Pareto-Informed Multi-Objective Neural Architecture Search

Ganyuan Luo[1], Hao Li[1], Yuren Zhou[2], and Zefeng Chen[1](✉)

[1] School of Artificial Intelligence, Sun Yat-sen University, Zhuhai, China
{luogy7, lihao75}@mail2.sysu.edu.cn, chenzef5@mail.sysu.edu.cn
[2] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
zhouyuren@mail.sysu.edu.cn

Abstract. This paper introduces a novel approach called Pareto-Informed Multi-Objective Neural Architecture Search (PiMO-NAS), which utilizes a solution generator based on Tchebycheff decomposition to explore the multi-objective evaluation function space in one-shot neural architecture search (NAS). We detail the application of PiMO-NAS in two distinct search spaces: Once-For-All (OFA) and AutoFormer, covering both convolutional neural networks and vision transformers. Our method begins with a continuous transformation of these discrete search spaces, followed by an iterative optimization of the Pareto front. We deploy a Gaussian Process surrogate model to estimate the function landscape and a recurrent solution generator, influenced by preference vectors, for exploration. To cope with local optima and maintain diversity, we introduce an adaptive sampler for balanced Pareto front formation. The efficacy of PiMO-NAS is demonstrated through comprehensive experiments within the AutoFormer and OFA-based search spaces. Our approach not only covers the genetic algorithm's search results in these spaces but also excels in the hypervolume metric. Experimental results on the ImageNet-1k dataset show that in the OFA-based search spaces, compared with the NSGANetV2 based on MSuNAS, the proposed PiMO-NAS was able to achieve similar performance in two-thirds of the number of iterations, thereby saving about 24% of the search time. On the one hand, in the AutoFormer-based search space, we successfully approached a strong baseline formed by a single-objective evolutionary algorithm with restricted parameter quantities, approximating the entire Pareto front in a comparable timeframe.

Keywords: Neural architecture search (NAS) · Surrogate-assisted search · Pareto set learning.

1 Introduction

In recent years, a paradigm shift in computer vision has been witnessed with the advent of neural network such as convolutional neural networks(CNNs) [18–20, 28,33] and vision transformers (ViTs) [12,25,36,37]. Pioneering the deep learning

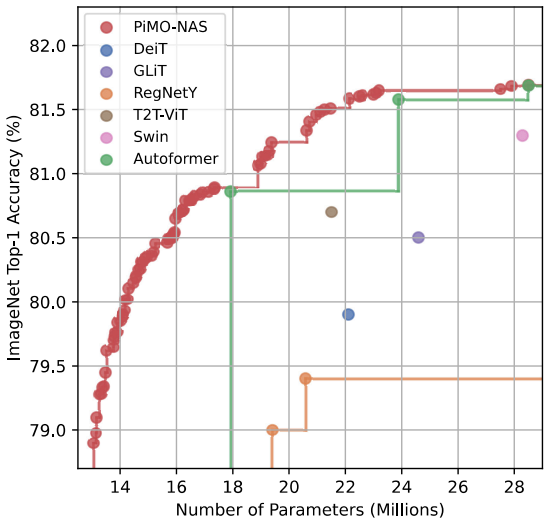

Fig. 1. PiMO-NAS, with a single 2.16 GPU-day search in the 'small' Autoformer space, outperforms previous models like DeiT and RegNetY on the Pareto front. In comparison, Autoformer spent 1.83 GPU days for each of its three searches.

revolution, CNNs have established themselves as a cornerstone in the field, thanks to their exceptional capability in handling image data. Meanwhile, ViTs have demonstrated remarkable capabilities in capturing long-range dependencies, a feature critical for understanding complex visual contexts, which marked a significant departure from CNNs [17]. However, the quest for efficient and powerful neural architectures in computer vision raises a critical question: how to balance the trade-off between multiple competing objectives (such as accuracy and model size)? As the success of CNNs and ViTs in visual recognition is contingent upon the meticulous architecture design, it requires substantial computational resources and expert knowledge. This challenge has prompted the exploration of neural architecture search (NAS) methods, which automates the process of architectural design, leading to models that are both efficient and performance [1,21,24,31].

Among these, one-shot NAS (OSNAS) method innovatively decouples the training and searching phases in the process of neural architecture optimization [2,6,24,29,30]. Some distinct approaches have showcased the versatility and efficiency of one-shot NAS in different contexts. For example, the AutoFormer [6] exemplifies the application of OSNAS in fine-tuning ViTs for different budgets with weight entanglement. This approach effectively identifies the optimal ViT configurations that balance computational efficiency and high performance in visual tasks. On the other hand, Once-For-All (OFA) [3] represents an application of OSNAS extending to CNNs. Developed by Cai *et al.*, the OFA trains a diverse and comprehensive supernet capable of adapting to a wide range of hardware constraints and efficiency requirements. With progressive shrinking technique, this adaptability is achieved without the necessity for model retraining.

This work primarily focuses on multi-objective neural architecture on one-shot NAS search space. Drawing inspiration from the key idea of Pareto set learning [23], we develop an innovative approach, named *Pareto-Informed Multi-Objective Neural Architecture Search* (*PiMO-NAS* for short), to address a crucial challenge in deploying neural networks across varied budgets.

In brief, our major contributions in this paper are summarized as three-fold.

1. A simplified yet effective method is developed to relax the traditionally rigid discrete search space of NAS, and an innovative integration of Pareto Set Learning into one-shot NAS is put forward.
2. A novel dynamic weight sampling approach is designed to address the challenge in terms of the presence of inequality in sub-problems caused by uneven improvements in hypervolume across different Tchebycheff decomposition vectors. This can endow with our proposed PiMO-NAS algorithm a strong capability of adapting to diverse search spaces.
3. Our proposed PiMO-NAS algorithm is systemically compared with existing state-of-the-art methods. This comparative analysis showcases the potential of PiMO-NAS, highlighting its superior performance in balancing multiple objectives in neural architecture search.

The structure of this paper is organized as follows. Section 2 delves into the foundations of multi-objective NAS and also provides a detailed exposition on Pareto Set Learning, outlining its principles and relevance in the context of NAS. Section 3 elaborates on the methodology of integrating Pareto Set Learning into the aforementioned search spaces and covers the details of our approach. Section 4 is dedicated to the experimental aspect of our research. Here, we present the design of our experiments and discuss the experimental results. Section 5 summarizes the key findings and contributions of our study.

