NEMO: Neuro-Evolution with Multiobjective Optimization of Deep Neural Network for Speed and Accuracy

Ye-Hoon Kim YEHOON.KIM@SAMSUNG.COM
Bhargava Reddy TB.REDDY@SAMSUNG.COM
Sojung Yun SOJUNG15.YUN@SAMSUNG.COM
Chanwon Seo CW0323.SEO@SAMSUNG.COM
SW Center, SEC, Seongchon-gil 56, Seocho-gu, Seoul, Korea 06765

Abstract

This paper proposes automatic machine learning approach (AutoML) of deep neural networks (DNNs) using multi-objective evolutionary algorithms (MOEAs) for the accuracy and run-time speed simultaneously. Traditional methods for optimizing DNNs have used Bayesian reasoning or reinforcement learning to improve the performance. Recently, evolutionary approaches for high accuracy has been adopted using a lot of GPUs. However, real-world applications require rapid inference to deploy on embedded devices while maintaining high accuracy. To consider both accuracy and speed at the same time, we propose Neuro-Evolution with Multiobjective Optimization (NEMO) algorithm. Experimental results show that proposed NEMO finds faster and more accurate DNNs than hand-crafted DNN architectures. The proposed technique is verified for the image classification problems such as MNIST, CIFAR-10 and human status recognition.

Keywords: AutoML, Meta Learning, Deep Neural Network, Deep Learning, Multi-objective Evolutionary Algorithms, Multi-objective Problems

1. Introduction

AlexNet (Krizhevsky et al., 2012) was the first neural network architecture which won first place in 2012 ImageNet classification task. Since the development of AlexNet, deep and diverse advanced Convolutional Neural Networks (CNNs) became a trend in image classification tasks (K.Simonyan and Zisserman, 2015; Szegedy et al., 2014; He et al., 2015). Those popular architectures have been developed by professional neural network engineers with tremendous heuristic experiments using huge computational resources and time. Recently, AutoML is gaining importance because of the ability to optimize neural networks while minimizing human resources (Craenendonck and Blockeel, 2015; Mendoza et al., 2016). Earlier AutoML techniques have solved traditional machine learning problems, however they have been rarely used for deep learning. Meta learning, which is similar to AutoML, has utilized various machine learning approaches such as Bayesian learning (Sant et al., 2016) or reinforcement learning (Zoph and Le, 2016; Baker et al., 2017).

Recently, evolutionary algorithms (EAs) which efficiently find global solution of NP-complete problems have been applied to architecture design of CNNs (Miikkulainen et al., 2017; Ha et al., 2016; Real et al., 2017; Shafiee et al., 2017). Traditional approaches of neuro-evolution (Stanley and Miikkulainen, 2002) have used evolutionary algorithms instead of stochastic gradient descent (SGD) for back-propagation of neural networks. However, the state-of-the-art DNNs consist of more than millions of parameters, so EAs may fail to find optimal solution in huge search space. Therefore recent trend in neuro-evolution research area is to maintain the back-propagation mechanism using convex optimization technique and find optimal network configuration such as the number of outputs and layers, or connections between layers.

Single-objective evolutionary approaches for automatic model design is one of the most notable research topics in recent years, since evolved arbitrary networks have shown competitive results with hand-crafted architecture. However, DNN techniques for practical applications on consumer-electronics requiring light-weight models and fast run-time have been an issue recently. Thus multi-objectives such as accuracy, run-time, energy consumption and model size should be considered simultaneously while designing DNNs. These problems are called as multi-objective optimization problems (MOPs). EAs for MOPs have been successfully researched for decades (Zitzler et al., 2001; Deb et al., 2002; Zhang and Li, 2007). Numerical MOPs such as DTLZ and ZDT problems (Huband, 2011) or combinatorial ones such as multi-objective knapsack problems can be solved by MOEAs accurately (Kim et al., 2006). The multi-objective neuro-evolution techniques were adopted to solve neural network design (Fieldsend and Singh, 2005; Kant et al., 2013). However inference speed was not considered as an objective and they were not applied to recent DNNs or image recognition problems.

