DVOLVER: EFFICIENT PARETO-OPTIMAL NEURAL NETWORK ARCHITECTURE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Automatic search of neural network architectures is a standing research topic. In addition to the fact that it presents a faster alternative to hand-designed architectures, it can improve their efficiency and for instance generate Convolutional Neural Networks (CNN) adapted for mobile devices. In this paper, we present a multi-objective neural architecture search method to find a family of CNN models with the best accuracy and computational resources tradeoffs, in a search space inspired by the state-of-the-art findings in neural search. Our work, called *Dvolver*, evolves a population of architectures and iteratively improves an approximation of the optimal Pareto front. Applying *Dvolver* on the model accuracy and on the number of floating points operations as objective functions, we are able to find, in only 2.5 days[1], a set of competitive mobile models on ImageNet. Amongst these models one architecture has the same Top-1 accuracy on ImageNet as NASNet-A mobile with 8% less floating point operations and another one has a Top-1 accuracy of 75.28% on ImageNet exceeding by 0.28% the best MobileNetV2 model for the same computational resources.

1 INTRODUCTION

The advent of deep neural networks ushered in qualitative leaps in several challenging machine learning tasks including image recognition (Krizhevsky et al., 2017) and language modeling (Sutskever et al., 2014). However, on top of being tedious and time consuming, deep neural network design requires deep technical knowledge. Automating this design process has reached important milestones in the last few years as advances in neural architecture search (Zoph & Le, 2016; Zoph et al., 2017) bridged the performance gap between manually designed architectures and those found via automatic search.

Traditional approaches usually rely on Reinforcement Learning (RL) (Baker et al., 2016; Zoph et al., 2017; Zoph & Le, 2016; Zhong et al., 2017), Genetic Algorithms (GA) (Precup & Teh, 2017; Xie & Yuille, 2017; Liu et al., 2017b; Real et al., 2018) or Sequential Model Based Optimization (SMBO) (Liu et al., 2017a; Brock et al., 2017), to search for the best performing architectures.

This search process is quite resource heavy which is why current work on architecture search shifted focus from merely outperforming manually designed architectures to accelerating the search process (Liu et al., 2017a; Pham et al., 2018; Liu et al., 2018). Nonetheless, very few works (Dong et al., 2018; Smithson et al., 2016; Ye-Hoon et al., 2017) attempted to take additional constraints such as memory consumption or inference time into consideration when evaluating candidate architectures, although it is of crucial importance when targeting mobile devices.

In this article, we introduce *Dvolver*, a principled multi-objective neural architecture search approach using Pareto optimality to navigate the search space. We instantiate this approach with a standard evolutionary algorithm (Deb et al., 2002) on the NASNet search space (Zoph et al., 2017) with validation accuracy and the total number of floating point operations (FLOPS) as objectives.

Our experiments show that (i) we do not sacrifice accuracy by looking for faster architectures (ii) we are able to explore the search space without loss in efficiency (iii) we are able to find architec-

[1] Using 20 NVIDIA V100 GPUs

tures that outperform NASNet-A mobile and certain configurations of MobileNetV2 architectures (Sandler et al., 2018) when accounting for both accuracy and floating point operations count.

2 RELATED WORK

Our work is founded on previous successful contributions focusing on single objective neural network architecture search that we extend for multiple criteria. Recently, independent papers came out with a similar approach but their search space is orders of magnitude smaller than the best studies so far.

Mono-objective Search Zoph et al. (2017) introduce a modular search space, based on reusable cells, which not only reduces the computational cost of the search process but also allows the search to be carried out on a small database and transfered to a larger one. They use RL to train a Recurrent Neural Network (RNN) that generates the structure of the cells. Real et al. (2018) showed that artificially-evolved architectures, on the same search space, are at least as good as their RL counterpart. On a similar note, Liu et al. (2017a) adopted SMBO coupled with a progressive exploration of a variation of said space, greatly reducing the search cost while maintaining competitive results. Their approaches achieved state of the art performances on ImageNet (Deng et al., 2009), with modules learned on CIFAR-10 (Krizhevsky & Hinton, 2009) but they only optimized for best validation accuracy. The best module is then combined in a scaled down network and compared with the best mobile architecture at the time. Even though, the results are impressive, this is sub-optimal and explicitly optimizing for validation accuracy and speed can lead to even better architecture.

