G-ICSO-NAS: Shifting Gears between Gradient and Swarm for Robust Neural Architecture Search

1st Xingbang Du†, 2nd Enzhi Zhang‡, 3rd Rui Zhong‡, 4th Yang Cao†, 5th Masaharu Munetomo‡
†*Information Systems Design Laboratory, Hokkaido University*, Sapporo, Japan
‡*Information Initiative Center, Hokkaido University*, Sapporo, Japan
xingbang.du.d1@elms.hokudai.ac.jp, {zhangenzhi, zhongrui, munetomo}@iic.hokudai.ac.jp, yang.cao.y4@elms.hokudai.ac.jp

***Abstract*—Neural Architecture Search (NAS) has become a pivotal technique in automated machine learning. Evolutionary Algorithm (EA)-based methods demonstrate superior search quality but suffer from prohibitive computational costs, while gradient-based approaches like DARTS offer high efficiency but are prone to premature convergence and performance collapse. To bridge this gap, we propose G-ICSO-NAS, a hybrid framework implementing a three-stage optimization strategy. The Warm-up Phase pre-trains supernet weights (w) via differentiable methods while architecture parameters (α) remain frozen. The Exploration Phase adopts a hybrid co-optimization mechanism: an Improved Competitive Swarm Optimizer (ICSO) with diversity-aware fitness navigates the architecture space to update α, while gradient descent concurrently updates w. The StablePhase employs fine-grained gradient-based search with early stopping to converge to the optimal architecture. By synergizing ICSO's global navigation capability with differentiable methods' efficiency, G-ICSO-NAS achieves remarkable performance with minimal cost. In the context of the DARTS search space, an accuracy of 97.46% is achieved on CIFAR-10 with a computational budget of just 0.15 GPU-Days. The method also exhibits strong transfer potential, recording accuracies of 83.1% (CIFAR-100) and 75.02% (ImageNet). Furthermore, regarding the NAS-Bench-201 benchmark, G-ICSO-NAS is shown to deliver state-of-the-art results across all evaluated datasets.**

***Index Terms*—Neural Architecture Search, Competitive Swarm Optimizer, Differentiable Architecture Search**

I. Introduction

Deep Convolutional Neural Networks (CNNs) have emerged as the predominant approach for image processing, demonstrating exceptional efficacy in various computer vision applications, largely attributed to the evolution of landmark architectures represented by VGG [1], ResNet [2], as well as MobileNet [3] and EfficientNet [4], which have established performance benchmarks for visual representation learning. Nevertheless, the conventional hand-crafting of neural architectures necessitates significant domain expertise and iterative testing, making the procedure notoriously time-consuming and resource-demanding.

Consequently, Neural Architecture Search (NAS) has emerged as an automated paradigm to alleviate the substantial burden of manual architecture design. While initial NAS strategies based on Evolutionary Algorithms (EAs) and Reinforcement Learning (RL) attained superior performance, their feasibility was severely compromised by an overwhelming computational burden. [5]. To address this efficiency bottleneck, Differentiable Architecture Search (DARTS) [6] was proposed, which relaxes the discrete search space into a continuous one, enabling efficient gradient-based optimization and dramatically accelerating the search process. Nevertheless, DARTS suffers from critical robustness challenges in its supernet training. The most prominent issue is the so-called "collapse" phenomenon [7], where the search process exhibits a strong bias toward parameter-free operations (e.g., none or skip_connect) in early stages, leading to degraded final performance. Additionally, the inherent instability of the bilevel optimization process often results in suboptimal discrete architectures.

In contrast, EAs demonstrate complementary strengths to gradient-based differentiable methods in the NAS domain. While evolutionary approaches offer superior exploration capabilities and robustness to local optima at the cost of computational efficiency, differentiable methods excel in search speed through gradient-based optimization but suffer from the aforementioned stability issues [8]. This fundamental complementarity in their computational efficiency and architecture quality trade-offs provides a strong theoretical foundation for their synergistic integration.

This study proposed G-ICSO-NAS, a novel three-stage method that navigates the NAS landscape by strategically shifting gears between gradient-driven descent and swarm-based exploration, striking a robust balance between computational efficiency and architecture quality. Our core innovations are as follows:

We designed Double-helix parameter update mechanism to achieve high quality of solution: During the exploration stage, architecture parameters α are updated by the ICSO algorithm through swarm-based evaluation (requiring only forward propagation), while super-net weights w are updated through standard epoch-based training. Compare to DARTS, ICSO's population-based evaluation of multiple α configurations preserves diversity.

