Non-smooth Bayesian learning for artificial neural networks

Mohamed Fakhfakh, Lotfi Chaâri, Bassem Bouaziz, Faiez Gargouri

Non-smooth Bayesian learning for artificial neural networks

Mohamed Fakhfakh[1,2*], Lotfi Chaari[2], Bassem Bouaziz[1] and Faiez Gargouri[1]

[1*]University of Sfax, MIRACL laboratory, Tunisia.
[2*]University of Toulouse, INP, IRIT, France.

*Corresponding author(s). E-mail(s): mohamed.fakhfakh@toulouse-inp.fr;
Contributing authors: lotfi.chaari@toulouse-inp.fr; bassem.bouaziz@isims.usf.tn;
faiez.gargouri@isims.usf.tn;

Abstract

Artificial Neural Networks (ANNs) are being widely used in supervised Machine Learning (ML) 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. Moreover, sparse regularizer is designed to zero superfluous weights and hence remove unneeded connections. However, if one wants to promote sparse networks, such as the $\boldsymbol{\ell_1}$ norm, one need use sparse regularizations, the optimization process becomes challenging since the error to be minimized is no longer differentiable. 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 datasets, two COVID-19 and two standard datasets (Fashion-MNIST and CIFAR-10). The results are verified by a comparative study with three different CNNs, one of which is deeper than the others in order to evaluate the effectiveness of our optimizer. 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%** in most of the experiments carried out. 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 subfield 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 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 over-parameterized, 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 ℓ_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 weigths 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 ℓ_1 regularizations are used. It is worth noting that ℓ_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 ℓ_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, Section 2 is dedicated to the state of the art. Then, the addressed problem is formulated in Section 3. The proposed efficient Bayesian optimization scheme is developed in Section 4 and validated in Section 5. Therefore, The discussion will be presented in section 6. Finally, The conclusion and future work are drawn in Section 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); Pajarinen 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 Połap (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.

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. (2020b). 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.

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 development and application of DNNs, so it is worthy of deduce the model parameters without losing performance. Sparsing neural networks is one of the methods to effectively reduce complexity which can improve efficiency and generalizability. The sparse optimization method can be used for various tasks to produce sparse solutions. The ℓ_1 penalty added to the classification cost can be interpreted as a convexification of the ℓ_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 ℓ_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 ℓ_0 problem is usually NP-hard Natarajan (1995). The ℓ_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. To the best of our knowledge, this is the first work which utilizes a Bayesian optimization based method for Deep learning architectures.

Table 1: Summary of Optimization Methods.

Categories	Method	Years	Purpose	Advantages	Disadvantages
First-Order	SGD Robbins and Monro (1951); 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.
	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.
	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.
	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.
	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.

Table 1: Summary of Optimization Methods.

Categories	**Method**	**Years**	**Purpose**	**Advantages**	**Disadvantages**
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.
	Hessian Free (HF) Method Martens et al. (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.
	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.
	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.
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.

Table 1: Summary of Optimization Methods.

Categories	Method	Years	Purpose	Advantages	Disadvantages
	DA Khishe and Safari (2019)	2019	Inspired by the dynamic and static swarming behaviors of dragonflies to resolves 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 impertinent 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 Połap (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 hyperarameter tuning, and interpretability tasks to the federated learning setting present roadblocks to the widespread adoption of FL in practical settings.

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 $\widehat{y}(x, W)$ 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 ℓ_1 regularization with M input data for the learning step, the weights vector can be estimated as

$$\begin{aligned}\widehat{W} &= \arg\min_W \mathcal{L}(W) \\ &= \arg\min_W \sum_{m=1}^{M} \|\widehat{y}(x^m;\, W) - y^{(m)}\|_2^2 + \lambda \|W\|_1,\end{aligned} \tag{1}$$

where λ 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) (ℓ_0, ℓ_p,...). The ℓ_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 Section 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 $\widehat{y}$, and assuming a quadratic error (first term in (1)), we define the likelihood distribution as

$$f\left(y;\, W, \sigma\right) \propto \prod_{m=1}^{M} \exp\left(-\frac{1}{2\sigma^2}\|\widehat{y}(x^m;\, W) - y^{(m)}\|^2\right), \tag{2}$$

where σ^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 ℓ_1 penalization. Under a Bayesian framework, the Laplace distribution can be used.

