) - H(z^{(s)}, q^{(s)}) \right] \right\}
\]

(14)

where \( H \) is the Hamiltonian defined in (5).

After convergence, Algorithm 2 provides chains of coefficients sampled according to the target distribution of each \( W^{k} \). These chains can be used to compute an MMSE (minimum mean square error) estimator after discarding the samples corresponding to the burn-in period.

It is worth noting that hyperprior distributions can be put on \( \lambda \) and \( \sigma \) in order to integrate them in the hierarchical Bayesian model. These hyperparameters can therefore be estimated from the data at the expense of some additional complexity.

5 Experimental validation

In order to validate the proposed method, five image classification experiments are conducted using four datasets: two COVID-19 datasets including Computed tomography (CT) images for simple (Angelov and Almeida Soares 2020) and challenging classification (Yang et al. 2020), and two standard datasets, namely Fashion-MNIST (Xiao et al. 2017) and
CIFAR-10 Recht et al. (2018). Table 2 illustrates the setting details of the different datasets.

In order to compare the proposed method with the state of the art, three kinds of optimizers are used: (i) MCMC-based methods, precisely the standard Metropolis-Hastings (MH) algorithm (Chib and Greenberg 1995) and its random walk variant (rw-MH), (ii) the most popular and widely used techniques: Adam, Adamax, SGD, Adadelta, and (iii) three metaheuristics algorithms: Improved Whale Trainer, Dragonfly Algorithm, and Salp Swarm Algorithm. The parameters setting of all these algorithms is detailed in Table 3.

As regards coding, we used python programming language with Keras and Tensorflow libraries on an Intel(R) Core(TM) i7-2720QM CPU 2.20GHZ architecture with 16 GB memory.

### 5.1 ConvNet models

Three CNN architectures are used in this study. Like the LeNet model (LeCun et al. 1998), the first one includes three convolutional (Conv-32, Conv-64 and Conv-128) and two fully-connected (FC-64 and FC-softmax). The second one has six convolutional (2XConv-32, 2XConv-64 and 2XConv-128) and three FC layers (FC-128, FC-64 and FC-softmax) that are organized similarly to VGG-Net (Muhammad et al. 2018). These architectures are shown in Table 4. The third one is a more deeper CNN used for comparison includes 15 convolutional layers and a FC-softmax layer (see Sect. 5.7 for more details).

All of them involve convolutional layers with $3 \times 3$ Kernel filters in addition to $2 \times 2$ max-pooling, with stride size equal to 1. All layers in the different configurations used ReLU as an activation function except the output layer.

As deep neural networks can easily overfit when trained with small datasets, the used CNNs are extended with three regularizing techniques:

### Table 2 Setting details of the used datasets.

| Dataset                  | Training set | Test set | # Classes |
|--------------------------|--------------|----------|-----------|
| CT images for simple classification | 1210         | 430      | 2         |
| CT images for challenging classification | 566          | 180      | 2         |
| Fashion-MNIST            | 48,000       | 12,000   | 10        |
| CIFAR-10                 | 50,000       | 10,000   | 10        |

### Table 3 Parameters setting for benchmark algorithms

| Algorithm | Parameters | Description                        | Values |
|-----------|------------|------------------------------------|--------|
| MH        | $\sigma$   | Stand. dev. of the proposal normal distribution | 3      |
|           | $\sigma$   | Stand. dev. of the proposal normal distribution | 5      |
|           | lr         | Learning rate                       | $10^{-3}$ |
| rw-MH     | $\beta_1$  | 1st moment estimates exponential decay rate | 0.9    |
|           | $\beta_2$  | 2nd moment estimates exponential decay rate | 0.999  |
|           | $\epsilon$ | Numerical stability constant         | $10^{-08}$ |
|           | lr         | Learning rate                       | $10^{-3}$ |
| Adam      | momentum   | Acceleration rate                   | 0.8    |
|           | decay      | Learning rate decay over each update | $10^{-6}$ |
|           | lr         | Learning rate                       | $10^{-3}$ |
| SGD       | rho        | Decay rate                          | 0.95   |
|           | $\epsilon$ | Numerical stability constant         | $10^{-08}$ |
|           | lr         | Learning rate                       | $10^{-3}$ |
| Adadelta  | $\beta_1$  | 1st moment estimates exponential decay rate | 0.9    |
|           | $\beta_2$  | 2nd moment estimates exponential decay rate | 0.999  |
|           | $\epsilon$ | Numerical stability constant         | $10^{-08}$ |
|           | minv       | Lower bound                         | $-2$   |
| IWT       | maxv       | Upper bound                         | 2      |
|           | size       | Number of particles                 | 30     |
|           | p_s        | Spiral parameter                    | 3      |
|           | minv       | Lower bound                         | $-2$   |
| DA        | maxv       | Upper bound                         | 2      |
|           | size       | Number of particles                 | 15     |
|           | minv       | Lower bound                         | $-2$   |
| SSA       | maxv       | Upper bound                         | 2      |
|           | size       | Number of particles                 | 30     |
Non-smooth Bayesian learning for artificial neural networks