Multi-dimensional fitness function is designed to avoid local optimal:A validation-accuracy-based function selects the best individual to guide the update of architecture parameters α, while swarm diversity and operation diversity functions, combined with loss-based fitness, jointly guide the swarm evolution direction.The validation-loss-based fitness ensures consistent progression toward high-accuracy architectures, while the

other two mitigate the risk of entrapment in local optima—an advantage unattainable when relying solely on validation loss as the optimization objective.

Adaptive termination strategy aims to reduce computational costs: In the stable stage, we employ the DARTS with a convergence testing mechanism for adaptive search termination.

Finally, the proposed method has been extensively validated across different datasets and search spaces: On CIFAR-10, the search achieves 97.46% accuracy with only 0.15 GPU-days. The discovered architecture demonstrates excellent transferability, attaining 83.10% accuracy on CIFAR-100 and 75.02% (Top-1) / 92.51% (Top-5) accuracy on ImageNet. Specifically within the search space of NAS-Bench-201, our method identifies the architecture with the highest test accuracy, surpassing all compared state-of-the-art methods.The code is publicly available at https://github.com/hikkidxbjp/G-ICSO-NAS.

II. Related Work

A. Evolution Algorithm for NAS

NAS marks a milestone in deep learning's transition from manual tuning to automated design. This field was inaugurated by reinforcement learning (RL)-based methods from Google Brain, demonstrating machines' capability to discover state-of-the-art architectures within vast search spaces. To improve efficiency, researchers subsequently introduced cell-based search spaces [9].

Concurrently, evolutionary algorithm (EA)-based approaches achieved remarkable results. Through mutation and tournament selection, EAs exhibit strong global search robustness, successfully yielding AmoebaNet—an architecture surpassing contemporary hand-designed models. These studies demonstrated EAs' potential [10] for discovering non-intuitive architectures. Although first-generation methods achieved peak performance, their prohibitive computational costs—often thousands of GPU hours—motivated more efficient paradigms, paving the way for parameter sharing techniques like ENAS [5] and differentiable search methods.

B. Differentiable architecture search

DARTS [6] pioneered the differentiable search paradigm by projecting discrete spaces into a continuous domain. This relaxation enables gradient-based updates, where the search efficiency is achieved by framing the learning process as a bilevel optimization problem that alternates between architecture parameters and model weights.Specifically, the output of each edge in the search cell is computed as a softmax-weighted combination of candidate operations:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathscr{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathscr{O}} \exp(\alpha_{o'}^{(i,j)})} o(x) \tag{1}$$

where $\mathscr{O}$ denotes the set of candidate operations, and α represents the architecture parameters. DARTS alternately optimizes the network weights w and architecture parameters α via gradient descent, enabling efficient end-to-end search.

Subsequent variants addressed its limitations: PC-DARTS [11] improved memory efficiency through partial channel sampling, enabling search on larger-scale datasets; P-DARTS [12] bridged the depth gap between search and evaluation via progressive depth increase and search space regularization; R-DARTS [7] analyzed failure mechanisms of the original algorithm and proposed Hessian eigenvalue monitoring to prevent performance degradation; and FBNet [13] incorporated hardware-aware latency prediction, enabling joint optimization of accuracy and inference speed for resource-constrained deployment.

C. Competitive Swarm Optimizer

To tackle high-dimensional optimization problems, the Competitive Swarm Optimizer (CSO) [14] was introduced as a specialized PSO variant. It eliminates the reliance on global best tracking in favor of a pairwise competition mechanism, where losing particles update their positions based on the winners. Instead, particles are randomly paired for competition: the loser learns from the winner, while the winner passes directly to the next generation. The loser particle updates its velocity and position as follows:

$$v_l(t+1) = R_1 \cdot v_l(t) + R_2 \cdot (x_w(t) - x_l(t)) + \phi R_3 \cdot (\bar{x}(t) - x_l(t)) \tag{2}$$

where x_w and x_l denote the winner and loser positions respectively, $\bar{x}$ is the swarm mean position, R_1, R_2, R_3 are random vectors in $[0,1]$, and ϕ controls the influence of the swarm centroid.

