DOI 10.31891/2307-5732-2020-281-1-7-11
UDC 004.93'12

P.M. RADIUK
Khmelnytskyi National University

APPLICATION OF A GENETIC ALGORITHM TO SEARCH FOR THE OPTIMAL CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE WITH WEIGHT DISTRIBUTION

In the past decade, a new way in neural networks research called Network architectures search has demonstrated noticeable results in the design of architectures for image segmentation and classification. Despite the considerable success of the architecture search in image segmentation and classification, it is still an unresolved and urgent problem. Moreover, the neural architecture search is also a highly computationally expensive task. This work proposes a new approach based on a genetic algorithm to search for the optimal convolutional neural network architecture. We integrated a genetic algorithm with standard stochastic gradient descent that implements weight distribution across all architecture solutions. This approach utilises a genetic algorithm to design a sub-graph of a convolution cell, which maximises the accuracy on the validation set. We show the performance of our approach on the CIFAR-10 and CIFAR-100 datasets with a final accuracy of 93.21% and 78.89%, respectively. The main scientific contribution of our work is the combination of genetic algorithm with weight distribution in the architecture search tasks that achieve similar to state-of-the-art results on a single GPU.

Keywords: convolutional neural networks, genetic algorithms, weight distribution, ablation study.

П.М. РАДЮК
Хмельницький національний університет

ЗАСТОСУВАННЯ ГЕНЕТИЧНОГО АЛГОРИТМУ ДЛЯ ПОШУКУ ОПТИМАЛЬНОЇ АРХІТЕКТУРИ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ З РОЗПОДІЛЕННЯМ ВАГ

За останнє десятиліття новий спосіб дослідження нейронних мереж під назвою «Пошук мережевих архітектур» продемонстрував позитивні результати в розробці архітектур для сегментації та класифікації зображень. Незважаючи на значний успіх пошуку архітектур в задачах сегментації та класифікації зображень, він все ще є невирішеною і актуальною проблемою. Більше того, пошук архітектур нейронних мереж є також дуже витратим з точки зору обчислювальних ресурсів. У цій роботі пропонується новий підхід на основі генетичного алгоритму для пошуку оптимальної архітектури згорткової нейронної мережі. Ми інтегрували генетичний алгоритм зі стандартним стохастичним градієнтом, що реалізує розподіл ваг у всіх архітектурних рішеннях. Цей підхід використовує генетичний алгоритм для проектування частини графу в якості згорткового шару, що забезпечує максимальну точність на валідаційному наборі даних. У цій роботі ми демонструємо ефективність нашого підходу на наборах даних CIFAR-10 та CIFAR-100 з кінцевою точністю 93,21 % та 78,89 % відповідно. Основним науковим внеском нашої роботи є поєднання генетичного алгоритму з розподілом ваг в задачах пошуку архітектури, що досягає точності класифікації зображення з використанням одного графічного процесора близької до найсучасніших результатів.

Ключові слова: згорткові нейронні мережі, генетичні алгоритми, розподілення ваг, абляція дослідження.

Introduction

In recent decades, artificial neural networks have produced outstanding results in computer vision with a wide variety of applied tasks such as object detection, image segmentation and classification and others. The design of neural network architectures for a specific task or dataset usually requires specific approaches and a large number of computational resources [1, 2]. Recently, a new way in neural networks research called Network Architectures Search (NAS) has demonstrated noticeable results in the design of architectures for image segmentation and classification. NAS approaches use a recurrent neural network (RNN) controller to generate a candidate network architecture, called child model, which is then trained to converge. After the training, researches measure the performance of the trained network architecture on the desired task or dataset. RNN controller receives the performance measurement as a signal to explore for a better architecture. After that, this process repeats over many computationally expensive iterations.

Analysis of recent research

Despite the considerable success of NAS in classification tasks, it is still highly computationally expensive. Recently, many studies have used the idea of parameter sharing [3] across all child models to reduce the need for training each child model from scratch, thereby eliminating most computational costs. Weight distribution, the analogue of parameter sharing for convolutional cells, has shown prominent result utilising reinforcement-based and gradient-based methods. The first method is an effective search for neural architecture using parameter sharing (ENAS) [4] based on reinforcement training with the RNN controller to create the candidate architecture. The second method is called a differentiable architecture search with various modifications (DARTS) [5], where each compound has a gradient-updating probability function. However, none of the publications has yet combined GAs with parameter sharing or its analogies to NAS.