• Batch normalization (Ioffe and Szegedy 2015): deals with the change of the feature space distribution along with the model during the training. The input of the layer is normalized to be zero-mean with unitary variance. This step not only acts as a regularizer, but also allows for faster training, higher learning rates, and less dependence on weights initialization.

• $\ell_1$ Regularization (Xu et al. 2010): $\ell_1$ regularization is the preferred choice when having a high number of features as it provides sparse solutions. It allows obtaining the computational advantage because features with zero coefficients can be avoided. In our case, the used regularization parameter was set to $\lambda = 0.001$.

• Dropout (Srivastava et al. 2014): random disabling of neurons during training with probability (or percentage) $p$. Temporarily ignoring some activations forces the other neurons to learn a more robust representation of the input data while reducing the sensitivity of specific neurons. In our study, the dropout rate is set by cross validation to $p = 0.35$.

5.2 Sparsity and stability analysis

In this section, we evaluate the sparsity and the stability of the estimation of the weights with different values of $\lambda$ and compare to Adam as a state of the art optimizer. The CNN_1 architecture is applied using the CT Covid-19 image database.

Table 5 reports accuracy, computational time, and $\ell_1$ norm of the estimated weights using different values of the regularization parameter $\lambda$ over 10 Monte Carlo runs. To further evaluate the sparsity level, Table 5 also reports the $\ell_0$ norm of the estimated weights.

Bold values indicate the importance of the obtained results of our approach compared to competing algorithms.
pseudo-norm values (number of non-zeros). Standard deviations over the 10 runs are provided in the table.

The obtained scores clearly indicate that our method provides estimates with a higher sparsity level in comparison to Adam: the number of non-zero weights is significantly lower than the Adam optimizer. The proposed method provides estimates with 14% higher sparsity level. Moreover, the reported low standard variation values in Table 5 clearly indicate good stability properties of the proposed method with respect to random sampling in the MCMC procedure, which confirms the good convergence properties. This stability holds for accuracy, sparsity, sensitivity, specificity, and computational time. Moreover, the same conclusions hold for all tested values. In other words, the same conclusions hold for different network sparsity levels.

5.3 Experiment 1: COVID-19 classification using CT images

This section studies the performance of our optimizer for classifying CT data into normal and Covid-19 cases using a public dataset of CT scans for SARS-CoV-2 identification. The dataset is made up of of 1252 CT scans of size $230 \times 230$ that are positive for SARS-CoV-2 infection (COVID-19) and 1230 CT scans for negative patients. These data have been collected from real patients in hospitals from São Paulo, Brazil.1

Table 6 reports accuracy, loss, sensitivity, specificity and computational time, for all optimizers with CNN_1 and CNN_2. The reported scores indicate that our ns-HMC outperforms the competing optimizers, including metaheuristic methods in terms of learning precision, and hence classification performances. Accuracy values even show a slight advantage in favor of the proposed method for both the CNNs. The lack of performance obtained by the metaheuristic methods (IWT, DA, and SSA) caused by early convergence experienced during the process of finding optimum value. This phenomenon is due to the lack of population diversity and known as premature convergence. The proposed method enjoys faster convergence properties while reaching global optimum due to the Bayesian formulation with lower loss rates.

