Bayesian Hyperparameter Optimization of Deep Neural Network Algorithms Based on Ant Colony Optimization

Sinda Jlassi[1], Imen Jdey[1,2(✉)], and Hela Ltifi[1,2]

[1] Faculty of Sciences and Techniques of Sidi Bouzid, University of Kairouan, Kairouan, Tunisia
Jlassi.Sinda@fstsbz.u-kairouan.tn, {imen.jdey, hela.ltifi}@fstsbz.rnu.tn

[2] REsearch Groups in Intelligent Machines, University of Sfax, National School of Engineers (ENIS), BP 1173, 3038 Sfax, Tunisia

Abstract. Within this paper we proposed a new method named BayesACO, to improve the convolutional neural network based on neural architecture search with hyperparameters optimization. At its essence BayesACO in first side uses Ant Colony Optimization (ACO) to generate the best neural architecture. In other side, it uses bayesian hyperparameters optimization to select the best hyperparameters. We applied this method on Mnist and FashionMnist datasets. Our proposed method proven competitive results with other methods of convolutional neural network optimization.

Keywords: Convolutional neural network · Neural architecture search · Hyperparameters optimization · Ant colony optimization · Bayesian hyperparameters optimization

1 Introduction

Deep learning is a new area of machine learning research which was introduced with the aim of bringing machine learning closer to its main goal artificial intelligence. These are algorithms inspired by the structure and functioning of the brain they can learn several levels of representation [28–30].

The performance of many deep learning methods is highly sensitive to many decisions including choosing the right neural structures, training procedures and methods of hyperparameters optimization; in order to get the desired result. This is a problem whether for new users or even experts. Therefore, Automated Machine Learning (AutoML) can improve performance while saving a lot time and money. AutoML fields aims to make these decisions in data-driven, objective and automated manner. The most helpful and remarkable method of AutoML is the Neural Architecture Search (NAS) method [9].

We choose one of the latest NAS method that uses the ant colony optimization algorithm to design its structure with minimal weights [7]. Where the general concept

J. Lladós et al. (Eds.): ICDAR 2021, LNCS 12823, pp. 585–594, 2021.
https://doi.org/10.1007/978-3-030-86334-0_38

of ant colony optimization algorithms, inspired by original ant demeanor [17], is a combination of prerequisites about a promising solution structure with basic in-formation about the priori obtained network structure [4]. It is branched into several types, which are of interest to us in this study, namely Ant Colony System (ACS), where a group of ant team up to explore good solutions for treatment providers, using an indirect form of pheromone mediated communication deposited at the edges of the travelling salesman problem (TSP) diagram while building solutions [8].

The hyperparameters optimization has a significant impact on the performance of the neural network. Many technologies have been applied successfully. The most common ones are grid search, random search, and Bayesian optimization [2]. Grid search, searches all possibilities. So, it takes a lot of time devoted to the search of hyperparameters. Whereas, random search is based on Grid search, with the aim of creating a network of excessive parameter values, to randomly choose a combination of them. Therefore, this process automatically takes a long time, so it cannot converge with global optimum or guarantee a stable and competitive result. These two methods need all the possible values for each parameter. On the other hand, Bayesian optimization just needs the order of values. The Bayesian optimization is looking for a global optimum with minimal stages [12].

In this work, we are interested in modeling a new optimization process for a convolutional neural network model. To verify the feasibility of our work we compared it with others previous methods of optimization, such as Deepswarm [7], Udeas [2], and LDWPSO [3].

The remainder of the paper is organized as follows: Sect. 2 presents the study background concerning the deep learning, the ant colony optimization and the bayesian hyperparameter optimization; Sect. 3 introduces our proposed method; Sect. 4 provides an evaluation of our method; Sect. 5 concludes the paper and explores possible future directions.

2 Study Background

2.1 Deep Learning

Deep learning supports computer models composed of different processing layers to explore representations of data with different advanced levels these technics have dramatically afflicted the technical level of various domains [28, 29].