Genetic algorithm (GA) is a search method based on natural selection and genetics [6]. GAs consist of four fundamental concepts: selection, cross-over, mutation, and replacement. Population, another essential component of GA, is utilised to generate new candidate solutions. A new generation is created in each iteration, using a three-step process of selection, cross-over and mutation. The next generation is then inserted into the population through the replacement phase. The algorithm starts with a random set that is evaluated at the beginning of training.

Evolution-based method for NAS has been used in lots of works, for instance [7] and [8], but without weight distribution. This study presents a genetic approach to optimise the convolutional neural network architecture on the validation set. Our work utilises the search space, defined by [4], and presents a convolution cell as a directed acyclic graph (DAG) with a fixed-length list of integers that are used to describe the DAG structure [9].

Benchmark datasets

The data used in the experiment is CIFAR-10, and CIFAR-100 datasets [10]. The CIFAR-10 dataset consists of 60000 32×32 colour images in 10 classes, with 6000 images per class. The CIFAR-100 dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each with 500 training images and 100 testing images per class. We pre-processed the data by normalising all images. Fig. 1 introduces instances of datasets.

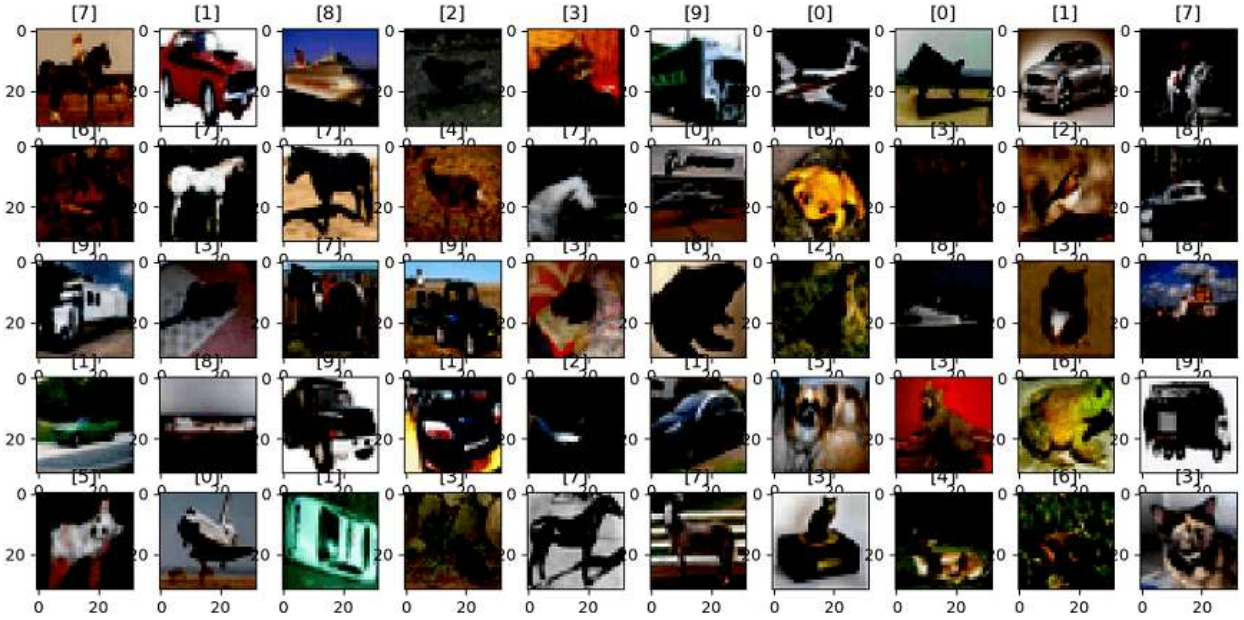


Fig. 1. Instances of the CIFAR-10 and CIFAR-100 datasets

A genetic approach to an architecture search

The main idea of our application is to combine evolution search with weight distribution. Having used ENAS search space, we present a convolutional cell as DAG, where each node represents an operation, and the edges – tensors. Additionally, we utilised the ENAS's primary attribute – parameter sharing, however, applied it only to the weights across all the child networks. The convolution search space exploits the idea that the same cell repeats several times, which is a standard practice in NAS [11].

Data: current_population x_p ;

Result: sampled_individual x_s ;

$x_0, x_1 = sampling(x_p)$;

$x_0, x_1 = crossover(x_o, x_1)$;

$x_s = mutation(x_0)$.