To further enhance search efficiency, several notable extensions have been proposed: SL-PSO [15] introduces a social learning mechanism allowing particles to learn from any better demonstrators to improve population diversity; and LLSO [16] employs a level-based structure where particles learn from superior hierarchy levels to better balance exploration and exploitation. These variants have demonstrated robust performance on high-dimensional benchmarks, establishing the learning-based swarm framework as a powerful alternative to classical PSO.

III. Methodology

In this section, we will first present a comprehensive overview of the proposed G-ICSO-NAS method. Subsequently, the detailed designs of the Warmup Stage, Exploration Stage, and Stable Stage are elaborated in sub-sections B, C, and D, respectively.

A. Overview Of G-ICSO-NAS

Figure 1 depicts the overall workflow of the G-ICSO-NAS method. It comprises three distinct phases: In the Warmup Stage, we adhere to the supernet design principles of DARTS to train and update the supernet weights w, while the architecture parameters α remain frozen. The Exploration Stage uses ICSO to search for optimal architecture parameters α and uses supernet training to update the super-net weights w via gradient descent. In the Stable Stage, the method reverts to a gradient-based DARTS approach and we terminate the search process upon satisfying the Hoeffding Inequality-based convergence test.

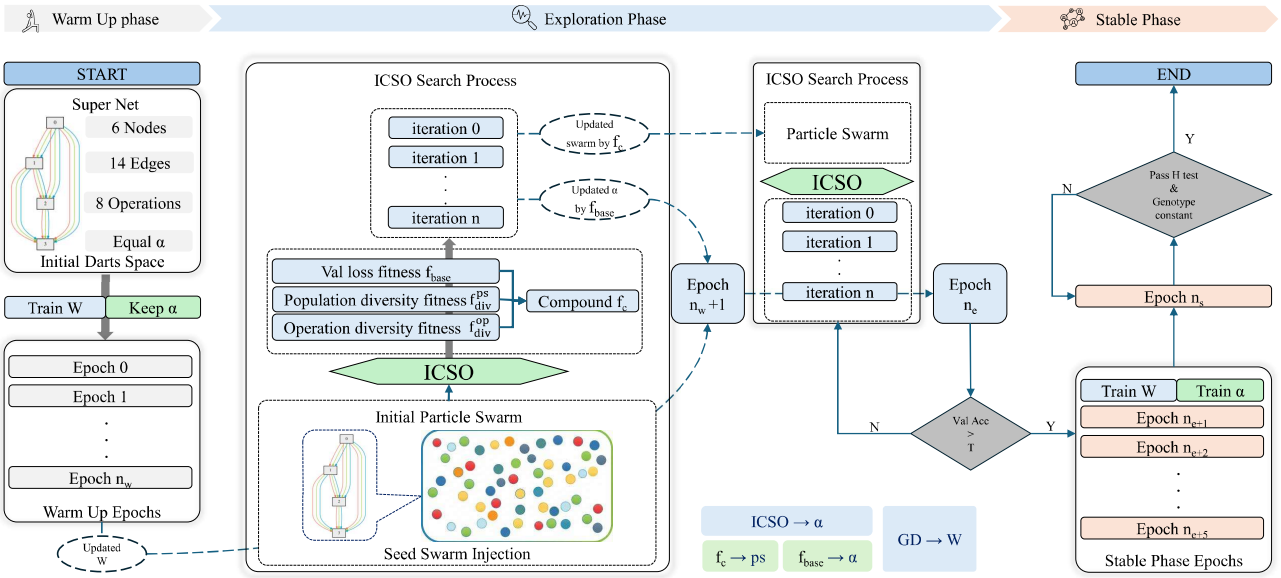

Fig. 1: The overview of G-ICSO-NAS. The warm-up stage trains super-net weights W with unchanged architecture weights α; the exploration stage alternates between ICSO-based α updates and gradient-based W updates; the stable stage performs gradient-based optimization with Hoeffding-based early stopping for convergence detection.

B. Warm-Up Stage

In this subsection, we elaborate on the first phase: the Warm-up Stage. As illustrated in the left panel of Fig. 1, a supernet is constructed based on the DARTS search space to serve as the initialization for architecture search. Each intermediate node j in the cell aggregates information from all predecessor nodes:

$$x^{(j)} = \sum_{i<j} \bar{o}^{(i,j)}(x^{(i)}) \tag{3}$$

where the mixed operation $\bar{o}^{(i,j)}$ is defined as a softmax-weighted sum over all candidate operations:

$$\bar{o}^{(i,j)}(x) = \sum_{o\in\mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o'\in\mathcal{O}}\exp(\alpha_{o'}^{(i,j)})} o(x) \tag{4}$$

where $\mathcal{O}$ denotes the set of candidate operations, and $\alpha_o^{(i,j)}$ represents the architecture weight for operation o on edge (i,j).

