Genetic Algorithm-Based Pruning for Efficient DenseNet Architectures

Jingeun Kim
Department of Computer Engineering
Gachon University
Gyeonggi-do, Republic of Korea
wlsrms27@gachon.ac.kr

Yong-Hyuk Kim
School of Software
Kwangwoon University
Seoul, Republic of Korea
yhdfly@kw.ac.kr

Yourim Yoon*
Department of Computer Engineering
Gachon University
Gyeonggi-do, Republic of Korea
yryoon@gachon.ac.kr

***Abstract*—CNNs have shown remarkable performance on a variety of computer vision problems. However, CNN-based models require a lot of computational resources, which have limitations of resource-constrained environments. To address this problem, various lightweight techniques have been developed, such as pruning of network structures. This paper employed a genetic algorithm (GA) to implement pruning with various pruning rates, aiming for the efficient DenseNet. We optimized the dense connectivity pattern of DenseNet-BC (*k*=12) using a GA-based pruning method with multi-dimensional encoding scheme. We demonstrate that the proposed method can perform similarly with fewer parameters than the baseline model.**

***Index Terms*—genetic algorithm, pruning, deep learning, computer vision**

I. Introduction

Convolutional neural networks (CNNs) have demonstrated remarkable performance in various computer vision tasks, such as image classification and object recognition. The emergence of models such as LeNet [1] and AlexNet [2], which have a simple structure, led to a visual geometry group network (VGGNet) [3], which has a deeper network depth. However, the problem of gradient vanishing or exploding as the layers become deeper led to the emergence of the residual network (ResNet) [4], which solved this problem using skip-connection. Then came densely connected convolutional networks (DenseNet), designed to be more parameter efficient than ResNet and reduce over-fitting. Notably, DenseNet has achieved state-of-the-art results in image classification with the lowest number of parameters compared to other networks [5].

Despite the successes of CNN-based models, it face limitations on low-end devices, such as mobile devices, due to their substantial computational and memory requirements [6]. To tackle these challenges, researchers have proposed methods to make CNNs lighter, including pruning [7], [8], quantization [9], [10], and knowledge distillation [11], [12]. Among these methods, pruning, which involves removing synapses connected to neurons, corresponds to a combinatorial optimization problem [13].

Previous work has employed evolutionary algorithms to address pruning methods for combinatorial optimization problems. EvoPruneDeepTL [14] utilizes a genetic algorithm to prune the fully-connected layers of a transfer learning model, aiming to enhance performance by reducing the number of neurons. In DeepPruningES [15], the Multi-Objective Evolution Strategy (MOES) algorithm is applied for filter pruning in deep convolutional neural networks (DCNNs) to decrease model complexity. Cho et al. [16] employed a memetic genetic algorithm for filter pruning to reduce forward convolution computation while minimizing the decrease in accuracy.

This paper focuses on optimizing the dense connectivity pattern of DenseNet-BC (*k*=12), which has the fewest parameters among the DenseNet variants, using a genetic algorithm (GA). The experiment aims to reduce the parameter number of DenseNet-BC (*k*=12) with various pruning rates while minimizing the degradation in network performance. We employed a heuristic method, the GA, to address the combinatorial optimization problem of pruning. To evaluate the proposed method, we compare the accuracy and number of parameters of DenseNet-BC (*k*=12) and pruned DenseNet-BC (*k*=12) on the CIFAR-10 dataset.

II. Background

A. Dataset

CIFAR-10 is a small resolution (32×32) RGB image and contains 60,000 pictures, of which 50,000 are the training set and 10,000 are the test set. It also has ten classes, with 6 thousand images in each class [17]. We used random horizontal flips and 32x32 random crop for data augmentation.

B. Genetic Algorithm

Genetic algorithms are optimization algorithms that mimic natural evolution. The theory of natural evolution is that new species can adapt to their environment better than their ancestors through evolution. GA is a meta-heuristic algorithm that solves problems through evolution. GA creates solutions and uses fitness measures to evolve them. The fitness function gives each solution a fitness value. Genetic operations such as crossover and mutation are executed on the selected solution based on fitness value [18]. This process is repeated until a termination condition is met to improve the solution [19].