A deep neural network, within deep learning, includes many categories, including convolutional neural networks. So convolutional refers to a computation, and it is a specialized type of linear operation. They are simply neural networks that use convolution rather than repeating the general matrix in at least one of their layers. Its first appearance was in the 1990s, when models were developed to recognize handwritten numbers and were of high performance [18]. Even CNN structures have seen increased development, with ImageNet's challenge error rate dropping below 4%. When developing AlexNet researchers from 8 layers to 152 layers [19]. CNN has also excelled in other computer vision tasks, such as human action recognition [20], object localization [21, 22], pedestrian detection [23], and face recognition [24]. CNN subsequently demonstrated that it was effective in natural language and speech processing, and achieved excellent results in stratification [25], sentence modeling [26], and speech recognition [27].

2.2 Ant Colony Optimization

The algorithm to improve the ant colony mainly aims to find the path closest to the target [8]. By secreting the so-called pheromone, in order to leave a trace of the following ants, and according to the nature of the ants, it follows the smell of the most recent and most pheromone in terms of quantity, which means the ant that took the shortest path towards the target. The choice is between each node and another with the rule of choosing a pseudo random procedure (based on probability). The environment altered in two various modes, one to update the local track, when the other to update the global track as shown in Fig. 1. The amounts of pheromone are updated in the edges, where ants move between the contract, with the aim of building a new solution. Automatically all ants apply updating pheromone offline. Global Trail Update comes when all nodes complete through the shortest path by updating the edges that have the best ingenuity on their way [4].

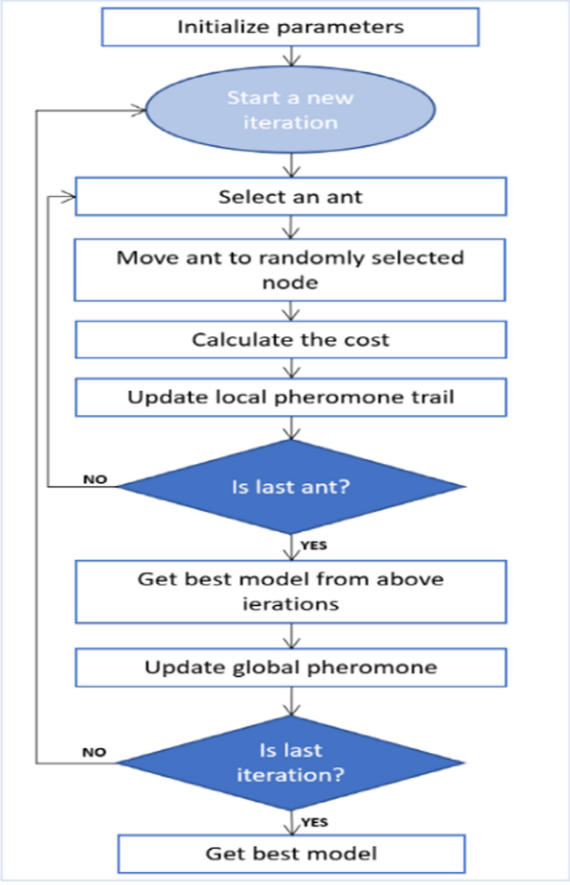

Fig. 1. Ant colony optimization steps

The contact weights are adjusted according to the number of ants, so different combinations of contact weight values are determined. For an independent Ant Colony Optimization (ACO) training application and ACO-BP hybrid training for forward neural networks training for categorization of patterns [10]. Through the global research of the ant colony, weights and bias of artificial neural network (ANN) and deep neural network (DNN) models were modified to achieve optimum performance in the prediction of capital cost [1]. ACO is used to form an ant clan that uses pheromone information to collectively search for the best neurological structure [7]. Is also used for recurrent neural network to develop the structure [11].

