EG-NAS: Neural Architecture Search with Fast Evolutionary Exploration

Zicheng Cai[1,2], **Lei Chen**[1], **Peng Liu**[2,3], **Tongtao Ling**[1], **Yutao Lai**[1]

[1]Guangdong University of Technology
[2]Ping An Technology (Shenzhen) Co., Ltd.
[3]The Hong Kong Polytechnic University
chenlei3@gdut.edu.cn

Abstract

Differentiable Architecture Search (DARTS) has achieved a rapid search for excellent architectures by optimizing architecture parameters through gradient descent. However, this efficiency comes with a significant challenge: the risk of premature convergence to local optima, resulting in subpar performance that falls short of expectations. To address this issue, we propose a novel and effective method called Evolutionary Gradient-Based Neural Architecture Search (EG-NAS). Our approach combines the strengths of both gradient descent and evolutionary strategy, allowing for the exploration of various optimization directions during the architecture search process. To begin with, we continue to employ gradient descent for updating network parameters to ensure efficiency. Subsequently, to mitigate the risk of premature convergence, we introduce an evolutionary strategy with global search capabilities to optimize the architecture parameters. By leveraging the best of both worlds, our method strikes a balance between efficient exploration and exploitation of the search space. Moreover, we have redefined the fitness function to not only consider accuracy but also account for individual similarity. This inclusion enhances the diversity and accuracy of the optimized directions identified by the evolutionary strategy. Extensive experiments on various datasets and search spaces demonstrate that EG-NAS achieves highly competitive performance at significantly low search costs compared to state-of-the-art methods. The code is available at https://github.com/caicaicheng/EG-NAS.

Introduction

Deep neural networks (DNNs), particularly convolution neural networks (CNNs) such as Inception-v1 (Szegedy et al. 2015), ResNet (He et al. 2016), and other pioneering network architectures (Simonyan and Zisserman 2014; Howard et al. 2017), as well as the modern architecture ConvNeXt (Liu et al. 2022), have played a crucial role in driving the advancement of computer vision and addressing various vision-related problems. However, the cost of human design of network architectures is increasing as neural networks advance further, leading to high-cost limitations that prevent researchers from practicing more creative ideas and discouraging their enthusiasm. The proposed Neural Architecture Search (NAS), as an automated method for searching network architectures and effectively addressing this problem, still faces challenges in terms of expensive computational power and long search times. For example, on the CIFAR-10 dataset, AmoebaNet-B (Real et al. 2018), an evolutionary strategy (EA) based approach, requires 3150 GPU days of search time, while NASNet-A (Zoph et al. 2018), a reinforcement learning (RL) based approach, requires 1800 GPU days of search time. To alleviate the high search costs in NAS, a gradient-based method called Differentiable Architecture Search (DARTS) (Liu, Simonyan, and Yang 2018) is proposed, which shares weights between architectures. Compared to costly search strategies such as AmoebaNet-B and NASNet-A, DARTS can perform the architecture search task in 0.4 GPU-Days, offering a more efficient alternative. Furthermore, PC-DARTS achieves even faster results, completing the network architecture search in just 0.1 GPU-Days. However, these gradient-based methods (Xiao et al. 2022a; Chen et al. 2019; Zela et al. 2019; Wang et al. 2021b) still face different dilemmas, and these dilemmas all ultimately point to one reason, i.e., premature convergence to a local optimum. In traditional optimization problems, evolutionary strategy (Hansen and Ostermeier 1996) is a well-responding measure due to its global search property, which makes it more capable of exploring differ-

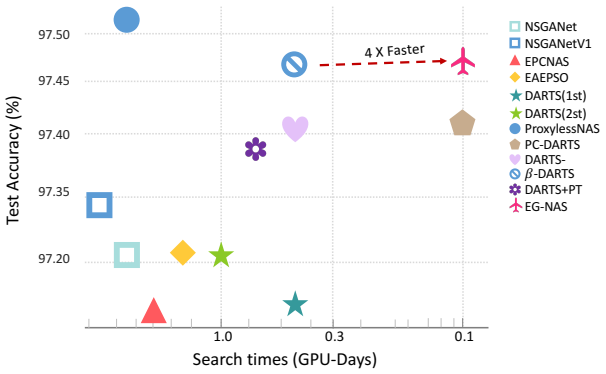

Figure 1: Speed-performance comparison of our proposed EG-NAS with various neural architecture search methods on CIFAR-10.

ent search directions to prevent the dilemma of falling into a local optimum. However, in the field of Neural Architecture Search (NAS), despite the proposal of numerous Evolutionary Algorithms (EA) methods to reduce search time and achieve some results (Lu et al. 2018; Huang et al. 2022), they have not met the expectations in effectively balancing the trade-off between search cost and performance. Therefore, striking a balance between premature convergence to local optima and search cost has become a major challenge for researchers.

In this paper, we propose a simple and efficient compound search algorithm, called EG-NAS, to address the above issues via the evolution strategy with gradient descent for neural architecture search. We adopt an improved evolutionary strategy to explore more diverse search directions and tune the architectural parameters, rather than relying exclusively on gradient descent (GD) to update the architectural and network parameters. This approach allows us to leverage the strengths of both gradient descent and evolutionary strategies, namely efficiency and global searchability. Furthermore, we redesign the fitness function to emphasize individual similarity rather than solely focusing on individual performance to enhance the diversity of search directions. When the current individual outperforms the old one, reducing the similarity between individuals fosters a more diverse evolution in the search direction. Conversely, increasing the similarity between individuals allows timely adjustments to the evolutionary direction toward the previously promising search direction. By adopting this evolutionary strategy, our approach ensures that the output represents the results obtained from exploring various search directions, alleviating the risk of being trapped in local optima. Moreover, it guarantees that the new search directions possess desirable performance. Finally, we iteratively update the architecture parameters with the new search direction obtained by the evolutionary strategy while optimizing the network parameters using gradient descent. This combination allows our approach to efficiently explore diverse architectures with excellent performance.