Subsequently, the supernet is trained to update the network weights w using training data, while the architecture parameters $\boldsymbol{\alpha}$ remain frozen (by setting the architecture learning rate to zero), ensuring equal probability for all operations:

$$\begin{aligned} w_{t+1} &\leftarrow w_t - \eta_w \nabla_w \mathscr{L}_{\text{train}}(w_t, \boldsymbol{\alpha}_{\text{init}}) \\ \boldsymbol{\alpha}_{t+1} &\leftarrow \boldsymbol{\alpha}_t = \boldsymbol{\alpha}_{\text{init}} \end{aligned} \tag{5}$$

Upon completion of the predefined warmup epochs, the warmup phase terminates. At this point, the supernet weights w have stabilized, preparing the method for the subsequent exploration phase.

C. Exploration Stage

In this subsection, we detail the Exploration Stage.

We begin by introducing the Improved Competitive Swarm Optimizer (ICSO), which is a simplified variant of L-ICSO [17], Improved Competitive Swarm Optimizer with Linear Population Reduction. The core advantage of L-ICSO is the novel triplet individual competition mechanism to enhance optimization performance. Analysis of the experimental results of CEC2017 benchmark functions demonstrates that L-ICSO possesses significant advantages in solving high-dimensional problems, validating both its convergence speed and solution quality through statistical tests.

Regarding the search within the DARTS space, due to the continuous relaxation of the architecture representation (where each edge is parameterized by a vector of operation weights), this search task is mathematically equivalent to a 224-dimensional optimization problem:

$$D = 2 \times K \times |\mathcal{O}| = 2 \times \left(\sum_{i=0}^{N-1} (i+2) \right) \times |\mathcal{O}| \tag{6}$$

where K denotes the number of edges per cell, N is the number of intermediate nodes, and $|\mathcal{O}|$ is the number of candidate operations.The architecture search is formulated as:

$$\min_{\boldsymbol{x}\in\mathbb{R}^D} \quad f(\boldsymbol{x}) = \mathscr{L}_{\text{val}}(w^*, \boldsymbol{x}) \tag{7}$$

where $\boldsymbol{x} = [\text{vec}(\boldsymbol{\alpha}_{\text{normal}}), \text{vec}(\boldsymbol{\alpha}_{\text{reduce}})]$ is the concatenated architecture parameter vector.

For this high-dimensional optimization problem, the L-ICSO algorithm, distinguished by its optimal balance of ef-

ficiency and effectiveness, is exceptionally well-suited in the exploration phase.

Secondly, to preserve architectural diversity, we discard the swarm reduction mechanism and introduce two novel diversity-aware fitness functions into ICSO. Specifically, the swarm Diversity Function is designed to maintain individual diversity throughout the iterative process:

$$f_{div}^{swarm}(\vec{x}) = \tanh\left(\min_{\boldsymbol{x}' \in \mathscr{H}} \frac{\|\boldsymbol{x} - \boldsymbol{x}'\|_2}{\sqrt{D}}\right) \tag{8}$$

where $\mathscr{H}$ is the historical swarm set and D is the parameter dimension.

The Operation Diversity Function is formulated to mitigate the collapse of the search space, suppressing the dominance of parameter-free operations (e.g., none and skip_connect) often observed in the early stages of DARTS:

$$f_{div}^{op}(\vec{x}) = \frac{-\sum_{o \in \mathscr{O}} p_o \log p_o}{\log |\mathscr{O}|} \tag{9}$$

where p_o is the selection frequency of operation o in the architecture, and $|\mathscr{O}|$ is the number of candidate operations.