Multi-objective Search Smithson et al. (2016) used Pareto optimality as a selection criteria in an SMBO-based hyperparameter search, and showed that the obtained Pareto front is close to the one obtained via exhaustive search, even with two orders of magnitude fewer architecture evaluations. However, their work focused on hyperparameter search and did not explore a more general search space.

In Ye-Hoon et al. (2017), a multi-objective genetic algorithm is implemented to search for fast and accurate architectures, initializing the search process from a given baseline network. They were able to find architectures that outperformed common baselines [2] both in speed and accuracy on MNIST and CIFAR10. Their method, while in principle similar to ours, explored the vicinity of a defined network and was not extended to a complex search space where the search process is initialized arbitrarily. Furthermore, it was not validated for learning architectures that transfer to large scale applications.

Dong et al. (2018) work is perhaps the closest to ours in intent and scale, as it performs multiple objective architecture search using Pareto optimality and uses a proxy dataset to avoid an expensive search on ImageNet. It even goes one step further to incorporate device-dependent metrics. However, their search space is orders of magnitude smaller than ours in addition to being heavily biased towards mobile architectures.

3 PROPOSED METHOD

For a given search space, our goal is to find all the neural network architectures that offer the best trade-offs between multiple independent criteria. We formulate the search process as a *multi-objective optimization* (MOO) problem (Section 3.1) where each objective to optimize is an independent scalar value. We refer to Figure 1 for a general overview of the search process.

The MOO controller selects child networks with different architectures. The child networks are evaluated on multiple criteria by the Evaluation Engine. The resulting objective values are then used by the controller as feedback so it can generates better architectures over time. At each iteration, *Dvolver* keeps the hall of fame of the best architectures found so far.

[2] namely Lenet (Lecun et al., 1998) and an all convolutional net (Springenberg et al., 2014)

3.1 MULTI-OBJECTIVE OPTIMIZATION AND PARETO DOMINANCE

We define the *search space* X as the set of all feasible solutions and the *objective space* as the set of all possible values of the objective function $f : X \rightarrow \mathbb{R}^k$. *Dvolver* searches for solutions by maximizing the objective function f over X with the total order relation defined next.

For a nontrivial multi-objective optimization problem, there is no solution that simultaneously optimizes all the objectives. In that case, the objective functions are said to be *conflicting*, and there exists a (possibly infinite) number of compromise solutions called *Pareto-optimal* solutions. The set of Pareto optimal outcomes is often called the *Pareto front*.

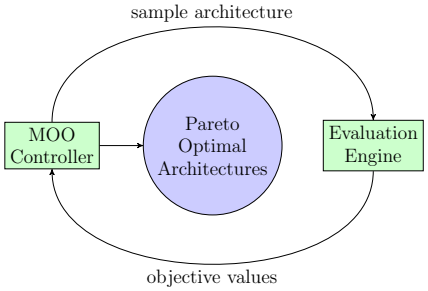


Figure 1: Overview of *Dvolver* Search Process. A controller generates a candidate architecture which is trained to convergence by the evaluation engine. Other criteria to optimize are evaluated while the network is training. Resulting metrics are fed back to the controller. At each iteration, the controller keeps track of the best performing architectures.

A simple approach to cope with multi-objective functions is to use scalarization which squashes all the objective functions into a single function falling back to single-objective optimization. While very simple in practice, this method suffers from many shortcomings such as (i) the difficulty to combine multiple unrelated criteria into a single value and (ii) inefficient sampling of the full Pareto front.

A much better solution is to use Pareto dominance partially ordered relation defined as follow: A solution v Pareto-dominates a solution u, if all objective values for v are greater than the objective values for u but a least one is strictly superior. A feasible solution is Pareto optimal, if it is not dominated by another solution. Without additional subjective preference information, all Pareto optimal solutions are considered equally good.

For the search process to be effective, it needs a totally ordered relation. Deb et al. (2002) defines the *crowding distance* value of a particular solution as the average distance of its two neighboring solutions. In addition to the Pareto dominance criteria, the crowding distance defines the totally ordered relation used by *Dvolver* to perform its search process.

3.2 SEARCH ALGORITHM

Most of the previous approaches, only keep track of the best solution found so far. On the contrary, *Dvolver* evolves a population of N individual architectures. At each iteration, the population evolves closer and closer toward the Pareto front, effectively sampling the full Pareto front in a single search process.