To demonstrate the effectiveness of our proposed EG-NAS, we conducted extensive experiments on different datasets and search spaces, showing significant competitive results compared to other methods. On CIFAR-10, our method achieves remarkable results, requiring only 0.1 GPU-Days for the search (see Fig. 1). The architectures discovered not only achieve 97.47% accuracy on CIFAR-10, but also demonstrate 74.4% top-1 accuracy when transferred to ImageNet. Moreover, we directly performed the search and evaluation on ImageNet, achieving an outstanding 75.1% top-1 accuracy with a search cost of just 1.2 GPU-Days on 2 RTX 4090 GPUs, which is the optimal search speed compared to state-of-the-art methods.

Related Work

The search strategy based on Evolutionary Algorithms (EA) is one of the most common approaches, relying on EA's global search capability to prevent premature convergence into local optima and effectively address large-scale NAS tasks with a discrete search space. The general process of the EA-based method involves defining the search space, initializing a population of network architectures, evaluating their fitness based on performance, selecting superior architectures, applying crossover and mutation, updating the population, and repeating the process for iterations. However, the challenge of balancing performance and search cost has been a persistent issue for EA-based methods (Cui et al. 2018; Xue et al. 2021). For example, (Liu et al. 2018) and (Real et al. 2018) proposed the methods, Hierarchical evolution and Amoeba-Net, which achieved a high accuracy of 96.25% and 97.45% on the CIFAR-10 dataset, respectively. However, the computation costs for their methods were significantly high, with 300 and 3150 GPU-Days, respectively, far exceeding the capabilities of most researchers. To reduce the excessive search cost, experts, (Lu et al. 2018; Huang et al. 2022; Yuan et al. 2023) in EA have incorporated efficient EA algorithms such as the efficient Non-dominated Sorting Genetic Algorithm, Particle Swarm Optimization (PSO), and so on, into NAS tasks, achieving some notable results. Despite the application of various efficient evolutionary algorithms to improve the efficiency of NAS, an excessive focus on efficiency sometimes hinders these algorithms from fully leveraging their strengths, thereby dramatically impacting the obtained architecture performance.

The high search cost barrier in NAS was first broken by Differentiable architecture search (DARTS) proposed by (Liu, Simonyan, and Yang 2018). DARTS proposed applying the continuous relaxation method to transform discrete operations into continuously differentiable weights, enabling efficient handling of the bi-level optimization objectives in architecture search through gradient descent. To further optimize the memory overhead and improve the search speed, (Xu et al. 2019) proposed PC-DARTS, which randomly selects only some of the channels to serve the computation in the search phase. Although memory overhead is alleviated, some practitioners, (Xie et al. 2018; Chen and Hsieh 2020; Wang et al. 2021a), have raised new questions about DARTS and provided the corresponding solutions. For example, (Hu et al. 2020b) proposed an angle-based metric that simplifies the original search space by eliminating unpromising candidates, thereby reducing the challenges faced by existing NAS methods in searching for high-quality architectures; (Wang et al. 2021a) proposed the node normalization and decorrelation discretization strategy to improve generality and stability; (Xiao et al. 2022b) introduced the Shapley value to evaluate the importance of operations; (Chen et al. 2020) adopted an incremental learning scheme to bridge the gap between search and evaluation. However, the critical issue highlighted by the researchers (Chen et al. 2019; Zela et al. 2019) remains that gradient-based methods suffer from the issue of premature convergence to local optima, significantly compromising the performance of architectures obtained during the search stage.

Methodology

Preliminaries

Differentiable Architecture Search (DARTS) DARTS (Liu, Simonyan, and Yang 2018) is a gradient-based method

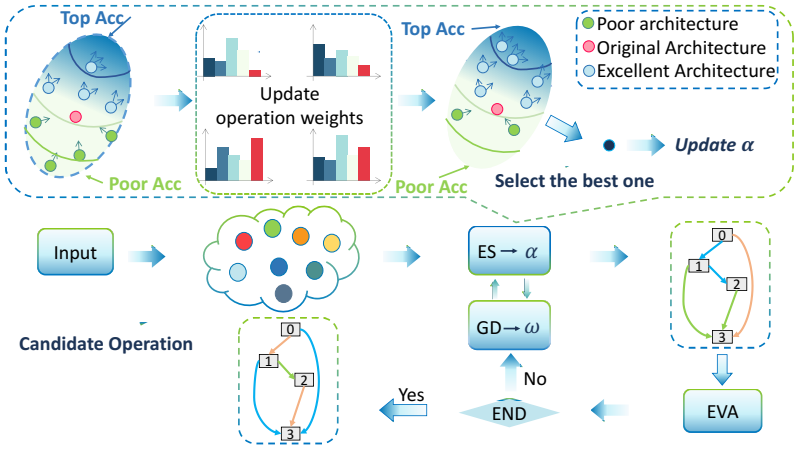

Figure 2: Illustration of EG-NAS main framework. Improved Evolutionary Strategy outputs the best one from evolved architectures as a direction to optimize $\boldsymbol{\alpha}$. After $\boldsymbol{\alpha}$ is updated, the new architecture is composed and evaluated. The EVA is shorthand for evaluation.