The behavior of the algorithms during the training step is displayed in Figs. 1, and 2 where the curves clearly indicate a convergence with high accuracy rate for most optimizers. A significant difference between training and loss curves may indicate potential overfitting obtained with some optimizers. This gap is reduced using the proposed optimizer. Interestingly, the same behavior is observed for both CNN models. Moreover, the accuracy increase between CNN_1 and CNN_2 is almost the same for all optimizers (see Table 6). The higher performance of the proposed method can be explained by a better exploration of the searching space due to the Bayesian formulation and the efficient sampling scheme, which also helps reducing the computational time. Indeed, ns-HMC sampling integrates a gradient information related to the geometry of the target distribution, which finally leads to a faster convergence of the used sampler.

It is worth noting that the curves irregularity for Bayesian techniques (proposed method, MH and rw-MH) are due to the random sampling effect. No monotonic behavior is expected.

5.4 Experiment 2: challenging case

A more challenging classification case is addressed in this experiment. The same CNNs are used for CT images

---

1 https://www.kaggle.com/plameduardo/sarscov2-ctscan-dataset.
Fig. 1 Experiment 1: Train and test curves using CNN_1.

(a): ns-HMC  (b): MH  
(c): rw-MH  (d): Adam  
(e): SGD  (f): Adadelta  
(g): Adamax  (h): IWT  
(e): DA  (f): SSA
Fig. 2 Experiment 1: Train and test curves using CNN_2

(a): ns-HMC
(b): MH
(c): rw-MH
(d): Adam
(e): SGD
(f): Adadelta
(e): Adamax
(f): IWT
(e): DA
(f): SSA
classification to identify COVID-19 infections from other pneumonia. In contrast to Experiment 1, this task is challenging due to the rich content of CT images and similarity between COVID-19 infection and other pneumonia. The COVID-CT dataset contains 349 CT images positive for COVID-19 belonging to 216 patients and 397 CT images that are negative for COVID-19. The dataset is open-sourced to the public. We used 566 images for the train and 180 images for the test with size of $230 \times 230$.

The reported scores in Table 7 indicate that the proposed method clearly outperforms the competing optimizers in both models to solve this challenging classification problem. Moreover, severe performance decrease is observed for some optimizers. IWT, DA, and SSA achieved an accuracy slightly better than gradient and MCMC-based methods. DA algorithm is the better performer compared to all the competing algorithms on this dataset, but has an accuracy less than our nc-HMC optimizer of around 6%. This is mainly due to this challenging classification, which leads to a more complex learning process.

### 5.5 Experiment 3: Fashion-MNIST image classification

In this scenario, the learning performance using the competing optimization algorithms is evaluated using the standard Fashion-MNIST dataset. A training set of 60,000 images is used, while the test set was made up of 10,000 images. Each example is a 28 grayscale image, associated with a label from 10 classes, with 7000 images per class. For the model training, we used 48,000 images for the train set and 12,000 for the test.

The obtained results for the fashion-MNIST dataset is given in Table 8. All the competing optimizers did not

### Table 7 Experiment 2: Results for CT image classification - challenging case - using CNN_1 and CNN_2 (computational time in min, accuracy, loss, sensitivity and specificity)

| Optimizers | CNN_1 | CNN_2 |
|------------|-------|-------|
| ns-HMC     | 40    | 53    |
| MH         | 71.4  | 92.4  |
| rw-MH      | 59    | 94.8  |
| Adam       | 58    | 81    |
| SGD        | 59    | 82.2  |
| Adadelta   | 61.8  | 87.6  |
| Adamax     | 60.6  | 90    |
| IWT        | 54    | 90    |
| DA         | 57    | 87    |
| SSA        | 51    | 83    |

Bold values indicate the importance of the obtained results of our approach compared to competing algorithms.

### Table 8 Experiment 3: Results for Fashion-MNIST image classification—challenging case—using CNN_1 and CNN_2 (computational time in min, accuracy, loss, sensitivity and specificity)

| Optimizers | CNN_1 | CNN_2 |
|------------|-------|-------|
| ns-HMC     | 70.5  | 308.4 |
| MH         | 166.2 | 745.8 |
| rw-MH      | 183.6 | 797.4 |
| Adam       | 156.6 | 444   |
| SGD        | 164.4 | 452.4 |
| Adadelta   | 169.8 | 439.8 |
| Adamax     | 149   | 448.2 |
| IWT        | 180.2 | 486   |
| DA         | 174   | 469   |
| SSA        | 165.7 | 453   |

Bold values indicate the importance of the obtained results of our approach compared to competing algorithms.

