EENA: Efficient Evolution of Neural Architecture

Hui Zhu[1,2], Zhulin An[1]*, Chuanguang Yang[1], Kaiqiang Xu[1], Erhu Zhao[1], Yongjun Xu[1]
[1]Institute of Computing Technology, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
{zhuhui,anzhulin,yangchuanguang,xukaiqiang,zhaoerhu,xyj}@ict.ac.cn

Abstract

Latest algorithms for automatic neural architecture search perform remarkable but are basically directionless in search space and computational expensive in the training of every intermediate architecture. In this paper, we propose a method for efficient architecture search called EENA (Efficient Evolution of Neural Architecture). Due to the elaborately designed mutation and crossover operations, the evolution process can be guided by the information have already been learned. Therefore, less computational effort will be required while the searching and training time can be reduced significantly. On CIFAR-10 classification, EENA using minimal computational resources (0.65 GPU−days) can design highly effective neural architecture which achieves 2.56% test error with 8.47M parameters. Furthermore, the best architecture discovered is also transferable for CIFAR-100.

1. Introduction

Convolutional Neural Network has a prominent performance in computer vision, object detection and other fields by extracting features through neural architectures which imitate the mechanism of human brain. Human-designed neural architectures such as ResNet [14], DenseNet [17], PyramidNet [13] and so on which contain several effective blocks are successively proposed to increase the accuracy of image classification. In order to design neural architectures adaptable for various datasets, more researchers have a growing interest in studying the algorithmic solutions based on human experience to achieve automatic neural architecture search [38, 23, 24, 27, 3, 4, 33].

Many architecture search algorithms perform remarkable but demand for lots of computational effort. For example, obtaining a state-of-the-art architecture for CIFAR-10 required 7 days with 450 GPUs of evolutionary algorithm [28] or used 800 GPUs for 28 days of reinforcement learning [38]. The latest algorithms based on reinforcement learning (RL) [27], sequential model-based optimization (SMBO) [22] and Bayesian optimization [19] over a discrete space are proposed to speed up the search process but the basically directionless search leads to a large number of architectures evaluations required. Although several algorithms based on gradient descent over a continuous space, such as DARTS [24] and NAO [26] address this problem to some extent, the training of every intermediate architecture is still computational expensive.

In this work, we propose a method for efficient architecture search called EENA (Efficient Evolution of Neural Architecture) guided by the experience gained in the prior learning to speed up the search process and thus consume less computational effort. The concept, guidance of experience gained, is inspired by Net2Net [5], which generate large networks by transforming small networks via function-preserving. There are several precedents [1, 35, 2] based on this for neural architecture search, but the basic operations are limited to the experience in parameters and are relatively simple, so the algorithms may degenerate into a random search. We absorb more basic blocks of classical networks, discard several ineffective blocks and even extend the guidance of experience gained to the prior architectures by crossover in our method. Due to the loss continue to decrease and the evolution becomes directional, robust and globally optimal models can be discovered rapidly in the search space.

Our experiments (Sect. 4) of neural architecture search on CIFAR-10 show that our method using minimal computational resources (0.65 GPU−days[1]) can design highly effective neural architecture that achieves 2.56% test error with 8.47M parameters. We further transfer the best architecture discovered on CIFAR-10 to CIFAR-100 datasets and the results perform remarkable as well.

Our contributions are summarized as follows:
(1) We are the first to propose the crossover operation guided by experience gained to effectively reuse the prior

*Corresponding author of this work

[1]All of our experiments were performed using a NVIDIA Titan Xp GPU.

learned architectures and parameters.

(2) We study a large number of basic mutation operations absorbed from typical architectures and select the ones that have significant effects.

(3) We achieve remarkable architecture search efficiency (2.56% error on CIFAR-10 in 0.65 GPU-days) which we attribute to the use of EENA.

(4) We show that the neural architectures searched by EENA on CIFAR-10 are transferable for CIFAR-100 datasets.

Part of the code implementation and several models we searched on CIFAR-10 of EENA is available at `https://github.com/ICCV-5-EENA/EENA`.

2. Related Work

In this section, we review human-designed neural architectures and automatic neural architecture search which are most related to this work.