widely used in NAS that leverages continuous relaxation and weight-sharing techniques to achieve a differentiable architecture search process, significantly reducing the computational cost. Following previous research (Pham et al. 2018; Bender et al. 2018), DARTS discovers the optimal cell architecture and constructs a supernet by repeatedly stacking normal and reduction cells. Note that, compared to normal cells, the reduction cells are located at 1/3 and 2/3 of the total depth of the network, where all operations adjacent to the stride of the input node are set 2. During the search process, each cell is regarded as a directed acyclic graph (DAG) with N nodes and E edges, where each node $x^{(i)}$ is represented by the feature map and each edge (i, j) represents an operation $o^{(i,j)}$ on the information flow transfer between different nodes. In DARTS, all the candidate operations are applied to the continuous relaxation approach to perform the gradient-based search. Specifically, the intermediate node is computed using a softmax mixture of candidate operations:

$$\overline{o}^{(i,j)}(x^{(i)}) = \sum_{o\in\mathcal{O}} \frac{exp(\boldsymbol{\alpha}_o^{(i,j)})}{\sum_{o'\in\mathcal{O}} exp(\boldsymbol{\alpha}_{o'}^{(i,j)})} o(x^{(i)}), \tag{1}$$

where $i < j$, all candidate operations are stored in $\mathcal{O}$, and $\boldsymbol{\alpha}_o^{(i,j)}$ represents the mixing weight of operation $o^{(i,j)}$ in the supernet construction. During the process of relaxation, architecture search optimizes network weight $\boldsymbol{\omega}$ and architecture parameters $\boldsymbol{\alpha}$ in a differentiable manner, establishing a bi-level optimization model for NAS:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} F(\boldsymbol{\omega}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) = \mathcal{L}_{val}(\boldsymbol{\omega}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \\ &\mathbf{s.t.}\ \ \omega^*(\boldsymbol{\alpha}) = argmin_{\boldsymbol{\omega}} \mathcal{L}_{train}(\boldsymbol{\omega}, \boldsymbol{\alpha}), \end{aligned} \tag{2}$$

where the optimization variables $\boldsymbol{\alpha}$ and $\boldsymbol{\omega}$ are updated via the gradient descent. After ending the search, the final architecture is composed of the operations with the largest architectural parameter $\boldsymbol{\alpha}$ on each edge, $o^{(i,j)} = argmax_{o'\in\mathcal{O}} \alpha_{\mathbf{o}}^{(\mathbf{i},\mathbf{j})}$.

Explore Search Directions with ES For gradient-based methods, the most common step in NAS is to calculate the gradient information and use it as a search direction, iterating until convergence. Although the method is simple and efficient, the search direction is limited by the gradient information, which makes the search direction single and easy to fall into a local dilemma. Covariance Matrix Adaptive Evolutionary Strategy (CMA-ES) (Hansen 2016; Loshchilov and Hutter 2016) is one of the most appreciated evolutionary algorithms in solving continuous black-box problems. Therefore, we apply the capability of CMA-ES, namely efficient convergence and global search, to explore different search directions, avoiding getting trapped in local optima. We begin by sampling N architectures, denoted as $\mathbf{x}_n$ for $n = 1, 2, \ldots, N$, from the Gaussian distribution with $\boldsymbol{\alpha}$ as the mean vector $\boldsymbol{m}^0$ and unit matrix $\mathbf{I}$ as the covariance matrix $\mathbf{C}^0$. In other words, $\mathbf{x}_n = \boldsymbol{\alpha} + \sigma\mathbf{y}$ with $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for $n = 1, 2, \ldots, N$. Subsequently, the sampled architectures $\mathbf{x}_n$ are used to initialize N CMA-ES based searches, where each $\mathbf{x}_n$, for $n = 1, 2, \ldots, N$, is considered as the initialized mean vector $\mathbf{m}^0$ of the n-th ES for the search direction exploration. More specifically, the initial search population of the n-th ES is sampled as follows:

$$\mathbf{z}_i^t = \mathbf{m}^t + \sigma_i \mathbf{y}_i, \mathbf{y}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C}^t), \tag{3}$$

where t starting with 0 is the number of iterations, $\mathbf{C}^0 = \mathbf{I}$, $i = 1, \ldots, \lambda$ and σ is the step size. After that, the mean vector $\mathbf{m}$ and the covariance matrix $\mathbf{C}^{t+1}$ are optimized by Eq. 4 and Eq. 5, which generate the new individual $\mathbf{z}_{i+1}^t$.

$$\mathbf{m}^{t+1} = \textstyle\sum_{i=1}^{\lfloor\lambda/2\rfloor} \beta_i \mathbf{z}_i^t, \tag{4}$$

where β_i represent the fitness weight assigned to each individual x_i.

$$\begin{aligned} \mathbf{C}^{t+1} = &(1 - c_1 - c_{\lfloor\lambda/2\rfloor})\mathbf{C} + c_1(\mathbf{p}\mathbf{p}^T) \\ &+ c_{\lfloor\lambda/2\rfloor} \textstyle\sum_{i=1}^{\lfloor\lambda/2\rfloor} \beta_i (\mathbf{z}_i^t - \mathbf{m}^{t+1})(\mathbf{z}_i^t - \mathbf{m}^{t+1})^T, \end{aligned} \tag{5}$$

where $\mathbf{p}$ is the evolutionary path. During the search, each individual is evaluated by the fitness function $f(\cdot)$ and sorted based on the fitness value of each individual. The best solutions $\lfloor \lambda/2 \rfloor$ are then utilized to update the search parameters, such as the covariance matrix $\mathbf{C}^{t+1}$ and the two learning rates c_1, $c_{\lfloor \lambda/2 \rfloor}$ for the next iteration in CMA-ES. Considering that the optimal individual may not necessarily be present in the final population, we not only update the variables to optimize the search direction but also maintain a record of the best individual $\mathbf{z}_i^*$ from each population by storing them in the set $\mathbf{P}$ based on their fitness values. Ultimately, during the final iteration phase, we select the individual $\mathbf{z}_{best}^*$ with the highest fitness value from the set $\mathbf{P}$ as the output, representing the optimal search direction for the n-th sampling.