2 Preliminaries

2.1 Multi-Objective NAS

The NAS aims to automate the process of designing optimal neural architectures, finding the best neural network parameters by performing optimization within a predefined search space. In search space, neural networks can be encoded as directed acyclic graphs (DAGs) of a series of hidden states and operations [14] $s^t = o^t(s^{t-1})$, where s^t represents the t^{th} hidden layer and o^t is a specific operation (e.g., fully connection, convolution, pooling and activation functions) for converting s^{t-1} to s^t. Sometimes, the encoding could be simplified to vectors. For instance, in AutoFormer, the neural network is encoded as a vector in the form of [*Embed dim, MLP ratio, Head Num, Depth Num*], allowing representation as discrete vectors [6].

At this point, the NAS process can be described as a problem of maximizing/minimizing specific objective functions (i.e., performance indicators, such as test set accuracy) within the search space. In practice, evaluating a neural architecture requires training on a training set D_{train} and then testing on a test set D_{test}, often consuming a significant amount of time. One-shot NAS partially mitigates this issue through methods like parameter sharing [30] and weight entanglement [6], but the evaluation process remains costly compared to common optimization problems. In real-world applications, accuracy is not the only objective to optimize; when balancing contradictory indicators such as model

complexity and error rate on test data, the NAS process becomes an expensive multi-objective optimization problem:

$$\min_{\alpha \in \mathbb{A}} f(\alpha) = (Error(\alpha | D_{train}, D_{test}), Params(\alpha), \ldots) \tag{1}$$

where α is a neural architecture in the search space $\mathcal{A} \subseteq \mathbb{R}^n$, and $f : \mathcal{A} \to \mathbb{R}^m$ is an m-dimensional vector-valued objective function.

In practical multi-objective NAS problems, it is often impossible to find a single solution that simultaneously optimizes multiple indicators such as computation cost and model accuracy. As a result, the problem evolves into the intricate challenge of identifying the Pareto front (PF) across varying trade-offs. Previous works have simplified the problem into a constrained single-objective optimization problem [6]. Alternatively, the problem can be further transformed into an unconstrained single-objective optimization problem through scalarization, as seen in the works of Cai *et al.* and Tan *et al.* [4, 34]. Another prevalent approach is to search for a set of non-dominated solutions to approximate the PF, subsequently applying multi-objective optimization algorithms, such as the NSGA algorithm [26, 27].

2.2 Pareto Set Learning (PSL)

In the field of expensive multi-objective optimization, earlier methods attempt to approximate the Pareto set with a finite set of solutions. In contrast, the PSL endeavors to predict the entire PF based on a surrogate model, thereby learning an approximation of the Pareto optimal solutions [23]. This process is facilitated by a set model that maps all valid trade-off preferences $\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m | \sum_{i=1}^m \lambda_i = 1\}$ to its corresponding Pareto solution, thus enabling the exploration of the entire approximate PF with ease by modifying the trade-off preferences.

Specifically, the PSL employs an augmented Tchebycheff approach to scalarize the multi-objective problem, and aims to find optimal parameters θ^* for the set model $h_{\boldsymbol{\theta}^*}(\boldsymbol{\lambda})$ which could generate a competitive solution set for augmented Tchebycheff scalarization $\mathcal{M}_{tch\_aug} = \{\boldsymbol{x}^*(\boldsymbol{\lambda}) | \boldsymbol{\lambda} \in \Lambda\}$, where

$$h_{\boldsymbol{\theta}^*}(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}} g_{tch\_aug}(\boldsymbol{x} | \boldsymbol{\lambda}), \forall \boldsymbol{\lambda} \in \Lambda. \tag{2}$$

The deployment of set models in PSL has not only made the traversal through the multi-dimensional space of preferences delineated by $\boldsymbol{\lambda}$ more straightforward but also more efficient.This efficiency gain is pivotal for enabling the utilization of hypervolume-based improvement strategies in batch selection, which is formalized as

$$\mathcal{B}_{hv} = \{\boldsymbol{x}^*(\boldsymbol{\lambda}) \mid \Delta HV(\boldsymbol{x}^*(\boldsymbol{\lambda}), \mathcal{M}_{psl}) > 0,\ \boldsymbol{\lambda} \in \Lambda\}, \tag{3}$$

where $\mathcal{B}_{hv}$ denotes the set of batch-selected solutions, and $\Delta HV(\boldsymbol{x}^*(\boldsymbol{\lambda}), \mathcal{M}_{psl})$ quantifies the hypervolume improvement contributed by the new solution $\boldsymbol{x}^*(\boldsymbol{\lambda})$

over the current Pareto set $\mathcal{M}_{psl}$. This approach to selection optimizes the trade-off exploration by quantitatively measuring the expansion of the PF, thereby providing a methodological advancement in the field of multi-objective optimization that is both effective and computationally viable.

3 Method

In this section, we delineate our methodological framework for PiMO-NAS. Initially, we expound upon the definition and composition of our search space, which originates from AutoFormer and OFA architectures. This space, characterized by its discrete nature, presents limitations for original PSL techniques. Subsequently, we introduce our novel framework designed to estimate the function landscape, employing a synergistic approach that integrates a surrogate model with a recurrent solution generator. Lastly, we propose an adaptive sampling strategy tailored to form more balanced PF.

3.1 Search Space

In order to validate the effectiveness of PiMO-NAS, while considering the inherent differences between CNNs and ViTs, we selected representative one-shot NAS spaces from Once-For-All for CNNs and AutoFormer for ViTs as our exploration grounds.

OFA Search Space As part of our methodology, we utilize the Once-For-All (OFA) approach, which offers a comprehensive search across several dimensions crucial for optimizing CNNs. These dimensions include depth, input resolution, width, and kernel size [3]. For a detailed exploration of these parameters within our framework, refer to Appendix A.1 where we follow the configuration settings of MSuNAS [26] and detail our parameter choices and methodology.

Figure 7 illustrates the search space used in our study, depicting the architectural elements and their variability across different configurations.

AutoFormer Search Space To optimize the ViT architecture, we employ an AutoFormer-based search methodology, which segments the architecture into critical dimensions such as Embedding Dimension, MLP Ratio, Head Number, and Depth [6]. Detailed parameter exploration for these dimensions is extensively described in Appendix A.2. Figure 8 shows the specialized search space utilized in our approach.