By integrating these two diversity metrics with the Basic Fitness Function which derived from the architecture's forward propagation loss,

$$f_{\text{base}}(\boldsymbol{x}) = \frac{\mathscr{L}_{\text{val}}(\boldsymbol{x}) - L_{\min}}{L_{\max} - L_{\min}} \tag{10}$$

where $\mathscr{L}_{\text{val}}(\boldsymbol{x})$ is the cross-entropy loss on the validation set, $L_{\min}$ and $L_{\max}$ are the expected loss bounds. Lower fitness indicates better architecture.We propose a Combined Fitness Function for swarm updates:

$$f_c(\vec{x}) = f_{\text{base}}(\boldsymbol{x}) - \lambda_{\text{swarm}} \cdot f_{div}^{swarm}(\vec{x}) - \lambda_{\text{op}} \cdot f_{div}^{op}(\vec{x}) \tag{11}$$

where λ_{swarm} and λ_{op} are the weights for particle swarm diversity and operation diversity, respectively. The subtraction encourages higher diversity since ICSO minimizes the fitness function.

Finally, Eq. (10) is utilized to select the optimal α parameters of the current iteration to guide the architecture update for the subsequent epoch, whereas Eq. (11) is employed to drive the evolution of the ICSO swarm, thereby ensuring sustained diversity.

The workflow of the refined ICSO algorithm is illustrated in Fig. 2. Initially, the fitness of the swarm is evaluated using Eq. (11), and the global optimum solution is recorded. Subsequently, the swarm indices are randomly shuffled, and the particles are partitioned into triplets. Within each triplet, the fitness values of the three particles are compared to categorize them into three hierarchical tiers: the Winner (x_w), the Second-Best (x_m), and the Loser (x_l). Finally, the Winner is directly preserved for the next generation without updating its architecture parameters α. The Second-Best particle has a 50% probability of remaining unchanged (direct replication) and a 50% probability of being updated. In the update case,

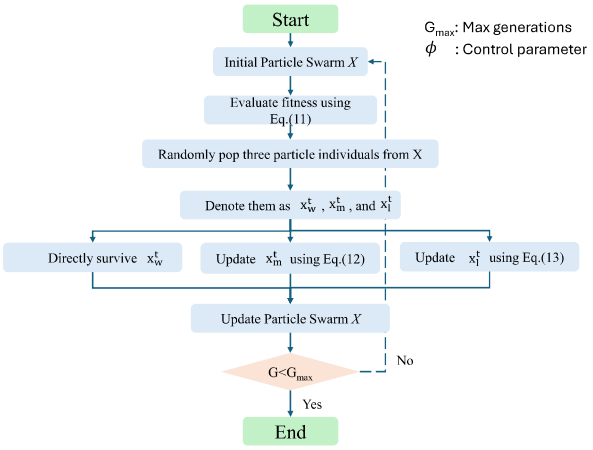

Fig. 2: ICSO. a triplet-based competitive swarm optimizer with diversity-aware fitness evaluation.

it primarily learns from the triplet's Winner while adjusting based on the swarm's centroid (x_{mean}):

$$\begin{aligned}
&\text{If rand}(0,1) < 0.5: \\
&\quad v_m^{t+1} = r_1 v_m^t + r_2 \cdot (x_w^t - x_m^t) + \phi \cdot r_3 \cdot (x_{mean}^t - x_m^t) \\
&\quad x_m^{t+1} = x_m^t + v_m^{t+1} \\
&\text{Else:} \\
&\quad x_m^{t+1} = x_m^t
\end{aligned} \tag{12}$$

Conversely, the Loser is mandatorily updated. Its trajectory is adjusted by learning from the triplet's Winner and incorporating information from the global best individual (x_{best}):

$$\begin{aligned}
v_l^{t+1} &= r_1 v_l^t + r_2 \cdot (x_w^t - x_l^t) + \phi \cdot r_3 \cdot (x_{best}^t - x_l^t) \\
x_l^{t+1} &= x_l^t + v_l^{t+1}
\end{aligned} \tag{13}$$

where ϕ is the control parameter that regulates the learning strength of the second-best or loser particle during the competitive update, and r_1, r_2, r_3 denote dimension-wise random vectors which introduce stochasticity into the update process.

