Efficient Evolution for Neural Architecture Search

Zihao Chen, Bin Li
University of Science and Technology of China, Hefei, China
Email: czhhbp@mail.ustc.edu.cn, binli@ustc.edu.cn

***Abstract*—The intensive consumption of resources by evolutionary algorithms makes it very time-consuming to search for network architectures. In this paper, We proposed a efficient evolution method for neural architecture search. Our method adopts the weight sharing strategy, in which a supernet is built to subsume all architectures, to speed up architecture evaluation. A universal choice strategy is designed to deal with the inaccurate evaluation caused by the methods that speeding up evaluation. Instead of searching for the best architecture,we search for the set of excellent architectures and derive the final architecture fromderive the target architecture according to commonalities of these architectures. The proposed method achieved better results(2.40% test error rate on CIFAR-10 with 3.66M parameters) compared to other the-state-of-art method using less than 0.4 GPU days.**

***Index Terms*—neural architecture search, efficient evolution, weight sharing strategy, inaccurate evaluation**

I. Introduction

Deep learning performs well on many tasks, of which the deep network architecture plays an important role [1]–[6]. Especially in image classification tasks, novel and sophisticated network architectures have been carefully designed by researchers, such as ResNet [4], InceptionNet [7],DenseNet [6], etc. Nevertheless, it is still very difficult to design a better network architecture for existing tasks or to design a suitable network architecture for new tasks. It is therefore not surprising that in recent years, the deep learning community has been working on automation design network architectures, which is known as Neural Architecture Search(NAS) and has achieved very good results [8]–[16].

Evolutionary Algorithm(EA) is one of the earliest methods applied to NAS and is still widely used today [8], [11], [17]–[21]. In the search procedure of EA, each sampled network architecture needs to be trained from scratch and evaluated on specific dataset so as to obtain the fitness. The computation is very resource-intensive and severely limits both of the search scale and evaluation fidelity of EA, which makes EA very difficult to get excellent results. For example, in order to obtain a state-of-the-art network architecture for CIFAR-10 and ImageNet using EA required multiple GPUs and thousands of GPU hours [20]. This kind of phenomenon does not only exist in EA but also in other methods based on reinforcement learning(RL) [9], [13], [22], gradient [23], [24], etc.

When searching network architecture for complex tasks, several method have been proposed to speed up evaluation procedure, such as searching on simpler proxy task [13], [22], [23], training a surrogate to predict weights [25] or performance [26] of each network architecture, forcing all possible network architecture to sharing weights [23], [27]–[29], low-fidelity evaluation [13], [22], [30], [31]. Though these methods improve the efficiency of search, there is a certain gap between the estimated performance and the true performance of the network architecture. Such kind of drawback prevents NAS from getting better results.

In order to make EA search mort efficient, we adopt single path one-shot(SPON) training strategy [27], [29], in which a supernet is defined to subsume all architectures(called paths), to reduce the computation. In the standard SPON, each path in the supernet is randomly sampled to be trained while rest weights not in the path will not be updated, and finally the best architecture is found out using a search algorithm on the trained supernet. We propose three changes to the standard SPON, first, path is no longer randomly sampled from whole supernet but from the population which is controlled by EA and second, architectures are evaluated during the training of supernet, and third, we don't search for the architecture with highest performance in trained supernet. In this way, time spent on evaluation of EA will be greatly reduced since architectures don't need to be trained from scratch for evaluation. Tournament selection evolutionary algorithm is used to evolve network architectures and more details will be described in method section.

Yet despite its efficiency, it prevents search from getting better results. There exists the gap between the accuracies of an architecture using inherited weights of supernet and optimized weights and it makes search a dilemma that the ranking correlation between architectures under one-shot training and the ones under stand-alone training is poor [28]. In addition to one-shot training strategy, other speeding up evaluation strategy referred above contribute to the gap. Searching network architectures according to wrong performance is no longer a good idea. Here we provide a new perspective on NAS. Instead of finding best architecture, it's worth trying to find commonalities of good models. We call it universal choice strategy(UCS), and it will be explained in detail later. Better results are obtained with universal choice strategy compared with searching for best architecture on the same supernet which has been trained in standard one-shot way.