Continuous Transformation of Search Space Similar to many NAS problems, both OFA and AutoFormer employ discrete search spaces to encode neural architectures. In our approach, we map the domain of discrete encoding simply onto the range $[0, 1]$, thus transforming the problem into minimizing a cost function in a continuous real space regarding an N-dimensional decision vector. To

restrict the ineffective space generated by zero-padding, we introduce a masking technique that will be further elaborated in Section 3.2.

3.2 Pareto-Informed Search

Task Formulation After implementing the aforementioned encoding and transformation of search space, we relaxed the decision space for both OFA-based and AutoFormer-based search spaces into a continuous domain. Consequently, the multi-objective NAS problem considered in this paper is transformed into a multi-objective optimization problem on a continuous space, and can be formalized as the following parametric dual-objective optimization problem:

$$\min_{x \in \mathbb{X}} \mathbf{F}(\mathbf{x}) = (Error(\mathbf{x}), Params(\mathbf{x})) \tag{4}$$

where $\mathbf{F}$ denotes the objective vector consisting of two performance indicators, $\mathbf{x}$ represents the decision vector embodied as $\mathbf{x} = (x_1, x_2, \ldots, x_n), x_i \in [0, 1]$, and $\mathbb{X} \subseteq \mathbb{R}^n$ is the decision space (search space).

Building on this, we developed PiMO-NAS for the iterative optimization of the PF. In a nutshell, each iteration of our proposed PiMO-NAS divides the fixed-length continuous neural architecture search process into three critical phases, which will be elaborated in the followings.

Surrogate-Model-based PF Estimation In the first phase of our approach, it is crucial to estimate the overall PF, which necessitates the use of a surrogate model. Drawing inspiration from Pareto set learning and referencing the analysis in MSuNAS, we opt for a Gaussian Process (GP) as our surrogate model.

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{5}$$

At the onset of our algorithm, we employ Latin Hypercube Sampling (LHS) to select a set of candidate models from the search space. Besides, our initial population also includes both all-ones and all-zeros decision vectors to estimate the functional boundaries. These models are then evaluated on the ImageNet [11] dataset to generates data for the training of our GP surrogate model.

Once the evaluations are completed, we normalize the results to align with the GP. The GP is then executed to perform regression analysis on this data, and the outcome of this regression is an estimate of the overall function landscape.

Exploration via Solution Generator In the second phase, we aim to thoroughly explore the landscape of the estimated objective function space. To achieve this, we utilize a solution generator, which is designed to operate under the guidance of a preference vector (denoted as λ). The solution generator's primary task is to produce optimal solutions for single-objective problems, as determined by the augmented Tchebycheff scalarization approach as Eq. (6), guided by the given preference vector $\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m | \sum_{i=1}^m \lambda_i = 1\}$.

$$\hat{g}_{\text{tch\_aug}}(\mathbf{x}|\mathbf{\Lambda}) = \max_{1\leq i\leq m}\left\{\lambda_i\left|\hat{f}_i(\mathbf{x}) - (z_i^* - \epsilon)\right|\right\} + \rho\sum_{i=1}^{m}\lambda_i\hat{f}_i(\mathbf{x}). \tag{6}$$

where $\rho > 0$ and $\epsilon > 0$ are small positive scalars used in [22], $z^* = (z_1^*, \ldots, z_m^*)$ is the ideal vector, and $\hat{f}(\mathbf{x})$ is the lower confidence bound(LCB) for minimization problems provided by GP surrogate model.

Given the block-wise stacked architecture inherent in both OFA-based and AutoFormer-based search spaces, we drew inspiration from works like [30, 34] to devise our solution generator. This generator is based on a Gated Recurrent Unit (GRU) architecture [10] as shown in Fig. 2.

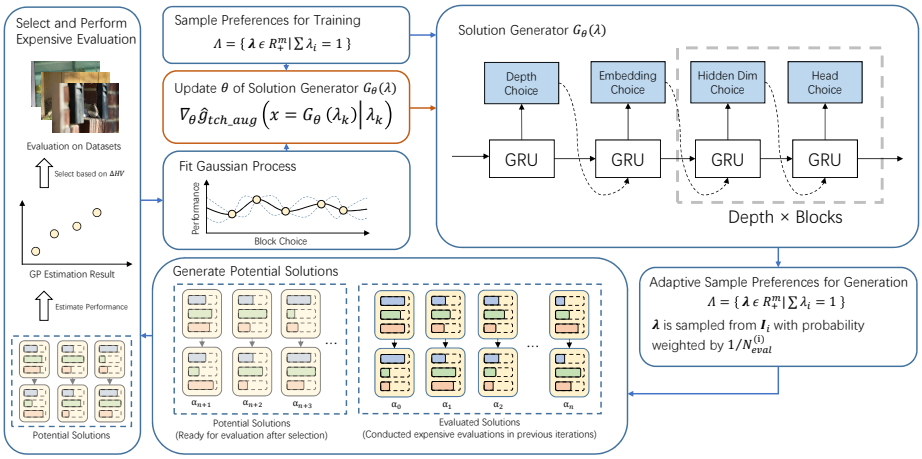

Fig. 2. Pareto-informed Multi-objective Neural Architecture Search: (1) GP surrogate model is utilized for estimating the function landscape from evaluated solutions. Note that we independently run a GP model for each objective function. (2) Solution Generator is responsible for generating the expected optimal solutions based on arbitrary target function preferences $\Lambda = \{\boldsymbol{\lambda} \in \mathbb{R}_+^m | \sum_{i=1}^m \lambda_i = 1\}$.

The generator $G_\theta(\cdot)$ initiates its operation by mapping a weight vector λ to an embedding using a linear layer. This embedding serves as the initial hidden state h_0 for the GRU. Subsequently, various linear heads, tailored to the specific stages of the model architecture, generate different types of decision variables. These heads operate in distinct stages, ensuring that the variable generation is aligned with the structural of the neural network blocks being considered.

Building upon the foundation laid by the GP approximation of the function landscape, our solution generator embarks on a journey of exploration, guided by weight vectors λ_k obtained via random sampling. The generator's objective is to probe the approximated terrain, seeking to enhance the lower bounds of performance for the solutions it generates through gradient descent:

$$\theta_{t+1} = \theta_t - \eta \sum_{k=1}^{K} \nabla_\theta \hat{g}_{\text{tch\_aug}}(x = G_\theta(\lambda_k)|\lambda_k), \tag{7}$$

In addition, it is noteworthy that the nature of our search space involves models with variable depth, while our encoding scheme employs fixed-length vectors. To address this discrepancy, we introduce a masking technique for the solutions generated beyond their designated depth limits. Specifically, a mask is applied to map the excess portions of the solution to zero. This approach ensures consistency with the zero-padding scheme previously established for managing the dimensionality of decision vectors in our search space.