---

2 https://www.kaggle.com/luisblanche/covidct.
perform well on this dataset which could be because of the size of the dataset. Our optimizer was the better performer with an accuracy up to 93% for both architectures, significantly better than all the competing optimizers. Indeed, as reported in Table 8, the computational time of all the competing algorithms is generally around 160 min for the CNN_1, more than twice the time needed for the proposed method. The same conclusion for the deep architecture CNN_2.

5.6 Experiment 4: CIFAR-10 image classification

In this scenario, the learning performance using the competing optimization algorithms is evaluated using the standard CIFAR-10 dataset. The CIFAR-10 dataset consists of 60,000 32 × 32 color images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images. The dataset is divided into five training batches and one test batch, each with 10,000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Classification results for the CIFAR-10 dataset are given in Table 9.

The same conclusion can be drawn as that of the Fashion-MNIST dataset. The proposed Bayesian optimizer showed great overall performance even if more classes are considered compared to all competing optimizers.

5.7 Comparison on deep CNN

This section studies the performances of our ns-HMC method using a deep CNN on the standard Fashion-MNIST dataset. The proposed deep CNN is deeper than CNN_1 and CNN_2. It is made up of 15 convolutional layers (5 × Conv3 × 3-32, 5 × Conv3 × 3-64, 5 × Conv3 × 3-128) and a FC-softmax layer. All of them involve convolutional layers with 3 × 3 Kernel filters in addition to 2 × 2 max-pooling, with stride size equal to 1.

The use of a deep CNN validates the effectiveness and robustness of our approach in terms of accuracy, loss, sensitivity, and specificity criteria as shown in Table 10. Furthermore, most of the competing optimizers clearly indicate an overfitting effect like SGD, and Adadelta, in contrast to the proposed method. Hence, one can easily notice the highest global accuracy of our Bayesian optimizer regardless of the depth architectures.

6 Discussion

In this paper, we proposed a novel optimization method built in a Bayesian framework. The proposed algorithm relies on a Hamiltonian Monte Carlo scheme to solve the subsequent
optimization problem involving sparsity constraints, while being adapted to large data problems under solid convergence guarantees. The proposed method has been validated on four different datasets in order to assess: (i) efficiency on a classification case where competing optimizers provide good results, (ii) fast convergence properties, and (iii) robustness with respect to the sample size.

A gold standard validation on the widely used both Fashion-MNIST and CIFAR-10 databases have also been performed. Furthermore, three kinds of comparisons are performed: (i) with respect to other state of the art optimizers (Adam, SGD and Adadelta), (ii) with respect to other MCMC-based techniques (MH and rw-MH), and (iii) with respect to the novel metaheuristics methods (IWT, DA, and SSA).

Though deep neural network has good expressive ability, its large model parameters which bring a great burden on calculation is still a problem remain to be solved. This problem hinders the widely deployment of DNNs-based applications, so it is primary of reducing the model parameters without losing performance. Sparsing neural networks is one of the methods to effectively reduce complexity which can improve efficiency and generalizability.

Empirical evidence shows that deep architectures often require to be over-parametrized (having more parameters than training examples) in order to be successfully trained (Brutzkus et al. 2017; Mhaskar and Poggio 2016). Indeed, the use of these networks is useful to extract more implicit characteristics that leads to good precision of the model, and hence reduce the overfitting effect. However, once input-output relations are properly represented by a complex network, such a network may form a starting point in order to find a simpler, sparser, but sufficient architecture (Brutzkus et al. 2017; Mhaskar and Poggio 2016). Three CNNs which one is deeper than others have been used with $\ell_1$ regularization to promote sparse networks. This allowed us to analyse how the proposed optimizer behaves when the complexity level of the network increases. From one side, experiments showed that our optimizer enjoys better sparsity levels in terms of $\ell_1$ and $\ell_0$ norms.