2.3 Bayesian Hyperparameter Optimization

We can say that the bayesian hyperparameters optimization algorithm is repeated $t - 1$ times [12]. Within this loop, it will increase the acquisition function and then update the pre-distribution. With each cycle, the pre-distribution is constantly updated, and based on the new setting, the dot at whom the acquisition function is incremented and collected to the training data set is organize. The entire process is duplicated until the maximum number of duplicates is reached or the difference between the current value and the optimum value obtained so far is less than a predetermined threshold [12]. Using bayesian optimization to control data, compare models for an ideal network [13]. Gaussian Process Model (GP) is an algorithm for integrating learning performance. Its performance is affected by some options like kernel type and handling super parameters. The advance is that, the algorithms take into account the variable cost of learning algorithm experiments, which can benefit from multiple cores of parallel experiments [14]. Where the only available standards are artificial test functions that do not represent practical applications. To alleviate this problem, a library of standards was introduced from the pre-eminent application to improve hyperparameters [15]. In [16], the authors define a new kernel for conditional parameter distances that clearly includes information about the relevant parameters in a particular structure, to link the collected performance data for different architectures. In the case of searching for structures that have parameters of different values. For example, we might want to research neural network architectures with an unknown number of layers.

The central idea of BO is to optimize the hyperparameters of the neural network. In this work, we suggest an improvement, that we called BayesACO.

3 Proposed Algorithm

The parameters embedded in our model are internal to the neural network, assessed automatically or learned from learning samples. Therefore, hyperparameters, which are external parameters determined by the neural network, have a great influence on the accuracy of the neural network, hence it is hard to detect these values. Our algorithm takes place in two phases as shown in Fig. 2:

- Initial phase: Neural architecture searches with ANT COLONY OPTIMIZATION to obtain the best weight that form the convolutional neural network structure which improves the weight of the CNN model in order to get optimal performance.

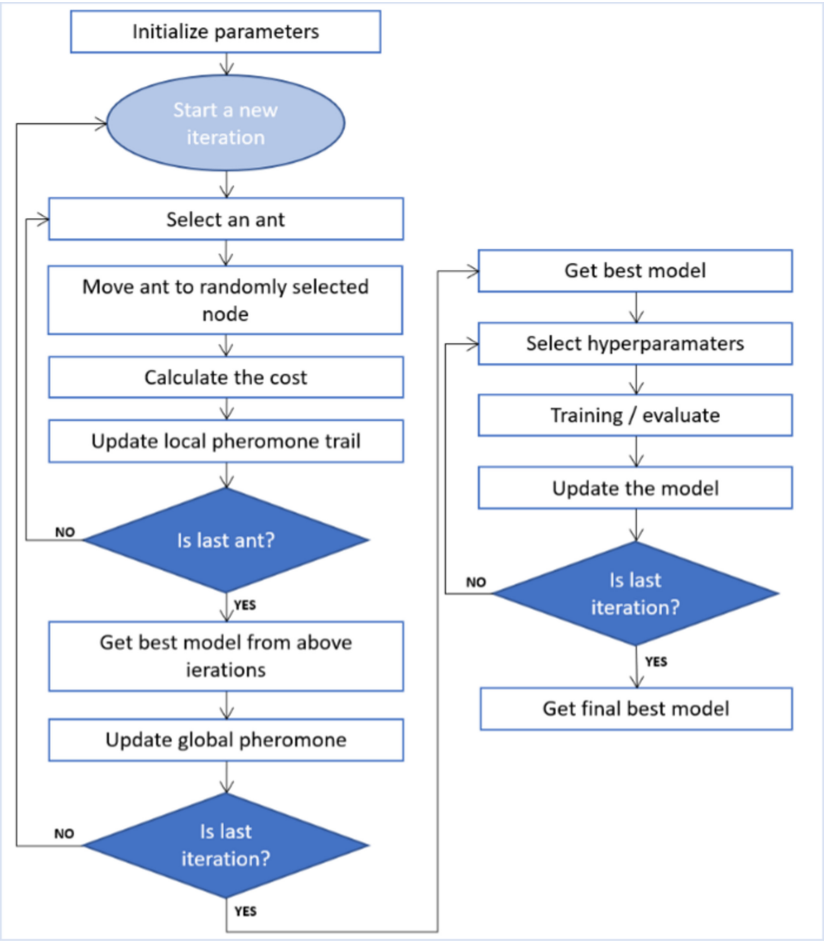

Fig. 2. BayesACO workflow

- Second phase: Using Bayesian Hyperparameters Optimization enables the optimization of certain number of hyperparameters. Thus, we can find better hyperparameters in less period of time because they are reflected on the best set of hyperparameters to an evaluation based on the former experiences.

