

G-EvoNAS: Evolutionary Neural Architecture Search Based on Network Growth

Juan Zou, Weiwei Jiang, Yizhang Xia, Yuan Liu, and Zhanglu Hou

Xiangtan University, Yuhu District, Xiangtan, Hunan, China
202221632973@smail.xtu.edu.cn

Abstract. The evolutionary paradigm has been successfully applied to neural network search(NAS) in recent years. Due to the vast search complexity of the global space, current research mainly seeks to repeatedly stack partial architectures to build the entire model or to seek the entire model based on manually designed benchmark modules. The above two methods are attempts to reduce the search difficulty by narrowing the search space. To efficiently search network architecture in the global space, this paper proposes another solution, namely a computationally efficient neural architecture evolutionary search framework based on network growth (G-EvoNAS). The complete network is obtained by gradually deepening different Blocks. The process begins from a shallow network, grows and evolves, and gradually deepens into a complete network, reducing the search complexity in the global space. Then, to improve the ranking accuracy of the network, we reduce the weight coupling of each network in the SuperNet by pruning the SuperNet according to elite groups at different growth stages. The G-EvoNAS is tested on three commonly used image classification datasets, CIFAR10, CIFAR100, and ImageNet, and compared with various state-of-the-art algorithms, including hand-designed networks and NAS networks. Experimental results demonstrate that G-EvoNAS can find a neural network architecture comparable to state-of-the-art designs in 0.2 GPU days.

Keywords: NAS · Evolutionary Algorithms · Network Growth

1 Introduction

In recent years, neural networks have experienced great success in solving various challenging tasks [12, 15, 18], such as image recognition [10, 12], object detection [7] and semantic segmentation [11]. Since the performance of deep neural networks majorly depends on their architecture, an increasing number of research efforts in the deep learning community are devoted to designing novel architectures, such as DenseNet [15], ResNet [12], and GoogleNet [39]. However, the design of novel network architectures strongly relies on the knowledge and experience of human experts and may necessitate multiple attempts before meaningful results are achieved [12]. Neural Architecture Search (NAS) is regarded as an effective method to automate the design of task-specific neural network

architectures [41], which enables interested users to quickly develop their own CNN app without sufficient engineering experience and domain expertise. It has been shown that NAS can implement competitive DNN architectures and human experts and discover new state-of-the-art architectures [49].

Present-day NAS methods mainly search local architectures (Cell) and then stack the searched Cells to build the entire model architecture or use artificially designed modules as basic search units to search the global neural network model. The former method simply repeatedly stacks Cells to build a model, resulting in the structure of each model layer being the same. Indeed, the functions of each neural network layer are different, so the resulting architecture could be more optimal. The global optimal solution cannot be obtained by reducing the complexity of the final model structure in exchange for fast convergence of the search process. The alternative method of directly searching the entire neural network architecture [9] generally uses artificially designed modules such as MobileNetV2 [34] as the basic building block. Using artificially designed block structures as basic units, this method performs well. Search methods targeting block structures are more straightforward than searching for neural network architectures in the global space, but the search space is greatly restricted.

Despite the fact that local search methods and methods based on manually designed modules can achieve excellent performance, the potential performance of globally operated search methods isn't overlooked [44]. The objective of this study is to search global neural network models more efficiently. A growth-centric network method is proposed. Through this method, a diverse inter-layer Block neural network model architecture can be obtained without directly searching the global space in a less time-consuming manner, thus enabling a quick and effective search of the entire model architecture.

In this paper, we propose an evolutionary neural architecture search framework based on network growth (G-EvoNAS). Specifically, we divide the evolution process into multiple growth stages and search for block structures in the network one by one in a growing manner. At the end of each growth stage, the candidate network in the current population is retained as the initial network for the next evolutionary stage, and the process is repeated. process until a sufficient number of blocks are found to form a complete network. Since only a single block is searched at each growth stage during the evolution process, the huge complexity caused by searching directly in the global space can be avoided. Lastly, concerning accelerated assessment, the weight-sharing strategy of the SuperNet is utilized to evaluate the performance of all progeny individuals without training. This not only reduces computational costs, but also, to alleviate the weight coupling of each network within the SuperNet, we preserve the superior individuals at each evolutionary stage from the population to conduct phasic pruning of the SuperNet, reducing the quantity of candidate networks within the SuperNet and enhancing the ranking accuracy of the network. An overview of our approach is shown in Figure 1.

The main contributions of our work are summarized as follows:

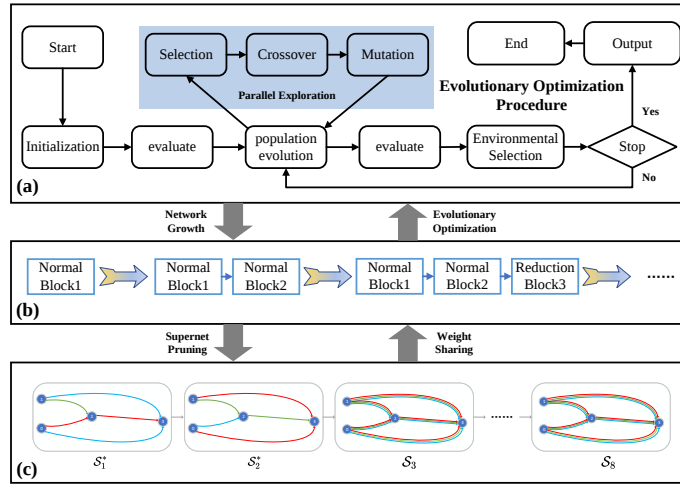


Fig. 1: Overall framework of G-EvoNAS. (a) Evolutionary search process. During the search process of population evolution, only the latest growing Block will be searched. (b) Network growth process, the network length will increase by one Block at each growth stage. (c) Supernet pruning, the SuperNet corresponding to the current growth stage will be searched. The block in is pruned. Figure (c) shows an example of pruning two blocks.