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

where λ is a hyperparameter to be set. This prior allows us to introduce exactly the same prior information as the ℓ_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:

$$\begin{aligned} f(W;\, y, \sigma, \lambda) &\propto f(y;\, W, \sigma) f(W;\, \lambda) \\ &\propto \prod_{m=1}^{M} \exp\left(-\frac{1}{2\sigma^2}\|\widehat{y}(x^m;\, W) - y^{(m)}\|^2\right) \times \\ &\prod_{k=1}^{N} \exp\left(-\frac{\|W^{[k]}\|_1}{\lambda}\right). \end{aligned} \tag{4}$$

It is clear that this posterior is not straightforward to handle in order to derive a closed-form expression of the estimate $\widehat{W}$. For this reason, we resort to a stochastic sampling approach in order to numerically approximate the posterior, and hence

to calculate an estimator for $\widehat{W}$. 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 thus 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 $K(q) = \frac{1}{2m}q^2$. Thus, the Hamiltonian $H(W, q)$ can be expressed as:

$$H(W, q) = E(W) + K(q), \tag{5}$$

and the dynamics of the particle can be specified by a set of coupled differential equations Neal et al. (2011),

$$\frac{dq}{dt} = \frac{\partial H}{\partial W} \tag{6}$$

$$\frac{dW}{dt} = -\frac{\partial H}{\partial q} \tag{7}$$

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 θ 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

$$\begin{aligned} f_\theta(W, q) &\propto \exp(-H(W, q)) \\ &\propto f(W;\ \theta) \exp(-K(q)). \end{aligned} \tag{8}$$

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 $N(0,I_N)$, 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 stepsize $\epsilon > 0$: 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_θ. The detailed ns-HMC scheme is given by algorithm 1, where L_f represent the number of leapfrog step and ϵ is the stepsize.

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

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

- Initialize with some $\boldsymbol{W}^{(0,0)}$, set the iteration number $r = 0$, L_f and ϵ;
for $r = 1, \ldots, S$ **do**
  - Sample $q^{(r,0)} \sim \mathcal{N}(\mathbf{0}, I_N)$;
  - Compute $q^{(r,\frac{1}{2}\epsilon)} = q^{(r,0)} - \frac{\epsilon}{2}\left[W^{(r-1,0)} - prox_{E_\theta}(W^{(r-1,0)})\right]$;
  - Compute $W^{(r,\epsilon)} = W^{(r-1,0)} + \epsilon q^{(r,\frac{1}{2}\epsilon)}$;
  for $l_f = 1$ *to* $L_f - 1$ **do**
    * Compute $q^{(r,(l_f+\frac{1}{2})\epsilon)} = q^{(r,l_f\epsilon)} - \frac{\epsilon}{2}\left[W^{(r,l_f\epsilon)} - prox_{E_\theta}(W^{(r,l_f\epsilon)})\right]$;
    * Compute $W^{(r,(l_f+1)\epsilon)} = W^{(r,l_f\epsilon)} + \epsilon q^{(r,(l_f+\frac{1}{2})\epsilon)}$;
  end
  - Compute $q^{(r,(L_f+\frac{1}{2})\epsilon)} = q^{(r,L_f\epsilon)} - \frac{\epsilon}{2}\left[W^{(r,L_f\epsilon)} - prox_{E_\theta}(W^{(r,L_f\epsilon)})\right]$;
  - Apply standard MH acceptation rule by taking to $q^* = q^{(r,\epsilon L_f)}$ and $W^* = W^{(r,\epsilon L_f)}$;
end

the initial point where the proximity operator is calculated first.

4.4 Hamiltonian Sampling

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

$$E_\theta^k(W^k) = \frac{\alpha}{2}\sum_{m=1}^{M} \|\widehat{y}(x^m;\, W) - y^{(m)}\|_2^2 + \|W^k\|_1. \tag{9}$$

The posterior in (4) can be reformulated as

$$f(W;\, y, \theta) \propto \exp\left(-\sum_{k=1}^{N} E_\theta^k(W^k)\right). \tag{10}$$

To sample according to this exponential posterior, and since direct sampling is not possible due to the form of the energy function E_θ^k, 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 ℓ_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 $G_{\mathcal{L}}(W^k)$ the gradient of the quadratic term of the loss function $\mathcal{L}$ 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

$$prox_{E_\theta^k}(z) = p \Leftrightarrow z - p \in \partial E_\theta^k(p). \tag{11}$$

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

$$\begin{aligned} & prox_{E_\theta^k}(z) = p \Leftrightarrow z - p \in \partial E_\theta^k(p) \\ & \Leftrightarrow z - p \in \partial\varphi(p) + \frac{\alpha}{2} G_{\mathcal{L}}(W^k) \\ & \Leftrightarrow z - \frac{\alpha}{2} G_{\mathcal{L}}(W^k) - p \in \partial\varphi(p) \\ & \Leftrightarrow p = prox_{\varphi}\left(z - \frac{\alpha}{2} G_{\mathcal{L}}(W^k)\right). \end{aligned} \tag{12}$$

