Efficient Bayesian Learning of Sparse Deep Artificial Neural Networks

Mohamed Fakhfakh[1,2(✉)], Bassem Bouaziz[1], Lotfi Chaari[2], and Faiez Gargouri[1]

[1] MIRACL Laboratory, University of Sfax, Sfax, Tunisia
{bassem.bouaziz,faiez.gargouri}@isims.usf.tn
[2] University of Toulouse, INP, IRIT, Toulouse, France
{mohamed.fakhfakh,lotfi.chaari}@toulouse-inp.fr

Abstract. In supervised Machine Learning (ML), Artificial Neural Networks (ANN) are commonly utilized to analyze signals or images for a variety of applications. They are increasingly performing as a strong tool to establish the relationships among data and being successfully applied in science due to their generalization ability, noise and fault tolerance. One of the most difficult aspects of using the learning process is optimization of the network weights.

A gradient-based technique with a back-propagation strategy is commonly used for this optimization stage. Regularization is commonly employed for the benefit of efficiency. This optimization gets difficult when non-smooth regularizers are applied, especially to promote sparse networks. Due to differentiability difficulties, traditional gradient-based optimizers cannot be employed.

In this paper, we propose an MCMC-based optimization strategy within a Bayesian framework. An effective sampling strategy is designed using Hamiltonian dynamics. The suggested strategy appears to be effective in allowing ANNs with modest complexity levels to achieve high accuracy rates, as seen by promising findings.

Keywords: Artificial neural networks · Optimization · Deep learning · LSTM · MCMC · Hamiltonian dynamics

1 Introduction

Machine learning (ML) [1] is an artificial intelligence subfield (AI). It has expanded at an incredible rate, drawing a large number of academics interested in studying how a system may learn to do a task. In reality, an ML system does not follow instructions but instead learns from experience, such as making predictions or decisions based on data and continuously improving performance by reviewing new data. ML research achieved outstanding results on several complex cognitive tasks, including Computer Vision [2–5], Medical diagnoses [6–9], Signal Processing [10,11], recommendation systems [12], etc. Deep Learning

T. Bouadi et al. (Eds.): IDA 2022, LNCS 13205, pp. 78–88, 2022.
https://doi.org/10.1007/978-3-031-01333-1_7

(DL) [13,14] architectures have proved their capacity to deal with progressively voluminous data during the previous two decades. Furthermore, it has gradually become the most extensively employed computational strategy in the field of machine learning, generating exceptional results on a variety of cognitive tasks, equal or even surpassing human performance in some cases. The capacity to learn from huge volumes of data is one of the benefits and challenges of deep learning.

In a similar vein, Convolutional neural networks (CNN) [2,15,16] are one of the state-of-art deep learning techniques. CNNs are designed to automatically and adaptively learn spatial hierarchies of features through backpropagation [17,18] 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 to learning data in order to minimize the error between ground truth and estimates.

Many optimization techniques have been presented in recent years [19]. The convexity and differentiability of the target loss function have a significant impact on the performance of the deployed algorithms. Hence, choosing an optimization strategy that seeks to find the global optima in the learning stage is generally challenging, especially when the number of parameters is large. A non-appropriate optimization technique may for instance lead the network to lie in a local minimum during training phase. Speeding up the optimization process is also a challenging issue for large databases.

In this context, Bayesian approaches have made significant progress in a number of areas over the years, and there are several practical benefits. The core concept is to use probabilities to represent all uncertainties throughout the model. One of the most significant benefits is the ability to incorporate prior information. Indeed, recent developments in Markov Chain Monte Carlo (MCMC) methods [20–24] facilitate the implementation of Bayesian analyses of complex data sets containing missing observations and handling multidimensional outcomes. The main goal of this paper is to highlight a Bayesian model for the minimization of the target cost function of a learning model through hyperparameters adjustment.

Specifically, we propose a Bayesian optimization method to minimise the target cost function and derive the optimal weights vector. Indeed, we demonstrate that using the proposed method leads to high accuracy results, which cannot be reached using competing.

The rest of this paper is organized as follows. The addressed problem is formulated in Sect. 2. The proposed efficient Bayesian optimization scheme is developed in Sect. 3 and validated in Sect. 4. Finally, conclusions and future work are drawn in Sect. 5.