Compound Fitness Function

The proposed fitness function, shown in Eq. 7 supplements the consideration of individual similarity or diversity. In other words, the fitness function $f(\cdot)$ is composed of the cross-entropy loss function $\mathcal{L}_1$ and the cosine similarity $\mathcal{L}_2$ as shown in Eq. 6.

$$\begin{aligned} \mathcal{L}_1(y_i, \hat{y}_i) &= -\sum_{i=1}^{C} y_i log(\hat{y}_i), \\ \mathcal{L}_2(\boldsymbol{\alpha}_t, \mathbf{z}_i^{t+1}) &= \frac{1}{2}[\frac{\boldsymbol{\alpha}_t \mathbf{z}_i^{t+1}}{\|\boldsymbol{\alpha}_t\| \left\|\mathbf{z}_i^{t+1}\right\|} + 1], \end{aligned} \tag{6}$$

where C is the number of classes and $\mathcal{L}_1$ represents the cross-entropy loss function used to showcase the performance of architecture $\mathbf{z}_i^{t+1}$. In $\mathcal{L}_1$, y_i and $\hat{y}_i$ refer to the predicted labels by architecture $\mathbf{z}_i^{t+1}$ and true labels, respectively. The essence of $\mathcal{L}_2$ is the cosine similarity function, which denotes the similarity between the current architecture $\mathbf{z}_i^{t+1}$ and the original architecture $\boldsymbol{\alpha}_t$. Note that for convenience in subsequent calculations, we have made a simple adjustment, so that the function $\mathcal{L}_2 \in [0, 1]$. When the $\mathcal{L}_2$ result tends to 0, it indicates that the two architectures are similar; otherwise, they are dissimilar.

$$f(\boldsymbol{\alpha}_t, \mathbf{z}_i^{t+1}) = \begin{cases} \zeta\mathcal{L}_1 - \eta\mathcal{L}_2 & \text{if } Acc(\boldsymbol{\alpha}_t) > Acc(\mathbf{z}_i^{t+1}) \\ \zeta\mathcal{L}_1 + \eta\mathcal{L}_2 & \text{else} \end{cases}, \tag{7}$$

where ζ and η are the weight coefficients of $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively. When the performance of the currently generated individual is inferior to the original individual, we make the $\mathcal{L}_2$ function tend towards 0, directing the new individual towards a more promising performance. Conversely, when the performance is better, we make the $\mathcal{L}_2$ function tend towards 1, encouraging the new individual to evolve towards increased architectural diversity. Assisted by this composite fitness function $f(\cdot)$, the evolution strategy generates more diverse individuals, evaluating a broader range of search directions, leading to a more effective balance between performance and diversity. Additionally, exploring and evaluating more diverse search directions is beneficial for alleviating the issue of premature convergence to local optima and obtaining superior architectures. To confirm the effectiveness of the compound fitness function $f(\cdot)$, we have conducted relevant ablation studies, and the corresponding comparative results are presented in Fig. 3. Detailed analysis of the results can be found in part of the ablation studies.

Evolution Strategy with Gradient Descent-based Architecture Search

To better guide the architecture search process, we propose exploring various search directions and selecting the optimal ones based on task performance with the evolutionary strategy. Compared to optimizing both the architecture and network parameters using gradient-based methods, our proposed approach updates only the network parameters through gradient descent, while the architecture parameters are updated using an evolutionary strategy to sample excellent search directions for guidance. To more visually illustrate our proposed approach, we provide the general steps of EG-NAS as shown in Fig. 2. The update process of $\boldsymbol{\alpha}$ can be presented in Algorithm 1 and in Eq. 8.

$$\boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \xi \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{\omega}(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \rightarrow \boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{t-1} + \xi \mathbf{s}_t, \tag{8}$$

where $\mathbf{s}_t$ represents the direction of $\boldsymbol{\alpha}$ updates at the t-th step during the optimization process, $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{val}(\boldsymbol{\omega}(\boldsymbol{\alpha}), \boldsymbol{\alpha})$ represents the search direction based on the gradient information, and ξ means the step size. To ensure that the search direction $\mathbf{s}$ used to update α is closely related to the task performance, we introduce task performance-based stabilization during the optimization:

$$\mathbf{s}_t = argmax\ Acc_{val}(\boldsymbol{\omega}_{t-1}, \mathbf{x}_n^*), n = 1, 2, \ldots, N, \tag{9}$$

where Acc_{val} means the validation accuracy, $\mathbf{x}_n^*$ is the search direction that yields the best fitness value by sampling the evolutionary strategy, and $\boldsymbol{\omega}_{t-1}$ is the network parameters at $(t-1)$ step. After completing the search, we adopt the same approach as DARTS to obtain the final architecture, where each edge is constructed with the operation that has the maximum weight. Compared to DARTS, which updates architecture parameters only once per round, our method optimizes architecture parameters in every sampling and evolutionary strategy step. As a result, in terms of architecture parameter optimization, EG-NAS has a time complexity of N times λ compared to traditional gradient descent methods.

Experiments and Analysis

In this part, we conduct extensive experiments to evaluate our approach, EG-NAS, on the DARTS search space with CIFAR-10, CIFAR-100, and ImageNet for image classification, as well as the NAS-Bench-201 search space with CIFAR-10, CIFAR-100, and ImageNet-16-120. All experiments were conducted on a single Nvidia RTX 3090, except for the ImageNet experiments, which were conducted on 2 RTX 4090. The ImageNet datasets mentioned in this paper refer to either ImageNet 1K or the ILSVRC2012 dataset, except for ImageNet-16-120 in NAS-Bench-201.