Experiments lead one to conclude that the proposed non-smooth Hamiltonian sampling scheme provides faster and more accurate convergence with lower overfitting effects. The obtained gain is not only due to the Bayesian formulation, but also to the efficient inference scheme. From another side, results showed that better accuracy is always obtained with the most sophisticated CNN in spite of the additional complexity.

The proposed method speeds up the learning time for both architectures. Indeed, for the investigated challenging classification problem (Experiment 2), a deeper network did not solve the overfitting problem with standard optimizers, while our method overcomes this limitation by providing more accurate optimization of the target criterion, and only need simple neural net architecture to produce high accuracy.

The sensitivity and specificity scores justify the stable behavior of our ns-HMC at different used datasets. Moreover, loss and accuracy obtained with our method do not fall when data complexity increases, in contrast to other competing optimizers.

Metaheuristic methods have attracted considerable attention in the last years, mainly due to their simple heuristics and ability to optimize nondifferentiable functions. Indeed, the performance of a metaheuristic can only be examined in a problem where these are applicable. They do not guarantee a global optimum, but it is near to a global best solution. We can further conclude from the results that our optimization strategy is insensitive to local minima unlike to metaheuristic which are already rarely conducted optimize DL method (Rere et al. 2016).

The main limitation of our proposed method concerns that is trained on CPU and not GPU which is commonly used for deep learning. Moreover, $\lambda$ is a hyperparameter to be set as a fixed value in our method. The choice of the best value of $\lambda$ depends on the used model for a training dataset. One of the challenges in the future is to extend our optimizer by estimating this hyperparameter.

7 Conclusion

In this paper, we proposed a new Bayesian optimization method to fit weights for sparse artificial neural networks. The proposed method relies on Hamiltonian dynamics with non-smooth regularizations, using a plug and play procedure. The proposed ns-HMC optimizer showed promising results with good classification performances and high generalization properties, in addition to low computational time in comparison with all the competing algorithms including the commonly used optimizers in DL and the recent metaheuristics methods.

The use of standard datasets (Fashion-MNIST and CIFAR-10) with multiple images confirms the stability of our optimizer in terms of accuracy. The different experiments carried out have showed the good generalization of our ns-HMC and their ability to be applied in various fields.

We can extend our experiments in the future by integrating the segmentation methods in Deep Learning by using our optimizer. Moreover, We will focus on investigating parallel implementation of the proposed method to further decrease computational time. Investigating the use of the proposed method on recurrent networks will also be considered.
References

Abualigah L, Diabat A, Mirjalili S, Abd Elaziz M, Gandomi AH (2021) The arithmetic optimization algorithm. Comput Methods Appl Mech Eng 376:113609

Alder BJ, Wainwright TE (1959) Studies in molecular dynamics I. General method. J Chem Phys 31(2):459–466

Alsalhan A, Alaouthman M, Alshdaifat E, Al-Ghuwairi A-R, Al-Dubai A (2021) Machine learning-driven optimization for svm-based intrusion detection system in vehicular ad hoc networks. J Ambient Intell Humaniz Comput Accesses 557:1–10

Angelov P, Almeida Soares E (2020) Sars-cov-2 ct-scan dataset: a large dataset of real patients ct scans for sars-cov-2 identification. medRxiv. https://doi.org/10.1101/2020.04.24.20078584

Anwar S, Hwang K, Sung W (2017) Structured pruning of deep convolutional neural networks. ACM J Emerg Technol Comput Syst (JETC) 13(3):1–18

Ashwini R, Shital R (2019) Deep neural network regularization for feature selection in learning-to-rank. IEEE Access 7:53988–54006

Avriel M (2003) Nonlinear programming: analysis and methods. Courier Corporation, Chelmsford

Berahas AS, Byrd RH, Nocedal J (2019) Derivative-free optimization of noisy functions via quasi-newton methods. SIAM J Optim 29(2):965–993

Bollapragada R, Byrd RH, Nocedal J (2019) Exact and inexact sub-sampled newton methods for optimization.IMA J Numer Anal 39(2):545–578

Bottou L, Curtis FE, Nocedal J (2018) Optimization methods for large-scale machine learning. Siam Rev 60(2):223–311

