Supplementary Material

Particle Swarm Optimization for Efficiently Evolving Deep Convolutional Neural Networks Using an Autoencoder-based Encoding Strategy

Gonglin Yuan, *Graduate Student Member, IEEE*, Bin Wang, *Graduate Student Member, IEEE*, Bing Xue, *Senior Member, IEEE*, and Mengjie Zhang, *Fellow, IEEE*

S-I. The Influence of Population Sizes on Particle Swarm Optimization (PSO)

In EAEPSO, the block structures are evolved by PSO. The population size is an essential parameter of PSO, affecting the consumed time and the performance of the final block structure. The section further investigates the influence of different population sizes on the CIFAR-10 dataset. The searched block structure's prediction accuracy on the evaluation dataset, and the computational cost of the PSO process (measured by GPU-days) are shown in Table S-I.

TABLE S-I
The searched block structure's performance and the computational cost using different population sizes.

Population Size	Accuracy	GPU-Days
10	74.09%	0.8
30	74.44%	2.0
50	75.06%	4.2

It is clearly shown that both the predictive accuracy and the GPU-days increase along with the increased population size. In the proposed EAEPSO, the particle size is set to be 30, which is a trade-off between the prediction performance and the consumed computational time. The population size can be set to be larger or smaller, considering the users' preference and available computing resources.

S-II. The Rational Exploration of Stacking the Blocks

In EAEPSO, PSO searches for a good dense block structure, and then different blocks that share the same structure are stacked together to determine the best number of blocks. In this section, we design experiments to show if the whole network's performance is closely related to the performance of one block. Specifically, five different dense block structures are randomly generated first. For each one, at most four blocks are stacked together to test the performance on the CIFAR-10 dataset considering the size of the input. Fig. S-I shows the prediction error rate with different numbers of blocks. Table S-II presents the single block's performance, the network's best performance, and the number of blocks of the best networks.

From Fig. S-I, we can see the prediction error rate decreases along with the number of blocks increasing when the number is smaller than four. From Table S-II, we can see that *Structure 5* has the lowest error rate when the network is composed of a single block, and it also achieves the best performance when consisting of 3 blocks. Generally, the network's best performance is consistent with the single block's performance. It is reasonable to select the best-performed block structure and then stack the blocks.

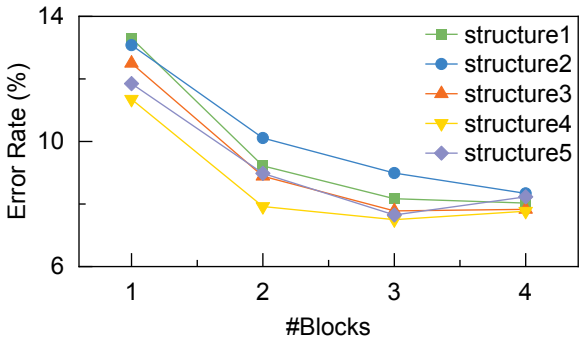

Fig. S-I. The prediciton error rates of networks stacked by different number of blocks, and there are five different block structures.

TABLE S-II
Different single blocks' prediction error rates, best networks' prediction error rates, and the corresponding numbers of blocks.

Structure	Single Block Error Rate	Lowest Network Error Rate	Corresponding Number of Blocks
1	13.30%	8.03%	4
2	13.08%	8.34%	4
3	12.50%	7.78%	3
4	11.85%	7.65%	3
5	11.35%	7.50%	3

S-III. Using PSO to Evolve Dense Blocks

PSO is usually used for addressing continuous optimization problems, and each particle, representing a potential solution, has velocity and position vectors [1]. In PSO, each particle's velocity and position are updated according to Equation (1) and Equation (2), where v_i is a vector representing the velocity of the i-th particle, and p_i expresses the position of the i-th particle. w is inertia weight, and c_1 and c_2 are two positive constants, which can be used to fine-tune the performance of PSO. r_1 and r_2 represent random numbers between 0 and 1, and p_p and p_g denotes the positions of $pBest_i$ and $gBest$, respectively.

