Macro Neural Architecture Search Revisited

Hanzhang Hu[1]*, **John Langford**[2], **Rich Caruana**[2], **Eric Horvitz**[2], **Debadeepta Dey**[2]
[1]Carnegie Mellon University; [2]Microsoft Research
hanzhang@cs.cmu.edu, {jcl,rcaruana,horvitz,dedey}@microsoft.com

Abstract

Neural architecture search (NAS) has recently shown promising results for automatically finding cost-efficient and accurate predictors. However, most recent work exclusively study the small search space of repeatable network modules (cells), instead of the more general overall network (macro), partially because the models found by macro-search typically require an order of magnitude more parameters compared to cell-search to have the same accuracy. In this work, we show through ablation study that this gap mainly exists due to the difference in initial models. In fact, by starting with the same condition as cell-search, the proposed macro-search can find a CIFAR-10 model that has 2.93% test error rate and uses 3.1 million parameters. Our macro-search algorithm has the advantage of being simple and fast. The search procedure randomly and incrementally grows the most cost-efficient models on the Pareto frontier. The proposed search takes only 6 GPU-days, which is much smaller than those of many existing methods, while achieving comparable or better results.

1 Introduction and Background

There has been keen interest in automated neural architecture search (NAS), ever since Zoph & Le (2017) showed that it is possible to find models that are more accurate and less computationally expensive than the best human designed models. While early NAS work (Zoph & Le, 2017; Real et al., 2017) consider **macro-search**, where each layer in the network can take inputs from any previous layer independent of how other layers choose their inputs, the most recent work (Liu et al., 2018b; Luo et al., 2018; Cai et al., 2018) only consider **cell-search**, where the algorithm only searches over how a few layers (typically $5 \sim 8$) are connected to each other, and use the found connection pattern, called a cell, as a repeatable module in a human designed skeleton network that consists of a sequence of cells. Macro-search and cell-search are compared in (Zoph et al., 2018; Liu et al., 2017; Pham et al., 2018; Elsken et al., 2018), which find that in comparison to macro-search, cell-search can find models that are more accurate with an order of magnitude fewer parameters. Zoph et al. (2018) also demonstrate that cells found on CIFAR-10 can be composed with larger skeleton outer networks to become state-of-the-art models on ImageNet (Russakovsky et al., 2015). Thereafter, no attempts have been made to reconcile the gap between macro and cell-search in terms of the performance of the found models, despite the fact that macro-search is more general and may be the only option on novel tasks and data-sets where expert knowledge about the outer skeleton does not exist.

In this work, we investigate why macro-search has been outperformed by cell-search, and find that the discrepancy in the initial model to be a dominating factor. In particular, when we start macro-search with the outer skeleton of Zoph et al. (2018) composed with one of the simplest cells in their search space, our macro-search found models for CIFAR-10 that takes 3.1 million parameters to reach 2.93% test error (averaged over 5 trials). This is close to the state-of-the-art, which is previously only achievable by cell-search, and to the best of our knowledge, it is the best macro-search result at the time of writing. The search is also much faster than many existing methods, taking only 6

*This work was part of an internship at Microsoft Research by Hanzhang Hu.

2nd Workshop on Meta-Learning at NeurIPS 2018, Montréal, Canada.

GPU-days. Our ablation study then shows that macro-search actually finds slightly better models than cell-search if they start with the same model. Furthermore, if we reduce the starting model complexity by half, then our macro-search result is degraded to 3.44% even when we triple the search resource. This sheds light on the current macro-cell gap: though macro-search can potentially utilize the extra freedom to find better models than cell-search, the benefits are overshadowed currently, because cell-search methods uses better initial models.

2 Search via Random Growth from Promising Models

We propose a simple algorithm, **RandGrow**, to grow networks in search of cost-efficient models. The search is defined by five components: the choice of parent models, the incremental model changes, the training procedure during search, the initial model, and the procedure for training the final model. The first three components are repeatedly looped over by parallel workers. The best model in validation goes through the final training procedure to report test error. We give a brief overview below:

Choice of Parent models. The goal of our algorithm is to find *cost-efficient* predictors. For simplicity, we take a greedy approach. Whenever we need to choose a parent model, we first plot the computational cost versus validation error of existing models, and then compute the lower convex hull of this curve. We stop the hull when it starts increasing. We then choose a model on the hull as the parent with probability defined as the following. If m_1 and m_2 are two adjacent models on the hull, with computational costs c_1 and c_2 ($c_1 < c_2$), then we set the probability weight of m_1 to be proportional to $c_2 - c_1$. The most accurate model (which has no following model on the curve) is always chosen with probability 0.5. This greedy approach is most similar to multi-objective architecture search (Elsken et al., 2018; Hsu et al., 2018) that focuses on models on the pareto-frontier of the performance scatter plot. Both methods modify the most cost-efficient models and convex hull is a subset of the pareto-frontier.