Fig. 2. The process of sampling

```
Data: import dataset;
Result: initialise network_architecture_weights;
for each_epoch do
        for each_batch do
            sample random_child_model from population;
            forward_pass;
            backward_pass;
        end
        generate next_generation from population;
        for each_child in next_generation do
            set child_architecture;
            evaluate architecture_accuracy;
        end
        update population using replacement;
end
```

Fig. 3. Application of the GA

We integrated the GA search with an SGD algorithm, for efficient architecture search. We also utilised weight distribution in order to reduce the number of training all child networks from scratch. The weight distribution requires the model coefficient to be adaptable to network architecture change. Such an approach requires an update of all the weights. We provide a sampling of different architectures from the current population in each batch to achieve the weights update. The process of sampling consists of three steps: 1) select randomly two individuals from the current population, 2) provide cross-over followed by a mutation that applied to the first offspring and 3) return a result. Fig. 2 presents the algorithm of the sampling process of our approach.

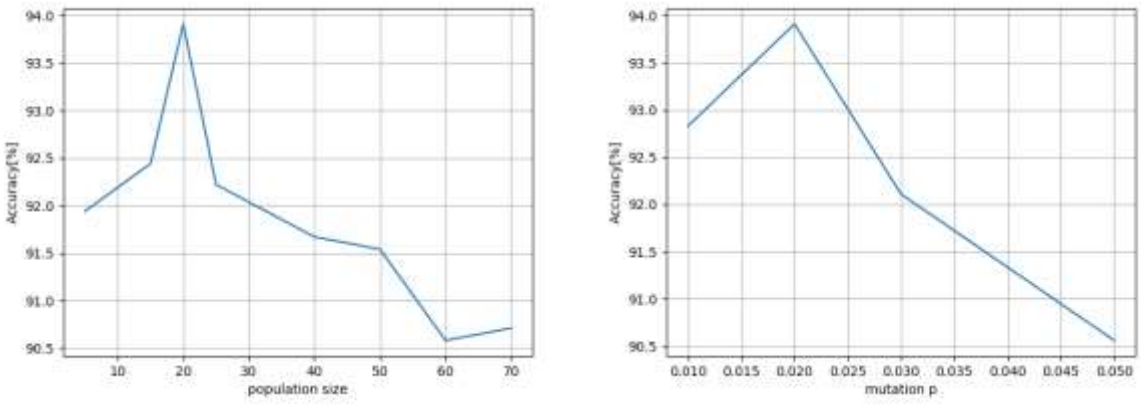
After completing the training process of one epoch, a new generation is created from the current population, using the following three steps: selection, crossing, and mutation. Once all individuals are established, the current population is then updated with the replacement method [12]. Fig. 3 shows the training process for our approach.

Training

First of all, we present an ablation study of genetic search on the CIFAR-10 dataset. Then, we show the performance of our application on the CIFAR-10 and CIFAR-100 datasets and describe the process of an architecture search. All calculations are performed with a single NVIDIA GTX 1080 graphics processing unit using machine learning framework TensorFlow v. 1.13.0 [13]. We applied a weight decay of 0.0001 with dropout probability of 0.3 [14]. The batch size is set to 256 images per gradient update [15].

The implementation of ablation study

The ablation study [16] was performed on CIFAR-10 dataset with a small model with convolutional cells of 2. In the first cell, the number of channels is 20. The momentum of the SGD optimiser is 0.9. To design a convolutional cell, we set 5 blocks. We also randomly selected 1000 images from the validation set to evaluate each individual from the total generation, which size is 20. On the whole, the training process took 320 epochs. Fig. 4 demonstrates the results of ablation study.



a) **b)**

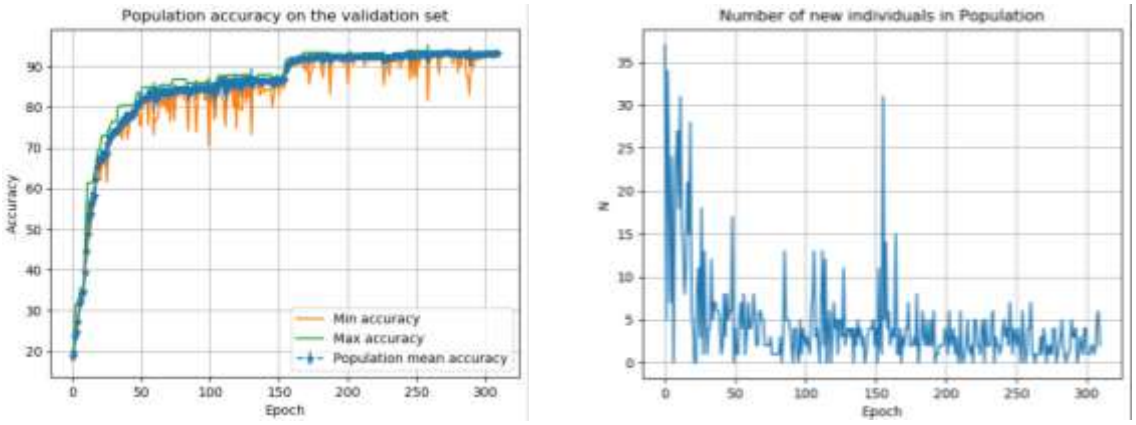
Fig. 4. Final search result over different parameters of the genetic algorithm: a) population size; b) mutation