Our method starts by generating a random population and for each generation, it generates a child population, evaluates each child by training the networks for a few epochs on CIFAR-10 and selects the N dominant architectures from the $2 \times N$ individuals (parents and children). Algorithm convergence is monitored with the hypervolume indicator defined as the volume under the Pareto front. In Algorithm 1, we present the *Dvolver* selection process inspired by NSGA-II (Deb et al., 2002).

At the end of an iteration, there are N parents and N children. We should select the best N individuals to form the parents' population for the next iteration. We compute the Pareto fronts for the $2 \times N$ individuals. The best fronts are considered first. If there are more than N individuals in the best front, the first N (with the help of the crowding distance) are selected and the selection ends. Otherwise, all the individuals from the front are selected and we carry on with the next best Pareto front until N individuals are selected.

Algorithm 1: *Dvolver* selection algorithm

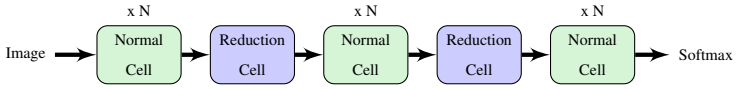
Inputs

- N individuals from parents' population
- N individuals from offsprings' population

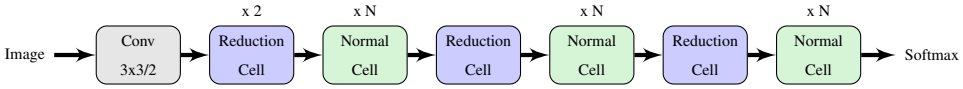
Output

- N selected individuals Inds_{sel}

$\text{Inds}_{sel} \leftarrow \varnothing$;
fronts $\leftarrow$ `ComputeParetoFronts`(parents $\cup$ offsprings);
$i \leftarrow 0$; ▷ Starts with Pareto front with rank 0
rem $\leftarrow N$; ▷ rem is the number of remaining individuals to be selected
while *rem* > 0 **do**
  $\text{Inds}_i \leftarrow$ `SortWithDominance`(fronts_i);
  if `Len`($Inds_i$) $<$ *rem* **then**
    $\text{Inds}_{sel} \leftarrow \text{Inds}_{sel} \cup \text{Inds}_i$;
    rem $\leftarrow$ rem $-$ `Len`(Inds_i);
    $i \leftarrow i + 1$; ▷ Move to the next Pareto front
  else
    $\text{Inds}_{sel} \leftarrow \text{Inds}_{sel} \cup$ `First`(rem, Inds_i); ▷ Highest crowding distance individuals first
  rem $\leftarrow 0$;



(a) CIFAR-10 Architecture



(b) ImageNet Architecture

Figure 2: Scalable architectures for image classification consist of two repeated motifs termed Normal Cell and Reduction Cell. This diagram highlights the model architecture for CIFAR-10 and ImageNet. The choice for the number of times the Normal Cells that gets stacked between Reduction Cells, N, can vary in our experiments.

3.3 Search Space

Our work is based on the work of Zoph et al. (2017), in which network architecture is composed of two levels: a micro level architecture defined by parametrized cells composed of B blocks and a macro architecture where those cells are stacked a predefined number of times, in order to create the final network. Figure 2 describes the macro architecture used for CIFAR-10 and ImageNet architectures.

There are two types of cells and *Dvolver* searches for both at the same time: (i) *Normal Cell* that returns a feature map of the same size as its input and (ii) *Reduction Cell* that returns a feature map of half the size of its input in the spatial dimensions.

A cell is composed of B *blocks* arranged in a Directed Acyclic Graph (DAG). Each block is a mapping of two input tensors to one output tensor and it can be represented by a 4-tuple, (I_1, O_1, I_2, O_2) where I_k specifies the input for the operation O_k. The result of the two operations O_1 and O_2 are summed together to form the output of the block. A block can take its input from previous blocks, the previous cell c_{k-1} or the previous previous cell c_{k-2}. Unused blocks are concatenated in the depth dimension to form the final output of the cell.

Moreover, we noticed that the original NASNet search space can greatly benefit from extra connections from any given block, the previous cell and the previous previous cell to the final concatenation layer. *Dvolver* optimizes for these extra connections in addition to the 4 parameters for each block.

For $B = 5$, the total number of parameters for each type of cell is 20 parameters for the blocks and 1 list of indexes of extra connections.

