E-NSSA: Efficient Neural Architecture Search using Sorting Algorithms

Muhammad Ali Hamza [1], Usama Ejaz [2], Heemin Park [3], Hyun-Chul Kim [4]

[1,2,3,4] Department of Software, Sangmyung University, Cheonan 31066, South Korea.

[1] alihamzaiub13@gmail.com,[2] usamaijaz404@gmail.com,[3] heemin@smu.ac.kr, [4] hyunchulk@gmail.com

Abstract

This paper introduces E-NSSA – an approach that explores the use of efficient non-dominated sorting algorithm in NSGA-Net for neural architecture search (NAS). The aim is to find a diverse set of trade-off network architectures that balance the dual objectives of minimizing error and computational complexity. Our result shows that using the efficient non-dominated sorting algorithm results in better performance in NSGA-Net. Our proposed approach achieves 1.36% higher accuracy and 0.6% better classification error rate on the CIFAR-10 dataset as compared with NSGA-Net. Our results are promising and demonstrate the potential of using efficient non-dominated sorting in NSGA for neural architecture search as our approach finds neural architectures with better accuracy and relatively less complicated architecture compared to NSGA-Net.

1. INTRODUCTION

The success of deep convolutional neural networks (CNNs) in image analysis tasks can be attributed to the development of many CNN architectures such as YOLO [1], AlexNet [2], VGG [3], GoogLeNet [4], ResNet [5], and DenseNet [6] for image classification. However, building such network architectures requires painstaking efforts and human ingenuity for years. Aside from that, Neural architecture search (NAS) methods are designed to automate this process of finding and building network architectures. The state-of-the-art methods such as, reinforcement learning (RL) [7], are inefficient in their use of the search space and require significant computational resources. Moreover, the Gradient-based methods such as, [8] are unable to deal with multiple conflicting objectives. Furthermore, most approaches search over a single computation block and repeat it to form a complete network. To address these challenges, we propose an efficient non-dominated sorting algorithm in NSGA-Net [9] for NAS that achieves better performance and can handle multiple conflicting objectives.

NSGA-Net [9] is a population-based evolutionary algorithm that comprises three steps: population initialization based on prior-knowledge from hand-crafted architectures, exploration through crossover and mutation, and exploitation utilizing hidden useful knowledge in a Bayesian network. The Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [10] is a core component of NSGA-Net.

In this paper, we propose an approach that uses four sorting algorithms such as, (i) Efficient Non-Dominated sorting [13], (ii) Fast Non-Dominated sorting [11], (iii) Naive Non-Dominated sorting [12] and (iv) Tree-based non-Dominated sorting [14] in NSGA-Net for neural architecture search (NAS). Moreover, we evaluated the performance of our approach on the CIFAR10 image classification task by minimizing both classification error and computational complexity. The computational complexity as the number of floating-point operations (FLOPs) a network performs during a forward pass. Our experimental results demonstrate that the performance of NSGA-Net can be improved by replacing non-dominated sorting with efficient non-dominated sorting as we get 1.36% higher accuracy than NSGA-Net. Additionally, our results find a diverse set of network architectures out-performed the architectures found by NSGA-Net in terms of improved accuracy and less size of FLOPs.

2. METHODOLOGY

In this section, we first implement baseline approach and uses it to compared with our proposed approach of E-NSSA. In E-NSSA, we implemented four sorting algorithms in the baseline approach.

2.1 EXPERIMENTAL SETUP

The CIFAR-10 [15] dataset is used and divide the original training set into separate training and validation sets for architecture search purposes. The original testing set of CIFAR-10 is used to obtain the final classification accuracy of models on the trade-off front after the search is complete. Moreover, the classification error rate is also measure on CIFAR-10 test dataset for incorrectly classified instances. In the NSGA-Net paper, it has been trained on for 320 epochs which takes more than one day to get training completed on one GPU machine. To enable us to perform several experiments with time efficient manner, we reduced no. of training epochs and trained the NSGA-Net on only 10 epochs using the original setup provided in the NSGA-Net paper [1].

In the encoding scheme of baseline NGSA-Net [9] model, every phase represents a gray box with a maximum of 6 nodes, 20 layers, and pooling after every phase as shown in the Figure 1. However, in E-NSSA, we made set the size of layers to 40 with a maximum of 12 nodes in encoding scheme, which would encourage population diversity. Furthermore, usage of max pooling after every phase would be a more optimized encoding scheme than normal pooling. Hence, we can find more prominent features in every phase instead of all features.

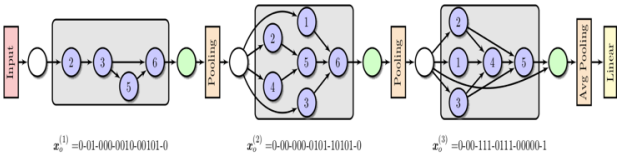

Figure 1. Encoding scheme

Furthermore, in our E-NSSA approach, we modified the NSGA-Net baseline architecture by adding 4 different non-dominated sorting algorithms such as efficient, naive, fast and tree-based sorting. Both approaches are trained on similar settings and results are shown in the Table 1.