Our approach works well in popular NAS search space [23]. Deterministic path one-shot training strategy allowed EA to search efficiently with a single GTX 1080Ti GPU and within 0.4 days, reducing the number of GPU-hours by more

The work is partially supported by the National Natural Science Foundation of China under grand No.U19B2044 and No.61836011.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

than 6,000x compared to AmobaNet-A [20] which evolves architecture too. In despite of efficiency, EENas products a couple of competitive convolution cell and reduction cell on CIFAR-10 dataset and achieves a test error of 2.40% with 3.66M parameters, compared to the same performance with 27.6M parameters of NASNet-A [22]. Then we transfer the best model on CIFAR-10 to CIFAR-100 dataset and achieved a competitive test error of 15.82% with 3.71M parameters, compared to AmobaNet-B [20] with a test error of 15.8% and 34.9M parameters.

Our contributions can be summarized as follows:

•**Efficiency**: We use weight sharing strategy to speed up the evaluation in EA. The architecture search can be completed during the training phase of the supernet. The time spent on searching on CIFAR-10 data set is less than 0.4 GPU days.

•**Universal Choice Strategy** A new idea is used to search for the population of excellent architectures rather than the best one and finally derive the target architecture according to commonalities of these architectures. This can help us save computation resource and obtain competitive results(2.40% test error on CIFAR-10 with 3.66M parameters).

•**Transferability** We show that the architectures found by EENas on CIFAR-10 is transferable to CIFAR-100.

II. Related Work

We note that EENas is based on NAS method proposed in [20], [23], [29]. These papers will be highlighted.

AmoebaNet [20] adopt tournament selection evolutionary algorithm [33] to search for convolution cells in NAS search space [23] with well designed mutation operator and achieves remarkable results. However the fact that there are 450 GPUs used for 7 days makes the demand for huge computation resources unacceptable. Darts [23] proposed a novel algorithm for differentiable network architecture search. A supernet that subsumes all architecture is built and network weights, togather with architecture parameters, are optimized. Architecture parameters determine the update degree of relative operation's weights. In other words, architecture parameters act as a controller to slow down the updating of parameters for bad operations. Such method has two kinds of drawbacks.First, the fact that all network weights of supernet need to be trained at the same time make it very resource intensive. Second, joint optimization further make architecture parameters and newwork weights coupled which misleads the architecture parameters distrubution since there are bias, caused by the greedy nature of the gradient based metho, in both parameters and weights. The first problem is solved by works [24], [34] via only sampling one or two architectures in supernet to be trained every time. The second problem remains. SPON [29] attempts to reduce the weight coupling in the supernet and then search on the trained supernet via EA. Evaluation of architectures isn't correct since there is gap between the accuracies of architectures under one-shot training and the ones under stand-alone training [28]. The inaccurate evaluation phenomenon not only exists in one-shot but also in other efficient evaluation methods [13], [22], [23], [26]–[31] as mentioned in the previous section. Wrong accuracy means that the optimal architecture obtained by searching may not have the highest accuracy and efficient search methods have to face such problem so as to get better results.

Evolutionary algorithms have been used to optimize the network parameters and hyperparameters of neural networks [18], [21], [35], and now lots of works focusing on NAS has emerged [11], [12], [14], [19], [20], [36], [37]. Multi-objective optimization has become popular in NAS and EA has shown superior performance [14], [19], [37].Among them, CARS [14] is very similar to this paper. The most important difference is that it uses the wrong accuracy evaluated by one-shot training to search for the optimal architecture.

III. Methods

In our approach, a supernet is built to subsume all architectures sharing weights so as to accelerate the search. Apart from this, the universal choice strategy is design to deal with the accuracy gap caused by inaccurate evaluation.

A. Search Space

All experiments adopt the NASNet search space [23], in which normal cell and reduction cell will be searched and then stacked repeatedly for N times(Fig.1, left) to construct a convolutional network. The difference between the two cells is that the normal cell does not downsample the input feature map, while the reduction cell does. This distinction is made by stride. As shown in Fig.1(middle), each cell contains K nodes which are respresented by a number, called NODEID. The first two nodes, respresented by minimum numbers 0 and 1, act as input nodes and receive the output from the previous two cells respectively, and the last node, respresented by maximum numbers K-1, acts as a output node and concatenates outputs of all hidden nodes. Every hidden node contains two elements, and every element consists of a node's number which determines which node's output is used as the hidden node's input and is called INNODE, and the tranform opration type(OP). The following OPs are used: 3x3 max pooling, 3x3 average pooling, skip connection(either identity or reduction connection), 3x3 and 5x5 separable convolution, 3x3 and 5x5 dilated separable convolution. Each separable convolution is always applied twice and each convolutional operation follows the ReLU-Conv-BN order [13], [23]. A valid cell is a directed acyclic graph [23] so that INNODE of one hidden node must smaller than its NODEID. We expanded the search space by allowing both inputs of a hidden node to come from the same one. NAS is equivalent to the discrete combinatorial optimization problem and we need to find the best four choices for every hidden node.