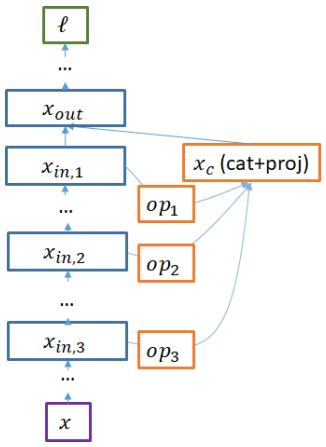

Figure 1: Incremental model change. Blue indicates existing layers, orange indicates increments. op_i are considered part of x_c.

Incremental Model Change. Fig. 1 illustrates a single incremental model change, x_c. Every iteration we randomly generate many of them. The number of changes at each iteration depends on the initial network, and we detail the relation later. To form x_c, we first randomly sample the target layer x_{out} from layers that were in the initial network. Then three input layers $x_{in,i}$ for $i = 1, 2, 3$ are chosen. To ensure that x_c has access to local layers, $x_{in,1}$ always chooses the deepest input of x_{out} that was in the initial network. $x_{in,2}$ and $x_{in,3}$ are sampled with replacement uniformly at random from all layers that are topologically earlier than x_{out}. Each input layer then uniformly randomly choose one of the five operations: 3x3 and 5x5 separable convolution, 3x3 max and average pooling, and identity. Following (Zoph et al., 2018; Real et al., 2018; Pham et al., 2018), separable convolutions are applied twice. The processed inputs are concatenated together and then projected to the same shape as the target layer x_{out} using a 1x1 convolution. The result is the incremental change x_c, which is added to x_{out}.

In our cell-search, our baseline, we use the same sampling procedure to form the incremental changes, except that we follow (Zoph et al., 2018) and choose inputs within the same cell or from the outputs of the previous two cells in cell-search.

Training During Search on CIFAR-10. The incremental changes have randomly initialized parameters while the parameters of the parent model are reused. We train the new network (parent network + incremental changes) with a batch size of 32 using SGDR (Loshchilov & Hutter, 2017) that starts at 0.025 and ends at 0 in 80 epochs for two cycles (160 epochs). Following (Elsken et al., 2018), we apply standard data augmentations of cropping-after-padding, left-right flipping, and pixel-wise normalization, along with cutout (DeVries & Taylor, 2017).

Initial Network. Since we aim to compare cell and macro-search, we start with a network that is compatible with both. Following common baselines (Zoph et al., 2018), we start on a network that has 3 blocks. Each block has $N = 6$ regular cells and there is a reduction cell in between blocks. The

Table 1: Comparison against state-of-the-art recognition results on CIFAR-10. Results marked with † are not trained with cutout. The first block represents approaches for macro-search. The second block represents approaches for cell-search.

Method	**# params** (mil.)	**Search** (GPU-Days)	**Test Error** (%)
Zoph & Le (2017)†	7.1	1680+	4.47
Zoph & Le (2017) + more filters†	37.4	1680+	3.65
Real et al. (2017)†	5.4	2500	5.4
ENAS macro (Pham et al., 2018)†	21.3	0.32	4.23
ENAS macro + more filters†	38	0.32	3.87
Lemonade I (Elsken et al., 2018)	8.9	56	3.37
RandGrow initial model ($N = 6$, $F = 32$)	0.4	–	4.6
RandGrow macro without DropPath	3.1	6	3.38
RandGrow macro	3.1	6	**2.93**
RandGrow macro (start at N=3, F=32)	2.7	18	3.44
NasNet-A (Zoph et al., 2018)	3.3	1800	2.65
AmoebaNet-A (Real et al., 2018)	3.2	3150	3.3
AmoebaNet-B (Real et al., 2018)	2.8	3150	2.55
PNAS (Liu et al., 2017)†	3.2	225	3.41
Heirarchical NAS (Liu et al., 2018a)†	15.7	300	3.75
ENAS cell (Pham et al., 2018)	4.6	0.45	2.89
ENAS cell (Pham et al., 2018)†	4.6	0.45	3.54
Lemonade II (Elsken et al., 2018)	3.98	56	3.50
Darts (Liu et al., 2018b)	3.4	4	2.83
Darts random (Liu et al., 2018b)	3.1	–	3.49
Cai et al. (2018)	5.7	8	2.49
Luo et al. (2018)†	3.3	0.4	3.53
RandGrow cell	2.0	6	3.21
RandGrow cell + more filters	3.5	6	3.05