In this paper, Neuro-Evolution with Multiobjective Optimization (NEMO) algorithm is proposed to automatically optimize CNNs in terms of accuracy and inference speed at the same time by applying MOEAs. CNN is a type of DNN composed of multiple layers including convolutional and fully connected layers. For evolution, the number of outputs (channels) of each layer on each pre-defined networks are encoded as a genotype representation. Pre-defined networks are varied in the combination of convolution and fully connected layers. Each combination of these genotype setting is decoded as a specific CNN and then evaluated by calculating fitness values on the train and validation processes. The experimental results show that accuracy and speed of optimized CNN models by proposed NEMO are better than hand-crafted CNNs on image classification tasks.

2. Neuro-Evolution with Multiobjective Optimization (NEMO)

2.1. Multi-objective Evolutionary Algorithms for Deep Neural Networks

To solve MOPs, single-objective approaches can be used. One of the method is to fix one objective while optimizing the other. The other method is a weighted-sum approach which converts multiple objectives to a single objective. However, single-objective approaches cannot handle trade-off problem between multi-objectives. Use of weighted-sum for MOPs does not guarantee diverse non-dominated solutions. To overcome the stated limits of single objective approaches, multi-objective approaches such as MOEA are necessary. Thus, the proposed method adopts NSGA-II framework, which ranks the solutions using non-dominated sorting and crowding distance sorting, as a MOEA. Non-dominated solutions are superior to every other solutions in all the objectives. The non-dominated sorting algorithm makes groups of solutions based on dominate and non-dominate concept, and the groups are defined as fronts. To decide the surviving solution in the same front, crowding distance is measured and the solution with top-N distances are survived. This method maintains the balance of exploration and exploitation during the evolution (Deb et al., 2002). Figure 1

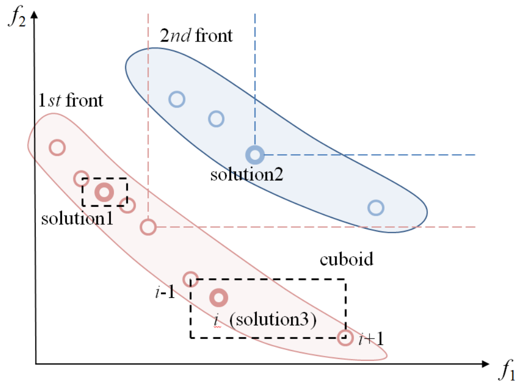


Figure 1: Ranking using non-dominated and crowding distance sorting.

shows example solutions in a certain generation. The nondominated sorting algorithm makes two fronts of red and blue solutions. The red front has higher priority than the blue front in case of minimization problems. In the same front, a solution with higher cuboid value has higher priority. The cuboid of ith solution is an normalized value of L1 distance between $i-1th$ and $i+1th$ solution. The solution with a bigger cuboid value has higher chance to explore unknown search space in evolution process. For example, in the red front solution3 has a priority than solution1.

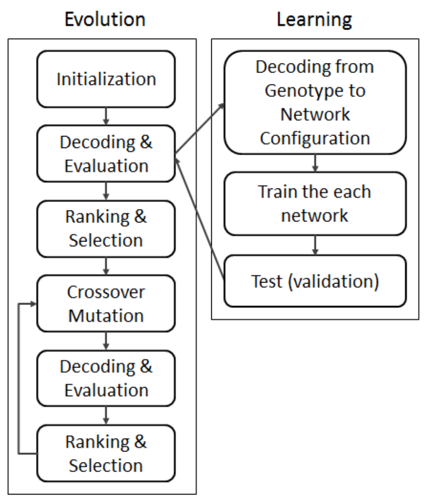
2.2. Proposed Neuro-Evolution with Multiobjective Optimization (NEMO)