C. DenseNet

DenseNet uses dense connectivity to enhance the information flow between layers. Dense connectivity means receiving

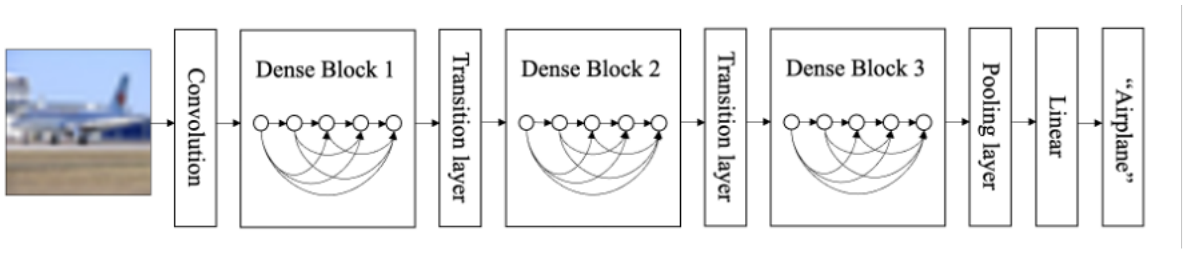

Fig. 1. DenseNet architectures

and concatenating all feature maps produced in each layer. Each layer generates k feature maps guided by the growth rate to regulate the amount of information it contributes to subsequent layers. Due to the substantial amount of information acquired at each layer, bottleneck layers are employed to reduce computational complexity. The structure of bottleneck layers involves placing batch normalization (BN), rectified linear unit (ReLU), and a 1×1 convolution layer (Conv) before BN, ReLU, and a 3×3 Conv. When utilizing such bottleneck layers, the model is referred to as DenseNet-B. For down-sampling the feature maps, the transition layer consists of BN, 1×1 Conv, and 2×2 average pooling layers. To enhance model compression, the feature maps of the transition layer are reduced by a factor denoted as θ, which was set to 0.5 in the experiments. When employing this approach, the model is referred to as DenseNet-C. Therefore, in this paper, the baseline model, referred to as DenseNet-BC (k=12), is a model that contains bottleneck layers, compression, and has a growth rate of 12.

III. Methodology

Figure 1 depicts the structure of DenseNet [5]. For the baseline model in our experiments, DenseNet-BC (k=12) consists of 16 Bottleneck blocks in each Dense block, with all Bottleneck blocks connected through Dense Connections.

To optimize the dense connectivity pattern of DenseNet, we applied a GA, an optimization technique inspired by Darwin's evolutionary theory. Using a multi-dimensional encoding of (N+2)×(N+2), we represented connections among N bottleneck blocks, inputs, and outputs within a dense block.

0	1	*p*	*p*	*p*	0	1	*p*	*p*	*p*	0	1	*p*	*p*	*p*
0	0	1	*p*	*p*	0	0	1	*p*	*p*	0	0	1	*p*	*p*
0	0	0	1	*p*	0	0	0	1	*p*	0	0	0	1	*p*
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 2. Example of multi-dimensional encoding

Figure 2 shows an example of multi-dimensional encoding in DenseNet with three bottleneck blocks and three dense blocks. We set a constraint to ensure a directed connection between bottleneck blocks always exists, maintaining a minimum information flow. The remaining matrix elements are filled with binary values based on pruning rate (p=50, 60, 70, 80, 90), where 1 indicates an enabled dense connection, and 0 indicates a disabled connection. In other words, DenseNet-BC (k=12) is represented by a 3×18×18 matrix, corresponding to its three dense blocks and 16 bottleneck blocks.