Bruno G, Antonelli D, Stadnicka D (2021) Evaluating the effect of learning rate, batch size and assignment strategies on the production performance. J Ind Prod Eng 38(2):137–147

Brutzkus, A., Globerson, A., Malach, E., and Shalev-Shwartz, S. (2017). Sgd learns over-parametrized networks that provably generalize on linearly separable data. arXiv preprint arXiv:1710.10174

Byrd RH, Hansen SL, Nocedal J, Singer Y (2016) A stochastic quasi-newton method for large-scale optimization. SIAM J Optim 26(2):1008–1031

Chaabene S, Bouaziz A, Bouaziz B, Hokelmann A, Ammar A, Chaari L (2021) Convolutional neural network for drowsiness detection using EEG signals. Sensors 21(5):1–19

Chaari L, Batatia H, Dobigeon N, Tourneret J (2014) A hierarchical sparsity-smoothness Bayesian model for l0+l1+l2 regularization. In: IEEE international conference on acoustics, speech and signal processing (ICASSP), Florence, Italy, pp 1901–1905

Chaari L, Tourneret J-Y, Chaux C, Batatia H (2016) A Hamiltonian Monte Carlo method for non-smooth energy sampling. IEEE Int Trans Signal Process 64(21):5585–5594

Chaari L, Tourneret J-Y, Batatia H (2017) A general non-smooth Hamiltonian Monte Carlo scheme using Bayesian proximity operator calculation. In: European signal processing conference EUSIPCO, Kos, Greece. IEEE, pp 1260–1264

Chang H-S, Learned-Miller E, McCallum A (2017) Active bias: training more accurate neural networks by emphasizing high variance samples. arXiv preprint arXiv:1704.07433

Chaux C, Combettes P, Pesquet J, Wajs V (2007) A variational formulation for frame-based inverse problems. Inverse Probl 23(4):1495

Cheng Y, Yu FX, Feris RS, Kumar S, Choudhary A, Chang S-F (2015) An exploration of parameter redundancy in deep networks with circulant projections. In: Proceedings of the IEEE international conference on computer vision (ICCV), Santiago, Chile, pp 2857–2865

Chib S, Greenberg E (1995) Understanding the metropolis-hastings algorithm. Am Stat 49(4):327–335

Devunooru S, Alsadoon A, Chandana P, Beg A (2021) Deep learning neural networks for medical image segmentation of brain tumours for diagnosis: a recent review and taxonomy. J Ambient Intell Humaniz Comput 12(1):455–483

Drewek-Ossowicka A, Pietrojaj M, Ruminski J (2021) A survey of neural network usage for intrusion detection systems. J Ambient Intell Humaniz Comput 12(1):497–514

Fakhfakh M, Bouaziz B, Gargouri F, Chaari L (2020a) Prognet: COVID-19 prognosis using recurrent and convolutional neural networks. Open Med Imaging J 12(1):1–7

Fakhfakh M, Chaâri L, Fakhfakh N (2020b) Bayesian curved lane estimation for autonomous driving. J Ambient Intell Humaniz Comput. 11(10):4133–4143

Fan Y, Yu J, Mei Y, Zhang Y, Fu Y, Liu D, Huang TS (2020) Neural sparse representation for image restoration. arXiv:2006.04357

Gen L, Yuantao G, Jie D (2020) The efficacy of l1 regularization in two-layer neural networks. https://doi.org/10.48550/arXiv.2010.01048

Gomez AN, Zhang I, Kamalakara SR, Madaan D, Swersky K, Gal Y, Hinton GE (2019) Learning sparse networks using targeted dropout. arXiv preprint arXiv:1905.13678

Goyal S, Singh R (2021) Detection and classification of lung diseases for pneumonia and COVID-19 using machine and deep learning techniques. J Ambient Intell Humaniz Comput Accesses 2685:1–21

Han S, Pool J, Tran J, Dally WJ (2015) Learning both weights and connections for efficient neural networks. arXiv preprint arXiv:1506.02626

Han L, Lin H, Jun L (2017) Remote sensing image classification based on convolutional neural networks with two-fold sparse regularization. In: IEEE international geoscience and remote sensing symposium (IGARSS), Fort Worth, TX, USA, 2153-7003, pp 992–995