Human-Designed Neural Architectures. Since convolutional neural networks were first used to deal with the problems in the field of computer vision, human-designed neural architectures constantly improve the classification accuracy on specific datasets. As the crucial factor affecting the performance of neural network, many excellent neural architectures have been designed. Chollet *et al*. [6] propose to replace Inception with depthwise separable convolutions to reduce the number of parameters. Grouped convolutions given by Krizhevsky *et al*. [20] is used to distribute the model over two GPUs and Xie *et al*. [36] further propose that increasing cardinality is more effective than going deeper or wider based on this. He *et al*. [14] solve the degradation problem of deep neural networks by residual blocks. Huang *et al*. [17] propose dense blocks to solve the vanishing-gradient problem and substantially reduce the number of parameters. Hu *et al*. [16] propose the Squeeze-and-Excitation block that adaptively recalibrates channel-wise feature responses by explicitly modelling interdependencies between channels. In addition to the improvements on convolutional layer, many other methods have been proposed to further optimize the network. Lin *et al*. [21] utilize global average pooling to replace the traditional fully connected layers to enforce correspondences between feature maps and categories and avoid overfitting. Some regularization methods, such as dropout [30], dropblock [11] and shake-shake [10] and so on, also improve the generalization ability of neural networks. Human-designed neural architectures rely heavily on expert experience, and it will be extremely difficult to design the most suitable neural architecture when faced with a new task for image. In this work, we fully draw on the excellent experience of predecessors and design an efficient neural architecture search method.

Automatic Neural Architecture Search. Many different search strategies have been proposed to explore the space of neural architectures, including random search, evolutionary algorithm (EA), reinforcement learning (RL), Bayesian optimization (BO) and gradient-based methods. Early approaches [32, 31] use evolutionary algorithms to optimize both the neural architecture and its weights. However, faced with the scale of contemporary neural architectures with millions of weights, recent approaches based on evolutionary algorithm have been imporved in some ways, such as using gradient-based methods for optimizing weights [29, 28, 23] or modifying the architecture by network morphisms [34, 5, 2]. The generation of a neural architecture can be regarded as the agent's action, thus, many approaches based on reinforcement learning [38, 39] have been proposed to deal with neural architecture search. Kandasamy *et al*. [19] derive kernel functions for architecture search spaces in order to use classic GP-based BO method, but compared with others, BO method shows no obvious advantages. In contrast to the gradient-free methods above, as gradient-based approaches, Liu *et al*. [24] propose a continuous relaxation of the search space and Luo *et al*. [26] use an encoder network mapping neural network architectures into a continuous space. We can notice that the algorithms of automatic neural architecture search pay more and more attention to the efficiency of algorithms (such as search time) besides focusing on the effect of neural architectures discovered. In this work, we propose a different method based on evolutionary algorithm and network morphisms for efficient architecture search and achieve remarkable results on the classification datasets.

3. Proposed Methods

In this section, we illustrate our basic mutation and crossover operations with an example of several connected layers which come from a simple convolutional neural network and describe the method of selection and discard of individuals from the population in the evolution process.

3.1. Search Space and Mutation Operations

As we mentioned in the Sect. 2, the birth of a better network architecture is usually achieved based on the local improvements. A good design of automatic neural architecture search should be based on a large number of excellent human-designed architctures. In addition, some of the existing methods [5, 35] based on function-preserving are briefly reviewed in this section and our method is built on them. Specifically, we absorb more blocks of classical networks such as dense block, add some effective changes such as noises for new parameters and discard several ineffective operations such as kernel widening in our method.

Our method explores the search space by mutation and crossover operations and every mutation operation refers to

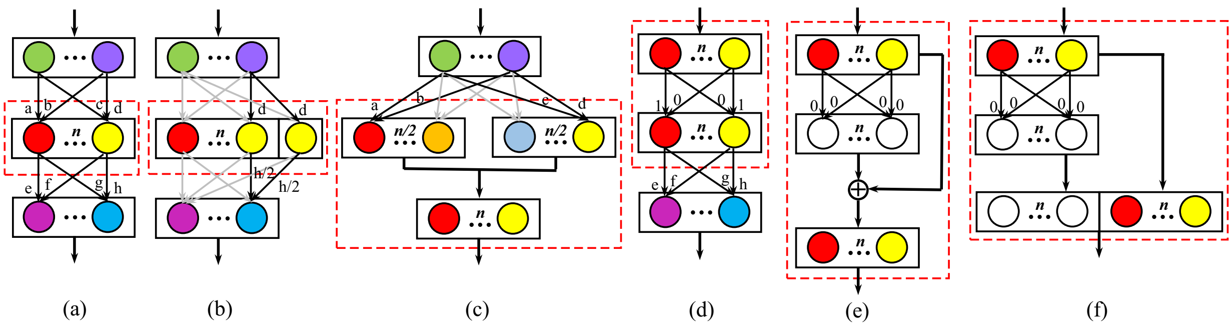

Figure 1. Visualization of the teacher network (a) and several mutation operations (b∼f). The rectangles and circles represent the convolutional layers and feature maps or filters, respectively. The same color means identical and white means 0 value. The parts in the red dashed box are equivalent.