$$v_i = w \cdot v_i + c_1 \cdot r_1 \cdot (p_p - p_i) + c_2 \cdot r_2 \cdot (p_g - p_i) \quad (1)$$

$$p_i = p_i + v_i \quad (2)$$

Algorithm S-1 shows the procedures of evolving dense block with the two fitness evaluation methods, which can be divided into three parts, i.e., the initialization part (lines 1-7), the basic fitness evaluation guided evolution part (lines 8-22), and the progressive fitness evaluation guided evolution part (lines 23-39). In the initialization part, the population is initialized according to $Algorithm\ 2$ first (line 1). After that, all the particles' velocities are initialized (line 3), and the fitness is evaluated based on the proposed basic fitness evaluation method (presented in $Algorithm\ 3$) to get their basic fitness (line 4). Then $pBest$ of each particle and $gBest$ for the population are determined (lines 5-7). After that, the basic fitness evaluation is used to guide the evolution (lines 8-22) and all particles are updated according to Equation (1) and Equation (2) (lines 13-14). If the fitness of the $gBest$ keeps increasing, all the particles will be evaluated by the basic fitness evaluation method and updated repeatedly (lines 10-15). On the contrary, if $gBest$'s fitness has not increased for five consecutive iterations, the current evolution is considered sufficient, and the particles will be evaluated by the proposed progressive fitness evaluation method (lines 23-39). Please note that n deep training dataset proportions are provided. Specifically, for each given training data proportion r_{prog}, the progressive fitness evaluation method (presented in $Algorithm\ 4$) is used to evaluate the fitness (line 27), and then the particles are updated (lines 28-29), and $pBest$ and the $gBest$ are updated based on the basic fitness and progressive fitness respectively (lines 28-32). Similarly, particles will be repeatedly evaluated and updated until $gBest$ has not been updated for five consecutive iterations. Finally, $gBest$ is selected and decoded to the network structure as the evolved dense block.

Algorithm S-1: Evolving block structure using PSO

Input: The population size N, the proportions of the progressive training data [r_{prog1},..., r_{progn}].
Output: The best block structure.

1 $P_0 \leftarrow$ Initialize population according to $Algorithm\ 2$;
2 **for** $i = 1;\ i \leq N;\ i \leftarrow i + 1$ **do**
3 $v_i \leftarrow$ randomly initialize the velocity;
4 $fitb_i \leftarrow$ evaluate basic fitness using $Algorithm\ 3$;
5 $pBest_i \leftarrow p_i$;
6 **end**
7 Select $gBest$;
8 $times \leftarrow 0$;
9 **while** $times < 5$ **do**
10 **for** $i = 1;\ i \leq N;\ i \leftarrow i + 1$ **do**
11 $fitb_i \leftarrow$ Evaluate basic fitness using $Algorithm\ 3$;
12 Update $pBest_i$;
13 $v_i \leftarrow$ Update velocity using Equation (1);
14 $p_i \leftarrow$ Update position using Equation (2);
15 **end**
16 Update $gBest$;
17 **if** $gBest$ changes **then**
18 $times \leftarrow 0$
19 **else**
20 $times \leftarrow times + 1$
21 **end**
22 **end**
23 **for** $k = 1;\ k \leq n;\ k \leftarrow k + 1$ **do**
24 $times \leftarrow 0$;
25 **while** $times < 5$ **do**
26 **for** $i = 1;\ i \leq N;\ i \leftarrow i + 1$ **do**
27 Evaluate basic fitness $fitb_i$ and progressive fitness $fitp_i$ using $Algorithm\ 4$ with proportion r_{progk} ;
28 $v_i \leftarrow$ Update velocity using Equation (1);
29 $p_i \leftarrow$ Update position using Equation (2);
30 update $pbest_i$ based on $fitb_i$;
31 **end**
32 Update $gbest$ based on $fitp$;
33 **if** $gbest$ changes **then**
34 $times \leftarrow 0$
35 **else**
36 $times \leftarrow times + 1$
37 **end**
38 **end**
39 **end**
40 Decode $gbest$ to dense block structure;
41 **Return** The evolved dense block structure.