initial channel size is $F = 32$. Our initial cell is inspired by ResNet's (He et al., 2016) residual unit: given an input x, it produces $h(x) + x$, where h is the 3x3 separable convolution operation in our search space. This cell is also the smallest non-trivial cell in the search space of (Zoph et al., 2018). At the start of each search, the initial model is trained from scratch as if it is a final model (more details below). The initial model has around 4.6% error rate on CIFAR-10. We recognize that this initial model for macro-search is more complex than early pioneering work (Zoph & Le, 2017; Real et al., 2017). Hence, we also conduct an ablation study using $N = 3$ for macro-search, and we find that this discrepancy in initial models is a crucial reason for macro-search to be outperformed by cell-search in current literature.

We choose one increment for each of normal and reduction cells in cell-search. Since there are $(N \times 3 + 2 =)20$ cells, we choose 20 increments for macro-search, so that the overall number of structural increments is kept the same at each iteration for comparing macro and cell-search fairly.

3 Experiments

Our experiments first showcase that macro-search can indeed find state-of-the-art models on CIFAR-10 that are currently only obtained via cell-search. Then we show that RandGrow macro-search can outperform RandGrow cell-search if they start from the same model. Finally, we use ablation study on the initial model to show how dominating initial model effects are on the search results.

Final training of the best models obtained. Following (Zoph & Le, 2017; Zoph et al., 2018; Real et al., 2018), we train from scratch the best models obtained via the search procedure (in terms of validation error), with batch size 32 using cosine learning rate that shrinks from 0.025 to 0 in 600 epochs. We apply path drop-out (Larsson et al., 2017) for the final training. Both training and validation data are used. We report the average test error of five independent training runs.

Table 2: Comparison against recognition results on CIFAR-100. RandGrow models are the ones found in CIFAR-10 search. The first block represents the best human designed models. The second represents models found through automated macro-search on CIFAR100.

Method	**# params** (mil.)	**Test Error** (%)
DenseNet-BC (k=40) (Gao Huang, 2017)	25.6	17.18
Shake-Shake (Gastaldi, 2017)	26.2	15.85
SMASHv2 (Brock et al., 2017)	16	20.6
Real et al. (2017)	40.4	23.7
Elsken et al. (2017)	22.3	23.4
Block-QNN-S (Zhong et al., 2018)	6.1	20.65
RandGrow macro	3.1	18.74

CIFAR-10 macro-search. RandGrow macro-search is run on 4 GPUs for 1.5 days for a total of 6 GPU-days. Table 1 displays for each automated search method the number of parameters in the final obtained model, search cost in GPU-days, and the test error on CIFAR-10. RandGrow macro-search achieves much lower error rate (2.93%) than *all previous macro-search based methods* (first block of Table 1), and only requires a fraction of parameters. In fact, the final obtained model is close to the state-of-the-art that is currently only achieved by cell-search based methods, which are listed in the second block of Table 1.

Comparison to cell-search. RandGrow macro and cell-searches start with the same model and are constrained to make the same amount of change to the overall model at each step. In Table 1, our macro-search outperforms cell-search by finding a more accurate and less expensive model (2.93% vs. 3.05%), indicating that the extra freedom of macro-search can indeed help find better models.

The importance of initial model. RandGrow is the closest to (Elsken et al., 2018), since both work warm start from existing models that are the most cost-efficient, and incrementally modify the networks at random. One major difference is the starting model. Elsken et al. (2018) starts at shallow models with three conv layers. RandGrow starts at one of the simplest model that the cell-search can start with, but it is still vastly more complex than the starting models of previous macro-search. In fact, when we set $N = 3$ instead of $N = 6$ in the initial model, we observe in Table 1 that the found model only achieves 3.44% error rate, even if we increase the search resource to 18 GPU-days. This suggests that the effect of initial model far outweighs the difference between macro and cell-searches, which only causes a change of 0.1% in error rates. We can further infer the impact of initial models from other existing works. For instance, (Zoph et al., 2018; Real et al., 2018; Pham et al., 2018; Liu et al., 2018b; Luo et al., 2018) all use the same or similar skeletons, and their results are similar. [2] By starting from a more complex skeleton and a more complex starting cell[3], (Cai et al., 2018) finds a more accurate model using only 8 GPU-days.