a random change for an individual. x is the input to the network, the guidance of experience gained in parameters is to choose a new set of parameters $\theta^{'}$ for a student network $g(x;\theta^{'})$ which transform from the teacher network $h(x;\theta)$ such that:

$$\forall x : h(x;\theta) = g(x;\theta^{'}). \tag{1}$$ [2]

Assume that the i-th convolutional layer to be changed is represented by a (k_1, k_2, c, f) shaped matrix $W^{(i)}$. The input for the convolution operation in layer i is represented as $X^{(i)}$ and the processing of BatchNorm and ReLU is expressed as φ. In this work, we consider the following mutation operations.

Widen a Layer. Fig. 1(b) is an example of this operation. $W^{(i)}$ is extend by replicating the parameters along the last axis at random and the parameters in $W^{(i+1)}$ need to be divided along the third axis corresponding to the counts of the same filters in the i-th layer. U is the new parameter matrix and $f^{'}$ is the number of filters in the layer i+1. Specifically, A noise δ is randomly added to every new parameter in $W^{(i+1)}$ to break symmetry.

$$g(j) = \begin{cases} j & j \le f \\ random\ sample\ from\{1, 2, \cdots, f\} & j > f \end{cases}, \tag{2}$$

$$U^{(i)}_{k_1,k_2,c,j} = W^{(i)}_{k_1,k_2,c,g_{(j)}}, \tag{3}$$

$$U^{(i+1)}_{k_1,k_2,j,f'} = \frac{W^{(i+1)}_{k_1,k_2,g(j),f'}}{card(x|g(x) = g(j))} \cdot (1+\delta),\ \delta \in [0, 0.05]\,. \tag{4}$$

Branch a Layer. Fig. 1(c) is an example of this operation. This operation adds no further parameters and will always be combined with other operations. U and V are the new parameter matrices which can be expressed as:

$$\begin{aligned} U^{(i)}_{k_1,k_2,c,j} &= W^{(i)}_{k_1,k_2,c,m} \quad m \in \left[0, \lfloor \frac{f}{2} \rfloor\right], \\ V^{(i)}_{k_1,k_2,c,l} &= W^{(i)}_{k_1,k_2,c,m} \quad m \in \left(\lfloor \frac{f}{2} \rfloor, f\right]. \end{aligned} \tag{5}$$

The convolutional layer will be reformulated as:

$$Concat\left(\varphi\left(X^{(i)} \cdot U^{(i)}_{k_1,k_2,c,j}\right), \varphi\left(X^{(i)} \cdot V^{(i)}_{k_1,k_2,c,l}\right)\right). \tag{6}$$

Insert a Single Layer. Fig. 1(d) is an example of this operation. The new layer weight matrix $U^{(i+1)}$ with a $k_1 \times k_2$ kernel is initialized to an identity matrix. $ReLU(x) = max\{x, 0\}$ satisfies the restriction for the activation function σ:

$$\forall x : \sigma(x) = \sigma\left(I\sigma(x)\right), \tag{7}$$

so this operation is possible and the new matrix can be expressed as:

$$U^{(i+1)}_{j,l,a,b} = \begin{cases} 1 & j = \frac{k_1+1}{2} \wedge l = \frac{k_2+1}{2} \wedge a = b \\ 0 & \text{otherwise} \end{cases}. \tag{8}$$

Insert a Layer with Shortcut Connection. Fig. 1(e) is an example of this operation. All the parameters of the new layer weight matrix $U^{(i+1)}$ are initialized to 0. The convolutional layer will be reformulated as:

$$Add\left(\varphi\left(X^{(i+1)}\right), \varphi\left(X^{(i+1)} \cdot U^{(i+1)}\right)\right). \tag{9}$$

Insert a Layer With Dense Connection. Fig. 1(f) is an example of this operation. All the parameters of the new layer weight matrix $U^{(i+1)}$ are initialized to 0. The convolutional layer will be reformulated as:

$$Concat\left(\varphi\left(X^{(i+1)} \cdot U^{(i+1)}\right), \varphi\left(X^{(i+1)}\right)\right). \tag{10}$$

[2] The '=' here doesn't mean completely equivalent, noise may be added to make the student more robust.

