Tech Science Press

ICCES, 2024, vol.32, no.1
DOI: 10.32604/icces.2024.013279

PROCEEDINGS

Application of Simplified Swarm Optimization on Graph Convolutional Networks

Ho-Yin Wong[1], Guan-Yan Yang[1,*], Kuo-Hui Yeh[2] and Farn Wang[1]

[1]Department of Electrical Engineering, National Taiwan University, Taipei City, 106319, Taiwan R.O.C.

[2]Institute of Artificial Intelligence Innovation, National Yang Ming Chiao Tung University, Hsinchu City, 300093, Taiwan R.O.C.

[*]Corresponding Author: Guan-Yan Yang. Email: f11921091@ntu.edu.tw

ABSTRACT

1 Introduction

This paper explores various strategies to enhance neural network performance, including adjustments to network architecture, selection of activation functions and optimizers, and regularization techniques. Hyperparameter optimization is a widely recognized approach for improving model performance [2], with methods such as grid search, genetic algorithms, and particle swarm optimization (PSO) [3] previously utilized to identify optimal solutions for neural networks. However, these techniques can be complex and challenging for beginners. Consequently, this research advocates for the use of SSO, a straightforward and effective method initially applied to the LeNet model in 2023 [4]. SSO optimizes various parameters, including kernel number, size, stride, number of units in fully connected layers, and batch size, but has not previously focused on activation functions, optimizers, or regularization. These elements are the focus of the current study. Additionally, as Graph Convolution Neural Networks (GCN) also incorporate convolutional structures, SSO is applied to GCN models as well. The primary contributions of this study are:

1. Demonstrating the applicability of SSO to GCN.
2. Highlighting importance of choosing activation functions and optimizers over structural modifications.
3. Providing a systematic approach to identify effective parameters through SSO.

2 The Proposed Method

The SSO algorithm is characterized by its straightforwardness. The update mechanism for SSO is detailed in Table 1.

Table 1. Update mechanism of SSO [3]

$0 <=$ random variable $< C_g$	$x_{i,j} = g_j$
$C_g <=$ random variable $< C_p$	$x_{i,j} = p_{i,j}$
$C_p <=$ random variable $< C_w$	$x_{i,j} = x_{i,j}$ (no change)
$C_w <=$ random variable $<= 1$	$x_{i,j} =$ random value

The update of each variable $x_{i,j}$ within a solution vector $X_i = (x_{i,1}, x_{i,2}, \ldots)$ is determined by a random variable between 0 and 1. Here, g_j represents the variable from the best solution, and $p_{i,j}$ denotes the variable from the original solution. The probabilities C_g, C_p and C_w dictate the likelihood that the updated $x_{i,j}$ will match g_j, $p_{i,j}$, $x_{i,j}$, and a random value, respectively, set at 0.4, 0.7, and 0.9 in this study. The experimental configurations for GCNs are detailed in Tables 2, highlighting the extensive range of variables or hyperparameters involved.

Table 2: Experiment setting

Variable	Hyperparameter	Range
x_1	No. of conv layers	1 - 5
x_2	Optimizer	Adam/AdamW/ Adagrad/Amsgrad
x_3	Activation function	Leaky ReLU/ReLU/ Tanh
x_4	Patience for early stop	100 - 200
x_5	No. of epochs	200 - 2000
$x_6 - x_{10}$	No. of neurons in 1st – 5th layers	16 - 128
x_{11}	Dropout rate	0.4 – 0.8
x_{12}	Learning rate	0.001 - 0.1
x_{13} - x_{17}	Weight decay in different conv layers	5e-4 – 0.01
x_{18}	Weight decay in fc layer	5e-4 – 0.01

3 Experiment

In this study, GCNs employed the Cora, Citeseer, and Pubmed datasets. All experiments were conducted using Python 3.10.12 and Pytorch 2.2.1 on Google Colab, executed on an Intel® Xeon® CPU @ 2.00GHz with 16 GB of RAM and an NVIDIA Tesla T4 GPU.

The application of the Simplified Swarm Optimization (SSO) algorithm involved three steps:

1. Conduct an initial run to identify the optimal activation function and optimizer without constraints.
2. With the activation function and optimizer fixed, determine the optimal network structure and other hyperparameters such as dropout rate or weight decay.
3. Optionally, implement a learning rate scheduler or data augmentation upon finding the optimal solution if deemed necessary.

The results, indicating the optimal activation function and optimizer, are presented in Figures 1 to 2.