Overall procedure of proposed NEMO is shown in Figure $2(a)$. The left side of the figure shows evolution, which is a general procedure of MOEAs. The right side shows learning, which includes genotype decoding for network definition and DNNs training and testing for evaluation. The learning procedure in NEMO is described in Figure $2(b)$ in detail. Decoding is the process of converting genotype to neural network structure. Genotype is represented by integers with an upper bound. These integers depict number of outputs for each layer and the number of layers. Although the layer configuration should be defined without any limitation for exploring global solution, heuristic initialization can be helpful to reduce search space. Each genotype representation is perturbed by real-coded crossover and mutation. For fitness evaluation, fitness functions for two-objective minimization problem are defined as follows:

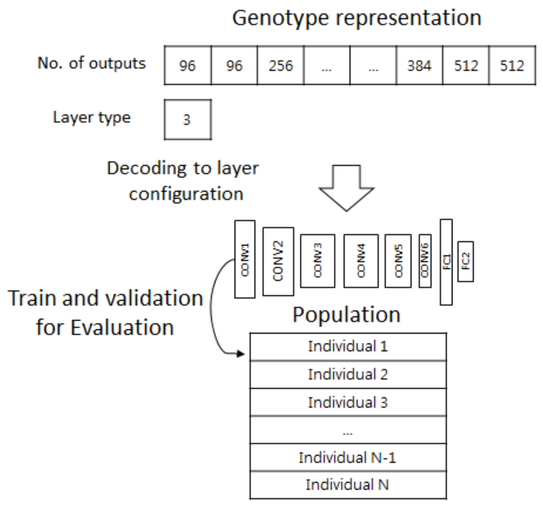
$$f_1 = \epsilon \tag{1}$$

$$f_2 = \alpha \cdot t_{inf} \tag{2}$$

where the error rate ϵ is a real value between 0 to 1 (1 - accuracy) and the unit of inference time t_{inf} is $msec$. α is scaling factor for balancing between two objectives. t_{inf} is the average inference time on forward pass.



(a) Overall procedure of NEMO.



(b) Learning procedure of genotype decoding and evaluation.

Figure 2: Overall procedure of NEMO.

3. EXPERIMENTAL RESULTS

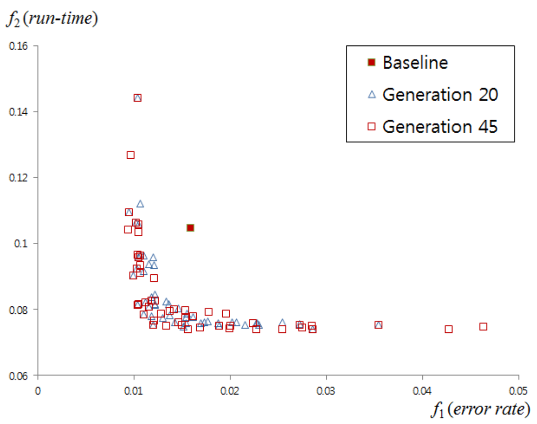
Benchmark image classification tasks such as MNIST and CIFAR-10 were used to verify the effect of NEMO on well-known problems. Moreover, to show the effectiveness of the proposed algorithm on real-world application, experimental result on drowsy behavior recognition task for driver monitoring system was shown.