Similar to the approach of Liu et al. (2017a), we simplify the search space and consider only the following operations:

- identity
- 3x3 average pooling
- 3x3 max pooling
- 3x3 depthwise-separable conv
- 5x5 depthwise-separable conv
- 7x7 depthwise-separable conv

The total number of combinations for the defined search space is about 10^{20} taking into account symmetries and possible redundant connections. Note that our search space is larger than the one defined in Liu et al. (2017a) but we show in Section 4 that *Dvolver* is capable of finding competitive architectures faster.

4 Experiments and Results

Here we present the results for *Dvolver* with the method described in Section 3.1 and search space in Section 3.3. Searching for neural network architecture is done on CIFAR-10 (see Section 4.1) and optimization is performed with the validation accuracy and the number of floating points operations as independent objectives. The best architectures are then fully trained on ImageNet (see Section 4.2).

4.1 Search Results

In this section, we describe our experiment to learn the Pareto front of convolutional cells for the search space described above. In particular, we conduct the search process on the CIFAR-10 dataset. We separate 5,000 images from the train set as a validation set used for architectures evaluation during the search process. We use a population of 32 individuals and a uniform mutation and a crossover probability of 0.1. The number of cells N is set to 2 and use 32 convolution filters. Each sampled architecture is trained for 72 epochs with a batch size of 150 and a learning rate with a linear decay from 0.1 to 0.039. We evaluate the candidate architectures in parallel on multiple GPUs and the search process takes 50 GPU-days to find competitive architectures.

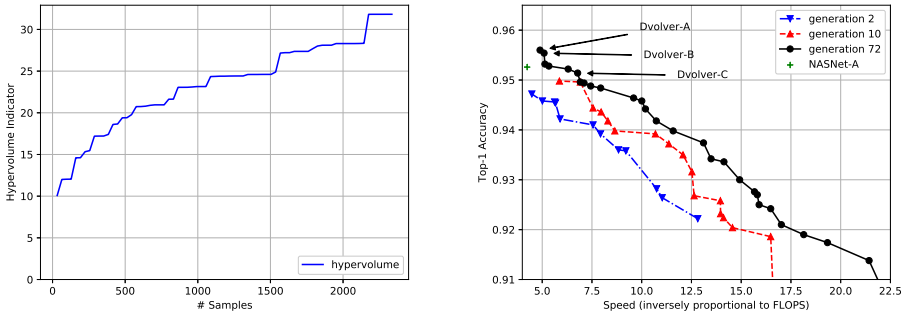
The Figure 3 shows the evolution of the hypervolume indicator as a function of the sampled architectures (Figure 3(a)) and the final Pareto front estimation with 2 intermediate fronts as an illustration of the evolution process. The horizontal segment in the Pareto front are particularly interesting. Cells situated on knees in the Pareto front are good trade-offs and this figure shows how the Pareto front can be a great tool when choosing a network architecture for a specific application. We manually select 3 architectures that we will be fully train on ImageNet in Section 4.2. They are called Dvolver-A, B and C.

The detailed architectures of the cells for Dvolver-A, B and C are presented in the diagrams in Figure 4. We can see that in most of them additional connections (see Section 3.3) are present which matches our observations during our preliminar experiments.

4.2 Results on Imagenet

Dvolver-A, B and C cells found in Section 4.1 are fully trained for different combination of cells, number of normal cells N and number of convolution cells F on ImageNet following the training procedure found in Real et al. (2018). Input image size is set to 224x224 and each network is trained for 330 epochs.

First, in Table 1, we select a few architectures with similar computational complexity as networks presented in Zoph et al. (2017) and show that we can find Pareto dominant architectures that can be more accurate and faster than architectures found with single objective optimization. Dvolver-A (N=4, F=44) has the same Top-1 accuracy of 73.64% with 8% less floating point operations than NASNet-A mobile. We also found two architectures with slightly more computations needs (3% and 7% more MACs) but with better Top-1 accuracy 74.37% and 74.81% respectively.



(a) Hypervolume indicator evolution

(b) Evolution of the Pareto front estimation

Figure 3: 3(a) shows the evolution of the hypervolume during the search process until it stops increasing. 3(b) represents the Pareto front estimation for a few generations during the search process. We define the speed $= \frac{2\times10^9}{\text{FLOPS}}$ which can be roughly interpreted as the number of inferences per second on a mid-range mobile device. The 3 selected cells Dvolver-A, B and C are shown on the Pareto front and NASNet-A is represented for reference.