Batch Selection with Adaptive Sampler In the final phase of each iteration, we focus on the generation and evaluation of candidate solutions derived from an extensive exploration of the function's landscape. We design an adaptive sampler to dynamically generate preference as exploration direction:

$$\lambda = (w, 1-w), p \in [0, 1] \tag{8}$$

where w is the weight of the first objective and We feed λ into solution generator. The sampler achieves this by dividing the weight p into K equal intervals and tracking the outcomes of candidate solutions generated from each interval. It maintains a record of the number of solutions from each interval that are accepted for costly evaluation $\mathbf{N}_{\text{eval}} = (N_{\text{eval}}^{(1)}, N_{\text{eval}}^{(2)}, \ldots)$, and the sampling of preference is converted into a two-stage process:

$$P(I = I_i) = \frac{1/N_{\text{eval}}^{(i)}}{\sum_{j=1}^{k} 1/N_{\text{eval}}^{(j)}}, \quad \text{for } i = 1, 2, \ldots, K \tag{9}$$

$$w \sim U(l^{(i)}, h^{(i)}) \tag{10}$$

where $N_{\text{eval}}^{(i)}$ represents the number of acceptance of $w \in I_i$ and K is the total number of distinct intervals $I_i \in [l^{(i)}, h^{(i)}]$. This discretization allows for a more controlled and focused exploration of the solution space.

Once the sampling process is completed, the preference vectors are input into the solution generator to obtain candidate solutions for the subproblems. These candidate solutions are then subjected to a batch selection process that filters the most promising subset to advance the search process. The batch selection procedure is guided by the expected hypervolume improvement within the framework of GP:

$$\Delta HV(\hat{\mathbf{F}}(X_{\text{cand}})) = HV(y_{i-1} \cup \hat{\mathbf{F}}(X_{\text{cand}})) - HV(y_{i-1}), \tag{11}$$

where X_{cand} are the candidate solutions generated by the solution generator from a batch of weight vectors λ. The term y_{i-1} represents the performance indicators of the solution set that has been evaluated in previous iterations, and $\hat{\mathbf{F}}(\cdot)$ denotes the estimated performance indicators provided by the GP model.

Summary of PiMO-NAS The complete framework of our PiMO-NAS algorithm is encapsulated and presented as Algorithm 1. In our comprehensive framework, we have incorporated the GP estimation, landscape exploration, and solution generation processes as mentioned above. Notably, unlike typical continuous optimization problems, the inherent discreteness of the original space in our problem leads to many slightly different solutions in the relaxed space being identical when evaluated by the function. To circumvent this, we applied a deduplication step when selecting solutions for evaluation, thereby avoiding redundant and costly assessment procedures.

Algorithm 1: PiMO-NAS Framework

Input: $\mathbf{F}(\mathbf{x})$ as Eq. (1), Exploration Iter T, Training Iter N_{iter}, Interval Seperation Num P
Output: $\{\mathbf{x}_t, \mathbf{y}_t\}$ and Solution Generator $G_{\theta_T}(\cdot)$

for $t = 1$ *to* T **do**
  Train GPs based on $\{\mathbf{x}_{t-1}, \mathbf{F}(x_{t-1})\}$;
  for $n = 1$ *to* N_{iter} **do**
    Sample preferences $\{\Lambda_k\}_{k=1}^{K} \sim \Lambda$;
    Update θ_t of solution generator G_{θ_t};
  end
  Sample P intervals I_i with adaptive sampler in Eq. (9);
  Sample P preferences $\{\Lambda^{(p)}\}_{p=1}^{P} \sim \Lambda$ in I_i;
  Generate solutions $x_p = \{G_\theta(\Lambda^{(p)})\}_{p=1}^{P}$;
  Find non-repetitive subset x_{cand} with highest ΔHV;
  $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} \cup x_{cand}$;
  $\mathbf{y}_t \leftarrow \mathbf{y}_{t-1} \cup \mathbf{F}(x_{cand})$;
end

4 Experiments

In this section, we introduce the experimental framework utilized to evaluate the proposed PiMO-NAS method. The central aim of our experiments is to demonstrate the effectiveness of PiMO-NAS by applying it within two specific one-shot NAS spaces: the AutoFormer space [6] and an OFA-based space shared with MSuNAS [26].

4.1 OFA-based Search Space

We applied PiMO-NAS to the same NAS space as MSuNAS, starting with a population of 100. In each iteration, a solution generator with a hidden layer of 128 dimensions was used. Throughout the training, we randomly sampled 20 preferences per iteration, over 2000 iterations, until convergence. Using the

Adaptive Sampler, 15 preferences were sampled to generate candidate solutions. The final selection involved screening 8 solutions based on expected hypervolume improvement for evaluation on the ImageNet-1K dataset. The exploration process, spanning 30 epochs, revealed comparative insights between PiMO-NAS and MSuNAS. It is important to highlight that in our implementation within MSuNAS, GPs were utilized as surrogate models. We emphasize that the choice of surrogate model type used in MSuNAS is kept consistent with our framework.

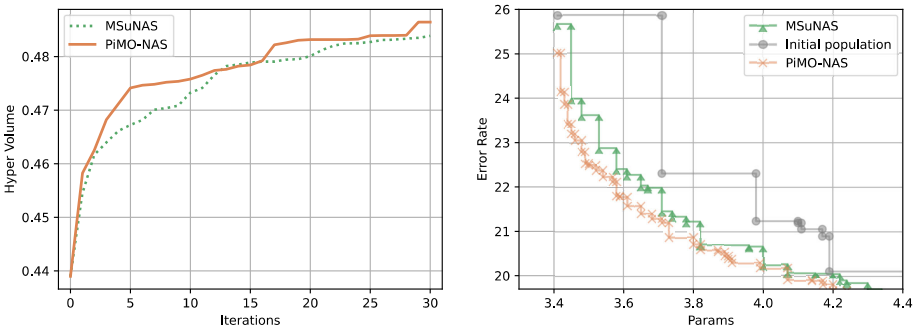

Fig. 3. Left: Hypervolume change during the search. The reference point was set at [8.426, 0.284], representing the worst values for each objective in the initial population. **Right:** Pareto front obtained at the end of the search.