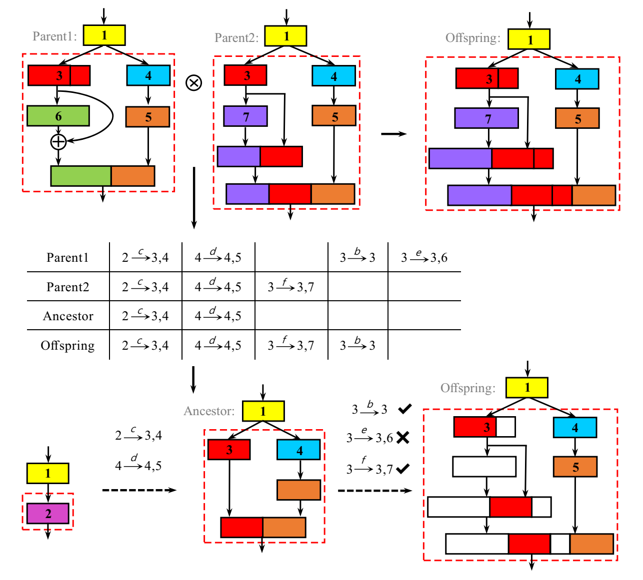

Figure 2. Visualization of the process that the parents produce an offspring by crossover operation. b∼f correspond to the mutation operations in Fig. 1 and 1∼7 represent the serial number of layers. The same colored rectangles represent the identical layers and white means 0 value. The parts in the red dashed box are equivalent.

In addition, many other important methods, such as separable convolution, grouped convolution and bottleneck etc. can be absorbed into the mutation operations. For these, we run several simple tests and notice that the search space is expanded but the accuracy of classification is not improved. Therefore, we finally abandoned these operations in our experiment.

3.2. Crossover Operation

Crossover refers to the combination of the prominent parents to produce offsprings which may perform even more excellent. The parents refer to the architectures with high fitness (accuracy) that have been already discovered and every offspring can be considered as a new exploration of the search space. Obviously, although our mutation operations reduce the computational effort of the repeated retraining, the exploration of the search space is still random without taking advantage of the experience already gained in prior architectures. It is crucial and difficult to find a crossover operation that can effectively reuse the parameters already trained and even produce the next generation guided by experience of the prior excellent architectures.

NEAT [32], as a existing method in the field of evolutionary algorithm, identify which genes line up with which by assigning the innovation number to each node gene. However, this method is limited to the fine-grained crossover for nodes and connections, and will destroy the parameters that have already been trained.

We notice that the architectures with high fitness all derive from the same ancestor of some point in the past (At worst, the ancestor is the initial architecture). Whenever a new architecture appears (through mutation operations), we record the type and the location of the mutation operation. Based on these, we can track the historical origins and find the common ancestor of the two individuals with high fitness. Then the offsprings inherit the same architecture (ancestor) and randomly inherit the different parts of architectures of the parents.

Fig. 2 is a visual example of the crossover operation in our experiments. Based on the records about the previous mutation operations for each individual (for Parent1, mutation c, d, b, e occurred at layer 2, 4, 3, 3, respectively and for Parent2, mutation c, d, f occurred at layer 2, 4, 3, respectively), the common ancestor of the parents (Ancestor with mutation c, d occurred at layer 2, 4) can be easily found. The mutation operations of the two parents different from each other are selected and added to the ancestor architecture according to a certain probability by the mutation operations (mutation b, f occurred at layer 3, 3 are inherited by Offspring and mutation e is randomly discarded).

3.3. The Selection and Discard of Individuals in Evolutionary Algorithm

The Selection of Individuals. Our evolutionary algorithm uses tournament selection [12] to select an individual for mutation: a fraction k of individuals is selected from the population randomly and the individual with highest fitness is final selected from this set. For crossover, the two individuals with the highest fitness but different architectures will be selected.

The Discard of Individuals. In order to constrain the size of the population, the discard of individuals will be accompanied by the generation of each new individual when the population size reaches N. We regulate aging and non-aging evolutions [28] via a variable λ to affect the convergence rate and overfit: Discarding the worst model with probability λ and the oldest model with $1 - \lambda$ within each round.