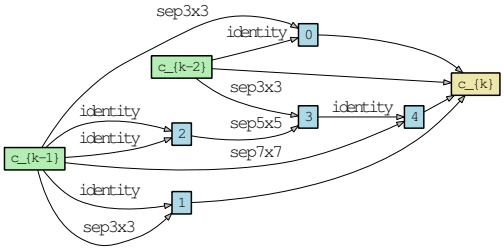
Table 1: Performance of Dvolver and NASNet on ImageNet. NASNet results are the ones we could reproduce in our setup on the same training conditions as the rest of our results. It is on par with the median value found by Ramachandran et al. (2017) but lower than the official results (Top-1 74%) found by Zoph et al. (2017). There are 2 FLOPS for 1 Mult-Adds (MACs).

Model	**Top-1 Acc.(%)**	**Top-5 Acc.(%)**	**#Mult-Adds**	**#Parameters**
NASNet-A (N=4, F=44)	73.64	91.53	564M	5.3M
Dvolver-A (N=3, F=48)	73.14	91.37	493M	4.3M
Dvolver-A (N=4, F=44)	73.64	91.49	**520M**	4.5M
Dvolver-A (N=4, F=48)	**74.81**	92.15	607M	5.25M
Dvolver-B (N=3, F=52)	**73.65**	91.72	**544M**	4.73M
Dvolver-C (N=3, F=44)	70.24	89.58	317M	2.76M
Dvolver-C (N=4, F=56)	**74.37**	91.86	582M	5.06M

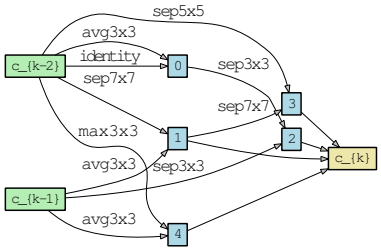
Inspired by the results of Ramachandran et al. (2017) on NASNet-A mobile, we replace all RELU activation functions by SWISH activation functions and compare our results with the best state-of-the-art architecture for mobile. We found that we can outperfom MobileNet-V2 in the high end mobile space with 75.28% Top-1 accuracy for the same number of floating point operations but stay behind on low-end mobiles. Table 2 summarizes our findings:

Table 2: Performance of Dvolver and other state-of-the-art models on ImageNet

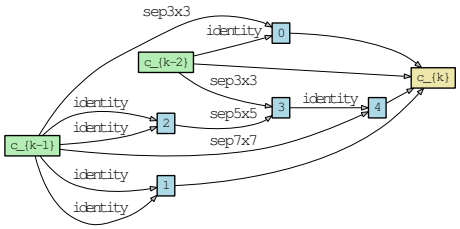
Model	**Top-1 Acc.(%)**	**Top-5 Acc.(%)**	**#Mult-Adds**	**#Parameters**
MobileNetV1-224 1.0	70.9	89.9	569M	4.24M
MobileNetV1-224 0.75	68.4	88.2	317M	2.59M
MobileNetV2-224 1.4	75.0	92.5	582M	6.06M
MobileNetV2-224 1.3	74.4	92.1	509M	5.34M
MobileNetV2-224 1.0	71.8	91.0	300M	3.47M
MobileNetV2-224 0.75	69.8	89.6	209M	2.61M
ShuffleNet (2x)	70.9	89.8	524M	5M
Dvolver-A (3, 48, swish)	74.35	91.99	493M	4.3M
Dvolver-B (3, 52, swish)	74.70	92.33	544M	4.73M
Dvolver-C (3, 44, swish)	70.95	90.05	317M	2.76M
Dvolver-C (4, 56, swish)	**75.28**	92.32	582M	5.06M