B. Genome Encoding and Mutation Operator

A network architecture can be represented by both normal cell and reduction cell, and the two cells are directly encoded into a single genome in the same scheme as indicated in Fig.1. In the genome, there are two kinds of gene: OP and INNODE. When generating offspring, a mutation operator [20]

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

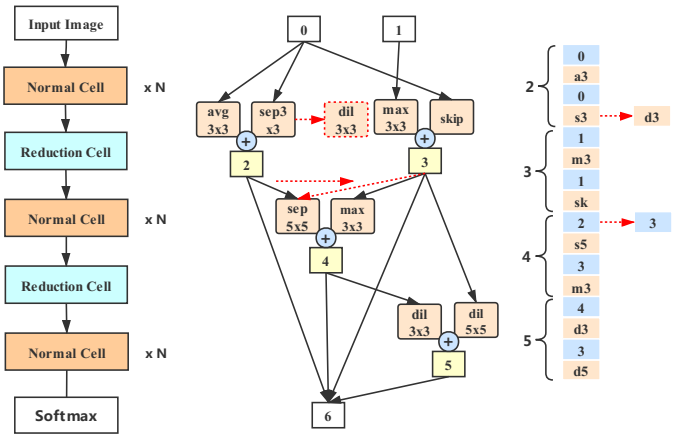


Fig. 1. Illustration of search space, genome encoding and mutation operator.The left figure shows the outer structure of network in search space. The genome (right) defines the cell architecture(middle). In the genome, blue squares represent INNODE gene and yellow squares represent OP gene. Red arrows indicate random changes in architecture

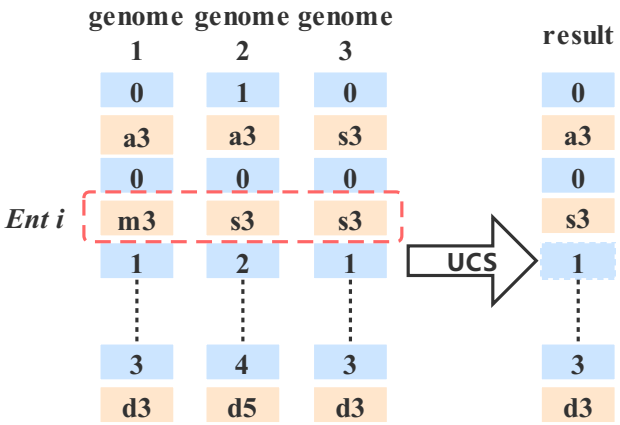


Fig. 2. Illustration of the universal choice strategy.

is adopt and further changed to transform parent architecture in a random way. In each cell genome, we randomly choose one OP gene and one INNODE gene and turn them into another allele according to the description of valid architecture in the previous subsection. The mutation operator is shown by the red dotted arrows in Fig1.

C. Evolutionary Algorithm and Universal Choice Strategy

The framework of EENas is summarized in Algorithm 1. Given a supernet S, it first randomly generate a population P of p different genomes and then update the ppopulation by binary tournament algorithm [38] for C generations. In each generation, two genomes are randomly selected from population and corresponding architectures are trained in strict fairness way [27], in which both architectures are trained alternately for T iterations, and then two architectures are evaluated on randomly sampled B batchs of validation data. This training and evaluation process is more fair compared to the way in [32]. After evaluation, the poor genome will be overwrited by the winning one and then mutates according to the mutation operator decribed above. Obviously, through the time spent on comparisons in tournaments has been greatly reduced, the accuracies of architectures evaluated in above way are inaccurate compared to ones trained from scratch, and the best individual is not known. It is not an efficient and wise approach to determine the best individual by training and evaluating all architectures in the population at last generation from scratch on the whole data set. In order to deal with inaccuracy and derive the final genome, we no longer search for a genome with the highest accuracy but designed the universal choice strategy(UCS) to abtain a universal genome jointly determined by a excellent genomes set.