4. Experiments

In this section, we report the performances of EENA in neural architecture search on CIFAR-10 and the feasibility of transferring the best architecture discovered on CIFAR-10 to CIFAR-100. In addition, we statistically analyze the effect of mutation and crossover operations. In our experiments, we start the evolution from initializing a simple convolutional neural network to show the efficiency of EENA and we use the methods of selection and discard mentioned

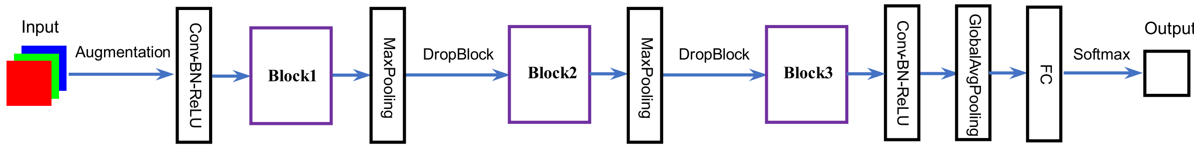

Figure 3. The initial model designed in our experiments.

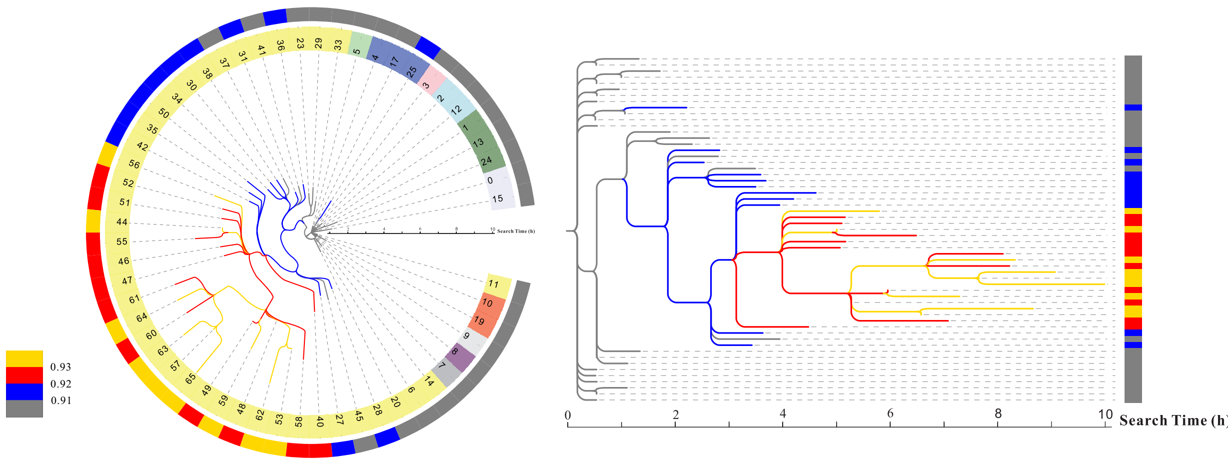

Figure 4. The phylogenetic tree visualized one search process on CIFAR-10. In the circle phylogenetic tree, the color of the outermost circle represents fitness, and the color of the penultimate circle represents ancestor. In the rectangular phylogenetic tree, the color on the right side represents fitness. From the inside to the outside in the left figure and from left to right in the right figure, along the direction of time axis, the connections represent the relationship from ancestors to offsprings.

in Sect. 3.3 to select individuals from the population and the mutation Sect. 3.1 and crossover in Sect. 3.2 operations to improve the neural architectures.

Initial Model. The initial model (the number of parameters is 0.67M) is sketched in Fig. 3. It starts with one convolutional layer, followed by three evolutionary blocks and two MaxPooling layers for down-sampling which are connected alternately. Then another convolutional layer is added, followed by a GlobalAveragePooling layer and a Softmax layer for transformation from feature map to classification. Each MaxPooling layer has a stride of two and is followed by a DropBlock [11] layer with $keep\_prob = 0.8$ ($block\_size = 7$ for the first one and $block\_size = 5$ for the second one). Specifically, the first convolutional layer contains 64 filters and the last convolutional layer contains 256 filters. An evolutionary block is initialized with a convolutional layer with 128 filters. Every convolutional layer mentioned actually means a Conv-BatchNorm-ReLU block with a kernel size of 3×3. The weights are initialized as He normal distribution [14] and the L2 regularization of 0.0001 is applied to the weights.