Datasets and Implementation Details

CIFAR-10 dataset contains 60,000 color images from 10 different categories. CIFAR-100 dataset consists of 100 different categories, including some finer-grained classes. The

Algorithm 1: **Main framework of EG-NAS**

Require: Architecture and Network parameters: $\boldsymbol{\alpha}$, $\boldsymbol{\omega}$; Sample times N; Epoch T; Step size ξ and σ_1; Population size λ.

For each edge (i, j), a mixed operation $o^{(i,j)}$ is formed by parameterizing it with $\boldsymbol{\alpha}^{(i,j)}$.
for $t = 1, 2, \ldots, T$ **do**
  Update network parameter $\boldsymbol{\omega}$ by $\nabla_{\boldsymbol{\omega}} L_{train}(\boldsymbol{\omega}, \boldsymbol{\alpha})$;
  for $n = 1, 2, \ldots, N$ **do**
    Initialize $\mathbf{x}_n = \boldsymbol{\alpha}_t + \sigma_n \mathbf{y},\ \mathbf{y} \sim \mathcal{N}(0, \mathbf{I})$;
    while $k < \lambda$ **do**
      Sample the population individuals $\mathbf{z}_k^{t+1}$ via Eq. 3;
      Evaluate λ individuals using Eq. 7 ;
      Select $\lfloor \lambda/2 \rfloor$ best individuals for updating and store the best in $\mathbf{P}$;
      $k = k + 1$;
    end while
    Select the best individual $\mathbf{z}_{best}^*$ from $\mathbf{P} = \{\mathbf{m}^t, \mathbf{z}_1^*, \ldots, \mathbf{z}_{\lambda-1}^*\}$ based on the fitness values.
    Store the optimal individual $\mathbf{z}_{best}^* \rightarrow \mathbf{x}_n^*$ in $\mathbf{D}$;
  end for
  Select the best search direction $\mathbf{s}_{t+1}$ based on the validation accuracy from $\mathbf{D} = \{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_N^*\}$;
  $\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + \xi \mathbf{s}_{t+1}$;
end for
Derive the final architecture parameter $\boldsymbol{\alpha}$.

same operation space $\mathcal{O}$ as DARTS is applied and a similar partial connection method as PC-DARTS is employed to reduce memory overhead. The training set of CIFAR-10 is divided into two parts of equal size, one for optimizing the network parameters by gradient descent and the other for obtaining the search direction. We train the supernet for 50 epochs (the first 15 epochs for warm-up) with a batch size of 256 and retrained the network from scratch from 600 epochs, with a batch size of 96. In ES, the population size λ is set as 25 and the coefficients ζ and η for $\mathcal{L}_1$ and $\mathcal{L}_2$ are set as 1.0 and 0.4, respectively. The step size ξ, the sample numbers N, and the initial channel number were assigned to 0.6, 5, and 16, respectively. For a fair comparison, the remaining search and assessment phases are set up to be consistent with DARTS.

ImageNet consists of 1000 classes with 1.2 million training and 50,000 validation images. We train the supernet for 50 epochs (the first 15 epochs for warm-up) with a batch size of 1024 in the search stage and retained the network from scratch from 250 epochs in the evaluation stage. In the ES algorithm, we set the population size λ to 50. We employ an SGD optimizer with a linearly decayed learning rate initialized at 0.5, a momentum of 0.9, and a weight decay of 3×10^{-5}. NAS-Bench-201 is a significant benchmark dataset containing 15,625 diverse neural network architectures with 5 different operations and 4 nodes per cell. It provides a standardized testing environment for evaluating NAS algorithms on CIFAR-10, CIFAR-100, and ImageNet16-120 datasets. In this search space, we obtain the performance of specific

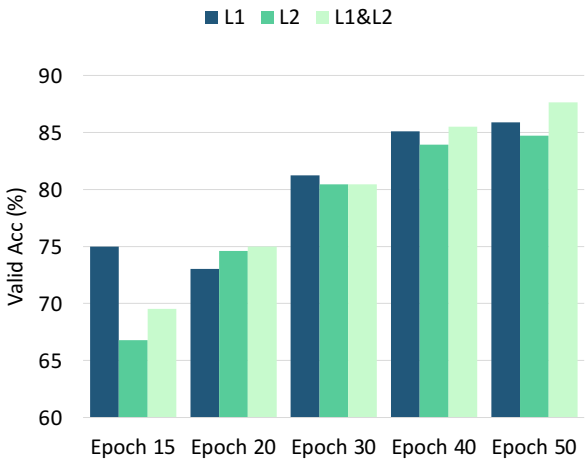

Figure 3: During the CIFAR-10 search phase, the impact of $\mathcal{L}_1$ and $\mathcal{L}_2$ on EG-NAS in evolutionary strategy. The valid acc (%) refers to the metric that evaluates the search direction with the optimal fitness value during the ES search.

tasks by directly evaluating the dataset, and we derive the mean and standard deviation of the best architectures by conducting independent runs with 4 different random seeds. More details about NAS-Bench-201 can be found in reference (Dong and Yang 2020).