3.1 Bayesian Hyperparameter Optimization

Throughout this phase, we are going to apply the Bayesian Hyperparameters Optimization. To begin first, we have to specify which function is to be optimized. Then we start with selecting the hyperparameters of the model in a random way. Afterwards, we train the model by evaluating and updating it until it gets the best performance. This stage is repeated with certain number of iterations which is specified by the user in a manner that each iteration depends on the previous one.

4 Application

Along this section, we will deal with hyperparameter and parameter optimization. Evaluating hyperparameters and model structure in CNN to get the best performance as possible is performed on the Mnist and FashionMnist datasets.

Table 1 shows the optimization parameters of our methodology. The number of filters in the convolution Node is optimized between 32 and 128, and the kernel size between 1 and 5 the learning rate of Dropout Node is between 0.1 and 0.3 the "stride" of Pooling Node between 2 and 3, and the type which max or average the size of Dense Node can be 64 or 128, and their activation function is ReLU or Sigmoid validation split between 0.0 and 0.3, batch size which 32, 64 or 128 and epochs number which 5, 10 or 20.

Table 1. Optimization parameters of BayeACO.

Parameter	Optimization value
filter_count	{32, 64, 128}
kernel_size	{1, 3, 5}
rate	{0.1, 0.3}
pool_type	{max, average}
stride	{2, 3}
output_size	{64, 128}
activation	{ReLU, Sigmoid}
validation split	{0.1, 0.3}
batch size	{32, 64, 128}
epochs	{5, 10, 20}

The main parameters of BayesACO use for optimization with mnist. The ant count is 16 and the maximum number of depths is 10. The epochs number is 15, the batch size equals 64, and the learning rate is equal to 0.1.

The first model that we present in Fig. 3 is composed of two convolutional layers and two fully connected layers and one max pooling, dropout and flatten layer.

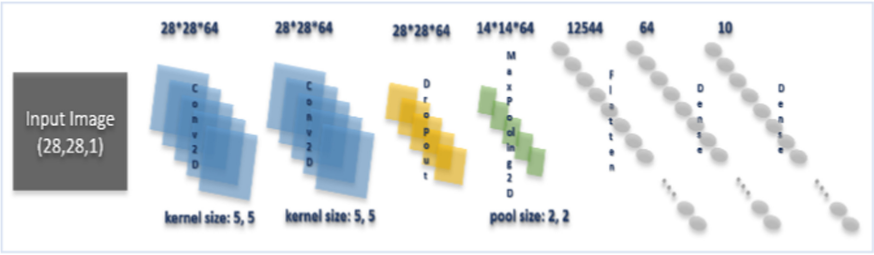

Fig. 3. The best architecture discovered with Mnist

From Fig. 4 The accuracy of training and testing increases with the number of epochs, this reflects that with each epoch the model learns more information. If the precision is reduced then we will need more information to teach our model and therefore we must increase the number of epochs and vice versa. Likewise, the learning and validation error decreases with the number of epochs. We also notice that the total misclassified images are 57 images, an error rate of 0.57% and the total well classified images is 9943 an accuracy rate of 99.43%.