As shown in Fig. 3, an early observation was that PiMO-NAS exhibited a faster rate of hypervolume increase compared to MSuNAS. This indicates PiMO-NASs efficient exploration and utilization of the surrogate model's estimated space. Specifically, after approximately 20 iterations (corresponding to 160 samples), the hypervolume achieved by PiMO-NAS approaches the results obtained by MSuNAS after 30 iterations. Furthermore, the Pareto solution set associated with PiMO-NAS not only exhibits a significantly larger hypervolume but also a denser distribution compared to that of MSuNAS at 30 iterations. Therefore, even when accounting for the overhead of training the solution generator, PiMO-NAS achieves a 1.24x speedup compared to the MSuNAS method for equivalent hypervolume results. Additionally, as the cost per solution evaluation increases, the relative expenditure of the Solution Generator in the total search time will further diminish. This reduction is expected to further extend the performance gap between PiMO-NAS and MSuNAS.

We also analyzed a total of 240 candidate solutions sampled by PiMO-NAS and MSuNAS as Fig. 4. We found that, compared with the nearly normal distribution sampling of different parameter-volume architectures by MSuNAS in the search space, PiMO-NAS, guided by GP evaluations, can sample more points with lower parameter volumes, effectively extending the Pareto front coverage. After introducing the adaptive sampler, PiMO-NAS can obtain a smoother sampling curve and significantly increase the sampling of large-volume models, com-

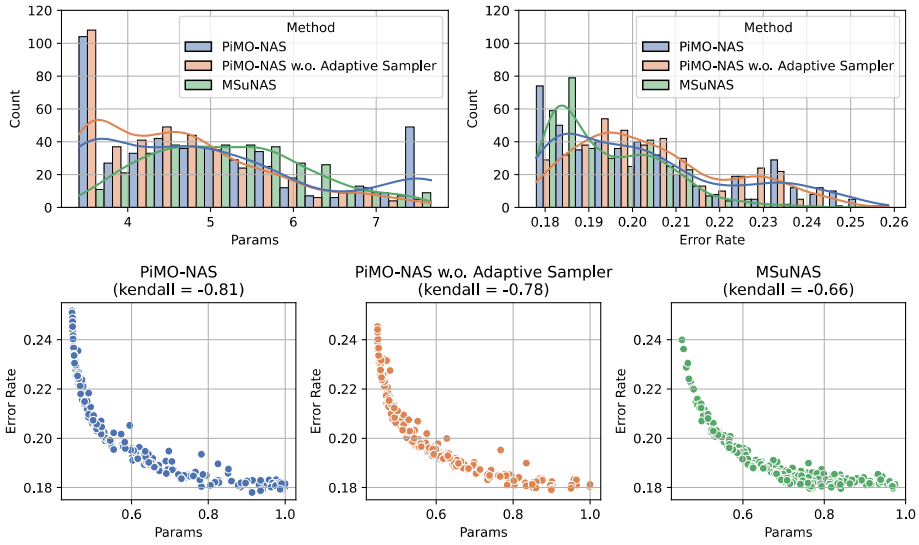

Fig. 4. Comparison of the search results of PiMO-NAS and MSuNAS. **Upper:** Distribution of Params and Error Rate of candidates sampled by PiMO-NAS and MSuNAS. **Lower:** The Kendall's Tau of points sampled by PiMO-NAS and MSuNAS.

pensating for the shortcoming in sampling density in the high parameter-volume region to some extent. Notably, we calculated the Kendall's rank correlation coefficient of the sampled solutions and found that the samples obtained by the PiMO-NAS method more purposefully explore trade-off solutions.

4.2 AutoFormer-based Search Space

We further explored the AutoFormer-based search space using the PiMO-NAS method, while employing the genetic algorithm based AutoFormer with a predefined parameter range from the original paper of AutoFormer as a strong baseline. Following the genetic algorithm's design in the original study, we used a population of 50 and conducted 20 iterations. For PiMO-NAS, we started with a sample size of 200, sampling 20 instances per iteration over 40 rounds, a design balancing sampling efficiency and the cost of GP. Both our approach and the AutoFormer's genetic algorithm sampled 1000 instances, with the results depicted in Fig. 5.

In our observations, after 1000 samples, the PiMO-NAS method effectively covered the search results of the genetic algorithm within the predefined parameter range. Notably, in both AutoFormer-Small and AutoFormer-Base spaces, PiMO-NAS managed to cover the optimal solutions obtained by three rounds of the genetic algorithm within its 1000 samples. This highlights the efficiency of PiMO-NAS in searching for Pareto-optimal solutions. This suggests that PiMO-NAS only required approximately 1.18 times the time to explore the entire Pareto

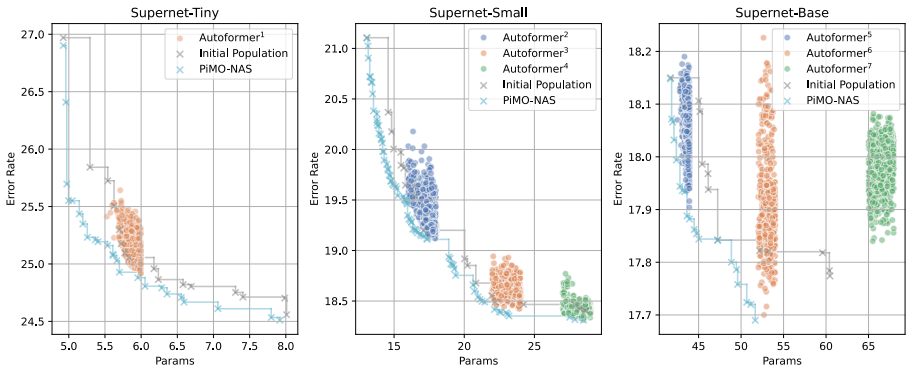

Fig. 5. Pareto front outcomes for PiMO-NAS on the AutoFormer-based search space. The variants of AutoFormer with distinct superscripts (i.e., 1 7) indicate the search results obtained under different parameter range restrictions, as detailed in Appendix B. In Supernet-Small space, the PF obtained by PiMO-NAS (with only 1,000 samples) can efficiently cover the results obtained by AutoFormer^2+AutoFormer^3+AutoFormer^4 (each of which consumes 1,000 samples, thus producing a combined total of 3,000 samples). A similar phenomenon occurs in Supernet-Base space.

front compared to the time AutoFormer took to find a single solution. Furthermore, the time required by AutoFormer to find solutions increases linearly with the number of solutions sought.