3.1. Results on Benchmark Image Recognition Dataset

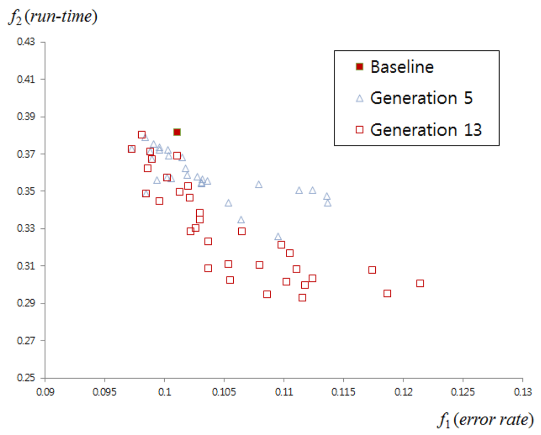
For MNIST classification task, LeNet architecture (LeCun et al., 1998) was employed as the baseline network. The scaling factor α was fixed as 300 and crossover, mutation rate were set to 0.1 in all the experiments. In total, sixty Tesla M40 GPUs were used for parallel evaluation. The detailed hyperparameters for each experiment are described in Table 1. Every individual was initialized with the baseline network configuration such as LeNet for quick convergence. On benchmark image recognition tasks, the number of outputs in each layer was perturbed with fixed layer configuration. Experimental results are described in Figure 3(a). The evolved solutions at generation 45 were better than baseline network. The solutions closer to y-axis represent more accurate and the ones closer to x-axis mean faster.

	Pop size	Max. no. of output	Max. iteration	Learning rate
MNIST	50	600	10,000	0.01
CIFAR-10	40	256	52,000	0.05
Drowsiness Recognition	60	600	9,000	0.01

Table 1: The Hyperparameters for Experimental Results



(*a*) Results on MNIST Dataset



(*b*) Results on CIFAR-10 Dataset

Figure 3: Experimental Results on Benchmark Dataset for Image Classification.

CIFAR-10 image classification problem was optimized by proposed evolution approach automatically. Every individual was initialized as baseline networks architecture proposed in (Springenberg et al., 2015). It was proven NEMO produces various networks with wide variation for MNIST and CIFAR-10 dataset in accuracy and speed perspective.

3.2. Results on Drowsiness Recognition for Driver Monitoring System

This task classifies the driver status into 3 labels, which are normal, drowsy and yawning. The input is cropped infra-red (IR) images of left eye and mouth. In this application, the number of outputs of each layer and the predefined layer configuration were perturbed for evolution in contrast to the image classification task in Section 3.1. The network types per each stream were defined as Table 2 for fast convergence. Every individual was initialized as baseline network as shown in Figure 4 which is a two-streamed CNN as described by (Reddy et al., 2017).

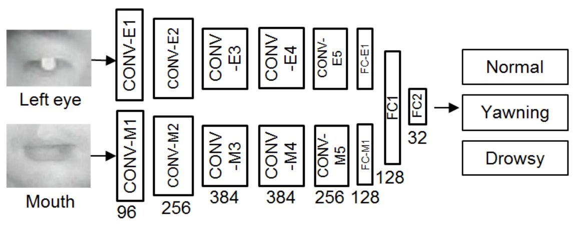


Figure 4: Baseline Architecture for Drowsy Behavior Recognition

Experimental results are described in Figure 5. The proposed NEMO found better networks than hand-crafted baseline network in both accuracy and speed. Also, more generations produced more optimized solutions. For instance, the solution pointed by arrow at generation 35 had performance of 6.6% higher accuracy and 2.1 times speed up than

Network type index	1	2	3	4	5	6	7
Total no. of layers	8	7	6	6	5	5	4
No. of conv layers	5	4	4	3	3	2	2
No. of fc layers	3	3	2	2	2	3	2

Table 2: Pre-defined Network Types for Genotype Representations.

baseline network. Moreover, diverse solutions gave the users a chance to select proper DNNs according to the needs.