Transfer to other data-sets. We also display in Table 2 the performance of human designed and macro-search models on CIFAR-100. Our model found in macro-search on CIFAR-10 achieves better accuracy than published macro-search results on CIFAR-100. We also transfer the macro-search model to ILSVRC2012 (Russakovsky et al., 2015) (more detail in appendix), and achieves 27.2% top-1 error rate using 6.88 million parameters and 830 million multi-add operations.

4 Discussion and Ongoing Work

The ablation study on initial models shows that they are extremely impactful to search speed and final model accuracy, we encourage future work to report initial model performances for fair evaluation of search algorithms. The successful macro-search on CIFAR-10 through RandGrow and ablation study on macro and cell-search together point future work to reconsider macro-search, because it is more general, and it may be the only viable option on novel data-sets and tasks where a reasonable outer skeleton architecture may not be available.

[2] (Zoph et al., 2018; Real et al., 2018) have similar performance, but are better than the others partially due to the thousands of GPU-days spent. (Pham et al., 2018; Liu et al., 2018b; Luo et al., 2018) have fast search, because of parameter sharing during search, a topic outside the scope of this work.

[3] (Cai et al., 2018) sets $N = 8$, and the starting cell is PyramidNet (Han et al., 2017)

References

Brock, Andrew, Lim, Theodore, Ritchie, James M., and Weston, Nick. Smash: one-shot model architecture search through hypernetworks. In *ICLR*, 2017.

Cai, Han, Yang, Jiacheng, Zhang, Weinan, Han, Song, and Yu, Yong. Path-level network transformation for efficient architecture search. In *ICML*, 2018.

DeVries, Terrance and Taylor, Graham. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.

Elsken, Thomas, Metzen, Jan Hendrik, and Hutter, Frank. Simple and efficient architecture search for convolutional neural networks. In *ICLR Workshop*, 2017.

Elsken, Thomas, Metzen, Jan Hendrik, and Hutter, Frank. Efficient multi-objective neural architecture search via lamarckian evolution. 2018.

Gao Huang, Zhuang Liu, Laurens van der Maaten Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

Gastaldi, X. Shake-shake regularization of 3-branch residual networks. In *ICLR Workshop*, 2017.

Han, Dongyoon, Kim, Jiwhan, and Kim, Junmo. Deep pyramidal residual networks. In *CVPR*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hsu, Chi-Hung, Chang, Shu-Huan, Juan, Da-Cheng, Pan, Jia-Yu, Chen, Yu-Ting, Wei, Wei, and Chang, Shih-Chieh. Monas: Multi-objective neural architecture search using reinforcement learning. *CoRR*, abs/1806.10332, 2018.

Larsson, Gustav, Maire, Michael, and Shakhnarovich, Gregory. Fractalnet: Ultra-deep neural networks without residuals. In *ICLR*, 2017.

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

Liu, Hanxiao, Simonyan, Karen, Vinyals, Oriol, Fernando, Chrisantha, and Kavukcuoglu, Koray. Hierarchical representations for efficient architecture search. In *ICLR*, 2018a.

Liu, Hanxiao, Simonyan, Karen, and Yang, Yiming. Darts: Differentiable architecture search. *CoRR*, abs/1806.09055, 2018b.

Loshchilov, Ilya and Hutter, Frank. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017.

Luo, Renqian, Tian, Fei, Qin, Tao, Chen, Enhong, and Liu, Tie-Yan. Neural architecture optimization. In *NIPS*, 2018.

Pham, Hieu, Guan, Melody Y., Zoph, Barret, Le, Quoc V., and Dean, Jeff. Efficient neural architecture search via parameter sharing. In *ICML*, 2018.

Real, Esteban, Moore, Sherry, Selle, Andrew, Saxena, Saurabh, Suematsu, Yutaka Leon, Tan, Jie, Le, Quoc, and Kurakin, Alex. Large-scale evolution of image classifiers. *CoRR*, abs/1703.01041, 2017.

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

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

Zhong, Zhao, Yan, Junjie, Wu, Wei, Shao, Jing, and Liu, Cheng-Lin. Practical block-wise neural network architecture generatio. In *CVPR*, 2018.

Zoph, Barret and Le, Quoc V. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

Zoph, Barret, Vasudevan, Vijay, Shlens, Jonathon, and Le, Quoc V. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.