- An alternative approach to reduce the global search complexity of NAS , i.e., progressively searching each block structure in the candidate network and allowing the network to grow from a single block to a comprehensive configuration.
- The weight sharing of the SuperNet is used to evaluate the candidate networks, while pruning the SuperNet based on the elite population during the network growth process. The weight coupling between individual subnets is alleviated by reducing the number of candidate networks in the SuperNet. The accuracy of network ranking is improved while accelerating individual performance evaluation.
- Our algorithm searches for competitive network architectures in a relatively short time. Using only 0.2 GPU days on the CIFAR10 and CIFAR100 datasets, the final network architectures achieve 97.52% and 83.38% classification accuracy. Afterwards, we transfer the network structure searched on CIFAR10 directly to the ImageNet dataset and find that G-EvoNAS can achieve 75.5% classification accuracy.

2 Related Work

2.1 EA-based NAS

The development of neural networks via evolutionary algorithms has garnered considerable interest. With the initial application of evolutionary algorithms

in neural network architecture search (NEAT), remarkable success has been achieved [19]. Evolutionary algorithms have been broadly utilized in the optimization of neural network parameters and hyperparameters [6, 9, 33], and currently, an abundance of research is concentrating on ENAS [7, 11]. [33] unveiled what was likely the inaugural large-scale application of a simple evolutionary algorithm, which returns networks matching the human-designed models. The evolutionary process is described in detail in [32] and the concept of age is introduced. The framework of our paper is based on evolutionary algorithms, where common genetic operators, tournament selection strategies, and environmental selection strategies match or even outperform their RL baselines in terms of speed and accuracy [32]. However, since a large number of candidate models often need to be trained from scratch for evaluation, most of these methods are still computationally intensive.

2.2 SMBO-based NAS

Several previous works have explored the Sequence Model-Based Optimization (SMBO). [29] involves Monte Carlo Tree Search (MCTS). The sequential model-based optimization approach [17] amplifies the efficiency of MCTS by unraveling a predictive model that directs node expansion. Other relevant works include the likes of [8, 27, 36, 50], which investigate MLPs instead of CNNs, the application of incremental methods in topology, the augmentation of the layer count, and the search for the syntactically specific area of the latent factor model respectively. Works such as [5, 14] progressively blossom CNNs sequentially; [21] gradually probe the entire network by adding nodes to the Cell and hastening performance evaluation via the surrogate model; and [20], which employs progressive search to alleviate the posterior decay.

2.3 Pruning SuperNet

The paradigm of Neural Architecture Search (NAS) [12, 41] is typically marked by high computational costs. To address this challenge, route improvements in search performance by sharing weights across varying models have been attempted [22, 31]. These methods, which leverage shared weights across a hyperparameterized network (hypergraph) that contains every model, can be further categorized into two groups.

The first, continuous relaxation methods [22], co-optimizes the weights and architectural selection factors of the SuperNet through gradient descent. However, the optimization of these selection factors introduces biases between sub-models inevitably. As sub-models with initial low performance will garner less training, they tend to be outperformed by other models. These methods heavily rely on the initial state, making it challenging to achieve optimal architecture. The second, one-shot methods [1, 4, 9], which guarantee the equitable treatment of all sub-models. After the SuperNet is trained via path loss or path sampling, the sub-models are sampled and assessed using weights inherited from the SuperNet.

Other relatable works include ASAP [30] and XNAS [28], models that introduce pruning during the training process of over-parameterized networks to improve NAS efficiency. Similar to these methods, we commence from an over-parameterized network, then reduce the search area to harvest an optimized architecture, and further improve the model’s ranking.

3 Methodology

In this section, we introduce G-EvoNAS in detail from the aspects of search space, search strategy and acceleration method. First, based on the existing micro search space, we propose a block search space, which increases the diversity of network modules. An evolutionary search strategy based on network growth is proposed in the block search space, which further reduces the search complexity. Finally, through SuperNet pruning, the accuracy of network ranking in the SuperNet is improved.

3.1 Block-wise Search Space

The primary focus of this section is to present the specifics of the search space employed in our study. As illustrated in Figure 1, the overall network architecture comprises of varying Blocks. All prospective networks possess a fixed outer structure. Each Block receives direct input from the preceding Block (as demonstrated in the figure) and input from its prior Block. Each component block exhibits distinct characteristics, yet, they generally can be categorized into two types: normal Blocks and downsampling Blocks. These vary in that their internal structure is different and that the internal stride of the Reduction Block lessens the magnitude of the image size, whereas the Normal Block retains it. From the illustration, it is evident that the Blocks located at the $1/3$ and $2/3$ markers of the completed network are designated as Reduction Blocks, while the remainder are Normal Blocks. The ultimate goal of architecture striving is to unearth the structure of diverse blocks.

We design new efficient networks by utilizing different topologies, allowing each Block to have its own structure. Although the block construction process is the same as in [22], the construction of the entire network is different. Specifically, the chunked search space is an extended version of the micro-search space proposed in [50] and [31]. To handle intermediate information, our method designs a new convolutional block after downsampling instead of repeating a normal block multiple times. This scheme overcomes the limitation of repeating the same normal block throughout the network, since the same kind of block may not be optimal for feature maps of different resolutions.