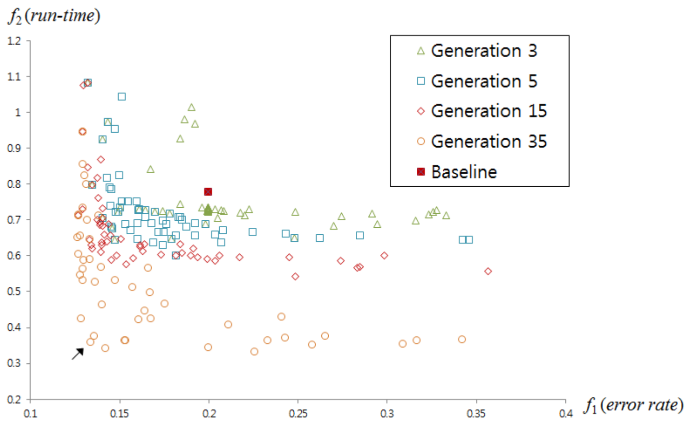


Figure 5: Results on Drowsy Behavior Recognition

4. Conclusion

This paper proposed AutoML technique, NEMO, which uses neuro-evolution with MOEAs adopted on DNNs. This scheme searched optimal DNNs automatically by EA. The combination of MOEAs and DNNs produced accurate and fast networks as shown in the experimental results. The state-of-the-art DNNs which are mostly heavy are hard to be used in real-world applications due to high latency. Fast neural network models which maintains high accuracies can be applied on small devices for practical use. The proposed AutoML approach can optimize networks using a lot of GPUs for high complex problems minimizing human effort. For future work, more objectives like power consumption or network size can be defined in optimization problem. Also, more network configurations such as connections between layers or kernel sizes can be encoded in genotype representations for diverse evolutions.

References

B. Baker, O. Gupta, N. Naik, and R. Raskar. Designing neural network architectures using reinforcement. *ICLR*, Mar 2017.

T. V. Craenendonck and H. Blockeel. Using internal validity measures to compare clustering algorithms. *ICML AutoML Workshop*, 2015.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2), 2002.

J. E. Fieldsend and S. Singh. Pareto evolutionary neural networks. *IEEE Trans. on Neural Networks*, 16(2), 2005.

D. Ha, A. Dai, and Q. V. Le. Hypernetworks. *arXiv*, 1609.09106, 2016.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv*, 1512.03385, 2015.

S. Huband. A review of multi-objective test problems and a scalable test problem toolkit. *ECU publications pre*, 2011.

A. Kant, B. Pranmohan, K. Suman, and S. Kumar. Comparison of multi-objective evolutionary neural network, adaptive neuro-fuzzy inference system and bootstrap-based neural network for flood forecasting. *Neural Computing and Applications 23(S1)*, 2013.

Y.-H. Kim, J.-H. Kim, and K.-H. Han. Quantum-inspired multiobjective evolutionary algorithm for multiobjective 0/1 knapsack problems. *IEEE Congress on Evolutionary Computation*, 2006.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS Proceedings*, 2012.

K.Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter. Towards automatically-tuned neural networks. *ICML AutoML Workshop*, 2016.

R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy, and B. Hodjat. Evolving deep neural networks. *arXiv*, 1703.00548, 2017.

E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Sematsu, and Q. Le. Large-scale evolution of image classifiers. *arXiv*, 1703.01041, 2017.

B. Reddy, Y.-H. Kim, S. Yun, C. Seo, and J. Jang. Real-time driver drowsiness detection for embedded system using model compression of deep neural networks. *CVPR Embedded Vision workshop*, Jul 2017.

L. G. Sant, A. M. Muniz, C. Schaff, and R. Garnett. Bayesian optimization for automated model selection. *ICML AutoML Workshop*, 2016.

M. J. Shafiee, E. Barshan, and A. Wong. Evolution in groups: A deeper look at synaptic cluster driven evolution of deep neural networks. *arXiv*, 1704.02081, 2017.

J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *ICML*, 2015.

K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *The MIT Press Journals Evolutionary Computation*, 10(2):99–127, 2002.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv*, 1409.4842, 2014.

Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6), Dec 2007.

E. Zitzler, M. Laumanns, and L. Thiele. Evolving neural networks through augmenting topologies. *TIK-Report*, 2001.

B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv*, 1611.01578, 2016.