Bloodline-preserving Evolutionary Neural Architecture Search for Image Classification

Project Proposal

Bowen Zheng Shijie Chen Shuxin Wang

Advisor: Hisao Ishibuchi

Southern University of Science and Technology

1 Background & Rationale

The great leap of computing resources in the past few decades made it possible to fully utilize the potential of neural networks. In recent years neural networks outperformed traditional methods in many fields of research, especially image classification. However, the state-of-the-art architectures are carefully designed and tuned by researchers for a specific problem. Therefore, people start to think about automate the design of neural networks in the hope of finding the best-performing network architecture efficiently.

Neural architecture search is a research field focusing on automating the design of neural networks. Currently there are a few popular approaches, including reinforcement learning, bayesian optimization, tree-based searching and genetic-based evolutionary algorithms.

This project focuses on NAS for image classification problems. The reason is that this area is well explored and there exists many high-performance hand-crafted neural architectures. They provide a good guidance and target to our project. In addition, neural networks for image classification are mostly built upon basic cells including convolution, polling, normalization, and activation layers. This helps to shrink our search space.

2 Problem Definition

Neural Architecture Search (NAS) refers to the process of automatically designing artificial neural network architectures [1]. Research topics in NAS are generally categorized into three categories:

1. ***Search Space*** The search space of NAS denotes the space in which the NAS algorithm tries to find a neural network architecture.
2. ***Search Strategy*** The search strategy of NAS denotes the search algorithm that is used to explore the *search space*.
3. ***Performance Estimation Strategy*** NAS algorithms will generate a large quantity of neural network architectures during search process. We need an efficient way to estimate the performance of each architecture to cut the demand for computational resource.

3 Objective

The objective of this project is to propose a novel evolutionary approach for neural architecture search targeted at image classification. The specific objectives are as follows:

1. Investigate existing evolutionary neural architecture search algorithms.
2. Find an efficient estimate approximation of neural networks to cut computational cost.
3. Propose an effective population selection strategy.
4. Compare performance against hand-crafted networks as well as other NAS algorithms.

4 Related Works

A lot of research works have been done in each of the three categories in NAS. Some proposed algorithms can design architectures that is on par of or even more capable than state-of-the-art hand-crafted networks.

4.1 Search Space

The search space of NAS determines the possible architectures a NAS algorithm can find.

The simplest search space is the simple multiple-layer structure, in which a neural network A is composed of multiple layers L_i connected to the neighboring layers. In this case, the search space can be described by (1) The maximum number of layers (2) The type and dimension of each layer and their hyper-parameters [2] [3].

In more recent studies, some researchers use a cell based search space in which the possible architecture of cells are explored. A cell is nothing but a smaller neural network. The entire neural network is constructed by connecting several pre-defined cells. A cell has less layers but allows more complex architectures like skip connections between any layers [4] [5]. The cell could be some hand-crafted neural networks that have already been proofed effective. The search space is therefore decreased to the possible arrangements of cells.

In contrast to the above direction, some researchers also tried to search for effective cells and connect them at last in a predefined manner [6] [4]. The search space is greatly decreased in that each cell is comparably small. This method can also be easily transferred to other datasets [6] since the structure of cells are not fixed.

Recently, some researchers managed to optimize the overall architecture as well as the cells at the same time and obtained state-of-the-art result [7].

4.2 Search Strategy

Many different search strategies can be applied to explore the search space discussed above. These methods includes bayesian optimization, evolutionary methods, reinforcement learning and gradient-based methods.

Evolutionary algorithms has been used to evolve neural networks since 1989 [8]. Earlier works use genetic algorithms to both optimize the structure of neural networks and train the networks [9]. However, with the birth of back-propagation (BP), recent works of neural-evolution use genetic algorithms only for optimizing neural architectures and use BP to train the networks [10]. In the context of NAS, the individuals in the genetic algorithm are neural network architectures and the genetic operations (crossover and mutation) are used to alter the architecture by adding/removing layers or change connectivity of nodes.

Genetic algorithms shows their diversity in how they sample parents, generate offspring and update population. Some work choose parents from a preto-optimal front [11] while others use tournament selection [12] [5] [10]. When generating offsprings, some algorithms randomly initialize weight of child networks. In comparison, Lamarckian inheritance is used in [11] so that child networks could inherit weight of its parent and the training cost is reduced. To update population, some algorithms abandon least capable individuals [10] while some delete the oldest individuals [5]. A more sophisticated policy is developed by [13] and [14] in which the age of the individuals are taken into account.