2.2 Results

Table 1 shows the results of our approach in comparison with NSGA-Net on CIFAR-10 [15] dataset. NSGA-Net achieves 85.84% accuracy when trained on 10 epochs. It achieves 3.0% error rate. Compared to NSGA-Net, our E-NSSA using efficient non-dominated sorting [13] improves the training accuracy by 1.36% and achieves an 86.2% accuracy score. Moreover, it also improves the error rate by 0.6% and gives a 2.4% error rate. Apart from that, our E-NSSA's fast non-dominated sorting algorithm also gets 0.16% higher training accuracy than NSGA-Net.

Approaches	**Sorting Algorithm**	**Error Rate**	**Accuracy**
NSGA-Net [9]	Non-Dominated (ND) [10]	3.0	85.84
E-NSSA	Tree-based ND [14]	3.1	84.51
	Naive ND [12]	2.8	85.88
	Fast ND [11]	2.6	86.0
	Efficient Non-Dominated [13]	**2.4**	**87.2**

Table 1. Training Results.

3. ARCHITECTURE SEARCH

Table 2 shows a comparison between NSGA-Net and our approach E-NSSA. For comparison, we show the results of NSGA-Net and our best-performed model (in terms of training accuracy) e.g., efficient non-dominated sorting [13]. The results are reported under the same settings as the top 3 best-found architectures for the initial 10 iterations only. In all 10 iterations, our approach found better neural architectures than NSGA-Net. The NSGA-Net found the best architecture with 84.65% accuracy and 65.15M FLOPs. As compared to NSGA-Net, our E-NSSA found the best architecture

among 10 iterations with 5.19% higher accuracy. However, the architecture becomes complicated as the FLOPs value also increases to 150.4M. Moreover, NSGA-Net's second-best architecture has 83.89% with 41.92 FLOPS. As compared to it, our approach outperforms with 1.48% higher accuracy and relatively simpler architectures by getting 85.37% accuracy and 41.57M FLOPs.

Found Architectures	**NSGA-Net [9]**		**E-NSSA**	
	FLOPs	**Accuracy**	**FLOPs**	**Accuracy**
Arch 1	65.15 M	84.65%	150.4 M	89.44%
Arch 2	41.92 M	83.89%	41.57 M	85.37%

Table 2 Top 2 architectures founds by each of approaches.

4. CONCLUSION

Our proposed methodology; E-NSSA explores the use of various sorting algorithms, such as efficient non-dominated sorting in NSGA-Net algorithm and improves its performance in terms of accuracy and FLOPs. We show that by implementing the efficient non-dominated sorting algorithm, we can achieve better performance as our test results achieves 1.36% higher accuracy and 0.6% better error rate as compared with NSGA-Net. Moreover, for the neural architecture search, our approach finds the architectures with better accuracy and relatively less complicated architecture. The neural architectures found by our approach are better than hand-crafted networks for classification on the CIFAR-10 dataset. However, all the experiments are performed in 10 epochs only. In the future, we can experiment with it by training on a large number of epochs.

REFERENCES

[1] Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classi- fication with Deep Convolutional Neural Networks. In NIPS.

[3] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Net- works for Large-scale Image Recognition. In ICLR.

[4] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In CVPR.

[6] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian QWeinberger. 2017. Densely connected convolutional networks. In CVPR.

[7] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin. 2017. Large-Scale Evolution of Image Classifiers.

[8] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. DARTS: Differentiable Architecture Search.

[9] Lu, Z., Whalen, I., Dhebar, Y., Deb, K., Goodman, E., Banzhaf, W., & Boddeti, V. N. (2020). NSGA-Net: Neural architecture search using multi-objective genetic algorithm (extended abstract). *IJCAI International Joint Conference on Artificial Intelligence*, *2021-Janua*, 4750–4754.

[10] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. 2000. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimiza- tion: NSGA-II. In International Conference on Parallel Problem Solving From Nature. Springer.

[11] Fast non-dominated sorting https://github.com/anyoptimization/pymoo/blob/main/pymoo/util/nds/fast_non_dominated_sort.py

[12] Naive non-dominated sorting https://github.com/anyoptimization/pymoo/blob/main/pymoo/util/nds/naive_non_dominated_sort.py

[13] Efficient non-dominated sorting https://github.com/anyoptimization/pymoo/blob/main/pymoo/util/nds/efficient_non_dominated_sort.py

[14] Tree Based non-dominated sorting https://github.com/anyoptimization/pymoo/blob/main/pymoo/util/nds/tree_based_non_dominated_sort.py

[15] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. [n. d.]. CIFAR-10 (Canadian Institute for Advanced Research). ([n. d.]). http://www.cs.toronto.edu/~kriz/cifar. html