3.2 Evolution Search

Many previous methods only search in the space of local cells, or search in the space of CNN based on handcrafted models. Although this is a simpler and

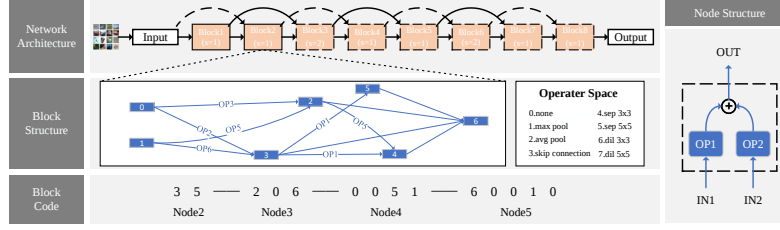


Fig. 2: Top: A design network architecture consisting of 8 blocks. Middle and bottom: Block 2 and its block code example. A block consists of 4 hidden nodes, 2 source nodes (node 0 and node 1) and one output node (node 6). Right side: nodes represent two inputs that are given an output by an add operation after the corresponding operation.

more direct approach, it gives up the possibility of finding a globally optimal model. While we believe that it is difficult to directly navigate an exponentially large search space, especially when starting the process without valid model knowledge, an alternative approach to reducing search complexity should be taken rather than simply narrowing down search space.

To address this issue, we employ a concept akin to [21] to optimize $Block_{1:N}$ through a sequential model. This approach significantly minimizes the complexity of our problem. Particularly, we parcel out the network search procedure into numerous stages for scrutinization and incrementally search the block structure within each aspirant network in the population one by one. Over the course of this procedure, the structure of the candidate network evolves from an individual block. To finalize the network, we initially fabricate the potential network structure of a singular block (i.e., composed of 1 block) in the premier growth stage and incorporate them into the population. Simultaneously, we train and assess all models within the population in a SuperNet (concurrently), developing for G generations until we achieve a converged population. Thereafter, in the second stage, we expand the population by appending a possible second block structure for each Model. At each evolutionary stage, we separately search for a corresponding block in all candidate networks in the population and extend it to the matching candidate network. Subsequently, we randomly sample the remaining blocks to form a complete candidate network to evaluate their potential. Those deemed elite Individuals proceed to the next generation population. At the climax of each evolutionary stage, the candidate network in the current population is retained as the base network for the subsequent evolutionary stage, and the process is iterated. For specific procedures, refer to Algorithm 1.

Genome Encoding. This paper provides a hybrid real-coded representation of the network structure under constraints, unlike most previous NAS work which uses binary adjacency matrix coding to represent the network structure. Each element of the adjacency matrix uses a decimal integer to encode the operation. Afterwards, by row-major vectorization (that is, flattening) the adjacency matrix, we can encode any architecture in the search space. See Figure 2 (bottom) for specific representation.

Algorithm 1 The overall framework of G-EvoNAS

Input: B (max num blocks), G (number of evolutionary iterations per block), S (SuperNet), P_{num} (population number), D_{train} (train set), D_{val} (validation set), E_w (warm up epoch), E_s (epoch interval of population evolution)

- 1: Warm-up(S, D_{train}, E_w)
- 2: $Network \leftarrow \{\emptyset\}$ //Set of candidate structures
- 3: $P_0 \leftarrow \text{InitializePopulation}(P_{num}, Network)$
- 4: **for** $b \leftarrow 1 : B$ **do**
- 5: $t \leftarrow 0$
- 6: **if** $b > 1$ **then**
- 7: $\text{PruneSupernet}(S, P_t^b)$
- 8: **end if**
- 9: $\text{GrowthNetworks}(P_t^b, Block_{1:P_{num}}^b)$
- 10: **while** $t < G$ **do**
- 11: $Q_t^b \leftarrow \text{Generate offspring from } P_t^b \text{ using genetic operators}$
- 12: $P_t^b \leftarrow P_t^b \cup Q_t^b$
- 13: $\text{TrainSupernet}(S, D_{train}, E_s, P_t^b)$
- 14: $\text{Evaluate}(P_t^b, S, D_{val})$ //Evaluate the fitness of individuals in P_t^b on D_{valid}
- 15: $P_{t+1}^b \leftarrow \text{Using pNSGAIII Select from } P_t^b$
- 16: $t \leftarrow t + 1$
- 17: **end while**
- 18: **end for**

Output: P_t^B as the final result.

Crossover and Mutation Operator. In the population-based search method, potential population individuals can be used in parallel to build a new network structure through crossover. This paper designs a constrained uniform crossover operator that can cross the connections and operations of the network at the same time. In order to enhance the diversity of the population (with different network architectures) and the ability to jump out of the local optimum, we design two main mutations. The difference between the two mutations is that connection mutations are used to change the connection method, while combinatorial operation mutations are used to change the operations inside the node. Figure 3 provides examples of crossover and mutation operators.