The workflow of the Exploration Stage proceeds as follows. First, an initial population is randomly generated within the search space:

$$\mathscr{P}_0 = \{\boldsymbol{x}_i \mid \boldsymbol{x}_i \sim U(\boldsymbol{l}, \boldsymbol{u}),\ i = 1, \ldots, N_{\text{pop}}\} \tag{14}$$

where $\boldsymbol{l}$ and $\boldsymbol{u}$ denote the lower and upper bounds of the search space. While all individuals within this population possess distinct architecture parameters, they share the supernet weights w inherited from the Warmup Stage. Subsequently, ICSO is employed to evolve the population based on the combined fitness function, yielding a new population:

$$\mathscr{P}_{t+1} = \text{ICSO}(\mathscr{P}_t, f(\mathscr{P}_t)) \tag{15}$$

which serves as the initialization for the next epoch. The optimal individual is then selected using the f_{base}, and its architecture parameters are used to update the super-net:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathscr{P}_{t+1}} f_{\text{base}}(\boldsymbol{x}) \tag{16}$$

$$\boldsymbol{\alpha}_{t+1} \leftarrow \boldsymbol{\alpha}_t + \eta_\alpha \cdot (\boldsymbol{x}^* - \boldsymbol{\alpha}_t) \tag{17}$$

Following the architecture update, the supernet undergoes training to update the network weights:

$$w_{t+1} \leftarrow w_t - \eta_w \nabla_w \mathscr{L}_{\text{train}}(w_t, \boldsymbol{\alpha}_{t+1}) \tag{18}$$

The updated weights are then integrated with the evolved population to initiate the next iteration. This cyclic process continues until the validation accuracy exceeds a predefined threshold T_{stab}, at which point the framework transitions into the Stable Stage.

D. Stable Stage

In this stage, the super-net reverts to the standard DARTS methodology, simultaneously updating both network weights w and architecture parameters $\boldsymbol{\alpha}$. However, to facilitate fine-grained convergence towards a precise local optimum, a significantly reduced architecture learning rate is employed:

$$\boldsymbol{\alpha}_{t+1} \leftarrow \boldsymbol{\alpha}_t - \eta_\alpha^{\text{stab}} \nabla_{\boldsymbol{\alpha}} \mathscr{L}_{\text{val}}(w_t, \boldsymbol{\alpha}_t) \tag{19}$$

where $\eta_\alpha^{\text{stab}}$ denotes the stable-phase architecture learning rate. After stabilizing for a predetermined minimum number of epochs, the method initiates an architecture convergence verification based on Hoeffding's Inequality [18]. Let $V_t = \|\boldsymbol{\alpha}_t - \boldsymbol{\alpha}_{t-1}\|_2$ denote the variation of architecture parameters at epoch t, and $\bar{V} = \frac{1}{n}\sum_{i=1}^{n} V_{t-n+i}$ be the mean variation within a sliding window of size n. The search terminates when:

$$\bar{V} < \varepsilon = \sqrt{\frac{D\ln(2/\delta)}{2n}} \tag{20}$$

where D is the dimension of $\boldsymbol{\alpha}$, δ is the significance level, and $1-\delta$ is the confidence level.

IV. Experiment and Result Analysis

In this section, we first assess the performance of G-ICSO-NAS within the DARTS search space using CIFAR-10, CIFAR-100, and ImageNet for image classification tasks. Subsequently, the effectiveness of the proposed method is further examined on the NAS-Bench-201 search space with CIFAR-10, CIFAR-100, and ImageNet-16-120 datasets.

A. Experiment environment

All experiments are carried out on a Dell Precision 3630 workstation equipped with an Intel Xeon E-2224 processor, 16 GB of system memory, and an NVIDIA RTX A4000 GPU with 16 GB VRAM. The experimental framework is implemented using Python 3.11.9 and PyTorch 2.7.1, with CUDA 12.8 support.

TABLE I: Key Experiment Settings

Stage	Parameters	Value
Warm up	batch size	64
	learning rate	0.025
	warm Up Epoch	5
	arch learning rate	0
Exploration	ICSO pop size	60
	ICSO ϕ	0.15
	diversity weight	0.3
	operation diversity weight	0.2
	ICSO generations	8
Stable	stable threshold	82%
	batch size	64
	stable arch learning rate	1e-5
	early stop alpha threshold	1e-3
	early stop confidence level	0.95

B. Datasets and Parameters

The CIFAR-10 dataset [19] contains 60,000 RGB images with a spatial resolution of 32×32, distributed across 10 mutually exclusive categories (e.g., airplane, automobile, bird, and cat). Each category includes 6,000 samples, which are divided into 50,000 images for training and 10,000 images for testing.