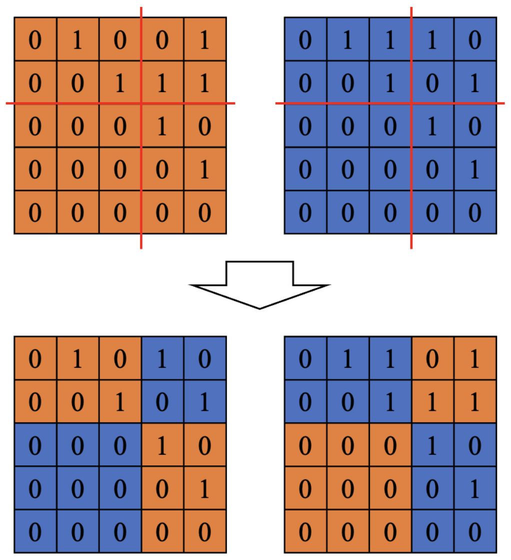

Fig. 3. Example of crossover in multi-dimensional encoding

The fitness value for evaluating each individual is set to be the accuracy on the test set after training the decoded model for one epoch. It then creates a pair of offspring using the two randomly selected parents. During crossover, it selects random points in each row and column of a two-dimensional matrix. Figure 3 shows an example of crossover in multi-dimensional encoding. It generates the first offspring by utilizing the first and third quadrants of the first parent and the second and fourth quadrants of the second parent. Simultaneously, it generates the second offspring using the second and fourth quadrants of the first parent and the first and third quadrants of the second parent.

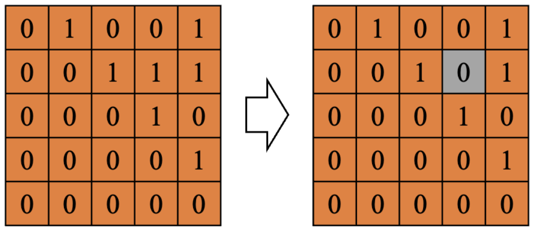

Fig. 4. Example of mutation in multi-dimensional encoding

Figure 4 depicts an example of a mutation in multi-dimensional encoding. The bit-flip mutation by selecting random points in each row and column of the two-dimensional matrix was used.

TABLE I
PARAMETERS AND VALUES USED IN THE IMPLEMENTATION OF GENETIC ALGORITHM

Parameter	Value
Pruning rate	50, 60, 70, 80, 90
Population size	20
Number of generations	10
Crossover probability	100%
Mutation probability	10%
elitism rate	20%

Table I shows the parameters and values used to implement the GA. The parameters such as pruning rate, population size, number of generations, crossover probability, mutation probability, and elitism rate must be determined to implement GA. The population size and the number of generations are set to 20 and 10, respectively. Crossover is applied with a 100% probability, and the mutation rate is set to 10%. To move the best individuals to the next generation, we copied the top 20% of the parent and transferred the best 80% of the offspring to the next generation. The proposed method in this paper was implemented using PyTorch [20] and experimented on an Intel Core i7-7700k CPU (4.20GHz), NVIDIA GeForce GTX 1090 GPU, and 64GB of memory.

IV. RESULTS

TABLE II
PERFORMANCE COMPARISON OF DENSENET-BC (k=12) AND PRUNED MODELS WITH VARIOUS PRUNING RATE

Model	#Params	Accuracy
DenseNet-BC (k=12)	0.80M	95.49
w/pruning (p=50%) DenseNet-BC (k=12)	0.51M	94.56
w/pruning (p=60%) DenseNet-BC (k=12)	0.55M	94.96
w/pruning (p=70%) DenseNet-BC (k=12)	0.61M	94.92
w/pruning (p=80%) DenseNet-BC (k=12)	0.68M	95.03
w/pruning (p=90%) DenseNet-BC (k=12)	0.78M	95.15

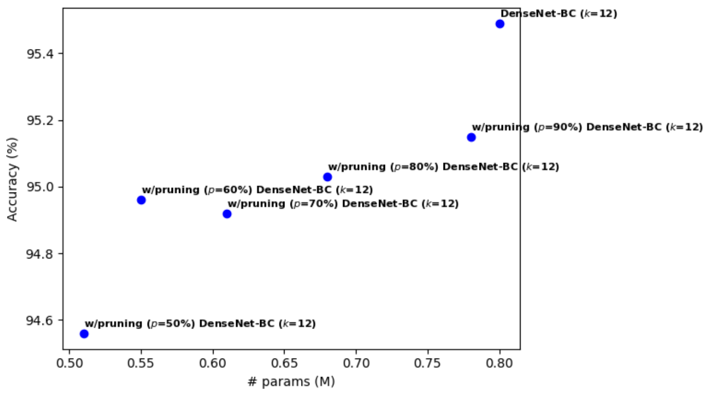

Fig. 5. Comparative analysis between DenseNet-BC (k=12) and pruned DenseNet-BC (k=12) with various pruning rate