There are other methods that are used to implement NAS, including bayesian optimization, reinforcement learning ,tree-based search and gradient-based methods. We don't discuss them here since we use evolutionary algorithms in our project.

4.3 Performance Estimation Strategy

One important issue in neural architecture search is the estimation of neural network performance. This is critical in the population update policy of evolutionary algorithms.

The simplest way to estimate performance is to train every searched network from scratch and test the desired performance metric, e.g. accuracy on validation set. However, the training of neural network is very time and computation consuming.

An alternative is to estimate performance on lower fidelities. More specifically, to train the network for shorter period of time [6] or on some subset of the dataset [15]. However, the estimate must ensure the result ranking of different networks must be the same as that of complete training. That is to say, there exist a trade-off between computational load and estimation fidelity.

Another approach to estimate performance is based on learning curve extrapolation [16]. This method accelerate estimation by stop poor performance networks at the early state of training based on statistical patterns of learning curves. Other researchers propose ways to predict neural network performance based on architectural and cell properties [12].

5 Methodology

We will develop an evolutionary neural architecture search algorithm based on the *age* of individuals. By incorporating *age* as a part of gene, we can prolong the existence of good individuals [14] and kill average individuals when their age reach a certain limit [13].

Combining the advantage of the above works, we propose a population update police based on *age* and *bloodline*. An individual is dropped from the total population if its *age* exceeds a predefined $lifetime$. However, we prolong its life if its offspring performs well (e.g. within top $P\%$ in ranking). In this way, we can preserve a good *bloodline* in the population.

To test the effectiveness of our algorithm, we will experiment on image classification datasets including CIFAR-10, CIFAR-100 and will possibly extend to IMAGENET.

6 System Design

For now, we follow the design of a cell based evolutionary NAS algorithm described in [14] except that we propose a population update policy to preserve good *bloodlines*. We will further examine initilization, crossover and mutation strategies to optimize our algorithm. We will also compare different performance estimation approaches to cut computation demand.

Algorithm 1 Aging Bloodlines Evolution:

```
1: population ← $\phi$
2: entireGen ← $\phi$
3: generation ← 0
4: while population size< $N$ do
5:     $newNetwork \leftarrow networkInit()$
6:     $trainNetwork(newNetwork)$
7:     add $newNetwork$ to poption and entireGen
8: end while
9: while genetation < $G$ do
10:    parent ← select a parent from population using tournament selection
11:    child ← mutate(parent)
12:    trainNetwork(child)
13:    add child to poption and entireGen
14:    if accuracy of child is better than $P\%$ individuals in population then
15:        extend the $lifetime$ of parent by $t$
16:    end if
17:    for all individual in population do
18:        update the age of individual
19:        remove current individual if its age reaches its $lifetime$
20:    end for
21: end while
22: return the network model with highest accuracy in entireGen
```

The *population* size is N and the algorithm evolve G generations in total. *entireGen* stores all network models that we generated.

In *networkInit*(), the initial network model is generated. Then its *age* is set to 1 and $lifetime$ is set to the default value.

P and t are hyper-parameters controlling the actual lifespan of an individual.

References

[1] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *arXiv preprint arXiv:1808.05377*, 2018.

[2] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

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

[4] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu, "Path-level network transformation for efficient architecture search," *arXiv preprint arXiv:1806.02639*, 2018.

[5] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *arXiv preprint arXiv:1802.01548*, 2018.

[6] 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*, pp. 8697–8710, 2018.

[7] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," *arXiv preprint arXiv:1901.02985*, 2019.

[8] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms.," in *ICGA*, vol. 89, pp. 379–384, 1989.

[9] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[10] 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*, pp. 2902–2911, JMLR. org, 2017.

[11] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," 2018.

[12] 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)*, pp. 19–34, 2018.

[13] G. S. Hornby, "Alps: The age-layered population structure for reducing the problem of premature convergence," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, (New York, NY, USA), pp. 815–822, ACM, 2006.

[14] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *CoRR*, vol. abs/1802.01548, 2018.

[15] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast bayesian optimization of machine learning hyperparameters on large datasets," *arXiv preprint arXiv:1605.07079*, 2016.

[16] T. Domhan, J. T. Springenberg, and F. Hutter, "Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.