4.3 Result on ImageNet

Finally, we conducted 20 iterations on the same space as MSuNAS with 100 random samples. In each iteration, we generated and evaluated eight candidate solutions. After completing the search, we fine-tuned the obtained solutions on the ImageNet-1k dataset for 200 epochs. These solutions were named PiMO-NAS-O-T/S/B where 'O' denotes the OFA-based search space. The solutions were then ranked according to their FLOPs. Table citeOFA search Result displays the performance of our models on the ImageNet-1k dataset. The networks discovered by PiMO-NAS demonstrated strong competitiveness with the MSuNAS method, while requiring significantly less search time.

In the AutoFormer search space, we named the search results PiMO-NAS-A-T/S/B, where 'A' denotes the AutoFormer-based search space, categorized according to the search space divisions. We successfully achieved results comparable to those obtained in the original AutoFormer paper, which used a genetic algorithm with a restricted parameter range. This corroborates that PiMO-NAS has effectively explored the Pareto front of the AutoFormer space after 1000 samples. It is important to note that while PiMO-NAS utilized slightly more search time, it achieved dozens to hundreds of Pareto solutions in one go on the AutoFormer search space, as opposed to the original algorithm where each solution required 1000 iterations. This implies that when multiple deployments are

Table 1. The comparison results on OFA-based Search Space on the ImageNet Classfication Task. The time required for supernet training and fine-tuning is not included.

Model	Type	Search Cost (GPU days)	Top-1 Acc. (%)	#Params (M)	Flops (M)
MobileNetV3 [19]	combined	-	75.2	5.4	219
ShuffleNetV2 [28]	manual		72.6	-	299
RCNet [46]	auto	9	74.7	4.7	471
SPOSNAS [16]	auto	13	74.8	5.3	465
MnasNet-A1 [34]	auto	3800	75.2	3.9	312
EEEA-Net-C1 [35]	auto	0.52	74.3	5.1	137
MOEA-PS [41]	auto	5.2	73.6	4.7	-
PiMO-NAS-O-T(Ours)	**auto**	**0.81**	**75.2**	**5.6**	**153**
FairNAS-A [9]	auto	12	77.5	5.9	392
MixPath-B [8]	auto	10.3	77.2	5.1	378
NSGANetV2-m [26]	auto	1	78.3	7.7	312
PiMO-NAS-O-S(Ours)	**auto**	**0.81**	**77.7**	**6.1**	**362**
FP-DARTS [39]	auto	2.44	76.3	7.2	598
PNAG-170 [15]	auto	0.7	80.3	10	606
NSGANetV2-xl [26]	auto	1	80.4	8.7	593
PiMO-NAS-O-B(Ours)	**auto**	**0.81**	**80.2**	**7.44**	**576**

Table 2. The comparison results on AutoFormer-Based Search Space on the ImageNet Classfication Task. The time required for supernet training and fine-tuning is not included. † indicates the results we reproduce using open-source code of AutoFormer.

Model	Type	Search Cost (GPU days)	Top-1 Acc. (%)	#Params (M)	Flops (G)
ViT-Ti [12]	manual	-	74.5	5.7	1.4
PVT-Tiny [38]	manual	-	75.1	13.2	1.9
DeiT-Tiny [37]	manual	-	72.2	5.7	1.2
ViTAS-C [32]	auto	32	74.7	5.6	1.3
AutoFormer-Tiny† [6]	auto	1.83	75.1	6	1.4
PiMO-NAS-A-T(Ours)	**auto**	**2.16**	**75.1**	**5.7**	**1.6**
T2T-VIT-14 [43]	manual	-	81.7	21.5	6.1
SWIN-T [25]	manual	-	82.3	29	4.5
GLiT-Small [5]	manual	-	80.5	24.6	4.4
CrossFormer-T [40]	manual	-	81.5	27.7	2.9
PoolFormer-S24 [42]	manual	-	80.3	21.3	3.4
TF-TAS-S [45]	auto	0.5	80.5	24.6	3.2
As-ViT-S [7]	auto	0.5	81.2	29	5.3
ViTAS-F [32]	auto	32	80.5	27.6	6.0
AutoFormer-Small† [6]	auto	1.83	81.6	23.9	4.8
PiMO-NAS-A-S(Ours)	**auto**	**2.16**	**81.6**	**21.4**	**4.3**
T2T-VIT-19 [43]	manual	-	82.2	39.2	9.8
EAT-B [44]	manual	-	82	86.6	14.8
ConViT-S+ [13]	manual	-	82.2	48	10.0
GLiT-Base [5]	manual	-	82.3	96.1	17.0
PoolFormer-M48 [42]	manual	-	82.5	73.4	11.6
AutoFormer-Base† [6]	auto	1.83	82.3	52.7	10.3
PiMO-NAS-A-B(Ours)	**auto**	**2.16**	**82.3**	**51.7**	**10.8**

needed, PiMO-NAS does not exhibit a significant time disadvantage compared to advanced Zero-Shot NAS methods like TF-TAS.

4.4 Ablation Study

We investigated the impact of incorporating an Adaptive Sampler, as demonstrated in 4.4. Initially, the introduction of the Adaptive Sampler slightly decelerated the increase in hypervolume; however, it ultimately facilitated superior hypervolume results over a prolonged period.

This effect is likely attributable to the Adaptive Sampler's ability to achieve smoother sampling of weight vectors. Such smoothing of the sampling process effectively redirected the exploratory efforts of PiMO-NAS towards underexplored areas, thus mitigating the "greediness" of the search algorithm. Over time, this strategy proves beneficial by promoting the formation of a more uniform and refined Pareto front. Furthermore, it provides a more thorough and accurate assessment of the overall function landscape, enabling more strategic exploration and better long-term outcomes in multi-objective optimization tasks.

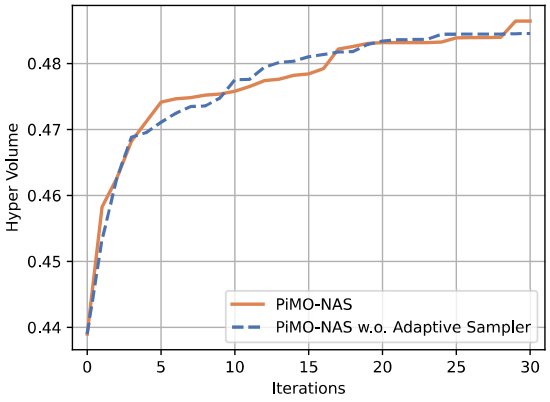

Fig. 6. The difference in hypervolume improvement in PiMO-NAS within the OFA-based space under Adaptive Sampling versus Random Sampling.