S-IV. Additional Performance Test for EAEPSO

A. Overall Performance on SVHN

The Street View House Number (SVHN) dataset [2] is built based on the images of numbers collected from Google street view. The images are provided as the size of 32 × 32. There are 73,257 digits in the training set and 26,032 digits in the test set. EAEPSO is performed on SVHN, and Table S-III presents the comparison results with six peer competitors regarding the model scale and the test error rate.

EAEPSO ranks the second in terms of the error rate, which is 0.05% less accurate than DenseNet (k=12) [8]. However, EAEPSO's model scale is only 46.4% of DenseNet(k=12) [8]. From the overall performance, we can say EAEPSO achieves very competitive performance on the SVHN dataset.

B. EAEPSO using ConvNeXt

To further examine the effectiveness and efficiency of the proposed method, we further change the backbone structure and build the search space based on ConvNeXt [9],

TABLE S-III
PERFORMANCE COMPARISON WITH PEER COMPETITORS ON THE SVHN DATASET.

Method	Number of Parameters	Error rate
Network in Network [3]	—	2.35%
Deeply Supervised Net [4]	—	1.92%
FractalNet [5]	38.6M	2.01%
Wide ResNet [6]	11.0M	1.85%
ResNet [7]	1.7M	2.01%
DenseNet(k=12) [8]	7.0M	1.67%
EAEPSO	3.05M	1.72%

which achieves excellent performance on ImageNet. However, considering our available computational resources, we perform EAEPSO using ConvNeXt backbone, termed as EAEPSO(Conv), on the CIFAR-10 dataset [10].

1) Search space and encoding strategy:

ConvNeXt is composed of a number of blocks. In EAEPSO(Conv), PSO searches for the depth of the whole network and the convolutional layer's kernel size of each ConvNeXt block. We use a string composed of 20 integers to represent the network structure that has at most 20 blocks; each integer represents the kernel size of a ConvNeXt block, and 0 is employed to represent the corresponding block is no-existing (the total number of blocks can be smaller than 20s).

2) The performance on CIFAR-10:

The performance of EAEPSO(Conv) on CIFAR-10 is shown in Table S-IV. EAEPSO(Conv) achieves an error rate of 3.08%, a model size of 3.2M, and a computational cost of 2.4 GPU-days. The error rate is smaller than 22 competitors and is slightly larger than six peer competitors, and the reason may be the backbone structure prefers larger datasets (such as ImageNet [11]), but the data volumes of CIFAR-10 and ImageNet are vastly different. However, the model size and the computational cost are relatively small. The results show EAEPSO(Conv) is a very competitive method.

S-V. EFFECTIVENESS OF THE PROPOSED AUTOENCODER

In order to examine the effectiveness of the proposed autoencoder, three different particle representation methods are compared in the proposed method, i.e., the representation without autoencoder transforming (NAE), the representation using a comparative autoencoder which only considers the reconstruction loss (CAE), and the representation transformed by the proposed autoencoder (EAE). The parameters are set to be the same for the three experiments: the population size is 30, and the number of generations is 20. For a fair comparison, we employ a standard fitness evaluation method instead the proposed hierarchical evaluation method. The fitness of the global best particle of the 10th generation and the 20th generation are presented in Table S-V.

EAE achieves the highest fitness in both the 10th generation, and the 20th generation, and NAE's performance is worse than CAE and EAE. The use of the autoencoder greatly affects the performance, and the proposed autoencoder can improve the performance compared with a vanilla autoencoder.

TABLE S-IV
PERFORMANCE COMPARISON WITH PEER COMPETITORS ON CIFAR-10.