Comparison with State-of-the-art NAS Methods

The performance of EG-NAS is compared with state-of-the-art NAS methods on the CIFAR-10 and CIFAR-100 datasets in Table 1. EG-NAS achieves an impressive test error rate of 2.53% with just 0.1 GPU-Days, surpassing the DARTS baseline significantly in search cost and accuracy. Although ProxylessNAS, P-DARTS, and $\beta-$DARTS perform better, EG-NAS explores architecture search in a distinct space with remarkably low search cost while maintaining exceptional performance. Table 2 presents the comparison results of EG-NAS with other methods on the ImageNet dataset. We trained the best-searched architecture on CIFAR-10 to evaluate its transferability to ImageNet, achieving a competitive top-1 test error rate of 25.6%, confirming the generalizability of EG-NAS. Additionally, we directly search ImageNet to evaluate the optimal architecture, obtaining a top-1 test error rate of 24.9%. This not only outperforms most NAS methods in terms of performance but is also noteworthy as we achieved this with an extremely low search cost (2.2 GPU-Days), which surpasses all other NAS methods in terms of search efficiency. On the NAS-Bench-201 dataset, EG-NAS achieves outstanding performance by conducting an architecture search on the CIFAR-10 dataset and evaluating on CIFAR-10, CIFAR-100, and ImageNet-16-120, achieving test accuracies of 93.56%, 70.91%, and 46.13%, respectively, as shown in Table 3. While our method may not outperform all NAS methods on all datasets, such as DARTS- which performs better on CIFAR-10 and CIFAR-

Architecture	Test Error (%)		Params	Search Cost
	CIFAR-10	CIFAR-100	(M)	(GPU-Days)
ResNet (He et al. 2016)	4.61	22.1	1.7	-
ENAS + cutout (Pham et al. 2018)	2.89	-	4.6	0.5
AmoebaNet-A (Real et al. 2018)	3.34	17.63	3.3	3150
NSGA-Net (Lu et al. 2018)	2.75	20.74	3.3	4.0
NSGANetV1-A2 (Lu et al. 2018)	2.65	-	0.9	27
EPCNAS-C (Huang et al. 2022)	3.24	18.36	1.44	1.2
EAEPSO (Yuan et al. 2023)	2.74	16.94	2.94	2.2
DARTS(1st) (Liu, Simonyan, and Yang 2018)	3.00	17.54	3.4	0.4
DARTS(2st) (Liu, Simonyan, and Yang 2018)	2.76	-	3.3	1.0
SNAS (moderate) + cutout (Xie et al. 2018)	2.85	17.55	2.8	1.5
ProxylessNAS + cutout (Cai, Zhu, and Han 2018)	2.02	-	-	4.0
GDAS (Dong and Yang 2019b)	2.93	18.38	3.4	0.2
BayesNAS (Zhou et al. 2019)	2.81	-	3.4	0.2
P-DARTS + cutout (Chen et al. 2019)	**2.50**	17.49	3.4/3.6	0.3
PC-DARTS + cutout (Xu et al. 2019)	2.57	16.90	3.6	**0.1**
DARTS- (Chu et al. 2020)	2.59	17.51	3.4	0.4
β-DARTS (Ye et al. 2022)	2.53	16.24	3.75/3.80	0.4
DrNAS (Chen et al. 2020)	2.54	-	4.0	0.4
DARTS+PT (Wang et al. 2021b)	2.61	-	3.0	0.8
EG-NAS	2.53	**16.22**	3.2	**0.1**

Table 1: Comparison of EG-NAS with state-of-the-art image classifiers on CIFAR-10 and CIFAR-100. The results of EG-NAS were obtained by repeated experiments with 4 random seeds.

100, EG-NAS still surpasses most of the NAS methods. In particular, it achieves exceptional performance with 46.13% on ImageNet-16-120, confirming the effectiveness of our method, EG-NAS, and its excellent generalizability in more complex datasets.

Architecture	Test error top-1(%)	Search cost (GPU-Days)	Params (M)
Inception-v1	30.1	-	6.6
MobileNet	29.4	-	4.2
DARTS(2st)	26.7	1.0	4.7
SNAS	27.3	1.5	2.8
ProxylessNAS†	24.9	8.3	7.1
GDAS	26.0	0.3	3.4
BayesNAS	26.5	0.2	3.9
PC-DARTS	25.1	0.1	4.7
DrNAS†	24.2	4.6	5.7
DARTS+PT†	25.5	3.4	4.7
NASNet-A	26.0	2000	3.3
NASNet-B	27.2	2000	3.3
NASNet-C	27.5	2000	3.3
AmoebaNet-A	25.5	3150	3.2
AmoebaNet-B	26.0	3150	3.2
AmoebaNet-C	24.3	3150	3.2
NSGANetV1-A2	25.5	27	4.1
EAEPSO	26.9	4.0	4.9
EPCNAS-C2	27.1	1.17	3.0
EG-NAS	24.9	0.1	5.3
EG-NAS†	25.6	2.2	5.2

Table 2: Comparison with state-of-the-art image classifiers on ImageNet. † means the results obtained by searching on ImageNet, otherwise on CIFAR-10.

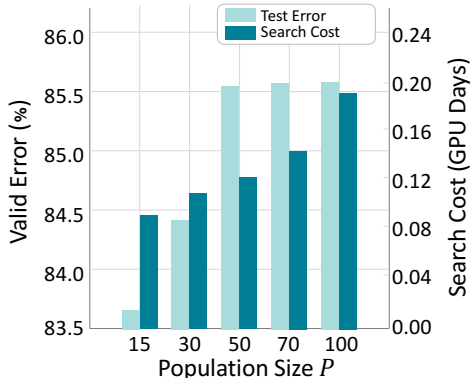

Figure 4: During the CIFAR-10 search phase, the effects of population size λ on EG-NAS in evolutionary strategy. The valid acc (%) refers to the metric that evaluates the search direction with the optimal fitness value during the ES search.