Potential Assessment of Candidates. In the process of evolution, we will gradually eliminate disadvantaged individuals in the population through environmental selection pNSGAI algorithm [47]. The factor we consider here is the potential of each individual architecture at this time. We define the potential of an individual as $P(\mathcal{N})$, containing the expected validation accuracy of the model and the expected model size:

$$P(\mathcal{N}) = E_{\mathcal{N} \in \{\mathcal{N} | \mathcal{N}_i = Block^b, \forall b \leq B\}} (Acc(\mathcal{N}), Size(\mathcal{N})). \quad (1)$$

Included in this process, the anticipated validation accuracy for model $\mathcal{N}$ is denoted by $Acc(\mathcal{N})$. The prospective model size for model $\mathcal{N}$ is represented by $Size(\mathcal{N})$. With uniform sampling of valid models, each chosen model uses a mini-batch to calculate its validation accuracy. The average accuracy across

each sampled model over the full validation set is used to estimate the expected validation accuracy for model $\mathcal{N}$. This process also incorporates the average model size of each sampled model to predict the expected model size for model $\mathcal{N}$. We discerned that given a sufficient count of sampled models, the potential of candidate models remains quite consistent and distinguishable among the models.

3.3 Supernet training and pruning

Inaccurate evaluation in NAS. The most time-consuming operation in the NAS process is performance evaluation. This is because to accurately evaluate the candidate network, it is necessary to train it from scratch and test its real performance until it converges. However, the performance of this method The evaluation cost is high and obviously unacceptable. In order to speed up the process of performance evaluation, recent works [22,31] do not recommend training candidate networks from scratch until convergence, because they believe that it is unwise to directly discard the previously trained network weights, which can be reused to train different candidate architectures simultaneously by using shared network parameters. Specifically, they define the search space $\mathcal{S}$ as a SuperNet, such that each candidate network $\mathcal{N}$ is its subnet. Let $\mathcal{W}$ represent the weight parameter of the SuperNet, and the training of the SuperNet is as follows:

$$\mathcal{W}^* = \arg \min_{\mathcal{W}} L_{train}(\mathcal{S}, \mathcal{W}; X, Y), \quad (2)$$

where L_{train} represents the training loss, while X and Y represent the input image and the corresponding label. Then, the performance of different candidate architectures is evaluated by sharing the optimal weight parameter $\mathcal{W}^*$ obtained after training. However, the optimal weight parameter $\mathcal{W}^*$ is optimal for the SuperNet, but it does not mean that $\mathcal{W}^*$ is the optimal weight parameter for each candidate architecture, because the subnet has not been fairly and fully trained, and the evaluation using $\mathcal{W}^*$ cannot Rank candidate models correctly because the search space is often large. Inaccuracies in assessment contribute to the ineffectiveness of existing NAS.

Supernet pruning. [4,20] show that when the search space is small and all candidates are fully and fairly trained, the evaluation can be accurate. In order to improve the accuracy of evaluation, we prune the SuperNet into blocks. Specifically, let $\mathcal{N}$ denote SuperNet. We divide $\mathcal{S}$ into B blocks according to the depth of the SuperNet and have:

$$\mathcal{S} = \mathcal{S}_1 \circ \dots \mathcal{S}_i \circ \mathcal{S}_{i+1} \dots \circ \mathcal{S}_B, \quad (3)$$

where $\mathcal{S}_i \circ \mathcal{S}_{i+1}$ represents the $(i+1)$ -th block initially connected to the i -th block in the SuperNet. We then prune each block of the SuperNet in turn using:

$$\mathcal{S}_{1:i}^* = \bigcup_{j=1}^{P_{num}} \mathcal{N}_j \quad (\mathcal{N}_j \in P_G^i) \quad i = 2, 3, \dots, B, \quad (4)$$

where $\mathcal{S}_{1:i}^*$ represents the SuperNet after pruning from the 1st block to the i -th block, $\mathcal{N}_j$ represents the subnet in $\mathcal{A}_{1:i}^*$, and P_G^i is the final population at the i -th growth stage. For more accurate evaluation subnet, merge the two parts of the SuperNet and then train the pruned SuperNet:

$$\mathcal{W}^* = \arg \min_{\mathcal{W}} L_{train}(\mathcal{S}_{1:i}^*, \mathcal{S}_{(i+1):B}; X, Y), \quad (5)$$

where $\mathcal{S}_{i:B}$ represents the unpruned remaining blocks in the SuperNet. In order to ensure the effective reduction of weight-sharing search space in block NAS, the pruning rate is analyzed as follows. Let C denotes the number of candidate networks for the i -th block. Then the size of the search space of the i -th block is $C_i, \forall i \in [1, B]$; the size of the search space $\mathcal{S}$ is $\prod_{i=1}^B C_i$. This shows that the size of the weight-sharing search space decreases exponentially:

$$Dropout\ rate = \prod_{j=1}^i P_{num} \cdot \prod_{j=i+1}^B C_j / (\prod_{i=1}^B C_i). \quad (6)$$

In our experiments, the weight-sharing search space in the SuperNet is significantly reduced, ensuring that each candidate architecture $\mathcal{N} \in \mathcal{S}$ is fully optimized.

4 Experiments Results

In this section, we execute experiments on CIFAR10 and CIFAR100 datasets. Having acquired the optimal neural network configuration with the best performance on CIFAR10 and CIFAR100, we implemented it for the ImageNet classification task. The experimental outcomes substantiate the generalization capability of the network we have constructed.

4.1 Results on CIFAR10 and CIFAR100

Training Details. Throughout the search process, we partitioned the original 50,000 training images into sets of 40,000 for training purposes and 10,000 for validation. We constructed a compact SuperNet composed of 16 channels and 8 regional blocks. The count of nodes within each Block is fixed at 7. Further, we incorporated the label smoothing method [40] to avoid overfitting. To optimize the SuperNet, we utilized SGD. The momentum was set at 0.9, and the weight decay was recorded as 3×10^{-4} . The primary learning phase adheres to a cosine annealing schedule [23] wherein the learning rate was reduced from 0.025 to 0 progressively. We specified the batch size of the training set and validation as 256 and 512 respectively.