Since prox$_\varphi$ is nothing but the soft thresholding operator Chaux et al. (2007), the proximity operator in (12) can be easily calculated once a single gradient step is applied (back-propagation) to calculate $G_{\mathcal{L}}(W^k)$.
We therefore propose the following leapfrog discretization scheme to be integrated in Algorithm 1, where $P_0 = prox_{E_\theta}(\boldsymbol{z}^{(0,0)})$:

$$\begin{aligned} q^{(r,(l+\frac{1}{2})\epsilon)} &= q^{(r,l\epsilon)} - \frac{\epsilon}{2}\left[2z^{(r,l\epsilon)} - \boldsymbol{z}^{(0,0)} - P_0\right] \\ z^{(r,(l+1)\epsilon)} &= z^{(r,l\epsilon)} + \epsilon q^{(r,(l+\frac{1}{2})\epsilon)} \\ q^{(r,(l+\frac{1}{2})\epsilon)} &= q^{(r,l\epsilon)} - \frac{\epsilon}{2}\left[2z^{(r,l\epsilon)} - \boldsymbol{z}^{(0,0)} - P_0\right]. \end{aligned} \tag{13}$$

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

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

Algorithm 2: Gibbs sampler using the ns-HMC optimizer to sample $\boldsymbol{z} = W^{[k]}$.

- Initialize with some $\boldsymbol{z}^{(0,0)}$, set the iteration number $r = 0$, L_f and ϵ;
- $P_0 = prox_{E_\theta}(\boldsymbol{z}^{(0,0)})$;
for $r = 1, \ldots, S$ **do**
  - Sample $q^{(r,0)} \sim \mathcal{N}(\mathbf{0}, I_N)$;
  - $q^{(r,\frac{1}{2}\epsilon)} = q^{(r,0)} - \frac{\epsilon}{2}\left[2z^{(r-1,0)} - \boldsymbol{z}^{(0,0)} - P_0)\right]$;
  - $z^{(r,\epsilon)} = z^{(r-1,0)} + \epsilon q^{(r,\frac{1}{2}\epsilon)}$;
  for $l_f = 1$ *to* $L_f - 1$ **do**
    * $q^{(r,(l_f+\frac{1}{2})\epsilon)} = q^{(r,l_f\epsilon)} - \frac{\epsilon}{2}\left[2z^{(r,l_f\epsilon)} - \boldsymbol{z}^{(0,0)} - P_0\right]$;
    * $z^{(r,(l_f+1)\epsilon)} = z^{(r,l_f\epsilon)} + \epsilon q^{(r,(l_f+\frac{1}{2})\epsilon)}$;
  end
  - $q^{(r,(L_f+\frac{1}{2})\epsilon)} = q^{(r,L_f\epsilon)} - \frac{\epsilon}{2}\left[2z^{(r,L_f\epsilon)} - \boldsymbol{z}^{(0,0)} - P_0\right]$;
  - Apply MH acceptation rule to (z^*, q^*) with $q^* = q^{(r,\epsilon L_f)}$ and $z^* = z^{(r,\epsilon L_f)}$;
end

probability

$$\min\{1, \exp[H(z^{(r)}, q^{(r)}) - H(z^{(*)}, q^{(*)})]\} \quad (14)$$

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 λ and σ 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 Go memory.

5.1 ConvNet Models

Two CNN architectures are used in this study. Like the LeNet model LeCun et al. (1998), the first one includes three convolutional and two fully-connected (FC). The second one has five convolutional and three FC layers that are organized similarly to VGG-Net Muhammad et al. (2018). These architectures are shown in Table 4. All of them involve convolutional layers with 3×3 Kernel filters in addition to 2×2 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 :

- 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.
- ℓ_1 Regularization Xu et al. (2010): ℓ_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 $\lambda = 0.001$.

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	48000	12000	10
CIFAR-10	50000	10000	10

Table 3 Parameters setting for benchmark algorithms.

Algorithm	Parameters	Description	Values
MH	σ	Stand. dev. of the proposal normal distribution	3
rw-MH	σ	Stand. dev. of the proposal normal distribution	5
Adam	lr	Learning rate	10^{-3}
	β_1	1st moment estimates exponential decay rate	0.9
	β_2	2nd moment estimates exponential decay rate	0.999
	ϵ	Numerical stability constant	$1e-08$
SGD	lr	Learning rate	10^{-3}
	momentum	Acceleration rate	0.8
	decay	Learning rate decay over each update	$1e-6$
Adadelta	lr	Learning rate	10^{-3}
	rho	Decay rate	0.95
	ϵ	Numerical stability constant	$1e-08$
Adamax	lr	Learning rate	10^{-3}
	β_1	1st moment estimates exponential decay rate	0.9
	β_2	2nd moment estimates exponential decay rate	0.999
	ϵ	Numerical stability constant	$1e-08$
IWT	minv	Lower bound	-2
	maxv	Upper bound	2
	size	Number of particles	30
	p_s	Spiral parameter	3
DA	minv	Lower bound	-2
	maxv	Upper bound	2
	size	Number of particles	15
SSA	minv	Lower bound	-2
	maxv	Upper bound	2
	size	Number of particles	30

- 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 λ 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 ℓ_1 norm of the estimated weights using different values of the regularization parameter λ over 10 Monte Carlo runs. To further evaluate the sparsity level, Table 5 also reports the ℓ_0 pseudo-norm values (number of non-zeros). Standard deviations

Table 4 Convnet with regularization techniques.