2 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. In this sense, smooth regularizers such as the ℓ_2 norm are used. In this case, gradient-based algorithms could still be used. However, if one aims at promoting sparse networks, sparse regularizations such as the ℓ_1 norm 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 $\widehat{y}(x, W)$ 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.

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

In Sect. 3 we present a method to efficiently estimate the weights vector without increase of learning complexity. The optimization problem in (1) is formulated and solved in a Bayesian framework.

3 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.

3.1 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 the prior knowledge on the weights vector W, we propose the use of a Laplace distribution in order to promote the sparsity of the neural network:

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

where λ is a hyperparameter to be fixed or estimated.

By adopting a Maximum A Posteriori (MAP) approach, we first need to express the posterior distribution. Based on the defined likelihood and prior, this posterior writes:

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

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.

3.2 Hamiltonian Sampling

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

$$E_\theta^k(W^k) = \frac{\alpha}{2} \sum_{m=1}^{M} \|\widehat{y}(x^m; W) - y^{(m)}\|_2^2 + \|W^k\|_1. \tag{5}$$

The posterior in (4) can therefore be reformulated as

$$f(W; y, \theta) \propto \exp\left(-\sum_{k=1}^{N} E_\theta^k(W^k)\right). \tag{6}$$

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 [25] strategy has been widely used in the literature to sample from 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 as proposed in [26]. More specifically, we use the plug and play procedure developed in [27]. Indeed, this strategy requires to calculate the proximity operator only at an initial point, and uses the shift property [28,29] to deduce the proximity operator during the iterative procedure [27, Algorithm 1].

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 standard definition of the proximity operator [28,29], we can write for a point z

$$\text{prox}_{E_\theta^k}(z) = p \Leftrightarrow z - p \in \partial E_\theta^k(p). \tag{7}$$

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

$$\text{prox}_{E_\theta^k}(z) = \text{prox}_{\varphi}\left(z - \frac{\alpha}{2} G_{\mathcal{L}}(W^k)\right). \tag{8}$$

Since prox_{φ} is nothing but the soft thresholding operator [29], the proximity operator in (8) can be easily calculated once a single gradient step is applied (back-propagation) to calculate $G_{\mathcal{L}}(W^k)$.

The main steps of the proposed method are detailed in Algorithm 1.

Algorithm 1: Main steps of the proposed Bayesian optimization.

- Fix the hyperparameters λ and σ;
- Initialize with some W_0;
- Perform one back-propagation step to provide an initialization for $G_{\mathcal{L}}(W_0)$;
- Compute $\text{prox}_{E_\theta}(W_0)$ according to (8);
- Use the Gibbs sampler in [27, Algorithm 1] until convergence;

After convergence, Algorithm 1 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.

4 Experimental Validation

In order to validate the proposed method, two image classification experiments are conducted using two different datasets: COVID-19 dataset including Computed tomography (CT) images [30], and a standard dataset, namely, CIFAR-10 [31]. For the sake of comparison, two kinds of optimizers are used: *i)* MCMC-based method, precisely the standard Metropolis-Hastings (MH) algorithm and the random walk Metropolis Hastings (rw-MH) [32], and *ii)* the most popular optimization techniques used in DL. : Adam and Adagrad [33]. One of the key hyper-parameters to set in optimizers in order to train a neural network is the learning rate. This parameter scales the magnitude of the weight updates in order to minimize the network's loss function. In the experiments, the learning rate is equal to 10^{-3}. In addition, the hyper-parameters β_1 and β_2 are equals to 0.9 and 0.999 respectively. They stand for the initial decay rates used when estimating the first and second moments of the gradient. As regards coding, we used python programming language with Keras and Tensorflow libraries on an Intel(R) Core(TM) i7-2720QM CPU 2.20 GHZ architecture with 16 Go memory. The same behavior with the computational time and accuracy which justify the effectiveness of our proposed MCMC method.