For comprehensive CIFAR10 training, we elongate the stacking architecture by adjusting the number of blocks and initial number of channels to 20 and 36

Table 1: PERFORMANCE COMPARISON OF G-EvoNAS WITH COMPETITORS IN TERMS OF THE CLASSIFICATION ACCURACY (%), NUMBER OF PARAMETERS AND THE SEARCH COST (GPU DAYS) ON CIFAR10 AND CIFAR100 DATASETS

Architecture	CIFAR10			CIFAR100			Search Level
	Acc (%)	P (M)	GDs	Acc (%)	P (M)	GDs	
MobileNetV2 [34]	94.56	2.1	-	77.09	2.1	-	manual
NASNet-A [50]	97.35	3.2	1800	82.19	3.2	1800	Repeated Cell
PNAS [21]	96.59	3.2	225	82.37	3.2	225	Repeated Cell
ENAS [31]	97.11	4.6	0.5	81.09	4.6	0.5	Repeated Cell
AmoebaNet-A[44] [32]	96.66	3.1	3150	81.07	3.1	3150	Repeated Cell
SNAS [43]	97.15	2.8	1.5	-	-	-	Repeated Cell
DARTS [22]	97.14	3.3	1	82.46	3.3	1	Repeated Cell
PDARTS [3]	97.50	3.4	0.3	84.08	3.6	0.3	Repeated Cell
PC-DARTS [45]	97.43	3.6	0.1	82.89	3.6	0.1	Repeated Cell
NSGANetV1-A3 [25]	97.78	2.2	27	82.77	2.2	27	Repeated Cell
CARS-I [47]	97.38	3.6	0.4	82.72	3.6	0.4	Repeated Cell
MOEA-PS [46]	97.23	3.0	2.6	81.03	5.8	5.2	Repeated Cell
Proxyless NAS [2]	97.92	5.7	1500	-	-	-	Artificial Block
NSGA-Net [24]	97.25	3.3	4	79.26	3.3	8	Artificial Block
CNN-GA [38]	96.78	2.9	35	79.47	3.3	1800	Artificial Block
AE-CNN [37]	95.30	2.0	27	79.15	5.4	36	Artificial Block
EPCNAS-C [16]	96.93	1.2	1.1	81.67	1.3	1.1	Artificial Block
SI-EvoNAS [48]	97.31	1.8	0.5	84.30	3.3	0.8	Complete Network
DNAS [44]	97.32	3.3	0.4	82.53	4.6	0.6	Complete Network
G-EvoNAS	97.52	3.2	0.2	83.38	3.3	0.2	Growth Network

respectively. The derived architecture will be trained from its inception for 600 epochs employing all training images, and test performance will be measured on the test set, utilizing the SGD optimizer with a momentum of 0.9. The initial learning rate is designated as 0.025 and a weight decay of 3×10^{-4} is applied. We employ a cosine annealing scheme [23] to progressively reduce the learning rate until it reaches 0. The batch size is standardized at 96, and the specifications of other parameters remain consistent with [22].

Comparison with State-of-the-art. Table 1 delineates the comparative analysis with state-of-the-art classification results on CIFAR10 and CIFAR100. In contrast to the manually engineered network, MobileNet V2 [34], the models we have sought after have considerably outperformed their networks. Adhering to a fair approach, we also juxtaposed with alternative neural architecture search methodologies. This included stacked Cell-based search algorithms such as NASNet-A [50], PNAS [21], AmoebaNet-A [32], DARTS [22], PDARTS [3], PC-DARTS [45], CARS-I [47], and others. Upon contrast with other networks, G-EvoNAS delivers competitive results on CIFAR10 and CIFAR100 with equivalent or lower search costs, although NSGANetV1-A3 [25] displays superior classification precision with fewer parameters than G-EvoNAS. However, the search

Table 2: COMPARISON BETWEEN G-EvoNAS AND OTHER NAS METHODS IN TERMS OF CLASSIFICATION ACCURACY (%), THE NUMBER OF PARAMETERS, AND THE CONSUMED SEARCH COST ON THE IMAGENET DATASET

Architecture	Test Acc(%)		Params(M)	Search Cost (GPU-days)	Search Level
	Top-1	Top-5			
MobileNetV2 [34]	72.0	90.4	3.4	-	manual
ShuffleNetV2 [26]	74.9	90.1	7.4	-	manual
AmoebaNet-A [32]	74.5	92.0	5.1	3150	Repeated Cell
NASNet-A [50]	74	91.6	5.3	1800	Repeated Cell
PNAS [21]	74.2	91.9	5.1	224	Repeated Cell
DARTS [22]	73.3	91.3	4.7	4	Repeated Cell
PDARTS [3]	75.6	92.6	4.9	0.3	Repeated Cell
SNAS [43]	72.7	90.8	4.3	1.5	Repeated Cell
ENAS [31]	74.3	91.9	5.1	0.5	Repeated Cell
CARS-I [47]	75.2	92.5	5.1	0.4	Repeated Cell
NSGANetV1-A3 [25]	76.2	93.0	5.0	27	Repeated Cell
MOEA-PS [46]	73.6	91.5	4.7	2.6	Repeated Cell
Proxyless NAS [2]	75.1	92.5	7.1	8.3	Artificial Block
MNASNet-A2 [41]	75.6	92.7	4.8	-	Artificial Block
EPCNAS-C2 [16]	72.9	91.5	3.0	1.2	Artificial Block
FBNet-C [42]	74.9	-	5.5	9	Artificial Block
SI-EvoNAS [48]	75.8	92.5	4.7	0.5	Complete Network
G-EvoNAS	75.5	92.5	4.6	0.2	Growth Network