The CIFAR-100 dataset [19] adopts the same image resolution (32×32) and overall sample size (60,000 images) as CIFAR-10, while posing a more challenging classification scenario with 100 distinct classes. Each class consists of 600 images, split into 500 training samples and 100 testing samples. The 100 fine-grained categories are further grouped into 20 coarse superclasses, making CIFAR-100 particularly suitable for both coarse- and fine-level recognition evaluation.

ImageNet [20] is a large-scale hierarchical image dataset comprising 1,000 object categories and approximately 1.28 million training images, along with 50,000 validation samples and 100,000 test images. In contrast to CIFAR datasets, ImageNet images exhibit varying resolutions and are typically resized or center-cropped to a fixed spatial dimension (e.g., 224×224) prior to model input. So it is widely regarded as a standard benchmark for evaluating large-scale visual recognition models. Architecture search is conducted on CIFAR-10 under the DARTS search space, detailed experimental settings for the search phase are listed in Table I.

NAS-Bench-201 [21] is adopted as a standardized evaluation benchmark to facilitate reproducible and computationally efficient comparisons among Neural Architecture Search (NAS) algorithms. It employs a predefined and finite search space containing 15,625 candidate neural architectures. For each architecture, exhaustive training and evaluation results on CIFAR-10, CIFAR-100, and ImageNet-16-120 are provided. The benchmark further offers a query-based interface that enables direct access to ground-truth performance indicators, such as accuracy, training time, and parameter count, under different training budgets. Following common practice, we conduct independent evaluations on NAS-Bench-201 using four distinct random seeds.

TABLE II: Comparison of Neural Architecture Search Methods on CIFAR-10 and CIFAR-100

Architecture/Method	Test Error (%)		Params	Search Cost	Method Type
	CIFAR-10	CIFAR-100	(M)	(GPU-Days)	
ResNet	4.61	22.1	1.7	-	Manual
ENAS + cutout	2.89	-	4.6	0.5	RL
AmoebaNet-A	3.34	18.93	3.2	3150	Evolution
NSGA-Net	2.75	20.74	3.3	4.0	Evolution
NSGANetV1-A2	2.65	-	0.9	27	Evolution
EPCNAS-C	3.24	18.36	1.44	1.2	Evolution
EAEPSO	2.74	16.94	2.94	2.2	Evolution
DARTS (1st)	3.00	-	3.3	1.5	Gradient
DARTS (2st)	2.76	-	3.3	4.0	Gradient
SNAS (moderate) + cutout	2.85	17.55	2.8	1.5	Gradient
ProxylessNAS + cutout	**2.02**	-	-	4.0	Gradient
GDAS	2.93	18.38	3.4	0.2	Gradient
BayesNAS	2.81	-	3.4	0.2	Gradient
PC-DARTS + cutout	2.57	16.90	3.6	**0.1**	Gradient
DARTS-	2.59	17.51	3.4	0.4	Gradient
DrNAS	2.54	-	4.0	0.4	Gradient
DARTS+PT	2.61	-	3.0	0.8	Gradient
G-ICSO-NAS	2.54	**16.90**	4.24	0.15	Hybrid

C. Experiment Result and Comparison on DARTS search space

The comparative performance of G-ICSO-NAS against representative NAS approaches on CIFAR-10 and CIFAR-100 is summarized in Table II.On CIFAR-10, G-ICSO-NAS achieves a test error of 2.54% while requiring only 0.15 GPU-days for architecture search, demonstrating clear advantages over established baselines such as DARTS baseline, SNAS, BayesNAS, and NSGA-Net. Although ProxylessNAS [22] report marginally lower error rates, their search costs are substantially higher, reaching 4.0, respectively, which exceeds that of G-ICSO-NAS by a notable margin.To further examine the transferability of the architectures discovered on CIFAR-10, the selected model is evaluated on CIFAR-100 and ImageNet. As summarized in Table II, G-ICSO-NAS achieves a test error of 16.90% on CIFAR-100, exhibiting performance comparable to PC-DARTS while outperforming several representative NAS baselines. Moreover, the corresponding evaluation results on ImageNet are reported in Table III, where G-ICSO-NAS demonstrates competitive performance in terms of both top-1 and top-5 error rates.

D. Experiment Result and Comparison on NAS-Bench-201 search space

The comparative evaluation of G-ICSO-NAS against representative state-of-the-art NAS approaches on the NAS-Bench-201 benchmark is reported in Table IV, covering experiments on CIFAR-10, CIFAR-100, and ImageNet-16-120.Across the NAS-Bench-201 search space, on CIFAR-10, CIFAR-100, and ImageNet-16-120, G-ICSO-NAS exhibits strong overall performance. The architecture identified based on CIFAR-10 achieves the best results on six evaluation metrics, outperforming a wide range of existing NAS methods.