Hanson K (2001) Markov Chain Monte Carlo posterior sampling with the hamiltonian method. In: Medical imaging 2001: image processing, vol 4322. International Society for Optics and Photonics, pp 456–467

Hawkins DM (2004) The problem of overfitting. J Chem Inf Comput Sci 44(1):1–12

Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. PMLR, pp 448–456

Jaini SNB, Lee D, Lee S, Kim M, Kwon Y (2021) Tool monitoring of end milling based on gap sensor and machine learning. J Ambient Intell Humaniz Comput 12(12):10615–10627

Jia H, Sun K, Zhang W, Leng X (2021) An enhanced chimp optimization algorithm for continuous optimization domains. Complex Intell Syst 8(1):65–82

Khisle M, Mohammadi H (2019) Passive sonar target classification using multi-layer perceptron trained by salp swarm algorithm. Ocean Eng 181:98–108

Khisle M, Mosavi M (2019) Improved whale trainer for sonar datasets classification using neural network. Appl Acoust 154:176–192

Khisle M, Safari A (2019) Classification of sonar targets using an MLP neural network trained by dragonfly algorithm. Wirel Pers Commun 108(4):2241–2260

Kingma D, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980

Konečný J, McMahan HB, Ramage D, Richtárik P (2016) Federated optimization: distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.05257