5 Conclusion

This work introduced the Pareto-Informed Multi-Objective Neural Architecture Search (PiMO-NAS) method, offering a significant stride in the domain of multi-objective NAS. By integrating Pareto set learning with a one-shot NAS framework, our approach adeptly balances conflicting objectives, such as accuracy and model size, within neural architecture search. Our experiments within the AutoFormer and OFA-based search spaces showcased that PiMO-NAS could efficiently cover the search results of conventional genetic algorithms, exemplifying its efficacy in discovering Pareto-optimal solutions.

PiMO-NAS was observed to outperform baselines in terms of hypervolume metrics, indicating its proficiency in exploring the search space more comprehensively. Moreover, the Adaptive Sampler introduced in our methodology promoted diversity and circumvented local optima, further solidifying the robustness of PiMO-NAS.

Ultimately, the models discovered by PiMO-NAS achieved competitive accuracy on the ImageNet-1k dataset, underlining the practicality of our method in real-world scenarios. The innovative blend of techniques within PiMO-NAS paves the way for future research in the efficient and effective exploration of complex neural architecture landscapes. Our findings and the developed method contribute a valuable addition to the field of NAS, promising to streamline the design of high-performing neural networks for diverse computational budgets.

This work acknowledges certain limitations, similar to those observed in studies like NSGA-NetV2, wherein the precision of Gaussian Process (GP) based landscape estimation of objective functions may vary across datasets, potentially impacting the quality of solution generation in PiMO-NAS. Additionally, the complexity of GP regression is tied to the number of samples, typically exhibiting a computational complexity of $O(n^3)$. This results in progressively longer iteration times for PiMO-NAS. In future research, we plan to optimize the cost of the Gaussian Process in this context and explore the potential of hybrid or adaptive surrogate models within the PiMO-NAS framework.

References

1. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing neural network architectures using reinforcement learning. arXiv preprint arXiv:1611.02167 (2016)
2. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Smash: one-shot model architecture search through hypernetworks. arXiv preprint arXiv:1708.05344 (2017)
3. Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S.: Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791 (2019)
4. Cai, H., Zhu, L., Han, S.: Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332 (2018)
5. Chen, B., Li, P., Li, C., Li, B., Bai, L., Lin, C., Sun, M., Yan, J., Ouyang, W.: Glit: Neural architecture search for global and local image transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12–21 (2021)
6. Chen, M., Peng, H., Fu, J., Ling, H.: Autoformer: Searching transformers for visual recognition (2021)
7. Chen, W., Huang, W., Du, X., Song, X., Wang, Z., Zhou, D.: Auto-scaling vision transformers without training. arXiv preprint arXiv:2202.11921 (2022)
8. Chu, X., Lu, S., Li, X., Zhang, B.: Mixpath: A unified approach for one-shot neural architecture search. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5972–5981 (2023)
9. 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)
10. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
12. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale (2021)

13. dAscoli, S., Touvron, H., Leavitt, M.L., Morcos, A.S., Biroli, G., Sagun, L.: Convit: Improving vision transformers with soft convolutional inductive biases. In: International conference on machine learning. pp. 2286–2296. PMLR (2021)
14. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
15. Guo, Y., Chen, Y., Zheng, Y., Chen, Q., Zhao, P., Huang, J., Chen, J., Tan, M.: Pareto-aware neural architecture generation for diverse computational budgets. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2247–2257 (2023)
16. 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)
17. Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., Tang, Y., Xiao, A., Xu, C., Xu, Y., et al.: A survey on vision transformer. IEEE transactions on pattern analysis and machine intelligence **45**(1), 87–110 (2022)
18. 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)
19. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1314–1324 (2019)
20. 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)
21. Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., Xing, E.P.: Neural architecture search with bayesian optimisation and optimal transport. Advances in neural information processing systems **31** (2018)
22. Knowles, J.: Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE transactions on evolutionary computation **10**(1), 50–66 (2006)
23. Lin, X., Yang, Z., Zhang, X., Zhang, Q.: Pareto set learning for expensive multiobjective optimization. Advances in Neural Information Processing Systems **35**, 19231–19247 (2022)
24. Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)
25. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021)
26. Lu, Z., Deb, K., Goodman, E., Banzhaf, W., Boddeti, V.N.: Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16. pp. 35–51. Springer (2020)
27. 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)
28. 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)

29. Peng, Y., Song, A., Ciesielski, V., Fayek, H.M., Chang, X.: Pre-nas: predictor-assisted evolutionary neural architecture search. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 1066–1074 (2022)
30. 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)
31. 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)
32. Su, X., You, S., Xie, J., Zheng, M., Wang, F., Qian, C., Zhang, C., Wang, X., Xu, C.: Vitas: Vision transformer architecture search. arXiv e-prints pp. arXiv–2106 (2021)
33. 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)
34. 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)
35. Termritthikun, C., Jamtsho, Y., Ieamsaard, J., Muneesawang, P., Lee, I.: Eeea-net: An early exit evolutionary neural architecture search. Engineering Applications of Artificial Intelligence **104**, 104397 (2021)
36. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International conference on machine learning. pp. 10347–10357. PMLR (2021)
37. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: International conference on machine learning. pp. 10347–10357. PMLR (2021)
38. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 568–578 (2021)
39. Wang, W., Zhang, X., Cui, H., Yin, H., Zhang, Y.: Fp-darts: Fast parallel differentiable neural architecture search for image classification. Pattern Recognition **136**, 109193 (2023)
40. Wang, W., Yao, L., Chen, L., Lin, B., Cai, D., He, X., Liu, W.: Crossformer: A versatile vision transformer hinging on cross-scale attention. In: International Conference on Learning Representations (2021)
41. 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)
42. Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., Yan, S.: Metaformer is actually what you need for vision. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10819–10829 (2022)
43. Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.H., Tay, F.E., Feng, J., Yan, S.: Tokens-to-token vit: Training vision transformers from scratch on imagenet. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 558–567 (2021)
44. Zhang, J., Xu, C., Li, J., Chen, W., Wang, Y., Tai, Y., Chen, S., Wang, C., Huang, F., Liu, Y.: Analogous to evolutionary algorithm: Designing a unified se-

quence model. Advances in Neural Information Processing Systems **34**, 26674–26688 (2021)
45. Zhou, Q., Sheng, K., Zheng, X., Li, K., Sun, X., Tian, Y., Chen, J., Ji, R.: Training-free transformer architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10894–10903 (2022)
46. Zong, Z., Cao, Q., Leng, B.: Rcnet: Reverse feature pyramid and cross-scale shift network for object detection. In: Proceedings of the 29th ACM International Conference on Multimedia. pp. 5637–5645 (2021)

A Search Space

A.1 Detailed Description of the Once-For-All Approach

The Once-For-All (OFA) methodology incorporates several key dimensions for searching optimal configurations in Convolutional Neural Networks (CNNs), specifically focusing on depth, input resolution, width, and kernel size. This approach is inspired by and follows the settings outlined in MSuNAS [26].