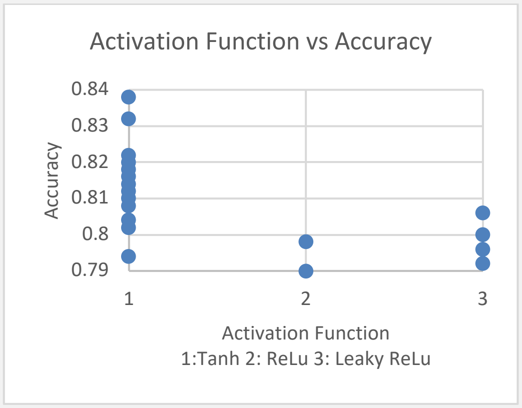

Fig. 1. Relationship between activation function and accuracy

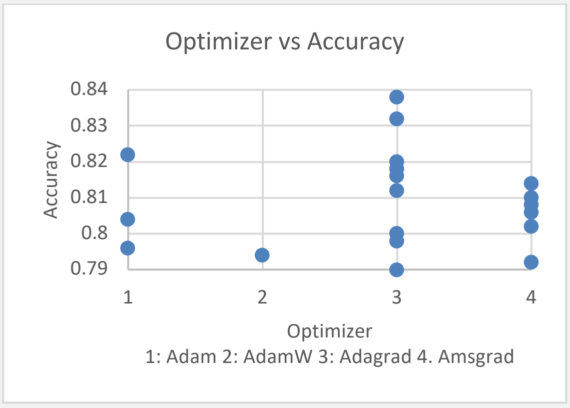

Fig. 2. Relationship between optimizer and accuracy

Following steps 2, the SSO algorithm identified an optimal solution with the best activation function (Tanh) and optimizer (Adagrad), as detailed in Tables 3.

Table 3. Result of GCN experiment

Dataset	Method	Accuracy
Cora	GCN [5] (ReLU)	0.8150
	SSO-GCN (Tanh)	**0.8348**
Citeseer	GCN [5] (ReLU)	0.7030
	SSO-GCN (Tanh)	**0.7262**
PubMed	GCN [5] (ReLU)	0.7900
	SSO-GCN (Tanh)	**0.7968**

4 Conclusion

This research demonstrates that SSO is an effective method for identifying optimal solutions in GCN. SSO systematically suggests crucial hyperparameters, such as activation functions and optimizers. Additionally, SSO's applicability extends to other neural network architectures. Currently, we are also conducting research to make SSO applicable to more different neural networks. For future work, we think SSO potentially can improve the test case generation [6], test prioritization [7], deep learning process [8], and blockchain latency [9, 10].

KEYWORDS

Deep learning; node classification; graph convolutional networks; simplified swarm optimization; hyperparameter optimization

Funding Statement: This work was partly supported by the National Science and Technology Council (NSTC) of Taiwan under grants MOST 110-2221-E-002-069-MY3, NSTC 112-2634-F-011-002-MBK, NSTC 113-2634-F-011-002-MBK, and NSTC 111-2221-E-A49-202-MY3. Additionally, it was supported by the 2024 CITI Visible Project: Questionnaire-based Technology for App Layout Evaluation, Academia Sinica, Taiwan, ROC. Moreover, this work received partial funding from the National Taiwan University under Grant 113L7256, within the framework of the Higher Education Sprout Project by the Ministry of Education, Taiwan. Guan-Yan Yang is supported by National Science and Technology Council Graduate Research Fellowship (NSTC-GRF), under Grant 113WFA0110119.

Acknowledgements: Ho-Yin Wong and Guan-Yan Yang are co-first author. CRediT authorship contribution statement: conceptualization, G.-Y.Y.; investigation G.-Y.Y. and H.-Y.W.; methodology, G.-Y.Y; resources, K.-H.Y.; software, G.-Y.Y. and H.-Y.W.; validation, G.-Y.Y., H.-Y.W. and F.W.; writing—original draft preparation, H.-Y.W. and G.-Y.Y.; writing—review and editing, G.-Y.Y., F.W. and K.-H.Y.; supervision, F.W. and K.-H.Y.; project administration, G.-Y.Y. and F.W.; funding acquisition, F.W. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29(3), 93-106*.
2. Hazan, E., Klivans, A., & Yuan, Y. (2017). Hyperparameter optimization: A spectral approach. *arXiv preprint arXiv:1706.00764.*
3. Lorenzo, P. R., Nalepa, J., Kawulok, M., Ramos, L. S., Pastor, J. R. (2017). Particle swarm optimization for hyper-parameter selection in deep neural networks. *In Proceedings of the genetic and evolutionary computation conference*. Berlin, Germany.
4. Wei-Chang Y., Yi-Ping L., Yun-Chia L., Chyh-Ming L., Chia-Ling H. (2023). Simplified swarm optimization for hyperparameters of convolutional neural networks. *Computers & Industrial Engineering, 177,* 109076.
5. Thomas N. K., Max W. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.
6. Huang, G. D., Wang, F. (2005, October). Automatic test case generation with region-related coverage annotations for real-time systems. *In International Symposium on Automated Technology for Verification and Analysis,* pp. 144-158. Berlin, Heidelberg: Springer Berlin Heidelberg.
7. Yang, G. -Y., Ko, Y. -H., Wang, F., Yeh, K. -H., Chang, H. -S., Chen, H. -Y. (2024). Automated vulnerability detection using deep learning technique. *ArXiv Preprint*. Retrieved from https://arxiv.org/abs/2410.21968.
8. Yeh, K. -H., Yang, G. -Y., Butpheng, C., Lee, L. F., Liu, Y. H. (2022). A secure interoperability management scheme for cross-blockchain transactions. *Symmetry, 14(12),* 2473.
9. Yang, G. -Y., Yeh, K. -H., Lee, L. -F. (2021, October). Towards a novel interoperability management scheme for cross-blockchain transactions. *In 2021 IEEE 10th Global Conference on Consumer Electronics (GCCE),* pp. 942-943. Kyoto, Japan.
10. Yang, G. -Y., Wang, F., Gu, Y. -Z., Teng, Y. -W., Yeh, K. -H., Ho, P. -H., Wen, W. -L. (2024). TPSQLi: test prioritization for SQL injection vulnerability detection in web applications. *Applied Sciences, 14(18),* 2076-3417.