Model	#Parameters	Error Rate	GPU-Days
FractalNet [5]	38.6M	5.22%	—
Maxout [12]	—	9.3%	—
ResNet-101 [13]	1.7M	6.43%	—
DenseNet (k=24) [8]	27.2M	3.74%	—
Highway Network [14]	—	7.72%	—
VGG [15]	20.04M	6.66%	—
CGP-CNN [16]	2.64M	5.98%	27
NAS [17]	2.5M	6.01%	22,400
Large-scale Evolution [18]	5.4M	5.4%	2,750
Block-QNN-S [19]	6.1M	4.38%	90
MetaQNN [20]	—	6.92%	100
EIGEN [21]	2.6M	5.4%	2
CNN-GA [22]	2.9M	4.78%	35
PNASNet-5 [23]	3.2M	3.41%	150
AmoebaNet-B [24]	34.9M	2.98%	3,150
EAS [25]	23.4M	4.23%	$<$10
NASNET-A [26]	27.6M	2.97%	2,000
AECNN [27]	2.0M	4.3%	27
DENSER [28]	10.81M	5.87%	—
GeNet from WRN [29]	—	5.39%	100
CoDeepNEAT [30]	—	7.3%	—
Hier. repr-n, evolution [31]	—	3.63%	300
EffPnet [32]	2.68M	3.58%	$<$3
DARTS [33]	3.4M	2.82%	1
NSGA-Net [34]	3.3 M	2.75%	4
LEMONADE [35]	13.1M	2.58%	90
NSGANetV1-A1 [36]	0.5M	3.49%	27
Proxyless NAS [37]	5.7M	2.08%	1,500
EAEPSO(Conv)	3.2M	3.08%	2.4

TABLE S-V
PERFORMANCE COMPARISON OF DEFFERENT PARTICLE REPRESENTATIONS.

Method	gbest (10th generation)	gbest (20th generation)
NAE	0.7853	0.7867
CAE	0.7876	0.7885
EAE	0.7888	0.7901

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *International conference on neural networks*, vol. 4, 1995, pp. 1942–1948.

[2] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[3] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[4] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-Supervised Nets," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Lebanon and S. V. N. Vishwanathan, Eds., vol. 38. San Diego, California, USA: PMLR, 09–12 May 2015, pp. 562–570. [Online]. Available: https://proceedings.mlr.press/v38/lee15a.html

[5] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," *arXiv preprint arXiv:1605.07648*, 2016.

[6] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

[7] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European conference on computer vision (ECCV)*, 2016, pp. 646–661.

[8] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[9] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 11 976–11 986.

[10] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," *University of Toronto*, 2009.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[12] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *International conference on machine learning*. PMLR, 2013, pp. 1319–1327.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[14] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the genetic and evolutionary computation conference*, 2017, pp. 497–504.

[17] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[18] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 2902–2911.

[19] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2423–2432.

[20] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.

[21] J. Ren, Z. Li, J. Yang, N. Xu, T. Yang, and D. J. Foran, "Eigen: Ecologically-inspired genetic approach for neural network structure searching from scratch," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9059–9068.

[22] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3840–3854, 2020.

[23] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.

[24] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, 2019, pp. 4780–4789.

[25] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[26] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[27] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated CNN architecture design based on blocks," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 4, pp. 1242–1254, 2019.

[28] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, "Evolving the topology of large scale deep neural networks," in *European Conference on Genetic Programming*. Springer, 2018, pp. 19–34.

[29] L. Xie and A. Yuille, "Genetic cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1379–1388.

[30] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy *et al.*, "Evolving deep neural networks," in *Artificial intelligence in the age of neural networks and brain computing*. Elsevier, 2019, pp. 293–312.

[31] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," *arXiv preprint arXiv:1711.00436*, 2017.

[32] B. Wang, B. Xue, and M. Zhang, "Surrogate-assisted particle swarm optimization for evolving variable-length transferable blocks for image classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 8, pp. 3727–3740, 2022.

[33] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations (ICLR)*, 2019.

[34] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-Net: neural architecture search using multi-objective genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 419–427.

[35] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," *arXiv preprint arXiv:1804.09081*, 2018.

[36] Z. Lu, I. Whalen, Y. Dhebar, K. Deb, E. D. Goodman, W. Banzhaf, and V. N. Boddeti, "Multi-objective evolutionary design of deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 2, pp. 277–291, 2021.

[37] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *International Conference on Learning Representations*, 2018.