CNN_1	CNN_2
Conv3x3-32:stride=1	Conv3x3-32:stride=1
BatchNormalization	BatchNormalization
MaxPool 2x2	Conv3x3-32:stride=1
Dropout(0.2)	BatchNormalization
	MaxPool 2x2
	Dropout(0.2)
Conv3x3-64:stride=1	Conv3x3-64:stride=1
BatchNormalization	BatchNormalization
MaxPool 2x2	Conv3x3-64:stride=1
Dropout(0.3)	BatchNormalization
	MaxPool 2x2
	Dropout(0.3)
Conv3x3-128:stride=1	Conv3x3-128:stride=1
BatchNormalization	BatchNormalization
MaxPool 2x2	Conv3x3-128:stride=1
Dropout(0.4)	BatchNormalization
	MaxPool 2x2
	Dropout(0.4)
Flattening	Flattening
FC-64	FC-128
Dropout(0.3)	Dropout(0.3)
	FC-64
	Dropout(0.2)
FC-softmax	FC-softmax

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 λ 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 230×230 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 Figures 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

[1] https://www.kaggle.com/plameneduardo/sarscov2-ctscan-dataset

Table 5 Accuracy, sensitivity, specificity, computational time (in minutes), ℓ_0 and ℓ_1 norms of the estimated weights for CNN_1 using Adam and the proposed method with different values of λ.

Optimizer	λ	$\|\cdot\|_0$	$\|\cdot\|_1$	Acc.	Time	Sens.	Spec.
ns-HMC	10^{-3}	149113	48988	89.68	37.28	88.21	86.95
		±9.07	±9.10	±0.04	±0.63	±0.06	±0.08
	10^{-2}	**142542**	**45476**	**90.02**	**38.79**	**89.02**	**88.57**
		±8.78	**±8.81**	**±0.02**	**±0.61**	**±0.4**	**±0.3**
	10^{-1}	152904	51232	89.42	38.47	87.81	86.11
		±9.11	±9.19	±0.05	±0.70	±0.5	±0.6
Adam	10^{-3}	180513	51727	86.91	52.12	84.36	81.25
		±9.77	±9.11	±0.07	±0.87	±0.9	±0.8
	10^{-2}	191229	67732	85.34	54.91	83.14	80.66
		±10.28	±10.63	±0.12	±1.08	±1.12	±1.09
	10^{-1}	189075	58823	85.49	53.85	84.09	82.49
		±10.15	±10.47	±0.09	±0.95	±1.05	±1.01

Table 6 Experiment 1: Results for CT image classification using CNN_1 and CNN_2 (Computational time in min, accuracy, loss, sensitivity and specificity).

	CNN_1					CNN_2				
Optimizers	Time (min)	Acc.	Loss	Sens.	Spec.	Time (min)	Acc.	Loss	Sens.	Spec.
ns-HMC	**37**	**0.90**	**0.10**	**0.89**	**0.87**	**58**	**0.92**	**0.09**	**0.90**	**0.89**
MH	79.2	0.84	0.18	0.81	0.77	133.8	0.86	0.16	0.85	0.80
rw-MH	64.8	0.85	0.17	0.84	0.79	95.4	0.87	0.14	0.86	0.82
Adam	52	0.87	0.12	0.86	0.83	85.2	0.88	0.11	0.87	0.85
SGD	53	0.88	0.13	0.85	0.80	87.1	0.86	0.15	0.84	0.81
Adadelta	56	0.86	0.12	0.84	0.81	90.6	0.87	0.11	0.85	0.83
Adamax	53	0.87	0.13	0.86	0.84	87,6	0.87	0.12	0.86	0.85
IWT	59	0.84	0.21	0.81	0.78	70,2	0.86	0.19	0.84	0.83
DA	61	0.83	0.25	0.82	0.79	73.2	0.85	0.22	0.83	0.81
SSA	57	0.86	0.18	0.85	0.83	61.2	0.88	0.17	0.87	0.86

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 classification to identify Covid-19 infections from other pneumonia. In contratst 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 [2]. We used 566 images for the train and 180 images for the test with size of 230×230.