Fig.2 shows an example to describe UCS. At the beginning, all the genomes in the initial population constitute a excellent genomes set W. At the end of tournament in every generation, UCS adds the winning genome to W or removes the poor genome from W if necessary. A universal genome G can be derived from W. All genes at the same location in all genomes among W make up a gene pool, in which the gene with the most appearing times is selected as the final gene at that location.In this way, a universal genome consists of those final genes can be derived from W. At the same time, in the last $\alpha \times C$ generations, the information entropy E of the genome is calculated and it is equal to the sum of all the information entropies of the gene pool at each location. In other words, UCS records both the information entropy E and the universal genome G in the last $\alpha \times C$ generations. And finally, the universal genome G with the minimum information entropy E is selsected as the final genome.

1) The reason for not choosing the universal genome G at the last generation but determining it based on the information entropy is that the information entropy reflects the diversity of the population to a certain extent. The lower the diversity, the more concentrated the gene distribution is at each location. Population based knowledge, reflecting the universal gene choices of genomes in the set W, helps to derive the final genome or two cell architectures. 2) And the reason for setting a threshold(α) is that it can ensure that the final genome with minimum information entropy appears in the later stages of evolution, and in this stage, W contains a lot of excellent genomes, even through they may not be the top ones in search space.

D. Efficiention Analysis

We discuss the efficiention of EENas. In a tournament, there are two sub-architectures trained alternately for for T iterations and then evaluated on B batchs of validation data. Assuming the average training and inference time for a architecture on one batch of data is T_{tr} and T_{inf}, respectively. Total time for a tournament is $T_{tou} = 2 \times (T \times T_{tr} + B \times T_{inf})$. Compared to tournament time, time spent on calculating the information entropy and deriving the universal genom can be ignored. What's more, when $B \ll T$, $T_{inf} \ll T_{tr}$. Since there are C generations, the total time T_{total} for deriving the final genome is $T_{total} \approx 2 \times C \times T \times T_{tr}$.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

Algorithm 1 Efficent Evolution for Neural Architecture Search(EENas)

Input: S(supernet), C(generation num), T(training iteration num), trainSet, valSet, B(batch num for evaluation).

Initionalize population P and excellent genomes set W;
for generation = 1 to C **do**
  Sample two architectures from P;
  Train two architectures in strict fairness way for T iterations on trainSet;
  Evaluate two architectures on the same B batchs of valSet data;
  Add the winning genome to W or remove the poor genome from W;
  Overwrite the loser genome with winner genome and then mutate it;
  if generation $\geq$ (1-α)$\times C$ **then**
    Record both the universal genome G and the information entropy E of W;
  end if
end for

Output: The universal genome G with the minimum information entropy E.

IV. Experiments

In this section, we test our method on CIFAR-10 image classification task and evaluate the transferability of the discovered architecture on CIFAR-100 dataset. Lastly, we try to verify whether our method is able to deal with the accuracy gap caused by approachs which speed up evaluation procedure. All experiments were performed using NVIDIA GTX 1080Ti GPUs.

A. Results on CIFAR-10 Classification

1) Dataset: We test our method on CIFAR-10 [39] image classification task.There are 50,000 training images and 10,000 testing images in the dataset. The size of images is 32×32 and the number of classes of the dataset is 10. We use 45,000 training images to form the training set, and use the remaining training images to form the validation set. The original testing images are used to form the test set.

2) Seaching and evaluation Details: In searching stage, we build a small supernet with 16 initial channels and set the cell repeatly stack time N to two and the number K of nodes in the cell is set to 7 as done in Darts [23]. The generation C is set to 800 and the threshold α is set to 1/5. In the tournament, the iteration num T for training a architecture is set to 50, and the batch num B for evaluating is set to 4. The batch size is set to 128 both for training and validation sets. We adopt Label Smoothing method [40] to avoid overfitting since small architectures usually abtain better accuracyies than large ones in early training stage and use SGD to optimize the weights of the supernet, with weight deacy 3×10^{-4}, momentum 0.9, initial learning rate 0.025 reduced to zero following cosine annealing schedule.