expense of G-EvoNAS is significantly lower (only comprising 0.2 GD, substantially lower than 27 GD). When considered against manual block-based search algorithms, G-EvoNAS performs superiorly to networks with comparable parameters like NSGA-Net [24], CNN-GA [38], AE-CNN [37], and demonstrates superior performance on CIFAR10 and CIFAR100 with a search cost merely 0.2 GD. Although Proxyless NAS [2] exhibits better classification precision than GENet on CIFAR10, GENet deploys fewer parameters ($3.2\text{M} < 5.7\text{M}$). Upon contrast with the algorithm that conducts a direct search of the complete network, G-EvoNAS utilizes less time for seeking superior networks on CIFAR10. While it falls short of SI-EvoNAS [48] on CIFAR100 by 0.92%, the search time of G-EvoNAS is merely 1/4 of it. The system thus discovers networks with similar model sizes and performances.

4.2 Results on ImageNet

Training Details. We adhere to the common practice of training networks on ImageNet [35]. We initially preprocess every image and secure a random crop of the image of dimensions 224×224 . The trimmed image segments are subsequently randomly flipped in a horizontal manner, after which the channel mean is deducted and divided by the channel standard deviation. We employ marginally less aggressive scale augmentation for lesser models, akin to the modifications

used in [13]. In the course of testing, we alter the size of the input image to 256×256 and deploy a center crop of size 224×224 as network input.

We utilize the SGD optimizer with a momentum of 0.9 and a weight decay of 3×10^{-5} . The initial learning rate is set to 0.1, and a cosine annealing scheme [23] is deployed to decrement the learning rate until it zeroes out. We use NVIDIA A100-SXM4-80GB for training with a batch size of 256. A maximum of 250 epochs are trained. Label smoothing of 0.1 and dropout with probability 0.3 were also amalgamated during training.

Comparison with State-of-the-art. We assess the transferability of the search architecture by tutoring it on the ILSVRC2012 dataset. The outcomes on ILSVRC2012 are depicted in Table 2. G-EvoNAS exceeds manually crafted models, inclusive of MobileNetV2 [34] and ShuffleNetV2 [26], along with preceding state-of-the-art NAS models. This is particularly the case for NASNet [50] and AmoebaNet [32], which are emblematic of RL-based and EA-based methodologies, respectively. They consume more GPUs and days than our method. Upon contrast with alternative advanced NAS models, the stacked Cell-based search algorithms NSGANetV1-A3 [25] and PDARTS [3] outrank G-EvoNAS in performance by 0.7% and 0.1% respectively. However, the model sizes of both are larger than that of G-EvoNAS, and the search duration of G-EvoNAS is shorter ($0.2 < 0.3 < 27$). Upon contrast with PNAS [21], GENet utilizes lesser parameters ($4.6\text{M} < 5.1\text{M}$) and augmentation in top-1 and top-5 classification. The rates underwent an augmentation of 1.3% and 0.6% respectively. In the artificial block-based search algorithm and the algorithm that directly searches the complete network, the comparison of the MNasNet-A2 [41] and SI-EvoNAS [48] networks, albeit the performance is marginally superior to G-EvoNAS, finds our model size to be smaller. The search cost is also less, especially for MNasNet-A2 [41], our search cost is exponentially reduced. Therefore, the G-EvoNAS search process is efficient and exhibits massive potential in searching for exceptional models.

4.3 Ablation Study

In this section, we investigate the effect of hyperparameters and discuss the effectiveness of our Supernet pruning and search methods.

Impact of hyperparameters We examine the influence of hyperparameters on our methodology on the CIFAR10 dataset. The hyperparameters encompass the interval training generation E_s , the populace size P_{num} , and the stage evolution generation G . We endeavored setting E_s to 50 and 100, maintaining $P_{num} = 10$ and $G = 30$. Experiments unveiled that the two resulting models did not exhibit sizable differences in reaching top-1 accuracy (less than 0.05%). Consequently, we select the interval training generation E_s to be 50 in our experiments to conserve computing resources. Pertaining to the effects of P_{num} and G , we exemplify the results in Figure 3 (a). It is observable that the top-1 accuracy of the model discovered by G-EvoNAS doesn't inflate with the increase of P_{num} and G . Consequently we select $P_{num} = 10$, $G = 30$ in our experiments for the potential of superior performance. We do not discern significant advancements when elevating these two hyperparameters any further in our experiment.

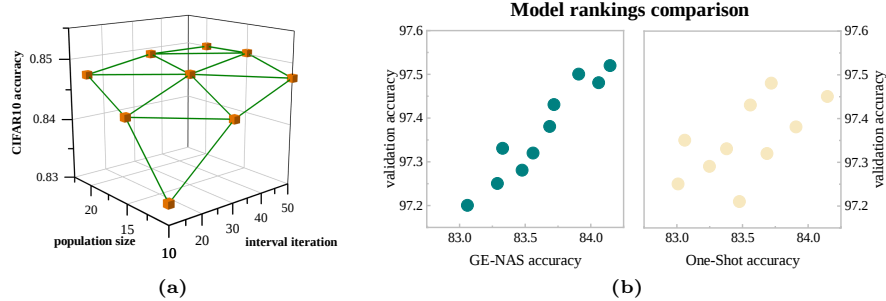


Fig. 3: (a)Hyperparametric analysis on CIFAR10. We chose population sizes of 10,15, and 20 and number of growth phase iterations of 10, 30, and 50. (b)Comparison of model rankings for G-EvoNAS (left) and One-Shot (right).