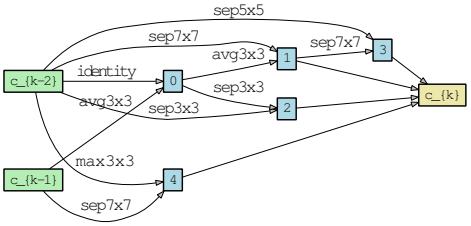
(a) Dvolver-A Normal Cell



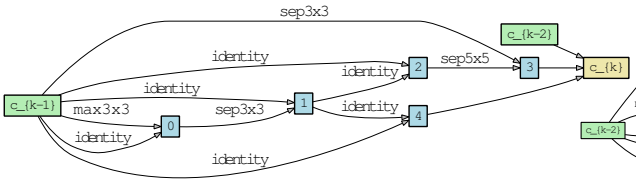
(b) Dvolver-A Reduction Cell



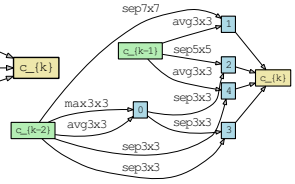
(c) Dvolver-B Normal Cell



(d) Dvolver-B Reduction Cell



(e) Dvolver-C Normal Cell



(f) Dvolver-C Reduction Cell

Figure 4: Dvolver-A, B and C cells (best viewed in color). Blue boxes are the output of the 5 blocks. Each block has 2 incoming arrows representing the 2 operations in the block. Labels on arrows describe the specific operation performed. The origin of the arrow is the input of for the operation. Green boxes represent the previous and previous previous cells and the yellow box represents the final depth concatenation at the end of the cell.

There are a lot of different mobile devices with extreme computation resources diversity. To keep the discussion simple, we consider two tiers of devices: the low-end mobile market with devices with less than 400M Mult-Adds for a specific computation and a high-end tier with devices with higher performance.

Our best architectures are compared with the state-of-the-art architectures in Figure 5. In particular, we show that optimizing for multiple criteria including the number of floating operations, *Dvolver* finds architectures with better trade-offs than NASNet architectures. The poor performances at low operation count, show a weakness in the design of the NASNet search space which seems to be less efficient as a better tailored search space, design with low-end mobile device in mind.

4.3 Search Efficiency

An important result of our work is the computational resources needed to perform the architecture search. *Dvolver* finds the full Pareto front, on a much bigger search space, 3 times faster than PNAS (Liu et al., 2017a). Table 3 presents the computational resources needed to perform the automatic search process with different method found in the field.

One explication for *Dvolver* speed compared to other method is that it actually generates a population of architectures and among them some are very fast (see Figure 3(b)). Intriguingly, these very fast architectures do not penalize the search process for the most accurate architectures. We think that the interaction between the different kind of individuals in the population is an essential property responsible for its effectiveness.

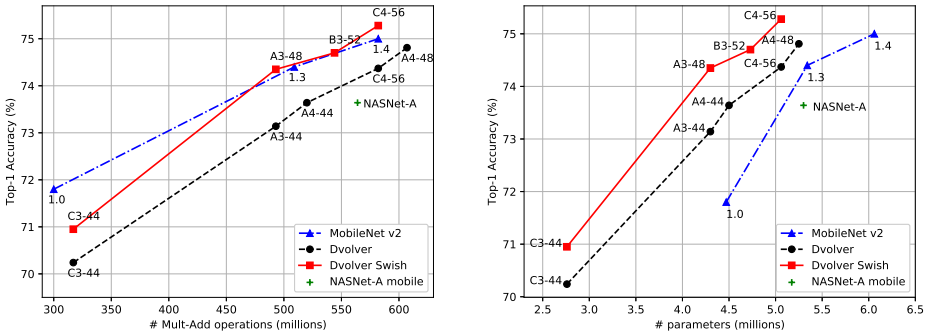


Figure 5: Accuracy versus computational demand (left) and number of parameters (right) for *Dvolver* and state-of-the-art architectures for mobile devices on ImageNet. Computational demand is measured in the number of floating-point multiply-add operations to process a single image. Labels on the figures for *Dvolver* are of the form Family, N and F. For example Dvolver-A with N=3 and F=48 is noted A3-48.

Table 3: Search efficiency for *Dvolver* and other state-of-the-art related methods

Method	**GPU.days**	**Search Space Size (Approx.)**
NASNet (Zoph et al., 2017)	1800	10^{28}
PNAS (Liu et al., 2017a)	150	10^{12}
AmoebaNet (Real et al., 2018)	3150	10^{28}
Dvolver	**50**	10^{20}

5 Conclusion

In this work, we propose a general method to search for Pareto-optimal neural architectures optimized for multiple criteria. In particular, we propose two novel cell structures. The first one achieves the same Top-1 accuracy on ImageNet as NASNet-A mobile with 8% less floating point operations and the second one outperforms the best MobileNetV2 model by 0.28% in Top-1 accuracy (75.28%) for the same computational resources.