Ablation Study

Impact of $\mathcal{L}_1$ and $\mathcal{L}_2$ To investigate the impact of $\mathcal{L}_1$ and $\mathcal{L}_2$, we conducted experiments with different fitness compositions on CIFAR-10. From Fig. 3, we can observe that in the early stages of the experiments, fitness functions solely based on performance achieved relatively good results. However, as the experiments progressed, fitness functions incorporating both $\mathcal{L}_1$ and $\mathcal{L}_2$ show superior growth and performance, as they explored more directions while considering performance. On the other hand, fitness functions focusing solely on diversity could explore more directions but faced challenges in convergence due to the lack of performance consideration.

Architecture	CIFAR-10		CIFAR-100		ImageNet16-120	
	valid	test	valid	test	valid	test
ResNet(He et al. 2016)	90.83	93.97	70.42	70.86	44.53	43.63
Random (baseline)	90.93±0.36	93.70±0.36	70.60±1.37	70.65±1.38	42.92±2.00	42.96±2.15
ENAS (Pham et al. 2018)	37.51±3.19	53.89±0.58	13.37±2.35	13.96±2.33	15.06±1.95	14.57±2.10
RandomNAS (Li and Talwalkar 2020)	80.42±3.58	84.07±3.61	52.12±5.55	52.31±5.77	27.22±3.24	26.28±3.09
SETN (Dong and Yang 2019a)	84.04±0.28	87.64±0.00	58.86±0.06	59.05±0.24	33.06±0.02	32.52±0.21
GDAS (Dong and Yang 2019b)	90.01±0.46	93.23±0.23	24.05±8.12	24.20±8.08	40.66±0.00	41.02±0.00
DSNAS (Hu et al. 2020a)	89.66±0.29	93.08±0.13	30.87±16.40	31.01±16.38	40.61±0.09	41.07±0.09
DARTS (1st) (Liu, Simonyan, and Yang 2018)	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
DARTS (2st) (Liu, Simonyan, and Yang 2018)	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
PC-DARTS (Xu et al. 2019)	89.96±0.15	93.41±0.30	67.12±0.39	67.48±0.89	40.83±0.08	41.31±0.22
iDARTS (Wang et al. 2021a)	89.86±0.60	93.58±0.32	70.57±0.24	70.83±0.48	40.38±0.59	40.89±0.68
DARTS- (Chu et al. 2020)	**91.03±0.44**	**93.80±0.40**	**71.36±1.51**	**71.53±1.51**	44.87±1.46	45.12±0.82
EG-NAS	90.12±0.05	93.56±0.02	70.78±0.12	70.91±0.07	**44.89±0.29**	**46.13±0.46**
Optimal	91.61	94.37	73.49	73.51	46.77	47.31

Table 3: Performance comparison of the NAS-Bench-201 benchmark. Note that EG-NAS only searched on the CIFAR-10 dataset, but all achieved competitive results on CIFAR-10, CIFAR-100, and ImageNet16-120. The average values are obtained over four independent search runs.

Influence of coefficients ζ and η of $\mathcal{L}_1$ and $\mathcal{L}_2$ To explore the influence of different values of the coefficients ζ and η in the fitness function $f(\cdot)$ of ES, we implement $\mathcal{L}_1$ and $\mathcal{L}_2$ assignment with different ζ and η. The validation accuracy and model parameter cost are demonstrated in Table 4. As the number of updates increases, the value of the $\mathcal{L}_2$ coefficient in the fitness function $f(\cdot)$, denoted as η, plays a critical role in Evolution Strategies (ES). When η is set too large, ES tends to prioritize exploring diverse search directions, leading to excessive dispersion and hindered convergence. On the contrary, if η is too small, ES focuses predominantly on optimizing performance, resulting in overly monotonous search directions. As the $\mathcal{L}_1$ coefficient ζ increases, the evolutionary strategy discovers more stable and superior optimization directions, with ζ=1.0 achieving the best performance.

Effect of Population Size λ on EG-NAS In this part, a series of experiments are conducted on the CIFAR-10 dataset to investigate the impact of the population size λ on EG-NAS. Based on the observations in Fig. 4, when the population size λ is set too small, it significantly constrains the diversity of the population, further limiting the discovery of high-performance architectures. With the increase in the population size λ, there is a greater number of individuals available for selection, leading to increased diversity within the population, making it easier to discover high-quality network architectures. However, as λ continues to increase, the performance of discovered architectures sees only marginal improvements, due to the constraints imposed by the cell search space. Concurrently, the increase in λ results in a significant escalation of search costs. To effectively strike a balance between search costs and architecture performance, we ultimately opted for a population size of $\lambda = 50$ as our final choice.

ζ	η=1.0		η=0.8	
	valid_acc (%)	params (M)	valid_acc (%)	params (M)
1.0	84.36	3.23	85.42	2.96
0.8	85.21	2.96	84.89	3.56
0.4	84.36	3.23	83.80	3.85
0.2	83.61	3.10	83.13	3.57
ζ	η=0.4		η=0.2	
	valid_acc (%)	params (M)	valid_acc (%)	params (M)
1.0	**85.53**	3.71	85.40	2.85
0.8	84.82	3.97	83.63	3.41
0.4	82.81	3.0	81.25	2.85
0.2	81.54	2.96	80.68	3.66

Table 4: On CIFAR-10, the effect of different coefficient ζ and η values of the composite fitness function Eq. 7 on the task performance is verified. The valid acc (%) refers to the metric that evaluates the search direction with the optimal fitness value during the ES search.

Conclusion and Future Work

In this paper, we propose a simple yet efficient compound approach based on gradient descent and an improved evolutionary strategy, termed EG-NAS, for neural architecture search, which alleviates the dilemma of premature convergence to local optima by gradient descent. Meanwhile, by reducing the similarity between individuals in the evolutionary strategy, we can effectively explore various search directions and avoid being trapped in local optima. Finally, we evaluate the selected search directions with better fitness values using validation accuracy to more accurately determine the relationship between search directions and task performance. In the future, our aim is to further investigate the effective integration of different algorithms and enhance the stability of hybrid algorithms to address a wide range of tasks.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China 62006044, and in part by Programme of Science and Technology of Guangzhou 202201010377.