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

[2] https://www.kaggle.com/luisblanche/covidct

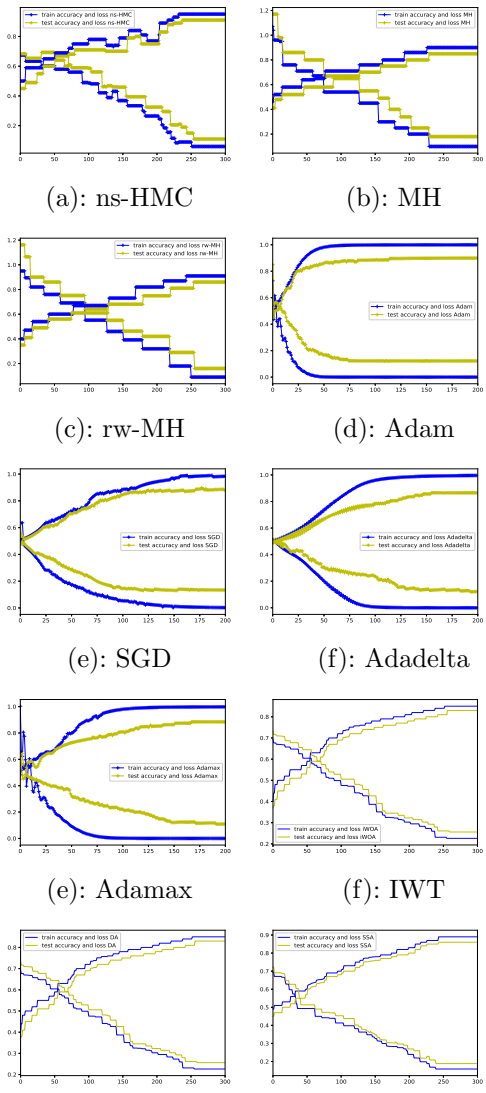

Fig. 1 Experiment 1: Train and test curves using CNN_1.

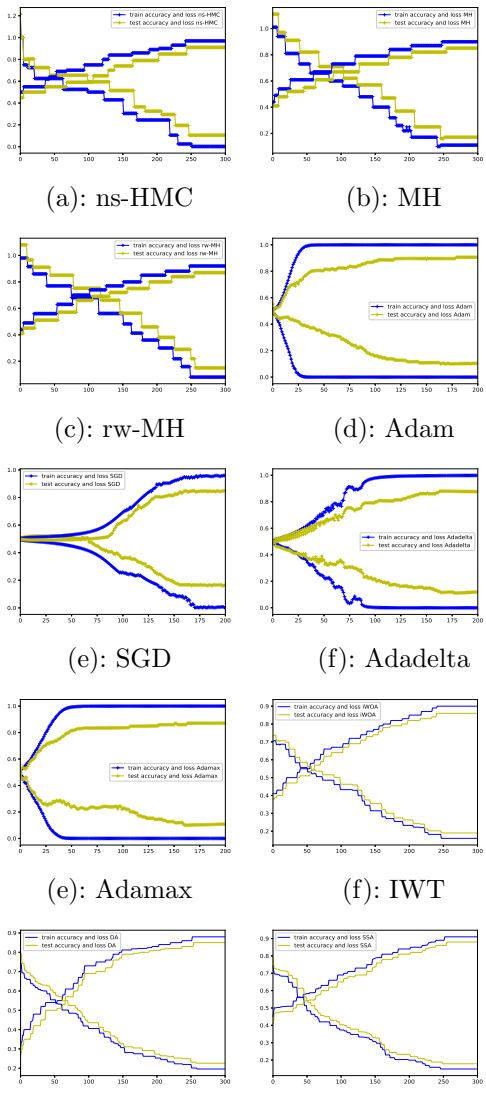

Fig. 2 Experiment 1: Train and test curves using CNN_2.

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 7,000 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 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 minutes for the CNN_1, more than

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).

	CNN_1					CNN_2				
Optimizers	Time (min)	Acc.	Loss	Sens.	Spec.	Time (min)	Acc.	Loss	Sens.	Spec.
ns-HMC	**40**	**0.84**	**0.26**	**0.82**	**0.80**	**53**	**0.88**	**0.22**	**0.86**	**0.85**
MH	71,4	0.73	0.38	0.71	0.69	92.4	0.76	0.34	0.74	0.72
rw-MH	59	0.76	0.36	0.75	0.72	94.8	0.77	0.32	0.75	0.74
Adam	58	0.71	0.43	0.69	0.68	81	0.73	0.36	0.72	0.71
SGD	59	0.65	0.45	0.64	0.62	82.2	0.68	0.42	0.67	0.65
Adadelta	61.8	0.67	0.42	0.65	0.63	87.6	0.70	0.38	0.69	0.67
Adamax	60.6	0.69	0.41	0.67	0.66	90	0.74	0.36	0.72	0.71
IWT	54	0.75	0.38	0.74	0.72	90	0.78	0.35	0.77	0.75
DA	57	0.78	0.36	0.77	0.76	87	0.81	0.33	0.80	0.76
SSA	51	0.76	0.37	0.76	0.75	83	0.79	0.36	0.78	0.77

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 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 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 four convolutional layers (5 X Conv3x3-32, 5 X Conv3x3-64, 5 X Conv3x3-128) and a FC-softmax layer. All of them involve convolutional layers with 3 X 3 Kernel filters in addition to 2 X 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: its *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

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).