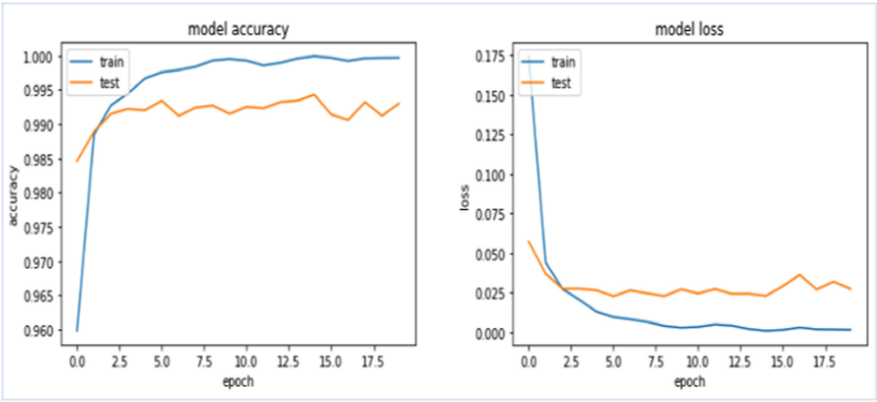

Fig. 4. Accuracy and Error for mnist model Fashion-MNIST dataset

The initial parameters of BayesACO use for optimization with fashionmnist. The ant count is 16 and the maximum number of depth is 15. The epochs number is 20, the batch size equals 64, and the learning rate is equal to 0.1.

The second model that we present in Fig. 5 is composed of three layers of convolution and two layers of averagepooling and a dropout and fully connected layer.

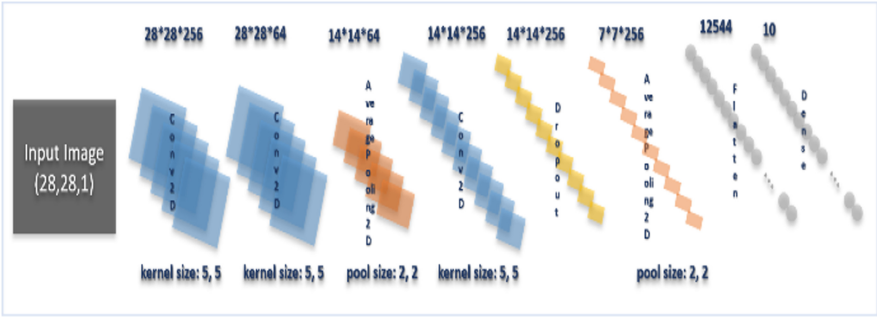

Fig. 5. The best architecture discovered with FashionMnist

Beyond this, we compare the differences between the results of our final method and other methods of improving the convolutional neural network such as Deepswarm, Udeas and LDWPSO.

The variations in results of the algorithms obviously indicate the effectiveness of our proposed methodology in term of cost which has been defined as the value of the test accuracy as it is clarified in the Table 2.

Table 2. Results of optimization methods.

Method	Accuracy	
	Mnist	FashionMnist
Lenet5	99%	81.6%
Resnet18	99.2%	92.1%
XgBoost	95.8%	89.8%
Deepswarm [7]	99.22%	93.4%
LDWPSO [3]	98.95%	–
uDeas [2]	99.1%	–
BayesACO	99.43%	93.8%

We can conclude that our result with the mnsit database is 3.63% higher compared to the xgboost architecture, and higher by 0.48 with the ldwpso optimization method and with the fashionmnsit database the precision obtained is 12.4% higher compared to the lenet5 architecture, and higher by 8.71% with the bayesian optimization method.

5 Conclusion

In this paper, we are interested in integrating the bayesian optimization of hyperparameters in the stages of an existing neural architecture search. This system was developed to optimize the convolutional neural network.

The combination of the hyperparameters optimization with the neural architecture search allows reducing human intervention because the process of extracting the network will become fully automated. Thus, we gain time and give us more accurate results.

As perspectives, we think it is important to run the proposed method on other databases. To evaluate the method in terms of time compared to other competing methods, and to develop this approach using more advanced techniques than those which already exist to obtain better results.