4.1 ConvNet Models

Two CNN architectures are used in this study. Like the LeNet model [34], the first one (CNN_1) includes three convolutional (Conv3 $\times$ 3-32, Conv3 $\times$ 3-64, Conv3 $\times$ 3-128), and two fully-connected (FC-64 and FC-softmax). The second one (CNN_2) has five convolutional (Conv3 $\times$ 3-32, Conv3 $\times$ 3-32, Conv3 $\times$ 3-64, Conv3 $\times$ 3-64, Conv3 $\times$ 3-128, Conv3 $\times$ 3-128) and three FC layers (FC-128,FC-64,FC-softmax) that are organized similarly to VGG-Net [35]. 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 [33]:

- Batch Normalization: deals with the feature space distribution variability 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 faster training, higher learning rates, and less sensitivity to weights initialization.
- ℓ_1 Regularization: ℓ_1 regularization is the preferred choice when having a high number of features as it provides sparse solutions. In our case, the regularization parameter was set to $\lambda = 0.001$.
- Dropout : random disabling of neurons during training with rate p. Temporarily ignoring some activation 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 $p = 0.35$.

4.2 Experiment 1: Challenging Case

A 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. 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 230×230.

The reported scores in Table 1 indicate that the proposed method clearly outperforms the competing optimizers in training both models to solve this challenging classification problem. Moreover, severe performance decrease is observed for some optimizers like Adagrad. This is mainly due to the challenging classification, which leads to a more complex learning process.

Table 1. Experiment 1: results for CT image classification using CNN_1 and CNN_2.

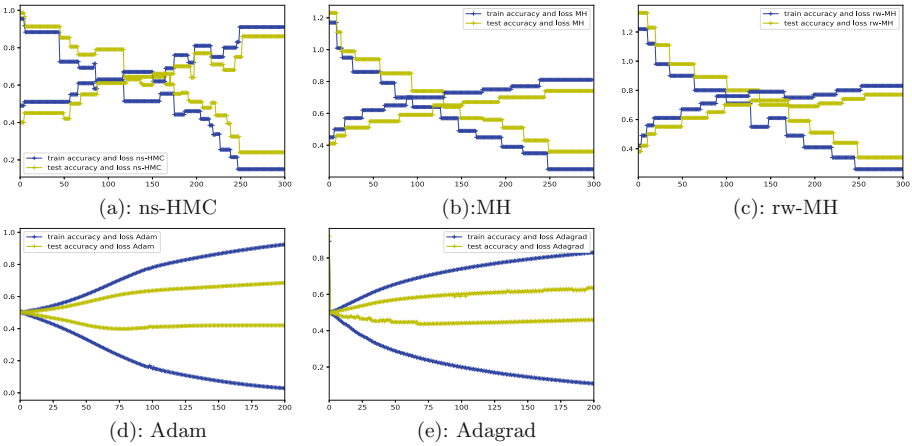
Optimizers	CNN_1			CNN_2		
	Comp. time (hrs)	Accuracy	Loss	Comp. time (hrs)	Accuracy	Loss
ns-HMC	**0.40**	**0.84**	**0.26**	**0.51**	**0.88**	**0.22**
MH	1.19	0.73	0.36	1.54	0.76	0.33
rw-MH	0.59	0.76	0.34	1.58	0.77	0.31
Adam	0.58	0.70	0.43	1.35	0.73	0.36
Adagrad	0.55	0.66	0.44	1.43	0.68	0.41

In order to confirm this performance decrease, Figs. 1 and 2 shows loss and accuracy curves obtained using the competing optimizers, and this for CNN_1 and CNN_2, respectively. The displayed curves clearly indicate an overfitting effect for classical optimizers, in contrast to the proposed method.

4.3 Experiment 2: 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 32×32 colour 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.