LeCun Y, Bengio Y, Hinton GE (2015) Deep learning. Nature 521(7553):436–444

Lee C-H, Xu X, Eun DY (2012) Beyond random walk and metropolis hastings samplers: why you should not backtrack for
unbiased graph sampling. ACM Sigmetrics Perform Eval Rev 40(1):319–330
Li T-M, Lehtinen J, Ramamoorthi R, Jakob W, Durand F (2015) Anisotropic gaussian mutations for metropolis light transport through hessian-hamiltonian dynamics. ACM Trans Graph (TOG) 34(6):1–13
Li T, Sahu AK, Talwalkar A, Smith V (2020) Federated learning: challenges, methods, and future directions. IEEE Signal Process Mag 37(3):50–60
Loris I, Nolet G, Daubechies I, Dahlen FA (2007) Tomographic inversion using-1-norm regularization of wavelet coefficients. Geophys J Int 170(1):359–370
Martens J et al (2010) Deep learning via hessian-free optimization. In: ICML, vol 27, pp 735–742
Mhaskar HN, Poggio T (2016) Deep vs. shallow networks: an approximation theory perspective. Anal. Anal J Int 14(06):829–848
Mocanu DC, Mocanu E, Stone P, Nguyen PH, Gibescu M, Liotta A (2018) Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. Nat Commun 9:1–12
Moreau J-J (1965) Proximité et dualité dans un espace hilbertien. Bull Soc Math France 93:273–299
Mosavi MR, Khishe M, Naseri MJ, Parvizi GR, Ayat M (2019) Multi-layer perceptor neural network utilizing adaptive best-mass gravitational search algorithm to classify sonar dataset. Arch Acousti 44:1–15
Muhammad U, Wang W, Chattha SP, Ali S (2018) Pre-trained vgg-net architecture for remote-sensing image scene classification. In: 24th international conference on pattern recognition (ICPR), Beijing, China. IEEE, pp 1622–1627
Natarajan BK (1995) Sparse approximate solutions to linear systems. SIAM J Comput 24(2):227–234
Neal RM et al (2011) Mcmc using hamiltonian dynamics. Handb Markov Chain Monte Carlo 2(4):341–363
Nocedal J, Wright SJ (2006) Sequential quadratic programming. Numer Optim Accesses 234k:529–562
Ostad-Ali-Araki K, Shayan M (2021) Subsurface drain spacing in the unsteady conditions by hydrus-3d and artificial neural networks. Arab J Geosci 14(18):1–14
Ostad-Ali-Araki K, Shayannejad M, Ghorbanizadeh-Kharazi H (2017) Artificial neural network for modeling nitrate pollution of groundwater in marginal area of zayandeh-rood river, Isfahan, Iran. KSCE J Civ Eng 21(1):134–140
Pajarnen J, Thai HL, Akroug R, Peters J, Neumann G (2019) Compatible natural gradient policy search. Mach Learn 108(8):1443–1466
Polap D (2021) Fuzzy consensus with federated learning method in medical systems. IEEE Access 9:150383–150392
Quiroz M, Villani M, Kohn R (2016) Slable mcmc for large data problems using data subsampling and the difference estimator. Riksbank Res Pap Ser 160:1–32
Recht B, Rebeschini P, Schmidt L, Shankar V (2018) Do cifar-10 classifiers generalize to cifar-10? arXiv preprint arXiv:1806.00451
Rere L, Fanany MI, Arymurthy AM (2016) Metaheuristic algorithms for convolution neural network. Comput Intell Neurosci 2016:1–14
Robbins H, Monro S (1951) A stochastic approximation method. Ann Math Stat 22(3):400–407
Robert G, Tweedie R (1996) Exponential convergence of langevin distributions and their discrete approximations. Bernoulli 2(4):341–363
Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536
Sajja TK, Kalluri HK (2021) Image classification using regularized convolutional neural network design with dimensionality reduction modules: Rcnn–drm. J Ambient Intell Humaniz Comput 12(10):9423–9434
Scardapane S, Commiello D, Hussain A, Uncini A (2017) Group sparse regularization for deep neural networks. Neurocomputing 241:81–89
Schraudolph NN, Yu J, Günter S (2007) A stochastic quasi-newton method for online convex optimization. In: Artificial intelligence and statistics. PMLR, pp 436–443
Shakshuki E, Yasar A, Malik H (2020) Applications of machine learning in pervasive systems. J Ambient Intell Humaniz Comput 11:5807–5808
Shanno DF (1970) Conditioning of quasi-newton methods for function minimization. Math Comput 24(111):647–656
Shi Y (2004) Particle swarm optimization. IEEE Connect 2(1):8–13
Sree V, Mapes J, Dua S, Lih OS, Koh JE, Ciaccio EJ, Acharya UR et al (2021) A novel machine learning framework for automated detection of arrhythmias in ecg segments. J Ambient Intell Humaniz Comput 12(11):10145–10162
Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhudinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
Sun S, Cao Z, Zhu H, Zhao J (2019) A survey of optimization methods from a machine learning perspective. IEEE Trans Cybern 50(8):3668–3681
Sutskever I, Martens J, Dahl G, Hinton G (2013) On the importance of initialization and momentum in deep learning. In: International conference on machine learning. PMLR, pp 1139–1147
Tartaglione, E, Lepsøy S, Fiandrotti A, Francini G (2018) Learning sparse neural networks via sensitivity-driven regularization. In: Proceedings of the 32nd international conference on neural information processing systems (NeurIPS). Montreal, Canada, pp 3882–3892
Wang Z, Mohamed S, Freitas N (2013) Adaptive hamiltonian and riemann manifold monte carlo. In: International conference on machine learning. PMLR, pp 1462–1470
Whitley D, Starkweather T, Bogart C (1990) Genetic algorithms and neural networks: optimizing connections and connectivity. Parallel Comput 14(3):347–361
Xiao H, Rasul K, Vollgraf R (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747
Xie C, Zhang F (2021) A new sequence optimization algorithm based on particle swarm for machine learning. J Ambient Intell Humaniz Comput 13(5):2601–2619
Xu Z, Zhang H, Wang Y, Chang X, Liang Y (2010) L 1/2 regularization. Sci China Inf Sci 53(6):1159–1169
Yang X, He X, Zhao J, Zhang Y, Zhang S, Xie P (2020) COVID-ct-dataset: a ct image dataset about COVID-19. arXiv preprint arxiv:2003.13865
Yurochkin M, Agarwal M, Ghosh S, Greenewald K, Hoang N, Khazaei Y (2019) Bayesian nonparametric federated learning of neural networks. In: International conference on machine learning. PMLR, pp 1462–1470
Yurochkin M, Agarwal M, Ghosh S, Greenewald K, Hoang N, Khazaei Y (2020) Bayesian nonparametric federated learning of neural networks. In: International conference on machine learning. PMLR, pp 1139–1147
Zeiler MD (2012) Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701
Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.