TABLE III: Comparison of Neural Architecture Search Methods on ImageNet

Architecture/Method	Test Error top-1(%)	Test Error top-5(%)	Params (M)
ShuffleNet-V2	25.1	9.9	7.4
MobileNet-V2	28.0	9.0	3.4
DARTS	26.9	9.0	4.9
DARTS(2st)	26.7	8.7	4.7
SNAS	27.3	9.2	4.3
GDAS	26.0	8.5	5.3
BayesNAS	26.5	8.9	3.9
PC-DARTS	25.1	7.8	5.3
SETN	26.7	8.6	5.2
NASNet-A	26.0	8.4	5.3
NASNet-B	27.2	8.7	5.3
NASNet-C	27.5	9.0	4.9
AmoebaNet-A	25.5	8.0	5.1
AmoebaNet-B	26.0	8.5	5.3
EoiNAS	25.6	8.3	5.0
MFENAS	26.06	8.18	5.98
SaDENAS	25.08	7.98	5.58
G-ICSO-NAS	**24.98**	**7.49**	5.8

E. Ablation study

Regarding the parameter diversity fitness and operation diversity fitness, extensive experiments demonstrate that setting the weights of both diversity functions to zero leads to considerable instability in the search results. A significant proportion of such cases exhibit an excessive presence of none and skip-connect operations, which adversely affect the final architecture's training performance.

About the Stable Stage and Early Stopping Strategy, experimental results indicate that after 10-15 epochs of ICSO-based search, the explored optimal architecture tends to stabilize. Continuing ICSO iterations beyond this point substantially increases computational cost without meaningful improvement. As shown in Table V, the removal of these specific acceleration

TABLE IV: Comparison of Neural Architecture Search Methods on NAS-Bench-201

Architecture/Method	CIFAR-10		CIFAR-100		ImageNet16-120	
	valid	test	valid	test	valid	test
ResNet	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	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	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	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	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	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)	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)	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	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	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-	**91.03±0.44**	93.80±0.40	71.36±1.51	71.53±1.51	44.87±1.46	45.12±0.82
G-ICSO-NAS	90.22±0.12	**94.00±0.08**	**72.17±0.38**	**72.63±0.47**	**45.21±0.26**	**45.66±0.34**
Optimal	91.61	94.37	73.49	73.51	46.77	47.31

TABLE V: Ablation Study of Stable Stage with Early Stopping

Stable Stage	Early Stopping Mechanism	Search Time Cost
✓	✓	0.15 GPU/Days
✗	✗	0.40 GPU/Days
✓	✗	0.25 GPU/Days

modules incurs increased search times ranging from 0.25 to 0.40 GPU-days.

V. Conclusion

In this work, we present G-ICSO-NAS, a neural architecture search framework that integrates swarm-based architecture search with gradient descent through an alternating update scheme. Comprehensive evaluations demonstrate that G-ICSO-NAS achieves a favorable trade-off between search efficiency and architectural performance. Future research will focus on improving the robustness and performance of this framework.

VI. Acknowledgment

This work is supported by the Hokkaido University EXEX Doctoral Fellowship.

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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

[4] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[5] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International conference on machine learning*. PMLR, 2018, pp. 4095–4104.

[6] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

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

[8] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.

[9] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[10] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International conference on machine learning*. PMLR, 2017, pp. 2902–2911.

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

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

[13] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 10 734–10 742.

[14] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE transactions on cybernetics*, vol. 45, no. 2, pp. 191–204, 2014.

[15] ——, "A social learning particle swarm optimization algorithm for scalable optimization," *Information Sciences*, vol. 291, pp. 43–60, 2015.

[16] Q. Yang, W.-N. Chen, J. Da Deng, Y. Li, T. Gu, and J. Zhang, "A level-based learning swarm optimizer for large-scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 578–594, 2017.

[17] R. Zhong, J. Yu, X. Du, E. Zhang, and A. G. Hussien, "Improved competitive swarm optimizer with linear population reduction for large-scale optimization," in *2025 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2025, pp. 1–8.

[18] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American statistical association*, vol. 58, no. 301, pp. 13–30, 1963.

[19] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

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

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