	CNN_1					CNN_2				
Optimizers	Time (min)	Acc.	Loss	Sens.	Spec.	Time (min)	Acc.	Loss	Sens.	Spec.
ns-HMC	**70.5**	**0.92**	**0.22**	**90**	**88**	**308.4**	**0.93**	**0.19**	**91**	**89**
MH	166.2	0.86	0.35	0.85	0.81	745.8	0.87	0.33	0.84	0.82
rw-MH	183.6	0.88	0.33	0.86	0.83	797.4	0.88	0.31	0.85	0.84
Adam	156.6	0.90	0.46	0.85	0.82	444	0.92	0.32	0.88	0.87
SGD	164.4	0.88	0.71	0.71	0.67	452.4	0.89	0.56	0.84	0.83
Adadelta	169.8	0.70	1.20	0.66	0.64	439.8	0.78	0.96	0.71	0.70
Adamax	149	0.91	0.49	0.88	0.82	448.2	0.91	0.26	0.88	0.87
IWT	180.2	0.82	0.37	0.79	0.73	486	0.83	0.36	0.80	0.79
DA	174	0.79	0.40	0.76	0.74	469	0.82	0.35	0.78	0.78
SSA	165.7	0.84	0.33	0.81	0.75	453	0.86	0.24	0.86	0.85

Table 9 Experiment 4: Results for CIFAR-10 image classification - challenging case - using CNN_1 and CNN_2 (Computational time in min, accuracy, loss, sensitivity and specificity).

	CNN_1					CNN_2				
Optimizers	Time (min)	Acc.	Loss	Sens.	Spec.	Time (min)	Acc.	Loss	Sens.	Spec.
ns-HMC	**85.7**	**0.91**	**0.25**	**89**	**87**	**331**	**0.92**	**0.21**	**90**	**87**
MH	172	0.83	0.41	0.80	0.78	763.2	0.84	0.36	0.83	0.81
rw-MH	192.2	0.85	0.36	0.84	0.81	814.1	0.86	0.35	0.84	0.83
Adam	161	0.89	0.42	0.83	0.81	429	0.90	0.36	0.87	0.86
SGD	169	0.86	0.75	0.69	0.65	459.7	0.86	0.60	0.83	0.80
Adadelta	174.7	0.75	0.92	0.68	0.65	453.3	0.79	0.81	0.72	0.70
Adamax	155	0.90	0.33	0.88	0.85	507.8	0.91	0.24	0.87	0.85
IWT	186	0.80	0.35	0.78	0.74	531	0.81	0.34	0.79	0.78
DA	179	0.82	0.35	0.77	0.75	519	0.84	0.30	0.78	0.76
SSA	172.7	0.83	0.31	0.81	0.77	503	0.85	0.27	0.83	0.80

Table 10 Results for Fashion-MNIST image classification using Deep CNN (Computational time in min, accuracy, loss, sensitivity and specificity).

Optimizers	Time (min)	Acc.	Loss	Sens.	Spec.
ns-HMC	**582**	**0.93**	**0.20**	**0.91**	**0.91**
MH	977	0.84	0.41	0.80	0.76
rw-MH	986	0.85	0.37	0.83	0.78
Adam	701	0.91	0.55	0.88	0.85
SGD	705	0.88	0.46	0.82	0.79
Adadelta	707	0.80	0.63	0.70	0.72
Adamax	706	0.92	0.44	0.90	0.88
IWT	681	0.88	0.33	0.82	0.79
DA	677	0.85	0.37	0.81	0.77
SSA	694	0.90	0.31	0.86	0.84

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).

The complexity of a neural network can be reduced by promoting sparse interconnection structures. 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). Two CNNs which one is deeper than others have been used with ℓ_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 ℓ_1 and ℓ_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, λ is a hyperparameter to be set as a fixed value in our method. The choice of the best value of λ 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 and most recent metaheuristics methods. The use of standard datasets (Fashion-MNIST and CIFAR-10) with multiple images confirms the stability results of our optimizer with a global accuracy more than 90% and a loss of less than 2.5%.

The experiments showed the generalization of our ns-HMC and their ability to be applied in various classification problems. 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., and Gandomi, A. H. (2021). The arithmetic optimization algorithm. *Computer methods in applied mechanics and engineering*, 376:113609.

Alder, B. J. and Wainwright, T. E. (1959). Studies in molecular dynamics. i. general method. *The Journal of Chemical Physics*, 31(2):459–466.

Alsarhan, A., Alauthman, M., Alshdaifat, E., Al-Ghuwairi, A.-R., and Al-Dubai, A. (2021). Machine learning-driven optimization for svm-based intrusion detection system in vehicular ad hoc networks. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–10.

Angelov, P. and Almeida Soares, E. (2020). Sars-cov-2 ct-scan dataset: A large dataset of real patients ct scans for sars-cov-2 identification. *medRxiv*.

Anwar, S., Hwang, K., and Sung, W. (2017). Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1–18.

Ashwini, R. and 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.

Berahas, A. S., Byrd, R. H., and Nocedal, J. (2019). Derivative-free optimization of noisy functions via quasi-newton methods. *SIAM Journal on Optimization*, 29(2):965–993.