Fig. 4,a shows the accuracy distribution depending on the population size. Higher population value leads to lower performance, which is caused by high population variability. However, low population size also leads to a decrease in accuracy, which reduces the ability of GA to explore new solutions. Fig. 4,b shows the result of a search depending on various mutation probabilities. From fig. 4, b, a high probability of mutation leads to poor performance. In terms of architecture search, the high probability of mutation increases the variability of the convolutional architecture, which also reduces the ability of GA to explore better solutions. Both population size and mutation probabilities represent a compromise between exploration and exploitation. Not only this trade-off affects the genetic search but also that SGD optimisation since we sample a different architecture for each gradient update from the current population.

Results of the experiments

Foremost, we provided experiments on the CIFAR-10 dataset. The hyper-parameters selected are based on the ablation study, where the mutation probability is 0.02, and the population size is 20. The values of other hyperparameters are equal to the results of the ablation study.

Fig. 5,a shows the distribution of min accuracy, max accuracy and mean accuracy of the population depending on the numbers of epochs. Error bar points the standard deviation of the population accuracy on the subset of the validation set. Fig. 5,b presents the number of new individuals inserted into the population at the end of each epoch. Having found the best architecture, we train the model from scratch for 600 epochs in total. The model in the final stage has four convolutional cells. The number of channels is increased to 48. For regularisation, we apply drop-path with a drop probability of 0.9. Table 1 presents the result of the final training.



a) **b)**

Fig. 5. The architecture search on the CIFAR-10 dataset: a) population mean accuracy; b) number of updates in population

Table 1

Compare CIFAR-10 Validation Accuracy

Model	Accuracy
BayesNAS [2]	95.63%
Single-Path NAS [11]	96.37%
ENAS [4]	96.46%
DART+ [5]	97.06%
Our model	93.21%

Table 1 compares the result on CIFAR-10 on a set of different human-designed networks and a set of search algorithms. We achieved an accuracy of 93.21% on the validation-set of CIFAR-10 dataset.

We performed the same architecture search on the CIFAR-100 dataset with the same hyperparameters as in the previous experiment. In order to fir into GPU memory, the number of channels in the search was to 48, and the batch size was reduced to 128. Fig. 6 describes the search results.

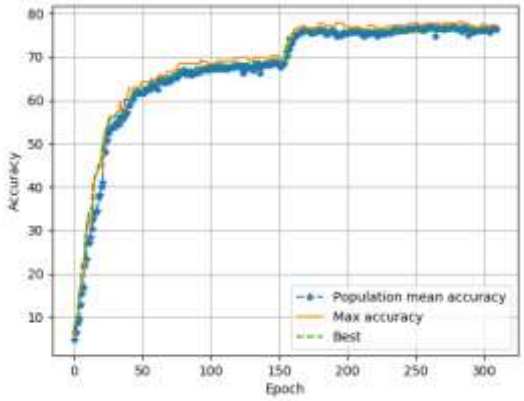


Fig. 6. Compare a population metric on a subset of a validation dataset to the best model's accuracy across the entire validation CIFAR-100 dataset

Fig. 6 shows a slight difference between the maximum accuracy of the population measured on a subset of a validation set and the best accuracy measured on a whole validation set. This slight difference indicates the high accuracy of prediction on a subset of the validation dataset. After finding the optimal architecture, we conducted a final training session with the same hyperparameters used on the CIFAR-10 dataset. The result of the final training on the CIFAR-100 dataset is 78.89%.

Conclusion

This study proposes a new approach based on genetic algorithms to search for optimal convolutional neural network architectures. In order to reduce search time, we implemented weight distribution to the approach. In this work, we also utilised ablation study to compromise exploration and exploitation that are inherent in genetic algorithms. Our work shows that ablation study demonstrates a significant effect on the validation accuracy, which can be considered as regularisation. We conducted experiments on the CIFAR-10 and CIFAR-100 datasets and achieved a final accuracy of 93.21% and 78.89% on the CIFAR-10 and CIFAR100 validation sets, respectively. The main contribution of our work is the combination of genetic algorithm with weight distribution in the architecture search tasks that achieve similar to state-of-the-art results on a single GPU.