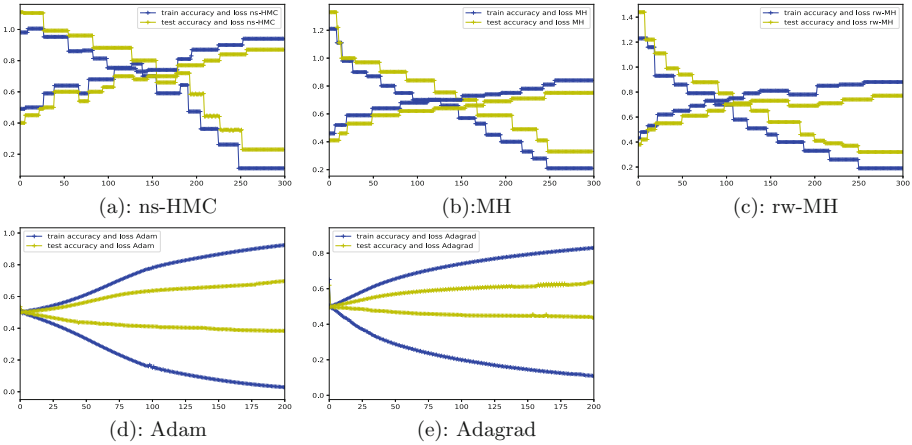
The reported scores in Table 2 indicate that the proposed method outperforms the competing optimizers in terms of learning precision, and hence classification performance. Furthermore, the competing optimizers do not perform



(a): ns-HMC (b):MH (c): rw-MH

(d): Adam (e): Adagrad

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



(a): ns-HMC (b):MH (c): rw-MH

(d): Adam (e): Adagrad

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

well to learn both CNNs on the CIFAR-10 dataset. This confirms the ability of the proposed method to allow different networks reaching high accuracy levels, in contrast to standard optimizers, even when regularization is use. The gain in terms of computational time using the proposed method is more important on this experiment.

Table 2. Experiment 2: results for CIFAR-10 image classification using CNN_1 and CNN_2.

Optimizers	CNN_1			CNN_2		
	Comp. time (hrs)	Accuracy	Loss	Comp. time (hrs)	Accuracy	Loss
ns-HMC	**1.17**	**0.92**	**0.22**	**5.13**	**0.93**	**0.19**
MH	2.77	0.86	0.35	12.43	0.87	0.33
rw-MH	3.06	0.88	0.33	13.29	0.88	0.31
Adam	2.60	0.90	0.46	7.40	0.92	0.32
SGD	2.73	0.88	0.71	7.54	0.89	0.56
Adagrad	2.78	0.75	0.81	7.22	0.78	0.64

5 Conclusion

In this paper, we proposed a new Bayesian optimization method for fitting weights for artificial neural networks. The suggested method uses Hamiltonian dynamics to solve the problem of sparse regularization optimization. Our results demonstrated the good performance of the proposed method in comparison with standard optimizers, as well as classical Bayesian ones. Moreover, the proposed technique allows simple networks to enjoy high accuracy and generalization properties. Future work will focus on testing our proposed optimizer with larger datasets, as well as proposing a distributed or parallel implementation.

References

1. Alpaydin, E.: Introduction to Machine Learning. MIT press, Cambridge (2020)
2. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. **25**, 1097–1105 (2012)
3. Lan, X., Zhang, S., Yuen, P.C., Chellappa, R.: Learning common and feature-specific patterns: a novel multiple-sparse-representation-based tracker. IEEE Trans. Image Process. **27**(4), 2022–2037 (2017)
4. Sainath, T.N., et al.: Deep convolutional neural networks for large-scale speech tasks. Neural Netw. **64**, 39–48 (2015)
5. Shao, R., Lan, X., Yuen, P.C.: Joint discriminative learning of deep dynamic textures for 3D mask face anti-spoofing. IEEE Trans. Inf. Forensics Secur. **14**(4), 923–938 (2018)
6. Kononenko, I.: Machine learning for medical diagnosis: history, state of the art and perspective. Artif. Intell. Med. **23**(1), 89–109 (2001)
7. Boudaya, A., et al.: EEG-based hypo-vigilance detection using convolutional neural network. In: Jmaiel, M., Mokhtari, M., Abdulrazak, B., Aloulou, H., Kallel, S. (eds.) ICOST 2020. LNCS, vol. 12157, pp. 69–78. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51517-1_6
8. Chaabene, S., Bouaziz, B., Boudaya, A., Hökelmann, A., Ammar, A., Chaari, L.: Convolutional neural network for drowsiness detection using EEG signals. Sensors **21**(5), 1734 (2021)
9. Safaei, A.A., Habibi-Asl, S.: Multidimensional indexing technique for medical images retrieval. Intell. Data Anal. **25**(6), 1629–1666 (2021)