Bollapragada, R., Byrd, R. H., and Nocedal, J. (2019). Exact and inexact subsampled newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2):545–578.

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311.

Bruno, G., Antonelli, D., and Stadnicka, D. (2021). Evaluating the effect of learning rate, batch size and assignment strategies on the production performance. *Journal of Industrial and Production Engineering*, 38(2):137–147.

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

Byrd, R. H., Hansen, S. L., Nocedal, J., and Singer, Y. (2016). A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031.

Chaabene, S., boudaya, A., Bouaziz, B., Hokelmann, A., Ammar, A., and Chaari, L. (2021). Convolutional neural network for drowsiness detection using EEG signals. *Sensors*, , 21(5).

Chaari, L., Batatia, H., Dobigeon, N., and 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)*, pages 1901–1905.

Chaari, L., Tourneret, J.-Y., and Batatia, H. (2017). A general non-smooth Hamiltonian Monte Carlo scheme using Bayesian proximity operator calculation. *European Signal Processing Conference EUSIPCO*, pages 1260–1264.

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

Chang, H.-S., Learned-Miller, E., and 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., and Wajs, V. (2007). A variational formulation for frame-based inverse problems. *Inverse Problems*, 23(4):1495.

Cheng, Y., Yu, F. X., Feris, R. S., Kumar, S., Choudhary, A., and 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*, pages 2857–2865.

Chib, S. and Greenberg, E. (1995). Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335.

Devunooru, S., Alsadoon, A., Chandana, P., and Beg, A. (2021). Deep learning neural networks for medical image segmentation of brain tumours for diagnosis: a recent review and taxonomy. *Journal of Ambient Intelligence and*

Humanized Computing, 12(1):455–483.

Drewek-Ossowicka, A., Pietrołaj, M., and Rumiński, J. (2021). A survey of neural networks usage for intrusion detection systems. *Journal of Ambient Intelligence and Humanized Computing*, 12(1):497–514.

Fakhfakh, M., Bouaziz, B., Gargouri, F., and Chaari, L. (2020a). Prognet: Covid-19 prognosis using recurrent and convolutional neural networks. *The Open Medical Imaging Journal*, 12(1).

Fakhfakh, M., Chaâri, L., and Fakhfakh, N. (2020b). Bayesian curved lane estimation for autonomous driving. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–11.

Fan, Y., Yu, J., Mei, Y., Zhang, Y., Fu, Y., Liu, D., and Huang, T. S. (2020). Neural sparse representation for image restoration. *ArXiv*, abs/2006.04357.

Gen, L., Yuantao, G., and Jie, D. (2020). The efficacy of l_1 regularization in two-layer neural networks.

Gomez, A. N., Zhang, I., Kamalakara, S. R., Madaan, D., Swersky, K., Gal, Y., and Hinton, G. E. (2019). Learning sparse networks using targeted dropout. *arXiv preprint arXiv:1905.13678*.

Goyal, S. and Singh, R. (2021). Detection and classification of lung diseases for pneumonia and covid-19 using machine and deep learning techniques. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–21.

Han, L., Lin, H., and 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)*, pages 992–995.

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

Hanson, K. (2001). Markov Chain Monte Carlo posterior sampling with the hamiltonian method. In *Medical Imaging 2001: Image Processing*, volume 4322, pages 456–467. International Society for Optics and Photonics.

Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12.

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

Jaini, S. N. B., Lee, D., Lee, S., Kim, M., and Kwon, Y. (2021). Tool monitoring of end milling based on gap sensor and machine learning. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–13.

Jia, H., Sun, K., Zhang, W., and Leng, X. (2021). An enhanced chimp optimization algorithm for continuous optimization domains. *Complex & Intelligent Systems*, pages 1–18.

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

Khishe, M. and Mosavi, M. (2019). Improved whale trainer for sonar datasets classification using neural network. *Applied Acoustics*, 154:176–192.

Khishe, M. and Safari, A. (2019). Classification of sonar targets using an mlp neural network trained by dragonfly algorithm. *Wireless Personal Communications*, 108(4):2241–2260.

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

Konečnỳ, J., McMahan, H. B., Ramage, D., and Richtárik, P. (2016). Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, C.-H., Xu, X., and Eun, D. Y. (2012). Beyond random walk and metropolis-hastings samplers: why you should not backtrack for unbiased graph sampling. *ACM SIGMETRICS Performance evaluation review*, 40(1):319–330.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. (2020). Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60.

Li, T.-M., Lehtinen, J., Ramamoorthi, R., Jakob, W., and Durand, F. (2015). Anisotropic gaussian mutations for metropolis light transport through hessian-hamiltonian dynamics. *ACM*

Transactions on Graphics (TOG), 34(6):1–13.

Loris, I., Nolet, G., Daubechies, I., and Dahlen, F. A. (2007). Tomographic inversion using l1-norm regularization of wavelet coefficients. *Geophysical Journal International*, 170(1):359–370.