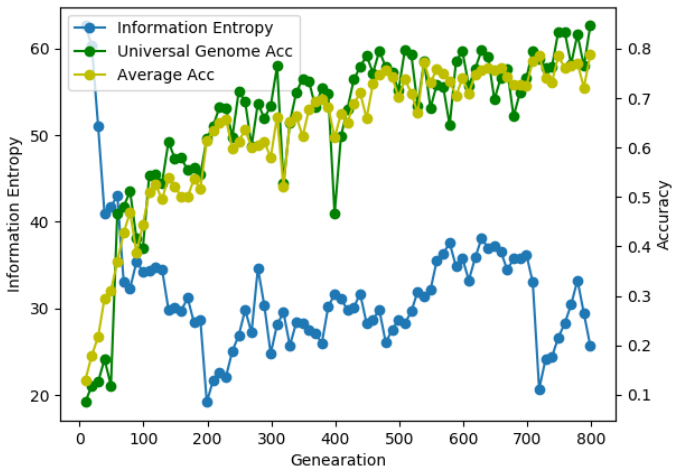


Fig. 3. Generational information entropy and accuracies of the excellent genome set.

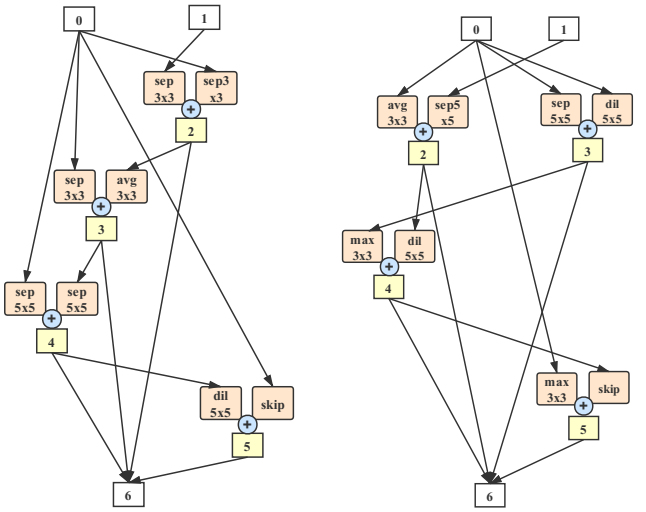
To evalute the final genome, we decode the final genome into cells and expand the stacked architecture by setting N and the number of initial channels to 6 and 36, respectively. The abtained architecture will be trained for 600 epochs from scratch using all training images and test performance on the test set, with batch size of 96. Label Smoothing method is not used again. Other hyperparameters follow the ones used in search stage. For fair comparison with other methods [16], [20], [23], we adopt the enhancements including scheduled path dropout [22], auxiliary towers [5] and cutout [41].

3) Results Analysis: We first represent the generational information entropy and accuracies of the excellent genome set W as indicated in Fig.3. As W is continuously updated in each generation, the information entripy of W drops sharply in the early stages of evolution and has been fluctuating since then. It is not wise to choose the universal genome of W with the smallest information entropy in the entire evolution process, because this may occur early in the evolution. We evaluate the universal genome of W and all genomes in W to abtain the average accuracy of W by evaluating corresponding architectures with weights inherited from supernet on whole validation set. It's obvious that the performance of the universal genome is slightly better than average performance of W.

The final genome is selected according to the universal choice strategy and decoded into both normal cell and reduction cell architecture as shown in Fig.4. We note that the displayed cells are the best of the ten independent search results and the standard variance of results is 0.15.

We compare the searched cells of our method with other state-of-the-art methods which adopt similar search space in Table I. Notely, EENas achieves the better results with model size around 3.7M, compared to those methods with the state of art. Despite competitive results, EENas reduces the number of GPU-hours by more than 6,000x compared to AmobaNet-A [20] which evolves architecture too. According to efficiention analysis and hyperparameters above, the searching time

978-1-7281-6926-2/20/$31.00 ©2020 IEEE



(a) *Normal Cell* (b) *Reduction Cell*

Fig. 4. The searched convolutional cells.

approximately equals to the average time of training a sub-architecture in supernet for about 205 epochs. Our experiments are performed on a single NVIDIA GTX 1080Ti GPU and the searching time is less than 0.4 GPU days.