Our key contribution is the iterative population-based approach which approximates the Pareto front in a single search process. Our search method is efficient in terms of number of sampled architectures and computational resources. It performs the full search for the Pareto front in 50 GPU.days, 3 times faster than PNAS (Liu et al., 2017a) on a larger search space.

Our work is based on a few assumptions that can be discussed. First, it relies on CIFAR-10 as proxy for ImageNet in the search process. This has several weaknesses: It is very hard to distinguish different architectures on this dataset as the differences between architecture's accuracies are very small and very noisy. In addtition, our method relies on the order to be conserved when transfering the architectures from CIFAR-10 to ImageNet and in practice it works but it is not always the case. Second, it is very expensive to fully train each candidate architecture from scratch even on CIFAR-10. To speedup the search process, we train for a few epochs with high learning rate and state that the ordering for this few epochs is the same as the one if all the architectures were fully trained. This is not always the case and it can inder the performance of the search process.

As a future work, we could apply our method on a different search space inspired by the lastest state-of-the-art mobile architectures to improve the performance on low-end mobile devices. Another way to improve our work is to enhance the selection phase by reducing the variance in the accuracy caused by the early stopping assumption.

REFERENCES

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *CoRR*, abs/1611.02167, 2016. URL http://arxiv.org/abs/1611.02167.

Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. SMASH: one-shot model architecture search through hypernetworks. *ICML*, abs/1708.05344, 2017. URL http://arxiv.org/abs/1708.05344.

Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017. URL https://doi.org/10.1109/4235.996017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pp. 248–255. IEEE Computer Society, 2009. ISBN 978-1-4244-3992-8. doi: 10.1109/CVPRW.2009.5206848. URL https://doi.org/10.1109/CVPRW.2009.5206848.

Jin-Dong Dong, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. DPP-Net: Device-aware Progressive Search for Pareto-optimal Neural Architectures. Technical report, 2018. URL https://arxiv.org/pdf/1806.08198.pdf.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, 2017. doi: 10.1145/3065386. URL http://doi.acm.org/10.1145/3065386.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):22782324, 1998. ISSN 0018-9219. doi: 10.1109/5.726791. URL http://dx.doi.org/10.1109/5.726791.

Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *CoRR*, abs/1712.00559, 2017a. URL http://arxiv.org/abs/1712.00559.

Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. *CoRR*, abs/1711.00436, 2017b. URL http://arxiv.org/abs/1711.00436.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. *CoRR*, abs/1806.09055, 2018. URL http://arxiv.org/abs/1806.09055.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *JMLR Workshop and Conference Proceedings*, pp. 4092–4101. JMLR.org, 2018. URL http://proceedings.mlr.press/v80/pham18a.html.

Doina Precup and Yee Whye Teh (eds.). *Large-Scale Evolution of Image Classifiers*, volume 70 of *Proceedings of Machine Learning Research*, 2017. PMLR. URL http://proceedings.mlr.press/v70/real17a.html.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017. URL http://arxiv.org/abs/1710.05941.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. *CoRR*, abs/1802.01548, 2018. URL http://arxiv.org/abs/1802.01548.

Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. URL http://arxiv.org/abs/1801.04381.

Sean C. Smithson, Guang Yang, Warren J. Gross, and Brett H. Meyer. Neural networks designing neural networks: Multi-objective hyper-parameter optimization. *CoRR*, abs/1611.02120, 2016. URL http://arxiv.org/abs/1611.02120.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014. URL http://arxiv.org/abs/1412.6806.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL http://arxiv.org/abs/1409.3215.

Lingxi Xie and Alan L. Yuille. Genetic CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 1388–1397. IEEE Computer Society, 2017. ISBN 978-1-5386-1032-9. doi: 10.1109/ICCV.2017.154. URL https://doi.org/10.1109/ICCV.2017.154.

kim Ye-Hoon, Reddy Bhargava, Yun Sojung, and Seo Chanwon. NEMO: Neuro-Evolution with Multiobjective Optimization of Deep Neural Network for Speed and Accuracy. *ICML'17 AutoML Workshop*, 2017.

Zhao Zhong, Junjie Yan, and Cheng-Lin Liu. Practical network blocks design with q-learning. *CoRR*, abs/1708.05552, 2017. URL http://arxiv.org/abs/1708.05552.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. URL http://arxiv.org/abs/1611.01578.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017. URL http://arxiv.org/abs/1707.07012.