Dataset. We randomly sample 10,000 images by stratified sampling from the original training set to form a validation set for evaluate the fitness of the individuals while using the remaining 40,000 images for training the individuals during the evolution. We normalize the images using channel means and standard deviations for preprocessing and apply a standard data augmentation scheme (zero-padding with 4 pixels on each side to obtain a 40×40 pixels image, then randomly cropping it to size 32×32 and randomly flipping the image horizontally).

Search on CIFAR-10. The initial population consists of 12 individuals, each formed by a single mutation operation from the common initial model. During the process of evolution, Individual selection is determined by the fitness (accuracy) of the neural architecture evaluated on the validation set. In our experiments, the size of k in selection of individuals is fixed to 3 and the variable λ in discard of individuals is fixed to 0.5. We don't discard any individual at the beginning to make the population grow to the size of 20. Then we use selection and discard together, that is to say, the individual after mutation or crossover operations will be put back into the population after training and at the same time the discard of individuals will be executed. The mutation and crossover operations to improve the neural architectures are applied in the evolutionary block and any mutation operation is selected by the same probability. The crossover operation is executed every 5 rounds, for which we select the two individuals as parents with the highest fit-

Table 1. Comparison against state-of-the-art recognition results on CIFAR-10. Results marked with † are NOT trained with Cutout [8]. The first block represents the performance of human-designed architectures. The second block represents results of various automatically designed architectures. Our method use minimal computational resources to achieve a low test error.

Method	Params (Mil.)	Search Time (GPU-days)	Test Error (%)
DenseNet-BC [17] †	25.6	–	3.46
PyramidNet-Bottleneck [13] †	26.0	–	3.31
ResNeXt + Shake-Shake [10] †	26.2	–	2.86
AmoebaNet-A [28]	3.2	3150	3.34
Large-scale Evolution [29] †	5.4	2600	5.4
NAS-v3 [38]	37.4	1800	3.65
NASNet-A [39]	3.3	1800	2.65
Hierarchical Evolution [23] †	15.7	300	3.75
PNAS [22] †	3.2	225	3.41
Path-Level-EAS [3]	14.3	200	2.30
NAONet [26]	128	200	2.11
EAS [2] †	23.4	10	4.23
DARTS [24]	3.4	4	2.83
Neuro-Cell-based Evolution [35]	7.2	1	3.58
ENAS [27]	4.6	0.45	2.89
NAC [18]	10	0.25	3.33
GDAS(FRC) [9]	2.5	0.17	2.82
Ours	8.47	0.65	2.56
Ours (adjust the number of channels)	54.14	–	2.21

ness but different architectures among the population. All the neural architectures are trained with a batch size of 128 using SGDR [25] with initial learning rate $l_{max} = 0.05$, $T_0 = 1$ and $T_{mult} = 2$. The initial model is trained for 63 epochs. Then, 15 epochs are trained after each mutation operation, one round of 7 epochs and another round of 15 epochs are trained after each crossover operation. One search process on CIFAR-10 is visualized in fig. 4. In the circle phylogenetic tree of EENA, the color of the outermost circle represents fitness, and the same color of the penultimate circle represents the same ancestor. In the rectangular phylogenetic tree, the color on the right side represents fitness. From the inside to the outside in the left figure and from left to right in the right figure, along the direction of time axis, the connections represent the relationship from ancestors to offsprings. We can notice that the fitness of the population increases steadily and rapidly via mutation and crossover operations. In addition, the population is quickly taken over by a highly prominent homologous group. After the search budget is exhausted or the highest fitness of the population no longer increase over 25 rounds, the individual with highest fitness will be extracted as the best neural architecture for post-training.

Post-Training of the Best Neural Architecture Obtained. We conduct post-processing and post-training towards the best neural architecture designed by the EENA. The model is trained on the full training dataset until convergence using Cutout [8] and Mixup [37] whose configurations are the same as the original paper (a cutout size of 16×16 and $\alpha = 1$ for mixup). Specifically, in order to reflect the fairness of the result for comparison, we don't use the latest method AutoAugment [7] which has significant effects but hasn't been widely used yet. The neural architectures are trained with a batch size of 128 using SGD with learning rate $l = 0.1$ for 50 epochs to accelerate the convergence process. Then we used SGDR with initial learning rate $l_{max} = 0.1$, $T_0 = 1$ and $T_{mult} = 2$ for 511 or 1023 epochs[3]. Finally, the error on the test dataset is reported. The comparison against state-of-the-art recognition results on CIFAR-10 is presented in Table 1. On CIFAR-10, Our method using minimal computational resources (0.65 GPU-days) can design highly effective neural cell that achieves 2.56% test error with small number of parameters (8.47M).