B. Transferring the discovered architecture to CIFAR-100

We evaluate the transferability of the discovered architecture on CIFAR-100 dataset. Training setup remains the same as those on CIFAR-10 dataset. Results are summaried in Table II.The network architecture stacked by the searched cells abtain better results compared to other state-of-the-art NAS methods with only 3.7M parameters. It suggests that the searched cells are transferable to CIFAR-100.

C. Facing Inaccurate Evaluation

1) Comparison: As discussed in the previous sections, there exists the gap between the accuracies of an architecture using inherited weights of supernet and optimized weights. Apart from weights sharing strategy, other strategies that speed up evaluation can cause such gap. The gap makes evaluation inaccurate. Searching for the best architectures based on inaccurate performance may miss the real one. UCS is designed to deal with inaccurate evaluation. And here we try to verify the validity of UCS.

For the same search space and supernet S as discribed above, we train S following single path one-shot(SPON) training strategy. Then we search for one architecture with highest accuracy following the SPON-EA algorithm [29] and one architecture corresponding to the universal genome following EENas algorithm from the trained supernet S. The evaluation results of two architectures is shown in Table III. The model with small size is abtained by SPON-EA, since small architectures's weights updates faster to a appropriate state during training of supernet than larger ones and generally get better evaluation results [28]. However, our method does not rely on evaluation values and abtain better architecture based on knowledge of population.

2) Why Does Universal Choice Strategy work?: UCS(universal choice strategy) updates a winning genomes set W. At the last stage of evolution, W contains excellent genomes. Each excellent genome owns the knowlodeg for the partial data set that it has experienced. So W owns the knowlodeg for the entire data set and can help to derive a universal genome adapting to the entire data set.

V. Conclusion

We have proposed EENas, an efficient evolutionary algorithm for searching neural network architectures. EENas builds a supernet that subsume all architectures and adopts weight sharing strategy to speed up architectures evaluation. The architecture search can be completed during the training phase of the supernet. Apart from this, we design a universal choice strategy to deal with the incourate evaluation caused by those fast evaluation methods. The universal choice strategy helps to derive a universal architecture based on the knowlodge of a excellent genome set for the entire data set. Experiments on CIFAR-10 and CIFAR-100 show that EENas is able to find the architecture with the state of art in less than 0.4 GPU days on GTX 1080Ti and the transfer ability.

References

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv: Computation and Language*, 2018.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv: Computer Vision and Pattern Recognition*, 2015.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *european conference on computer vision*, pp. 21–37, 2016.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *computer vision and pattern recognition*, Conference Proceedings, pp. 770–778.

[5] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, Conference Proceedings.

[6] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, Conference Proceedings, pp. 4700–4708.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," pp. 1–9, 2015.

[8] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, and N. Duffy, "Evolving deep neural networks."

[9] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," 2017.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," pp. 1097–1105, 2012.

[11] 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*. JMLR. org, Conference Proceedings, pp. 2902–2911.

[12] K. G. Kapanova, I. Dimov, and J. M. Sellier, "A genetic approach to automatic neural network architecture optimization," *Neural Computing and Applications*, vol. 29, no. 5, pp. 1481–1492, 2016.

[13] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

TABLE I
COMPARISON WITH STATE-OF-THE-ART IMAGE CLASSIFIERS ON CIFAR-10.

Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	Search Method
DenseNet-BC [6]	3.46	25.6	-	mannual
ENAS + cutout [16]	2.89	4.6	0.45	RL
NASNet-A + cutout [22]	2.65	3.3	2000	RL
SNAS (moderate) + cutout [42]	2.85	2.8	1.5	Gradient
DARTS (first) + cutout [23]	2.94	2.9	1.5	Gradient
DARTS (second) + cutout [23]	2.83	3.4	4	Gradient
NAONet [43]	3.18	10.6	200	Gradient
NAONet + cutout [43]	2.11	128	200	Gradient
PNAS [44] + cutout	3.41	3.2	225	Gradient
AmoebaNet-A + cutout [20]	3.12	3.1	3150	Evolution
NSGANet + cutout [20]	3.85	3.3	8	Evolution
LEMONADE + cutout [19]	3.05	4.7	80	Evolution
CARS-I + cutout [14]	2.62	3.6	0.4	Evolution
Our	**3.48**	**3.7**	**¡0.4**	**Evolution**
Our+cutout	**2.40**	**3.7**	**¡0.4**	**Evolution**

TABLE II
COMPARISON WITH STATE-OF-THE-ART IMAGE CLASSIFIERS ON CIFAR-100.