10. Dong, Yu., Deng, L.: Deep learning and its applications to signal and information processing [exploratory dsp]. IEEE Signal Process. Mag. **28**(1), 145–154 (2010)
11. Zhang, X.-L., Ji, W.: Deep belief networks based voice activity detection. IEEE Trans. Audio Speech Lang. Process. **21**(4), 697–710 (2012)
12. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web. LNCS, vol. 4321, pp. 325–341. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72079-9_10
13. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015)
14. Zhang, Y.D., Morabito, F.C., Shen, D., Muhammad, K.: Advanced deep learning methods for biomedical information analysis: an editorial. Neural Netw. Off. J. Int. Neural Netw. Soc. **133**, 101–102 (2020)
15. Yamashita, R., Nishio, M., Do, R.K.G., Togashi, K.: Convolutional neural networks: an overview and application in radiology. Insights Imaging **9**(4), 611–629 (2018)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
17. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
18. Leung, H., Haykin, S.: The complex backpropagation algorithm. IEEE Trans. Signal Process. **39**(9), 2101–2104 (1991)
19. Lin, T., Kong, L., Stich, S., Jaggi, M.: Extrapolation for large-batch training in deep learning. In: International Conference on Machine Learning, pp. 6094–6104. PMLR (2020)
20. Neal, R.M.: Probabilistic inference using Markov chain Monte Carlo methods. Department of Computer Science, University of Toronto Toronto, Ontario, Canada (1993)
21. Andrieu, C., Doucet, A., Holenstein, R.: Particle markov chain monte carlo methods. J. Royal Stat. Soc. Ser. B (Stat. Methodol.) **72**(3), 269–342 (2010)
22. Robert, C., Casella, G.: Monte Carlo Statistical Methods. Springer, Heidelberg (2013). https://doi.org/10.1007/978-1-4757-4145-2
23. Chaari, L., Batatia, H., Dobigeon, N., Tourneret, J.Y.: A hierarchical sparsity-smoothness bayesian model for l0+l1+l2 regularization. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1901–1905 (2014)
24. Chaari, L.: A bayesian grouplet transform. Signal Image Video Process. **13**, 871–878 (2019)
25. Hanson. K.M.: Markov Chain Monte Carlo posterior sampling with the hamiltonian method. In: Medical Imaging 2001: Image Processing, vol. 4322, pp. 456–467. International Society for Optics and Photonics (2001)
26. Chaari, L., Tourneret, J.-Y., Chaux, C., Batatia, H.: A Hamiltonian Monte Carlo method for non-smooth energy sampling. IEEE Trans. Signal Process. **64**(21), 5585–5594 (2016)
27. Chaari, L., Tourneret, J.Y., Batatia, H.: A plug and play Bayesian algorithm for solving myope inverse problems. In: European Signal Processing Conference EUSIPCO, pp. 742–746 (2018)
28. Moreau, J.-J.: Proximité et dualité dans un espace hilbertien. Bull. de la Société mathématique de France **93**, 273–299 (1965)

29. Chaux, C., Combettes, P.L., Pesquet, J.Y., Wajs, V.R.: A variational formulation for frame-based inverse problems. Inverse Prob. **23**(4), 1495 (2007)
30. Angelov, P., Soares, E.A.: Sars-cov-2 ct-scan dataset: a large dataset of real patients ct scans for sars-cov-2 identification. medRxiv (2020)
31. Recht, B., Roelofs, R., Schmidt, L., Shankar, V.: Do cifar-10 classifiers generalize to cifar-10? arXiv preprint arXiv:1806.00451 (2018)
32. Lee, C.H., Xu, X., Eun, D.Y.: Beyond random walk and metropolis-hastings samplers: why you should not backtrack for unbiased graph sampling. ACM SIGMETRICS Perf. Eval. Rev. **40**(1), 319–330 (2012)
33. Sun, S., Cao, Z., Zhu, H., Zhao, J.: A survey of optimization methods from a machine learning perspective. IEEE Trans. Cybern. **50**(8), 3668–3681 (2019)
34. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
35. Muhammad, U., Wang, W., Chattha, S.P., Ali, S.: Pre-trained vggnet architecture for remote-sensing image scene classification. In 24th International Conference on Pattern Recognition (ICPR), pp. 1622–1627 (2018)