References

1. Zhang, H., et al.: Developing a novel artificial intelligence model to estimate the capital cost of mining projects using deep neural network-based ant colony optimization algorithm. Res. Policy **66**, 101604 (2020)
2. Yoo, Y.J.: Hyperparameter optimization of deep neural network using univariate dynamic encoding algorithm for searches. Knowl.-Based Syst. **178**, 74–83 (2019)

3. Serizawa, T., Fujita, H.: Optimization of convolutional neural network using the linearly decreasing weight particle swarm optimization. arXiv:2001.05670 (2020)
4. Katiyar, S., Ibraheem, N., Ansari, A.Q.: Ant colony optimization: a tutorial review. In: National Conference on Advances in Power and Control, pp. 99–110 (2015)
5. Wu, J., Chen, X.Y., Zhang, H., Xiong, L.D., Lei, H., Deng, S.H.: Hyperparameter optimization for machine learning models based on bayesian optimization. J. Electr. Sci. Technol. **17**(1), 26–40 (2019)
6. Andonie, R.: Hyperparameter optimization in learning systems. J. Membr. Comput., 1–13 (2019)
7. Byla, E., Pang, W.: DeepSwarm: optimising convolutional neural networks using swarm intelligence. In: Zhaojie, J., Yang, L., Yang, C., Gegov, A., Zhou, D. (eds.) Advances in Computational Intelligence Systems: Contributions Presented at the 19th UK Workshop on Computational Intelligence, Portsmouth, UK, 4–6 September, 2019, pp. 119–130. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-29933-0_10
8. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)
9. Hutter, F., Kotthoff, L., Vanschoren, J. (eds.): Automated Machine Learning. TSSCML, Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5
10. Mavrovouniotis, M., Yang, S.: Training neural networks with ant colony optimization algorithms for pattern classification. Soft Comput. **19**(6), 1511–1522 (2014)
11. Desell, T., Clachar, S., Higgins, J., Wild, B.: Evolving deep recurrent neural networks using ant colony optimization. In: Ochoa, G., Chicano, F. (eds.) Evolutionary Computation in Combinatorial Optimization, pp. 86–98. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-16468-7_8
12. Zhang, X., Chen, X., Yao, L., Ge, C., Dong, M.: Deep neural network hyperparameter optimization with orthogonal array tuning. In: Advances in Neural Information Processing, Vancouver, BC, Canada, pp. 287–295 (2019)
13. MacKay, D.J.C.: Probable networks and plausible predictions—a review of practical Bayesian methods for supervised neural networks. Netw. Comput. Neural Syst. **6**(3), 469–505 (1995)
14. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems, Lake Tahoe, Nevada, pp. 2951–2959 (2012)
15. Eggensperger, K., et al.: Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In: NIPS Workshop on Bayesian Optimization in Theory and Practice, 10 December 2013
16. Swersky, K., Duvenaud, D., Snoek, J., Hutter, F., Osborne, M.A.: Raiders of the lost architecture: Kernels for Bayesian optimization in conditional parameter spaces. arXiv:1409.4011 (2014)
17. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE Comput. Intell. Mag. **1**(4), 28–39 (2006)
18. Lecun, Y., Bottou, L., Bengio, Y., Haffnern, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE (1998)
19. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision (2016)
20. Ji, S., Xu, W., Yang, M., Yu, K.: 3D convolutional neural networks for automatic human action recognition. US Patent 8,345,984 (2013)
21. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE (2015)
22. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. Adv. Neural Inf. (2015)

23. Angelova, A., Krizhevsky, A., Vanhoucke, V., Ogale, A., Ferguson, D.: Real-time pedestrian detection with deep network cascades (2015)
24. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE (2015)
25. Kim, Y.: Convolutional neural networks for sentence classification. arXiv:1408.5882. (2014)
26. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint (2014)
27. Abdel-Hamid, O., Mohamed, A., Jiang, H., Deng, L., Penn, G., Yu, D.: IEEE/ACM Transactions on Audio, Speech, and Language Processing (2014)
28. Bengio, Y.: Learning deep architectures for AI. Found. Trends Mach. Learn. **2**(1), 1–127 (2009)
29. Deng, L., Yu, D.: Deep learning: methods and applications. Found. Trends Signal Process. **7**(3–4), 197–387 (2014)
30. Jdey, I., Bouhlel, M.S., Dhibi, M.: Comparative study of two decisional fusion techniques: dempester Shafer theory and fuzzy integral theory in radar target recognition. Fuzzy Sets Syst. **241**, 68–76 (2014)