Martens, J. et al. (2010). Deep learning via hessian-free optimization. In *ICML*, volume 27, pages 735–742.

Mhaskar, H. N. and Poggio, T. (2016). Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14(06):829–848.

Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., and Liotta, A. (2018). Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, 9:1–12.

Moreau, J.-J. (1965). Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299.

Mosavi, M. R., Khishe, M., Naseri, M. J., Parvizi, G. R., and Ayat, M. (2019). Multi-layer perceptron neural network utilizing adaptive best-mass gravitational search algorithm to classify sonar dataset. *Archives of Acoustics*, 44.

Muhammad, U., Wang, W., Chattha, S. P., and Ali, S. (2018). Pre-trained vggnet architecture for remote-sensing image scene classification. In *24th International Conference on Pattern Recognition (ICPR)*, pages 1622–1627.

Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. *SIAM journal on computing*, 24(2):227–234.

Neal, R. M. et al. (2011). Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2.

Nocedal, J. and Wright, S. J. (2006). Sequential quadratic programming. *Numerical optimization*, pages 529–562.

Ostad-Ali-Askari, K. and Shayan, M. (2021). Subsurface drain spacing in the unsteady conditions by hydrus-3d and artificial neural networks. *Arabian Journal of Geosciences*, 14(18):1–14.

Ostad-Ali-Askari, K., Shayannejad, M., and Ghorbanizadeh-Kharazi, H. (2017). Artificial neural network for modeling nitrate pollution of groundwater in marginal area of zayandeh-rood river, isfahan, iran. *KSCE Journal of Civil Engineering*, 21(1):134–140.

Pajarinen, J., Thai, H. L., Akrour, R., Peters, J., and Neumann, G. (2019). Compatible natural gradient policy search. *Machine Learning*, 108(8):1443–1466.

Połap, D. (2021). Fuzzy consensus with federated learning method in medical systems. *IEEE Access*, 9:150383–150392.

Quiroz, M., Villani, M., and Kohn, R. (2016). Slable mcmc for large data problems using data subsampling and the difference estimator. *Riksbank Research Paper Series*, (160):1–32.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. (2018). Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451*.

Rere, L., Fanany, M. I., and Arymurthy, A. M. (2016). Metaheuristic algorithms for convolution neural network. *Computational intelligence and neuroscience*, 2016.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Roberts, G. and Tweedie, R. (1996). Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Sajja, T. K. and Kalluri, H. K. (2021). Image classification using regularized convolutional neural network design with dimensionality reduction modules: Rcnn–drm. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–12.

Scardapane, S., Comminiello, D., Hussain, A., and Uncini, A. (2017). Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89.

Schraudolph, N. N., Yu, J., and Günter, S. (2007). A stochastic quasi-newton method for online convex optimization. In *Artificial intelligence and statistics*, pages 436–443. PMLR.

Shakshuki, E., Yasar, A., and Malik, H. (2020). Applications of machine learning in pervasive systems. *Journal of Ambient Intelligence and Humanized Computing volume*, 11:5807–5808.

Shanno, D. F. (1970). Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656.

Shi, Y. (2004). Particle swarm optimization. *IEEE connections*, 2(1):8–13.

Sree, V., Mapes, J., Dua, S., Lih, O. S., Koh, J. E., Ciaccio, E. J., Acharya, U. R., et al. (2021). A novel machine learning framework for automated detection of arrhythmias in ecg segments. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–18.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Sun, S., Cao, Z., Zhu, H., and Zhao, J. (2019). A survey of optimization methods from a machine learning perspective. *IEEE transactions on cybernetics*, 50(8):3668–3681.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR.

Tartaglione, E., Lepsøy, S., Fiandrotti, A., and Francini, G. (2018). Learning sparse neural networks via sensitivity-driven regularization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 3882–3892.

Wang, Z., Mohamed, S., and Freitas, N. (2013). Adaptive hamiltonian and riemann manifold monte carlo. In *International conference on machine learning*, pages 1462–1470. PMLR.

Whitley, D., Starkweather, T., and Bogart, C. (1990). Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel computing*, 14(3):347–361.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xie, C. and Zhang, F. (2021). A new sequence optimization algorithm based on particle swarm for machine learning. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–19.

Xu, Z., Zhang, H., Wang, Y., Chang, X., and Liang, Y. (2010). L 1/2 regularization. *Science China Information Sciences*, 53(6):1159–1169.

Yang, X., He, X., Zhao, J., Zhang, Y., Zhang, S., and Xie, P. (2020). Covid-ct-dataset: a ct image dataset about covid-19. *arXiv preprint arxiv:2003.13865*, 3.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., and Khazaeni, Y. (2019). Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, pages 7252–7261. PMLR.

Zaheer, R. and Shaziya, H. (2019). A study of the optimization algorithms in deep learning. In *2019 Third International Conference on Inventive Systems and Control (ICISC)*, pages 536–539. IEEE.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.