References

Bender, G.; Kindermans, P.-J.; Zoph, B.; Vasudevan, V.; and Le, Q. 2018. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, 550–559. PMLR.

Cai, H.; Zhu, L.; and Han, S. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*.

Chen, X.; and Hsieh, C.-J. 2020. Stabilizing differentiable architecture search via perturbation-based regularization. In *International Conference on Machine Learning*, 1554–1565. PMLR.

Chen, X.; Wang, R.; Cheng, M.; Tang, X.; and Hsieh, C.-J. 2020. Drnas: Dirichlet neural architecture search. *arXiv preprint arXiv:2006.10355*.

Chen, X.; Xie, L.; Wu, J.; and Tian, Q. 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE/CVF international conference on computer vision*, 1294–1303.

Chu, X.; Wang, X.; Zhang, B.; Lu, S.; Wei, X.; and Yan, J. 2020. Darts-: robustly stepping out of performance collapse without indicators. *arXiv preprint arXiv:2009.01027*.

Cui, X.; Zhang, W.; Tüske, Z.; and Picheny, M. 2018. Evolutionary stochastic gradient descent for optimization of deep neural networks. *Advances in Neural Information Processing Systems*, 31.

Dong, X.; and Yang, Y. 2019a. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3681–3690.

Dong, X.; and Yang, Y. 2019b. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1761–1770.

Dong, X.; and Yang, Y. 2020. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*.

Hansen, N. 2016. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*.

Hansen, N.; and Ostermeier, A. 1996. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, 312–317.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Conference on Computer Vision and Pattern Recognition*, 770–778.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hu, S.; Xie, S.; Zheng, H.; Liu, C.; Shi, J.; Liu, X.; and Lin, D. 2020a. Dsnas: Direct neural architecture search without parameter retraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12084–12092.

Hu, Y.; Liang, Y.; Guo, Z.; Wan, R.; Zhang, X.; Wei, Y.; Gu, Q.; and Sun, J. 2020b. Angle-based search space shrinking for neural architecture search. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIX 16*, 119–134. Springer.

Huang, J.; Xue, B.; Sun, Y.; Zhang, M.; and Yen, G. G. 2022. Particle Swarm Optimization for Compact Neural Architecture Search for Image Classification. *IEEE Transactions on Evolutionary Computation*, 1–1.

Li, L.; and Talwalkar, A. 2020. Random search and reproducibility for neural architecture search. In *Uncertainty in Artificial Intelligence*, 367–377. PMLR.

Liu, H.; Simonyan, K.; Vinyals, O.; Fernando, C.; and Kavukcuoglu, K. 2018. Hierarchical Representations for Efficient Architecture Search. In *International Conference on Learning Representations*.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. DARTS: Differentiable Architecture Search. *arXiv preprint arXiv:1806.09055*.

Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T.; and Xie, S. 2022. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11976–11986.

Loshchilov, I.; and Hutter, F. 2016. CMA-ES for hyperparameter optimization of deep neural networks. *arXiv preprint arXiv:1604.07269*.

Lu, Z.; Whalen, I.; Boddeti, V.; Dhebar, Y. D.; Deb, K.; Goodman, E. D.; and Banzhaf, W. 2018. NSGA-NET: A Multi-Objective Genetic Algorithm for Neural Architecture Search. *CoRR*, abs/1810.03522.

Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient Neural Architecture Search via Parameter Sharing. *CoRR*, abs/1802.03268.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2018. Regularized Evolution for Image Classifier Architecture Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33.

Simonyan, K.; and Zisserman, A. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *Computer Science*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9.

Wang, H.; Yang, R.; Huang, D.; and Wang, Y. 2021a. idarts: Improving darts by node normalization and decorrelation discretization. *IEEE Transactions on Neural Networks and Learning Systems*.

Wang, R.; Cheng, M.; Chen, X.; Tang, X.; and Hsieh, C.-J. 2021b. Rethinking architecture selection in differentiable nas. *arXiv preprint arXiv:2108.04392*.

Xiao, H.; Wang, Z.; Zhu, Z.; Zhou, J.; and Lu, J. 2022a. Shapley-NAS: discovering operation contribution for neural

architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11892–11901.

Xiao, H.; Wang, Z.; Zhu, Z.; Zhou, J.; and Lu, J. 2022b. Shapley-NAS: Discovering Operation Contribution for Neural Architecture Search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11892–11901.

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2018. SNAS: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2019. Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737*.

Xue, K.; Qian, C.; Xu, L.; and Fei, X. 2021. Evolutionary Gradient Descent for Non-convex Optimization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 3221–3227.

Ye, P.; Li, B.; Li, Y.; Chen, T.; Fan, J.; and Ouyang, W. 2022. b-darts: Beta-decay regularization for differentiable architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10874–10883.

Yuan, G.; Wang, B.; Xue, B.; and Zhang, M. 2023. Particle Swarm Optimization for Efficiently Evolving Deep Convolutional Neural Networks Using an Autoencoder-based Encoding Strategy. *IEEE Transactions on Evolutionary Computation*, 1–1.

Zela, A.; Elsken, T.; Saikia, T.; Marrakchi, Y.; Brox, T.; and Hutter, F. 2019. Understanding and robustifying differentiable architecture search. *arXiv preprint arXiv:1909.09656*.

Zhou, H.; Yang, M.; Wang, J.; and Pan, W. 2019. BayesNAS: A Bayesian Approach for Neural Architecture Search. *CoRR*, abs/1905.04919.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8697–8710.