Table II and Figure 5 shows the accuracy and number of parameters of the baseline model, DenseNet-BC (k=12), and the pruned model with various pruning rates. When pruning the baseline model through the proposed method (p=50), the number of parameters was reduced by up to 36.25%, while the accuracy decreased by only 0.93%. The proposed method (p=60) reduced the number of parameters by 31.25%, with only a 0.53% decrease in accuracy compared to the baseline model. The proposed method (p=70) reduces the number of parameters by 23.75% and only decreases the accuracy by 0.57%. The proposed method (p=80) reduces the number of parameters by 15% compared to the baseline model, with only a 0.43% decrease in accuracy. Finally, the results of the proposed method (p=90) showed a reduction in the number of parameters by up to 2.5%, with the accuracy decreasing by 0.34% compared to the baseline model. The results demonstrate that the proposed method can find efficient models with a minimal loss of accuracy ($<$1%) compared to the baseline model while reducing the number of parameters.

V. CONCLUSIONS & FUTURE WORK

This paper proposes a GA-based pruning method, with DenseNet-BC (k=12) as the baseline model. We compare the accuracy and number of parameters with the baseline model, demonstrating that our proposed pruning method reduces the number of parameters and shows performance similar to that of the baseline model. However, there is a limitation to this research. It requires a high computational cost to find an efficient model. To solve this problem, future works which consider a surrogate model to approximate the fitness value are required. It is also necessary to optimize different well-known models to validate that the methodology is effective for pruning and analyze whether it achieves higher performance than other heuristic algorithms.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government

(MSIT) (No. 2021R1F1A1048466, No. 2022R1F1A1066017). This work was also supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea (NRF-2022S1A5C2A07090938).

References

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

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

[4] 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*, pp. 770–778, 2016.

[5] 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*, pp. 4700–4708, 2017.

[6] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, "Towards optimal structured cnn pruning via generative adversarial learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2790–2799, 2019.

[7] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.

[8] N. Lee, T. Ajanthan, and P. H. Torr, "Snip: Single-shot network pruning based on connection sensitivity," *arXiv preprint arXiv:1810.02340*, 2018.

[9] J. Yang, X. Shen, J. Xing, X. Tian, H. Li, B. Deng, J. Huang, and X.-s. Hua, "Quantization networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7308–7316, 2019.

[10] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.

[11] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1365–1374, 2019.

[12] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4133–4141, 2017.

[13] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient transfer learning," *arXiv preprint arXiv:1611.06440*, vol. 3, 2016.

[14] J. Poyatos, D. Molina, A. D. Martinez, J. Del Ser, and F. Herrera, "Evoprunedeeptl: An evolutionary pruning model for transfer learning based deep neural networks," *Neural Networks*, vol. 158, pp. 59–82, 2023.

[15] F. E. Fernandes Jr and G. G. Yen, "Pruning deep convolutional neural networks architectures with evolution strategy," *Information Sciences*, vol. 552, pp. 29–47, 2021.

[16] H. H. Cho, H. J. Byun, M. K. Kim, J. Huh, and B.-R. Moon, "Evolutionary pruning of deep convolutional networks by a memetic ga with sped-up local optimization and glcm energy z-score," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pp. 715–718, 2023.

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

[18] G. Renner and A. Ekárt, "Genetic algorithms in computer aided design," *Computer-aided design*, vol. 35, no. 8, pp. 709–726, 2003.

[19] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, pp. 65–85, 1994.

[20] B. Amos and J. Z. Kolter, "A PyTorch Implementation of DenseNet." https://github.com/bamos/densenet.pytorch. Accessed: [November 21, 2023].