Table 2. The effect of mutation for a random selected individual. The first column represents the percentage that the fitness becomes better after mutation. The second column represents the percentage that the fitness becomes worse. The last column represents that the fitness is unchanged.

The Fitness after Mutation for a Random Individual		
Better	**Worse**	**No Change**
118/310 (38.06%)	182/310 (58.71%)	10/310 (3.23%)

Table 3. The comparison of the effect of mutation and crossover. The second column represents the percentage of generating the Top1 fitness of the population from the individuals of Top2 fitness after mutation or crossover. The last column represents the percentage of generating the Top5 fitness of the population from the individuals of Top2 fitness after mutation or crossover.

Operations	**Top1**	**Top5**
Mutation (Top2)	28/120 (23.33%)	79/120 (65.83%)
Crossover (Top2)	18/55 (32.73%)	42/55 (76.36%)

Fine-Tuning of the Best Neural Network. In order to prove the good scalability of the network designed by EENA, we further fine-tune the best neural network by adjusting the number of the channels. We change the number of the filters in the three evolutionary blocks of the convolutional neural network to 1, 2, 4 times and use the same post-processing and post-training method which is mentioned above. The result after fine-tuning on CIFAR-10 is also presented in Table 1. We can notice that a better performance (2.21% test error) can be achieved by simply widening the discovered convolutional neural network.

The Effect of Mutation and Crossover Operations. We conduct several experiments and do a statistical analysis

[3]We did not conduct extensive hyperparameter tuning due to limited computation resources.

Table 4. Comparison against state-of-the-art recognition results on CIFAR-100. The first block represents the performance of human-designed architectures. The second block represents the results of several automatically designed architectures.The last block represents the performance of transferring the best architecture discovered on CIFAR-10 to CIFAR-100.

Method	Params (Mil.)	Search Time (GPU-days)	Test Error (%)
DenseNet-BC [17]	25.6	–	17.18
ResNeXt + Shake-Shake [10]	26.2	–	15.20
AmoebaNet-B [28]	34.9	3150	15.80
Large-scale Evolution [29]	40.4	2600	23.70
NASNet-A [39]	50.9	1800	16.03
PNAS [22]	3.2	225	17.63
NAONet [26]	128	200	14.75
Neuro-Cell-based Evolution [35]	5.3	1	21.74
GDAS(FRC) [9]	2.5	0.17	18.13
Ours (transferred from CIFAR-10)	8.49	–	17.71

about the changes of the fitness for an individual after mutation and crossover operations. We collect the results of 310 random individuals for mutation and count the number of the offsprings with better fitness or worse fitness separately. Then we roughly estimate the effect of this operation according to the statistical results. The effect of mutation for a random selected individual is presented in Table 2. We also collect the results of 120 individuals with Top2 fitness for mutation and 55 pairs of individuals with Top2 fitness for crossover. Then we count the number of the offsprings with Top1 and Top5 fitness in the population separately. The comparison of the effect of mutation and crossover is presented in Table 3. We can notice that the mutation and crossover are both effective but crossover operation is more likely to produce individuals with high fitness (Top1 or Top5 fitness).

Comparison to Search Without Crossover. Unlike random search by mutation operations, crossover as a heuristic search makes the exploration directional. In order to further verify the effect of the crossover operation, we conduct another experiment removing the crossover operation from the search process and all the other configurations remain unchanged. We run the experiment 5 times for 0.65 hours, then report a mean classification error of 3.44% and a best classification error of 2.96%. The result is worse than that of the original experiment with crossover. Thus, associating this result with the effect of mutation and crossover operations which is mentioned above, we confirm that the crossover operation is indeed effective.

Transfer the Best Architecture Searched on CIFAR-10 to CIFAR-100. We further try to transfer the best architecture of highest fitness searched on CIFAR-10 to CIFAR-100 and the results perform remarkable as well. For CIFAR-100, several hyper-parameters are modified: $block\_size = 3$ for the first DropBlock layer, $block\_size = 2$ for the second and the cutout size is 8×8. The comparison against state-of-the-art recognition results on CIFAR-100 is presented in Table 4.

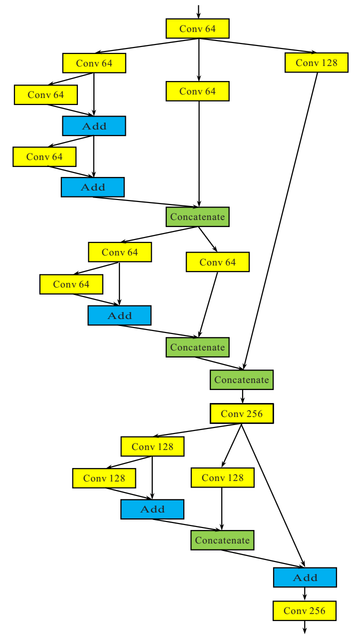

Figure 5. The best architecture discovered by EENA. 64, 128 and 256 is the number of filters.

5. Conclusions and Ongoing Work

We design an efficient method of neural architecture search based on evolution with the guidance of experience gained in the prior learning. This method takes repeatable CNN blocks (cells) as the basic units for evolution, and achieves a state-of-the-art accuracy on CIFAR-10 and others with few parameters and little search time. We notice that the initial model and the basic operations are extremely impactful to search speed and final accuracy. Therefore, we are trying to add several effective blocks such as SE block [16] as mutation operations combined with other methods that might perform effective such as macro-search [15] into our experiments.

Appendix

Here we plot the best architecture of CNN cells discovered by EENA in Fig. 5.

References

[1] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang. Reinforcement learning for architecture search by network transformation. *CoRR*, abs/1707.04873, 2017.

[2] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang. Efficient architecture search by network transformation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2787–2794, 2018.

[3] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu. Path-level network transformation for efficient architecture search. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 677–686, 2018.

[4] H. Cai, L. Zhu, and S. Han. Proxylessnas: Direct neural architecture search on target task and hardware. *CoRR*, abs/1812.00332, 2018.

[5] T. Chen, I. J. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[6] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1800–1807, 2017.

[7] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018.

[8] T. Devries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.

[9] X. Dong and Y. Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1761–1770, 2019.

[10] X. Gastaldi. Shake-shake regularization. *CoRR*, abs/1705.07485, 2017.

[11] G. Ghiasi, T. Lin, and Q. V. Le. Dropblock: A regularization method for convolutional networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 10750–10760, 2018.

[12] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Proceedings of the First Workshop on Foundations of Genetic Algorithms. Bloomington Campus, Indiana, USA, July 15-18 1990.*, pages 69–93, 1990.

[13] D. Han, J. Kim, and J. Kim. Deep pyramidal residual networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6307–6315, 2017.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016.

[15] H. Hu, J. Langford, R. Caruana, E. Horvitz, and D. D. and. Macro neural architecture search revisited. 2018.

[16] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 7132–7141, 2018.

[17] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269, 2017.

[18] P. Kamath, A. Singh, and D. Dutta. Neural architecture construction using envelopenets. *CoRR*, abs/1803.06744, 2018.

[19] K. Kandasamy, W. Neiswanger, J. Schneider, B. Póczos, and E. P. Xing. Neural architecture search with bayesian optimisation and optimal transport. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 2020–2029, 2018.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1106–1114, 2012.

[21] M. Lin, Q. Chen, and S. Yan. Network in network. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[22] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L. Li, L. Fei-Fei, A. L. Yuille, J. Huang, and K. Murphy. Progressive neural architecture search. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part I*, pages 19–35, 2018.

[23] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu. Hierarchical representations for efficient architecture search. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[24] H. Liu, K. Simonyan, and Y. Yang. DARTS: differentiable architecture search. *CoRR*, abs/1806.09055, 2018.

[25] I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[26] R. Luo, F. Tian, T. Qin, E. Chen, and T. Liu. Neural architecture optimization. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 7827–7838, 2018.

[27] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient neural architecture search via parameter sharing. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 4092–4101, 2018.

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

[29] 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, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2902–2911, 2017.

[30] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[31] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009.

[32] K. O. Stanley and R. Miikkulainen. Efficient reinforcement learning through evolving neural network topologies. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, 9-13 July 2002*, pages 569–577, 2002.

[33] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 6105–6114, 2019.

[34] T. Wei, C. Wang, Y. Rui, and C. W. Chen. Network morphism. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 564–572, 2016.

[35] M. Wistuba. Deep learning architecture search by neurocell-based evolution with function-preserving mutations. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part II*, pages 243–258, 2018.

[36] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5987–5995, 2017.

[37] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[38] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[39] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8697–8710, 2018.