We structure the CNN into five distinct stages, with each stage undergoing thorough parameter exploration for depth, width, and kernel size as illustrated in Figure 7. The resolution parameter is adjusted between 192 to 256, with incremental steps of 4, offering 17 unique settings. Depth levels considered are 2, 3, or 4, while the expansion ratio is chosen from the set {3, 4, 6}, and the kernel size options are {3, 5, 7}. Notably, a fixed-length encoding scheme is employed for representing the neural network architectures. When the depth parameter does not equal 4, zero padding is applied to extend the block's encoding to a consistent length of 8, ensuring uniformity in representation across different configurations.

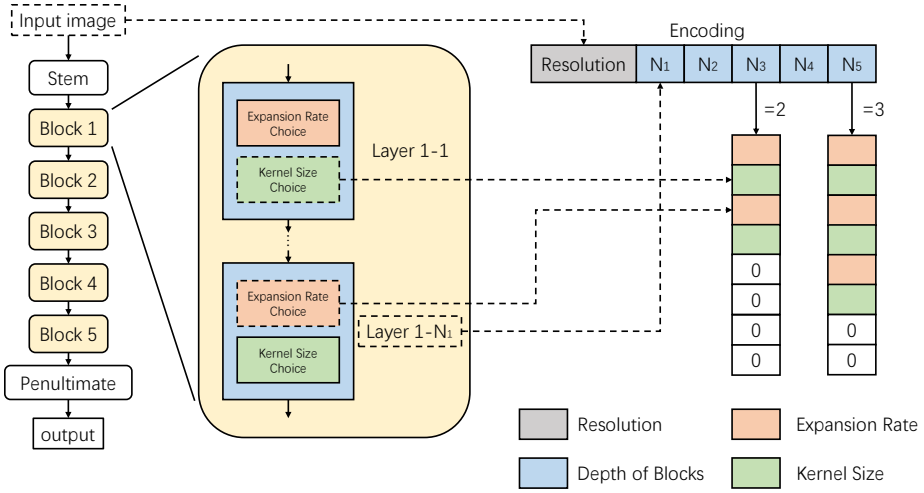

Fig. 7. OFA-based Search Space: The candidate architecture in the search space contains five blocks. Each block, denoted as b^{th} block, comprises N_b layers with a specific expansion rate and kernel size.

A.2 Detailed Description of the AutoFormer-based Search Space

In our approach, the AutoFormer-based search space is used to decompose the Vision Transformer (ViT) architecture into several fundamental dimensions,

specifically Embedding Dimension, MLP Ratio, Head Number, and Depth as Table 3.

The Depth and Embedding Dimension parameters are crucial as they directly influence the overall complexity of the ViT model. For each block stacked vertically in the model's depth, the MLP ratio and the Head Number parameters are explored independently. To standardize the configuration representation across different models, zero-padding is utilized to transform the parameter encoding into a fixed-length decision vector, as depicted in Figure 8.

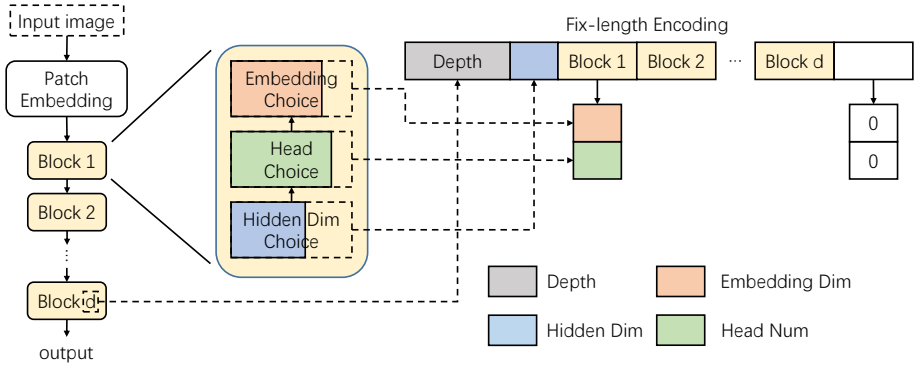

Fig. 8. AutoFormer Search Space: In this search space, the Vision Transformer architecture is conceptualized as a series of stacked blocks. Each block features a specific embedding dimension, while the number of attention heads and the hidden dimensions of the multi-layer perceptron within each block can be adjusted variably.

Table 3. Search space of AutoFormer. Tuples of three values represent the lowest value, the highest value, and steps.

	Supernet-tiny	Supernet-small	Supernet-base
Embed Dim	(192, 240, 24)	(320, 448, 64)	(528, 624, 48)
MLP Ratio	(3.5, 4, 0.5)	(3, 4, 0.5)	(3, 4, 0.5)
Head Num	(3, 4, 1)	(5, 7, 1)	(8, 10, 1)
Depth Num	(12, 14, 1)	(12, 14, 1)	(14, 16, 1)

B Parameter Restriction in AutoFormer

We replicated AutoFormer based on the source code and pre-trained networks provided at https://github.com/microsoft/Cream/tree/main/Autoformer. The neural architecture search (NAS) process within AutoFormer is conducted

by constraining the search parameter ranges and employing an evolutionary algorithm (EA). We set up seven different parameter restrictions based on the figures provided in the AutoFormer literature and descriptio in the project's GitHub issue https://github.com/microsoft/Cream/issues/74 for completing the replication process. The detailed settings of these seven parameter groups are shown in Table 4 and correspond one-to-one with Figure 5.

Table 4. AutoFormer Parameter Restriction

Supernet-Tiny		Supernet-Small		Supernet-Base	
Code	#Params	Code	#Params	Code	#Params
1	[5M, 6M]	2	[16M, 18M]	5	[41M, 44M]
		3	[22M, 24M]	6	[52M, 54M]
		4	[17M, 29M]	7	[65M, 68M]