Architecture	Test Error (%)	Params (M)	Search Cost (GPU days)	Search Method
ENAS + cutout [16]	17.27	4.6	0.45	RL
NASNet-A + cutout [22]	16.6	3.3	2000	RL
NAONet [43]	15.7	10.8	200	Gradient
NAONet + cutout [43]	14.8	128	200	Gradient
PNAS [44] + cutout	17.6	3.2	225	Gradient
AmoebaNet-B [20]	17.7	34.9	3150	Evolution
AmoebaNet-B + cutout [20]	15.8	34.9	3150	Evolution
NSGANet + cutout [20]	20.7	3.3	8	Evolution
Our	**17.6**	**3.7**	**0.4**	**Evolution**
Our+cutout	**15.8**	**3.7**	**0.4**	**Evolution**

TABLE III
COMPARISON WITH THE METHON SEARCHING FOR ARCHITECTURE WITH HIGHEST ACCURACY ON THE SAME SUPERNET TRAINED IN SINGLE PATH ONE-SHOT WAY.

Architecture	Test Error (%)	Params (M)
SPON-EA [29]	3.6	2.3
Our	3.2	3.2

[14] Z. Yang, Y. Wang, X. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, and C. Xu, "Cars: Continuous evolution for efficient neural architecture search," *arXiv preprint arXiv:1909.04977*, 2019.

[15] D. Stamoulis, R. Ding, D. Wang, D. Lymberopoulos, B. Priyantha, J. Liu, and D. Marculescu, "Single-path nas: Designing hardware-efficient convnets in less than 4 hours," *arXiv preprint arXiv:1904.02877*, 2019.

[16] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *arXiv preprint arXiv:1802.03268*, 2018.

[17] F. Ahmadizar, K. Soltanian, F. AkhlaghianTab, and I. Tsoulos, "Artificial neural network development by means of a novel combination of grammatical evolution and genetic algorithm," *Engineering Applications of Artificial Intelligence*, vol. 39, pp. 1–13, 2015.

[18] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 131–162, 2007.

[19] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," 2019.

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

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

[22] 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*, Conference Proceedings, pp. 8697–8710.

[23] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," 2019.

[24] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Conference Proceedings, pp. 10 734–10 742.

[25] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Smash: One-shot model architecture search through hypernetworks," 2018.

[26] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," *arXiv: Learning*, 2017.

[27] X. Chu, B. Zhang, R. Xu, and J. Li, "Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search," *arXiv: Learning*, 2019.

[28] R. Luo, T. Qin, and E. Chen, "Understanding and improving one-shot neural architecture optimization." *arXiv: Learning*, 2019.

[29] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

path one-shot neural architecture search with uniform sampling," *arXiv preprint arXiv:1904.00420*, 2019.
[30] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast bayesian optimization of machine learning hyperparameters on large datasets," *arXiv: Learning*, 2016.
[31] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of imagenet as an alternative to the cifar datasets." *arXiv: Computer Vision and Pattern Recognition*, 2017.
[32] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *arXiv preprint arXiv:1701.08734*, 2017.
[33] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," vol. 1, pp. 69–93, 1991.
[34] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," *arXiv preprint arXiv:1812.00332*, 2018.
[35] M. Kim and L. Rigazio, "Deep clustered convolutional kernels," pp. 160–172, 2015.
[36] E. Dufourq and B. A. Bassett, "Eden: Evolutionary deep networks for efficient machine learning," *arXiv: Machine Learning*, 2017.
[37] Z. Lu, I. Whalen, V. N. Boddeti, Y. Dhebar, K. Deb, E. D. Goodman, and W. Banzhaf, "Nsga-net: neural architecture search using multi-objective genetic algorithm," pp. 419–427, 2019.
[38] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of genetic algorithms*. Elsevier, 1991, vol. 1, pp. 69–93.
[39] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10," *Unpublished manuscript*, vol. 40, no. 7, pp. 1–9, 2010.
[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
[41] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.
[42] S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: stochastic neural architecture search," in *international conference on learning representations*, Conference Proceedings.
[43] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," in *Advances in neural information processing systems*, Conference Proceedings, pp. 7816–7827.
[44] 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)*, Conference Proceedings, pp. 19–34.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE