EIGEN: Ecologically-Inspired GENetic Approach for Neural Network Structure Searching from Scratch

Jian Ren¹, Zhe Li², Jianchao Yang³, Ning Xu⁴, Tianbao Yang², David J. Foran¹
¹Rutgers University  ²The University of Iowa  ³Bytedance Al Lab  ⁴Amazon Go

Abstract

Designing the structure of neural networks is considered one of the most challenging tasks in deep learning, especially when there is few prior knowledge about the task domain. In this paper, we propose an Ecologically-Inspired GENetic (EIGEN) approach that uses the concept of succession, extinction, mimicry, and gene duplication to search neural network structure from scratch with poorly initialized simple network and few constraints forced during the evolution, as we assume no prior knowledge about the task domain. Specifically, we first use primary succession to rapidly evolve a population of poorly initialized neural network structures into a more diverse population, followed by a secondary succession stage for fine-grained searching based on the networks from the primary succession. Extinction is applied in both stages to reduce computational cost. Mimicry is employed during the entire evolution process to help the inferior networks imitate the behavior of a superior network and gene duplication is utilized to duplicate the learned blocks of novel structures, both of which help to find better network structures. Experimental results show that our proposed approach can achieve similar or better performance compared to the existing genetic approaches with dramatically reduced computational cost. For example, the network discovered by our approach on CIFAR-100 dataset achieves 78.1% test accuracy under 120 GPU hours, compared to 77.0% test accuracy in more than 65, 536 GPU hours in [36].

1. Introduction

Deep Convolutional Neural Networks (CNN) have achieved tremendous success among many computer vision tasks [14, 25, 39]. However, a hand-crafted network structure tailored to one task may perform poorly on another task. Therefore, it usually requires extensive amount of human efforts to design an appropriate network structure for a certain task.

Recently, there are emerging research works [2, 3, 6, 22, 32, 51] on automatically searching neural network structures for image recognition tasks. In this paper, we focus on optimizing the evolution-based algorithms [30, 33, 42, 44] for searching networks from scratch with poorly-initialized networks, such as a network with one global pooling layer, and with few constraints forced during the evolution [36] as we assume no prior knowledge about the task domain. Existing work along this line of research suffers from either prohibitive computational cost or unsatisfied performance compared with hand-crafted network structures. In [36], it costs more than 256 hours on 250 GPU for searching neural network structures, which is not affordable for general users. In [44], the final learned network structure by their genetic approach achieves about 77% test accuracy on CIFAR-10, even though better performance as 92.9% could be obtained after fine-tuning certain parameters and modifying some structures on the discovered network. In [27], they firstly aim to achieve better performance with the reduced computational cost by the proposed aggressive selection strategy in genetic approach and more mutations operation to increase diversity which is decreased by the proposed selection strategy. In their work, they reduce computational cost dramatically from more than 64, 000 GPU hours (GPUH) to few hundreds GPUH. However, their approach still suffers performance sacrifice, for example, 90.5% test accuracy compared to 94.6% test accuracy from [36] on CIFAR-10 dataset.

Inspired by a few key concepts in ecological system, in this paper, we try to improve the genetic approach to achieve better test performance compared to [36] or competitive performance to hand-crafted network structures [18] under limited computation cost [27], but without utilizing pre-designed architectures [29, 30, 31, 51]. Inspired by primary, secondary succession from ecological system [38], we enforce a poorly initialized population of neural network structures to rapidly evolve to a population containing network structures with dramatically improved performance. After the first stage of primary succession, we perform fine-grained search for better networks in a population during the secondary succession stage. During the succession stages, we also introduce an accelerated extinction algorithm to improve the search efficiency. In our approach, we apply the
mimicry [16] concept to help inferior networks learn the behavior from superior networks to obtain the better performance. In addition, we also introduce the gene duplication concept to further utilize the novel block of layers that appear in the discovered network structure.

The contribution of this paper is four-fold and can be summarized as follows:

- We proposed an efficient genetic approach to search neural network structure from scratch with poorly initialized network and without limiting the searching space. Our approach can greatly reduce the computation cost compared to other genetic approaches, where neural network structures are searched from scratch. This is different from some recent works [13, 31, 34] that significantly restricts the search space.

- We incorporate primary and secondary succession concepts from ecological system into our genetic framework to search for optimal network structures under limited computation cost.

- We explore the mimicry concept from ecological system to help search better networks during the evolution and use the gene duplication concept to utilize the discovered beneficial structures.

- Experimental results show that the obtained neural network structures achieves better performance compared with existing genetic-based approaches and competitive performance with the hand-crafted network structures.

2. Related Work

There is growing interest on automatic searching of neural network architectures from scratch. Methods based on reinforcement learning (RL) show promising results on obtaining the networks with the performance similar or better than human designed architectures [3, 48, 50]. Zoph et al. propose to searching in cells, including a normal cell and a reduction cell, where the final architecture is based on stacking the cells [51]. The idea of cell based searching is widely adopted in many studies [9, 10, 29, 31, 34, 49]. In order to reduce high computational cost, efforts have been done to avoid training all networks during the searching process from scratch [4, 5, 7, 9, 11, 13, 23, 47]. However, these works require strict hand-designed constraints to reduce computation cost, and comparison with them are not the focus of this paper.

On the other hand, there emerges a few studies [36, 43, 44] targeting on network searching using evolutionary approaches. In order to have a fair comparison with the RL and evolutionary based approaches, Real et al. [35] conduct the study where the RL and evolutionary approaches are performed under the same searching space. Experiments show the evolutionary approach converges faster than RL.

Therefore, in this paper we focus on the genetic-based approaches for searching optimal neural network structures. Suganuma et al. propose the network searching based on Cartesian genetic programming [17]. However, a pre-defined grid with the fixed row and column is used as the network has to fit in the grid [43]. The studies that have the searching space similar to us are introduced in [27, 36, 44], where the network searching starts from poorly-initialized networks and uses few constraints during the evolution. Since in this paper we focus on achieving better performance with limited computational cost through a genetic approach, we will highlight the differences between our work with the similar studies [27, 36, 44] in the following from two aspects: reducing computation cost and improving performance.

In [36], the authors encode each individual network structure as a graph into DNA and define several different mutation operations such as IDENTITY and RESET-WEIGHTS to apply to each parent network to generate children networks. The essential part of this genetic approach is that they utilize a large amount of computation to search the optimal neural network structures in a huge searching space. Specifically, the entire searching procedure costs more than 256 hours with 250 GPUs to achieve 94.6% test accuracy from the learned network structure on CIFAR-10 dataset, which is not affordable for general users.

Due to prohibitive computation cost, in [44] the authors impose restriction on the neural network searching space. In their work, they only learn one block of network structure and stack the learned block by certain times in a designed routine to obtain the best network structure. Through this mechanism, the computation cost is reduced to several hundreds GPU hours, however, the test performance of the obtained network structure is not satisfactory, for example, the found network achieves 77% test accuracy on CIFAR-10, even though fine-tuning parameters and modifying certain structures on the learned network structure could lead to the test accuracy as 92.9%.

In [27], they aim to achieve better performance from automatically learned network structure with limited computation cost in the course of evolution, which is not brought up previously. Different from restricting the search space to reduce computational cost [12, 44], they propose the aggressive selection strategy to eliminate the weak neural network structures in the early stage. However, this aggressive selection strategy may decrease the diversity which is the nature of genetic approach to improve performance. In order to remedy this issue, they define more mutation operations such as add_fully_connected or add_pooling. Finally, they reduce computation cost dramatically to 72 GPUH on CIFAR-10. However, there is still performance loss in their
approach. For example, on CIFAR-10 dataset, the test accuracy of the found network is about 4% lower than [36].

At the end of this section, we highlight that our work is in the line of [27]. Inspired from ecological concepts, we propose the Ecologically-Inspired GENetic approach (EIGEN) for neural network structure search by evolving the networks through rapid succession, and explore the mimicry and gene duplication along the evolution.

3. Approach

Our genetic approach for searching the optimal neural network structures follows the standard procedures: i) initialize population in the first generation with simple network structures; ii) evaluate the fitness score of each neural network structure (fitness score is the measurement defined by users for their purpose such as validation accuracy, number of parameters in network structure, number of FLOP in inference stage, and so on); iii) apply a selection strategy to decide the surviving network structures based on the fitness scores; iv) apply mutation operations on the survived parent network structures to create the children networks for next generation. The last three steps are repeated until the convergence of the fitness scores. Note that in our genetic approach, the individual is denoted by an acyclic graph with each node representing a certain layer such as convolution, pooling and concatenation layer. A children network can be generated from a parent network through a mutation procedure. A population includes a fixed number of networks in each generation, which is set as 10 in our experiments. For details of using genetic approach to search neural network structures, we refer the readers to [27]. In the following, we apply the ecological concepts of succession, extinction, mimicry and gene duplication to the genetic approach for an accelerated search of neural network structures.

3.1. Evolution under Rapid Succession

Our inspiration comes from the fact that in an ecological system, the population is dominated by diversified fast-growing individuals during the primary succession, while in the secondary succession, the population is dominated by more competitive individuals [38]. Therefore, we treat all the networks during each generation of the evolution process as a population and focus on evolving the population instead of on a single network [36].

With this treatment, we propose a two-stage rapid succession for accelerated evolution, analogous to the ecological succession. The proposed rapid succession includes a primary succession, where it starts with a community consisting of a group of poorly initialized individuals which only contains one global pooling layer, and a secondary succession which starts after the primary succession. In the primary succession, a large search space is explored to allow the community grow at a fast speed, and a relatively small search space is used in the secondary succession for fine-grained search.

In order to depict how the search space is explored, we define mutation step-size \( m \) as the maximum mutation iterations between the parent and children. The actual mutation step for each child is uniformly chosen from \([1, m]\). In the primary succession, in order to have diversified fast-growing individuals, a large mutation step-size is used in each generation so the mutated children could be significantly different from each other and from their parent. Since we only go through the training procedure after finishing the entire mutation steps, the computation cost for each generation will not increase with the larger step-size. In the secondary succession, we adopt a relative small mutation step-size to perform a fine-grained search for network structures.

Each mutation step is randomly selected from the nine following operations including:

- **INSERT-CONVOLUTION**: A convolutional layer is randomly inserted into the network. The inserted convolutional layer has a default setting with kernel size as \(3 \times 3\), number of channels as 32, and stride as 1. The convolutional layer is followed by batch normalization [21] and Rectified Linear Units [25].

- **INSERT-CONCATENATION**: A concatenation layer is randomly inserted into the network where two bottom layers share the same size of feature maps.

- **INSERT-POOLING**: A pooling layer is randomly inserted into the network with kernel size as \(2 \times 2\) and stride as 2.

- **REMOVE-CONVOLUTION**: The operation randomly remove a convolutional layer.

- **REMOVE-CONCATENATION**: The operation randomly remove a concatenation layer.

- **REMOVE-POOLING**: The operation randomly remove a pooling layer.

- **ALTER-NUMBER-OF-Channels, ALTER-STRIDE, ALTER-FILTER-SIZE**: The three operations modify the hyper-parameters in the convolutional layer. The number of channels is randomly selected from a list of \{16, 32, 48, 64, 96\}; the stride is randomly selected from a list of \{1, 2\}; and the filter size is randomly selected from \{1 \times 1, 3 \times 3\}.

During the succession, we employ the idea from previous work [27] that only the best individual in the previous generation will survive. However, instead of evaluating the population in each generation after all the training iterations, it is more efficient to extinguish the individuals that may possibly fail at early iterations, especially during
the primary succession where the diversity in the population leads to erratic performances. Based on the assumption that a better network should have better fitness score at earlier training stages, we design our extinction algorithm as follows.

To facilitate the presentation, we denote \( n \) as the population size in each generation, \( T_1 \) and \( T_2 \) as the landmark iterations, \( f_{g,i,T_1} \) and \( f_{g,i,T_2} \) as fitness scores (validation accuracy used in our work) of the \( i \)th network in the \( g \)th generation after training \( T_1 \) and \( T_2 \) iterations, \( v_{g,T_1} \) and \( v_{g,T_2} \) as threshold to eliminate weaker networks at \( T_1 \) and \( T_2 \) iterations in the \( g \)th generation. In the \( g \)th generation, we have fitness scores for all networks \( \mathcal{F}_{g,T_1} = \{f_{g,i,T_1}, i = 1, \ldots, n\} \) and \( \mathcal{F}_{g,T_2} = \{f_{g,i,T_2}, i = 1, \ldots, \hat{n}\} \) after training \( T_1 \) and \( T_2 \) iterations, respectively. Note that \( \hat{n} \) can be less than \( n \) since weaker networks are eliminated after \( T_1 \) iterations. The thresholds \( v_{g,T_1} \) and \( v_{g,T_2} \) are updated at \( g \)th iteration as

\[
v_{g,T_1} = \max\left( S(\mathcal{F}_{g,T_1}), p, v_{g-1,T_1} \right)
\]

and

\[
v_{g,T_2} = \max\left( S(\mathcal{F}_{g,T_2}), q, v_{g-1,T_2} \right)
\]

where \( S(\cdot) \) is a sorting operator in decreasing order on a list of values and the subscripts \( p \) and \( q \) represents \( p \)th and \( q \)th value after the sorting operation, \( p \) and \( q \) are the hyperparameters.

For each generation, we perform the following steps until the convergence of the fitness scores: (i) train the population for \( T_1 \) iterations, extinguish the individuals with fitness scores less than \( v_{g,T_1} \); (ii) train the remaining population for \( T_2 \) iterations, and distinguish the population with fitness scores less than \( v_{g,T_2} \); (iii) the survived individuals are further trained till convergence and the best one is chosen as the parent for next generation. The details for the extinction algorithm are described in Algorithm 1.

### 3.2. Mimicry

In biology evolution, mimicry is a phenomenon that one species learn behaviours from another species. For example, moth caterpillars learn to imitate body movements of a snake so that they could scare off predators that are usually prey items for snakes [16]. The analogy with mimicry signifies that we could force inferior networks to adopt (learn) the behaviors, such as statistics of feature maps [37, 45] or logits [8, 19], from superior networks in designing neural network structure during the evolution.

In our approach, we force the inferior networks to learn the behavior of a superior network by generating similar distribution of logits in the evolution procedure. Since learning the distribution of logits from the superior network gives more freedom for inferior network structure, compared to learning statistics of feature maps. This is in fact the knowledge distillation proposed in [19]. More specifically, for the given training image \( x \) with one-hot class label \( y \), we define \( t \) as the logits predicted from the pre-trained superior network, and \( s \) as the logits predicted by the inferior network. We use the following defined \( \mathcal{L}_K \) as the loss function to encode the prediction discrepancy between inferior and superior networks as well as the difference between inferior networks prediction and ground truth annotations during the evolution:

\[
\mathcal{L}_K = (1 - \alpha)\mathcal{L}_C(y, \mathcal{H}(s)) + \alpha T^2 \mathcal{L}_C \left( \mathcal{H} \left( \frac{s}{T} \right), \mathcal{H} \left( \frac{t}{T} \right) \right)
\]

![Figure 1: Example of duplication. The image on the left shows the structure discovered after the rapid succession, where each block includes a number of layers with the same size of feature maps. The image in the middle and right are two examples of the duplication that the Block 2 undergoes different combination to create new architectures.](image-url)
<table>
<thead>
<tr>
<th>Model</th>
<th>PARAMS.</th>
<th>C10+</th>
<th>C100+</th>
<th>Comp Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAXOUT [15]</td>
<td>-</td>
<td>90.7%</td>
<td>61.4%</td>
<td>-</td>
</tr>
<tr>
<td>Network In Network [28]</td>
<td>-</td>
<td>91.2%</td>
<td>64.3%</td>
<td>-</td>
</tr>
<tr>
<td>ALL-CNN [40]</td>
<td>1.3 M</td>
<td>92.8%</td>
<td>66.3%</td>
<td>-</td>
</tr>
<tr>
<td>DEEPLY SUPERVISED [26]</td>
<td>-</td>
<td>92.0%</td>
<td>65.4%</td>
<td>-</td>
</tr>
<tr>
<td>HIGHWAY [41]</td>
<td>2.3 M</td>
<td>92.3%</td>
<td>67.6%</td>
<td>-</td>
</tr>
<tr>
<td>RESNET [18]</td>
<td>1.7 M</td>
<td>93.4%</td>
<td>72.8%</td>
<td>-</td>
</tr>
<tr>
<td>DENSENET</td>
<td>25.6 M</td>
<td>96.5%</td>
<td>82.8%</td>
<td>-</td>
</tr>
<tr>
<td>Teacher Network</td>
<td>17.2 M</td>
<td>96.0%</td>
<td>82.0%</td>
<td>-</td>
</tr>
<tr>
<td>GENETIC CNN [44]</td>
<td>-</td>
<td>92.9%</td>
<td>71.0%</td>
<td>408 GPUH</td>
</tr>
<tr>
<td>LS-Evolution [36]</td>
<td>5.4 M</td>
<td>94.6%</td>
<td>-</td>
<td>64,000 GPUH</td>
</tr>
<tr>
<td>LS-Evolution [36]</td>
<td>40.4 M</td>
<td>-</td>
<td>77.0%</td>
<td>65,536 GPUH</td>
</tr>
<tr>
<td>AG-Evolution [27]</td>
<td>-</td>
<td>90.5%</td>
<td>-</td>
<td>72 GPUH</td>
</tr>
<tr>
<td>AG-Evolution [27]</td>
<td>-</td>
<td>-</td>
<td>66.9%</td>
<td>136 GPUH</td>
</tr>
<tr>
<td>EIGEN</td>
<td>2.6 M</td>
<td>94.6%</td>
<td>-</td>
<td>48 GPUH</td>
</tr>
<tr>
<td>EIGEN</td>
<td>11.8 M</td>
<td>-</td>
<td>78.1%</td>
<td>120 GPUH</td>
</tr>
</tbody>
</table>

Table 1: Comparison with hand-designed architectures and automatically discovered architectures using genetic algorithms. The C10+ and C100+ columns indicate the test accuracy achieved on data-augmented CIFAR-10 and CIFAR-100 datasets, respectively. The PARAMS. column indicates the number of parameters in the discovered network.

where $\mathcal{H}(.)$ is the softmax function, $\mathcal{L}_C$ is the cross-entropy of two input probability vectors such that

$$
\mathcal{L}_C(y, \mathcal{H}(s)) = - \sum_k y_k \log \mathcal{H}(s_k), \quad (4)
$$

$\alpha$ is the ratio controlling two loss terms and $T$ is a hyper-parameter. We adopt the terms from knowledge distillation [19] where student network and teacher network represent the inferior network and superior network, respectively. We fix $T$ as a constant. While the target of neural network search is to find the optimal architecture, mimicry is particularly useful when we want to find a small network for applications where inference computation cost is limited.

### 3.3. Gene Duplication

During the primary succession, the rapid changing of network architectures leads to the novel beneficial structures decoded in DNA [36] that are not shown in the previous hand-designed networks. To further leverage the automatically discovered structures, we propose an additional mutation operation named duplication to simulate the process of gene duplication since it has been proved as an important mechanism for obtaining new genes and could lead to evolutionary innovation [46]. In our implementation, we treat the encoded DNA as a combination of blocks. For each layer with the activation map defined as $N \times D \times W \times H$, where $N, D, W, H$ denote the batch size, depth, width and height, respectively, the block includes the layers with activation map that have the same $W$ and $H$. As shown in Figure 1, the optimal structure discovered from the rapid succession could mutate into different networks by combining the blocks in several ways through the duplication. We duplicate the entire block instead of single layer because the block contains the beneficial structures discovered automatically while simple layer copying is already an operation in the succession.

### 4. Experimental Results and Analysis

In this section, we report the experimental results of using EIGEN for structure search of neural networks. We first describe the experiment setup including datasets preprocessing and training strategy in Subsection 4.1 and show the comparison results in Subsection 4.2. Following that, we analyze the experimental results in Subsection 4.3 with regard to each component of our approach.

#### 4.1. Experiment Setup

**Datasets.** The experiments are conducted on two benchmark datasets including CIFAR-10 [24] and CIFAR-100 [24]. The CIFAR-10 dataset contains 10 classes with 50,000 training images and 10,000 test images. The images have the size of $32 \times 32$. The data augmentation is applied by a Global Contrast Normalization (GCN) and ZCA whitening [15]. The CIFAR-100 dataset is similar to...
The discovered network architecture using the proposed method on CIFAR-10 dataset that includes convolutional layers, concatenation layers and global pooling layer.

(a) The discovered network architecture using the proposed method on CIFAR-10 dataset that includes convolutional layers, concatenation layers and global pooling layer.

(b) The detailed architecture of the BLOCK shown in (a).

Figure 2: Discovered neural network structure for CIFAR-10 dataset.

CIFAR-10 except it includes 100 classes.

Training Strategy and Details. During the training process, we use mini-batch Stochastic Gradient Descent (SGD) to train each individual network with the batch size as 128, momentum as 0.9, and weight-decay as 0.0005. Each network is trained for a maximum of 25,000 iterations. The initial learning rate is 0.1 and is set as 0.01 and 0.001 at 15,000 iterations and 20,000 iterations, respectively. The parameters in Algorithm 1 are set to $T_1 = 5,000$, $T_2 = 15,000$, $T_3 = 5,000$, $p = 5$, and $q = 2$. For the mimicry, we set $T$ to 5 and $\alpha$ to 0.9 in Eq. 3. The teacher network is an ensemble of four Wide-DenseNet ($k = 60, l = 40$) [20]. The fitness score is validation accuracy from validation set. The primary succession ends when the fitness score saturates and then the secondary succession starts. The entire evolution procedure is terminated when the fitness score converges. Training is conducted with TensorFlow [1].

We directly adopt the hyper-parameters developed on CIFAR-10 dataset to CIFAR-100 dataset. The experiments are run on a machine that has one Intel Xeon E5-2680 v4 2.40GHz CPU and one Nvidia Tesla P100 GPU.

4.2. Comparison Results

The experimental results shown in Table 1 justify the proposed approach are competitive with hand designed networks. Compared with the evolution-based algorithms, we can achieve the best results with the minimum computational cost. For example, we obtain similar results on the two benchmark datasets compared to [36], but our approach is 1,000 times faster. Also, the number of parameters of the networks found by our approach on the two datasets are more than two times smaller than LS-Evolution [36].

We show the discovered network architecture using our proposed method on CIFAR-10 dataset in Figure 2, where Figure 2a shows the engine network and Figure 2b represents the detailed architecture in the BLOCK of Figure 2a.

4.3. Analysis

Effect of Primary Succession. We show the results on different mutation step-size for the primary succession in Figure 3. The solid lines show average test accuracy of the best networks among five experiments and the shaded area represents the standard deviation $\sigma$ in each generation among five experiments. Larger mutation step-size, such as 100, leads to the faster convergence of fitness score com-
(a) Primary succession using the mutation step-size as 1, 2, 10 and 100.

(b) Primary succession using the mutation step-size as 50, 100 and 200.

Figure 3: The effect of different mutation step-size for the primary succession on CIF AR-10. The solid lines show the average test accuracy over five experiments of the individuals with the highest accuracy in each generation. The shaded area around each line has a width of standard deviation $\pm \sigma$. In general, the larger mutation step-size, the faster the convergence of fitness score.

Figure 4: The progress of rapid succession on CIFAR-10 (a) and CIFAR-100 (b). The blue line is the test performance of the best individual in each generation. The gray dots show the number of parameters of the individuals in each generation. The red line indicates where primary succession ends.

Effect of Secondary Succession. We further analyze the effect of the secondary succession during the evolution process. After the primary succession, we utilize the secondary succession to search the networks with a smaller searching space. We adopt small mutation step-size for the purpose of fine-grained searching based on the survived network from previous generation. Figure 4 shows the example evolution on CIFAR-10 and CIFAR-100 during the rapid succession. We use mutation step-size 100 and 10 for primary succession and secondary succession, respectively. The blue line in the plots shows performance of the best individual in each generation. The gray dots show the number of parameters for the population in each generation, and the red line indicates where the primary succession ends. The accuracy on the two datasets for the secondary succession shown in Table 2 demonstrates that small mutation step-size is helpful for searching better architectures in the rapid succession.

Analysis on Mimicry. In order to analyze the effect of mimicry, we consider the situation where only primary and secondary succession are applied during the evolution. Both the duplication and mimicry are disabled. We denote the method as EIGEN w/o mimicry and duplication. We compare EIGEN w/o mimicry and duplication with the approach where mimicry is enabled and denote it as EIGEN w/o duplication. The comparison between EIGEN w/o
Table 2: The results of secondary succession during the evolution. After the primary succession, the smaller mutation step-size is adopted to search the better network architectures. The accuracy on both CIFAR-10 and CIFAR-100 are improved.

<table>
<thead>
<tr>
<th>Succession</th>
<th>C10+</th>
<th>C100+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary Succession</td>
<td>93.3%</td>
<td>74.7%</td>
</tr>
<tr>
<td>Secondary Succession</td>
<td>93.7%</td>
<td>76.9%</td>
</tr>
</tbody>
</table>

Table 3: Analysis of mimicry during the rapid succession.

<table>
<thead>
<tr>
<th>Method</th>
<th>C10+</th>
<th>C100+</th>
</tr>
</thead>
<tbody>
<tr>
<td>EIGEN w/o mimicry and duplication</td>
<td>92.4%</td>
<td>74.8%</td>
</tr>
<tr>
<td>EIGEN w/o duplication</td>
<td>93.7%</td>
<td>76.9%</td>
</tr>
</tbody>
</table>

Table 4: Analysis of the gene duplication operation on CIFAR-10 and CIFAR-100. The performance on the two datasets is improved with more parameters on the networks discovered from gene duplication.

<table>
<thead>
<tr>
<th>Method</th>
<th>C10+ (PARAMS.)</th>
<th>C100+ (PARAMS.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EIGEN w/o duplication</td>
<td>93.7% (1.2 M)</td>
<td>76.9% (6.1 M)</td>
</tr>
<tr>
<td>EIGEN</td>
<td>94.6% (2.6 M)</td>
<td>78.1% (11.8 M)</td>
</tr>
</tbody>
</table>

Table 5: Analysis of mimicry after the gene duplication.

<table>
<thead>
<tr>
<th>Method</th>
<th>C10+</th>
<th>C100+</th>
</tr>
</thead>
<tbody>
<tr>
<td>EIGEN network w/o mimicry</td>
<td>93.3%</td>
<td>73.9%</td>
</tr>
<tr>
<td>EIGEN network w mimicry</td>
<td>94.6%</td>
<td>78.1%</td>
</tr>
</tbody>
</table>

5. Discussion and Conclusions

In this paper, we propose an Ecologically-Inspired Genetic Approach (EIGEN) for searching neural network architectures automatically from scratch, with poor initialization networks, such as a network with one global pooling layer, and few constraints forced during the searching process. Our searching space follows the work in [27, 36] and we introduce rapid succession, mimicry and gene duplication in our approach to make the search more efficient and effective. The rapid succession and mimicry could evolve a population of networks into an optimal status under the limited computational resources. With the help of gene duplication, the performance of the found network could be boosted without sacrificing any computational cost. The experimental results show the proposed approach can achieve competitive results on CIFAR-10 and CIFAR-100 under dramatically reduced computational cost compared with other genetic-based algorithms.

Admittedly, compared with other searching neural network algorithms [31, 34] which aim to searching network under limited computation resource, our work has the slightly higher error rate. But our genetic algorithm requires little prior domain knowledge from human experts, and is more “complete-automatic” compared with other semi-automatic searching neural network approaches [31, 34], which require more advanced initialization, carefully designed cell-based structures and much more training iterations after the searching process. Such comparison, although unfair, still indicates that more exploration is needed to improve the efficiency for genetic-based approaches in searching neural networks from scratch for the future study.

References


[33] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. Evolving


[38] Sarda Sahney and Michael J Benton. Recovery from the most profound mass extinction of all time. *Proceedings of the Royal Society of London B: Biological Sciences*, 275(1636):759–765, 2008. 1, 3  