References

1. Romanuke V.V. An efficient technique for size reduction of convolutional neural networks after transfer learning for scene recognition tasks / V.V. Romanuke // Applied Computer Systems. – 2018. – Volume 23. – Issue 2. – P. 141–149. – DOI: https://doi.org/10.2478/acss-2018-0018

2. Zhou H. BayesNAS: A bayesian approach for neural architecture search / H. Zhou, M. Yang, J. Wang // 2019 International Conference on Machine Learning, arXiv:1905.04919 [cs.LG]. – 2019. – P. 7603–7613.

3. Savarese P. Learning implicitly recurrent CNNs through parameter sharing / P. Savarese, M. Maire // 2019 International Conference on Learning Representations, arXiv:1902.09701 [cs.LG]. – 2019.

4. Pham H. Efficient neural architecture search via parameter sharing / H. Pham, M.Y. Guan, B. Zoph // 35th International Conference on Machine Learning, arXiv:1802.03268 [cs.LG]. – ICML, 2018. – Volume 80. – P. 6522–6531.

5. Liang H. DARTS+: Improved differentiable architecture search with early stopping / H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, Z. Li // arXiv:1909.06035 [cs.CV]. – 2019.

6. Deb K. Introduction to genetic algorithms / K. Deb // Sadhana. – Academy Proceedings in Engineering Sciences, 1999. – Volume 24. – No. 4. – P. 293–315. – DOI: https://doi.org/10.1007/BF02823145

7. Chen Y. Reinforced evolutionary neural architecture search / Y. Chen, Q. Zhang, Ch. Huang, L. Mu, G. Meng, X. Wang // arXiv:1808.00193 [cs.NE]. – 2018.

8. Litzinger S. Compute-efficient neural network architecture optimisation by a genetic algorithm / S. Litzinger, A. Klos, W. Schiffmann // Artificial Neural Networks and Machine Learning. – ICANN, 2019: Deep Learning. ICANN 2019. Lecture Notes in Computer Science, Springer, Cham, 2019. – Volume 11728. – DOI: https://doi.org/10.1007/978-3-030-30484-3_32

9. Costa M.G.F. Using convolutional neural networks with direct acyclic graph architecture in segmentation of breast lesions in US images / M.G.F. Costa, J.P.C. Mendes, W.C. A Pereira, C.F.F.C. Filho // VIII Latin American Conference on Biomedical Engineering and XLII National Conference on Biomedical Engineering. CLAIB 2019. IFMBE Proceedings, Springer, Cham, 2019. – Volume 75. – DOI: https://doi.org/10.1007/978-3-030-30648-9_99

10. Krizhevsky A. Learning multiple layers of features from tiny images / A. Krizhevsky – 2009. – URL: https://www.cs.toronto.edu/~kriz/CIFAR-.html

11. Stamoulis D. Single-Path NAS: Designing hardware-efficient ConvNets in less than 4 hours / D. Stamoulis, R. Ding, D. Wang, D. Lymberopoulos, B. Priyantha, J. Liu, D. Marculescu // arXiv:1904.02877 [cs.LG]. – 2019.

12. Lozano M. Replacement strategies to maintain useful diversity in steady-state genetic algorithms / M. Lozano, F. Herrera, J.R. Cano // Advances in Soft Computing. – Springer, Berlin, Heidelberg, 2005. – Volume 32. – DOI: https://doi.org/10.1007/3-540-32400-3_7

13. Abadi M. TensorFlow: Large-scale machine learning on heterogeneous distributed systems / M. Abadi, A. Agarwal, P. Barham. // OSDI'16 Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation. – 2016. – P. 265–283. – ISBN: 978-1-931971-33-1.

14. Romanuke V.V. Appropriateness of DropOut layers and allocation of their 0.5 rates across convolutional neural networks for CIFAR-10, EEACL26, and NORB datasets / V.V. Romanuke // Applied Computer Systems. – 2017. – Volume 22. – Issue 1. – P. 54–63. – DOI: https://doi.org/10.1515/acss-2017-0018

15. Radiuk P.M. Impact of training set batch size on the performance of convolutional neural networks for diverse datasets / P.M. Radiuk // Information Technology and Management Science. – 2017. – Volume 20. – Issue 1. – P. 20–24. – DOI: https://doi.org/10.1515/itms-2017-0003

16. Meyes R. Ablation Studies in artificial neural networks / R. Meyes, L. Melanie, C.W. de Puiseau, T. Meisen // arXiv:1901.08644 [cs.NE]. – 2019.

Рецензія/Peer review : 17.1.2020 р. Надрукована/Printed : 14.2.2020 р.
Стаття рецензована редакційною колегією