Effectiveness of our search method Given the complexity in conveniently seeking the optimal solution in the enormous space of global search, we embraced an evolutionary algorithm based on network growth for the search. With the intention to attest the effectiveness of our search methodology, we utilized the traditional EA method as a benchmark for the search by using One-Shot to train the model in the same global hypergraph. We executed a traditional EA with parameters identical to our methodology and searched directly for the comprehensive model. It was discerned that the top-1 accuracy of the model degenerated to 96.6%, a decline of 0.9% compared to G-EvoNAS. Due to this, it failed to parallel our search strategy. We ascertain that the performance discrepancy originates from the effectiveness of our network thriving method. In the G-EvoNAS approach, a single block is searched at one instance, and each architecture is evaluated in batches of a size equal to 512. Subsequently, the mean accuracy and average model size are quantified to symbolize the potential of each block. In contrast, the classic EA method necessitates simultaneous search of all blocks and the performance of the entire network must be evaluated. This leads to reduced search efficacy, especially in a large search space. Seeking the globally optimal solution demands superior performance of the search algorithm. As visible in Figure 4, the efficacy of our novel evolutionary search methodology is clearly demonstrated.

Effectiveness of Supernet Pruning A distinct advantage of the One-Shot technique is that the shared weights of the hypergraph can be employed to conveniently predict the performance of various architectures. Nonetheless, earlier models [9,21] exhibit inadequate results when ranking models. To ascertain the degree to which SuperNet pruning can mitigate this issue, we performed the following comparison. Firstly, we select a bunch of models from the individuals in the final population under a partitioned search space, train them from scratch, and evaluate their independent top-1 accuracy. Subsequently, we engage One-Shot to traditionally train the hypergraph beneath a block search space. Finally, we present the model rankings of G-EvoNAS and One-Shot using the

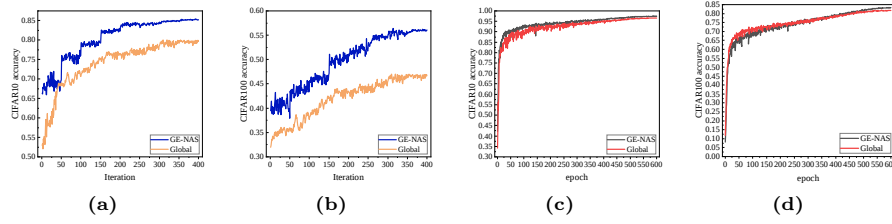


Fig. 4: Comparing the effectiveness of G-EvoNAS using network growing and SuperNet pruning on CIFAR10 and CIFAR100. In (a) and (b), the orange line shows the best fitness in each generation during the evolution iteration. The blue line marks the best validation accuracy of the models that were sampled by the global search in each iteration of the parameter sharing method. In (c) and (d), we plot the validation accuracy of the two networks found by both methods during the training. Each network is trained for 600 epochs.

accuracy obtained by model inference in hypergraphs trained by both methods. The divergence is visible in Figure 3 (b). The Kendall correlation coefficients for One-Shot and G-EvoNAS stand at 0.38 and 0.87 respectively. Hence, the models under G-EvoNAS SuperNet pruning can be ordered based on the precision of shared weight evaluation, which is visibly superior compared to that of One-Shot. Repeated experiments also unveiled analogous phenomena.

5 Conclusion

The Neural Architecture Search (NAS) holds the capacity to autonomously devise high-performance models. Diverging from preceding approaches that merely search local architectures, this paper proposes a block search space founded on the global scale. Simultaneously, with the objective to effectively explore superior models in the block search space, we propose an evolutionary neural architecture search based on network growth. Throughout this evolutionary journey, G-EvoNAS optimally harnesses the knowledge garnered in the latest generational evolution, specific to architecture and parameters. While utilizing pruned SuperNets to augment individual evaluation efficiency and maintain individual ranking consistency, experiments conducted on benchmark datasets reveal that the proposed G-EvoNAS can identify models within a shorter timeline, and with superior performance in model size/latency and accuracy parameters. It outstrips the performance of existing state-of-the-art models.

References

1. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Smash: one-shot model architecture search through hypernetworks. arXiv preprint arXiv:1708.05344 (2017)
2. Cai, H., Zhu, L., Han, S.: Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332 (2018)
3. Chen, X., Xie, L., Wu, J., Tian, Q.: Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1294–1303 (2019)
4. Chu, X., Zhang, B., Xu, R.: Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In: Proceedings of the IEEE/CVF International Conference on computer vision. pp. 12239–12248 (2021)
5. Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., Yang, S.: Adanet: Adaptive structural learning of artificial neural networks. In: International conference on machine learning. pp. 874–883. PMLR (2017)
6. Fernandes, F.E., Yen, G.G.: Automatic searching and pruning of deep neural networks for medical imaging diagnostic. *IEEE Transactions on Neural Networks and Learning Systems* **32**(12), 5664–5674 (2020)
7. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 1440–1448 (2015)
8. Grosse, R., Salakhutdinov, R.R., Freeman, W.T., Tenenbaum, J.B.: Exploiting compositionality to explore a large space of model structures. arXiv preprint arXiv:1210.4856 (2012)
9. Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., Sun, J.: Single path one-shot neural architecture search with uniform sampling. In: Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16. pp. 544–560. Springer (2020)
10. Han, K., Wang, Y., Shu, H., Liu, C., Xu, C., Xu, C.: Attribute aware pooling for pedestrian attribute recognition. arXiv preprint arXiv:1907.11837 (2019)
11. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
13. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
14. Huang, F., Ash, J., Langford, J., Schapire, R.: Learning deep resnet blocks sequentially using boosting theory. In: International Conference on Machine Learning. pp. 2058–2067. PMLR (2018)
15. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
16. Huang, J., Xue, B., Sun, Y., Zhang, M., Yen, G.G.: Particle swarm optimization for compact neural architecture search for image classification. *IEEE Transactions on Evolutionary Computation* (2022)
17. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17–21, 2011. Selected Papers 5. pp. 507–523. Springer (2011)

18. Kenton, J.D.M.W.C., Toutanova, L.K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of naacL-HLT. vol. 1, p. 2 (2019)
19. Levine, J.A.: Non-exercise activity thermogenesis (neat). *Nutrition reviews* **62**(suppl_2), S82–S97 (2004)
20. Li, X., Lin, C., Li, C., Sun, M., Wu, W., Yan, J., Ouyang, W.: Improving one-shot nas by suppressing the posterior fading. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition. pp. 13836–13845 (2020)
21. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: Proceedings of the European conference on computer vision (ECCV). pp. 19–34 (2018)
22. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055* (2018)
23. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016)
24. Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y., Deb, K., Goodman, E., Banzhaf, W.: Nsga-net: neural architecture search using multi-objective genetic algorithm. In: Proceedings of the genetic and evolutionary computation conference. pp. 419–427 (2019)
25. Lu, Z., Whalen, I., Dhebar, Y., Deb, K., Goodman, E.D., Banzhaf, W., Boddeti, V.N.: Multiobjective evolutionary design of deep convolutional neural networks for image classification. *IEEE Transactions on Evolutionary Computation* **25**(2), 277–291 (2020)
26. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European conference on computer vision (ECCV). pp. 116–131 (2018)
27. Mendoza, H., Klein, A., Feurer, M., Springenberg, J.T., Hutter, F.: Towards automatically-tuned neural networks. In: Workshop on automatic machine learning. pp. 58–65. PMLR (2016)
28. Nayman, N., Noy, A., Ridnik, T., Friedman, I., Jin, R., Zelnik, L.: Xnas: Neural architecture search with expert advice. *Advances in neural information processing systems* **32** (2019)
29. Negrinho, R., Gordon, G.: Deeparchitect: Automatically designing and training deep architectures. *arXiv preprint arXiv:1704.08792* (2017)
30. Noy, A., Nayman, N., Ridnik, T., Zamir, N., Doveh, S., Friedman, I., Giryes, R., Zelnik, L.: Asap: Architecture search, anneal and prune. In: International Conference on Artificial Intelligence and Statistics. pp. 493–503. PMLR (2020)
31. Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J.: Efficient neural architecture search via parameters sharing. In: International conference on machine learning. pp. 4095–4104. PMLR (2018)
32. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proceedings of the aaai conference on artificial intelligence. vol. 33, pp. 4780–4789 (2019)
33. Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q.V., Kurakin, A.: Large-scale evolution of image classifiers. In: International conference on machine learning. pp. 2902–2911. PMLR (2017)
34. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018)
35. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014)

36. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. *Evolutionary computation* **10**(2), 99–127 (2002)
37. Sun, Y., Xue, B., Zhang, M., Yen, G.G.: Completely automated cnn architecture design based on blocks. *IEEE transactions on neural networks and learning systems* **31**(4), 1242–1254 (2019)
38. Sun, Y., Xue, B., Zhang, M., Yen, G.G., Lv, J.: Automatically designing cnn architectures using the genetic algorithm for image classification. *IEEE transactions on cybernetics* **50**(9), 3840–3854 (2020)
39. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 1–9 (2015)
40. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 2818–2826 (2016)
41. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 2820–2828 (2019)
42. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 10734–10742 (2019)
43. Xie, S., Zheng, H., Liu, C., Lin, L.: Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926* (2018)
44. Xu, K., He, G.: Dnas: A decoupled global neural architecture search method. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 1979–1985 (2022)
45. Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.J., Tian, Q., Xiong, H.: Pc-darts: Partial channel connections for memory-efficient architecture search. *arXiv preprint arXiv:1907.05737* (2019)
46. Xue, Y., Chen, C., Słowik, A.: Neural architecture search based on a multi-objective evolutionary algorithm with probability stack. *IEEE Transactions on Evolutionary Computation* (2023)
47. Yang, Z., Wang, Y., Chen, X., Shi, B., Xu, C., Xu, C., Tian, Q., Xu, C.: Cars: Continuous evolution for efficient neural architecture search. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 1829–1838 (2020)
48. Zhang, H., Jin, Y., Cheng, R., Hao, K.: Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance. *IEEE Transactions on Evolutionary Computation* **25**(2), 371–385 (2020)
49. Zhang, M., Li, H., Pan, S., Chang, X., Zhou, C., Ge, Z., Su, S.: One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **43**(9), 2921–2935 (2020)
50. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016)