Searching Efficient Semantic Segmentation Architectures via Dynamic Path Selection

Yuxi Liu[1,2*] **Min Liu**[1,2*] **Shuai Jiang**[1,2] **Yi Tang**[3†] **Yaonan Wang**[1,2]
[1]School of Artificial Intelligence and Robotics, Hunan University
[2]National Engineering Research Center of Robot Visual Perception and Control Technology
[3]Department of Data and Systems Engineering, The University of Hong Kong
yuxi_liu@hnu.edu.cn yiitang@hku.hk

Abstract

Existing NAS methods for semantic segmentation typically apply uniform optimization to all candidate networks (paths) within a one-shot supernet. However, the concurrent existence of both promising and suboptimal paths often results in inefficient weight updates and gradient conflicts. This issue is particularly severe in semantic segmentation due to its complex multi-branch architectures and large search space, which further degrade the supernet's ability to accurately evaluate individual paths and identify high-quality candidates. To address this issue, we propose Dynamic Path Selection (DPS), a selective training strategy that leverages multiple performance proxies to guide path optimization. DPS follows a stage-wise paradigm, where each phase emphasizes a different objective: early stages prioritize convergence, the middle stage focuses on expressiveness, and the final stage emphasizes a balanced combination of expressiveness and generalization. At each stage, paths are selected based on these criteria, concentrating optimization efforts on promising paths, thus facilitating targeted and efficient model updates. Additionally, DPS integrates a dynamic stage scheduler and a diversity-driven exploration strategy, which jointly enable adaptive stage transitions and maintain structural diversity among selected paths. Extensive experiments demonstrate that, under the same search space, DPS can discover efficient models with strong generalization and superior performance.

1 Introduction

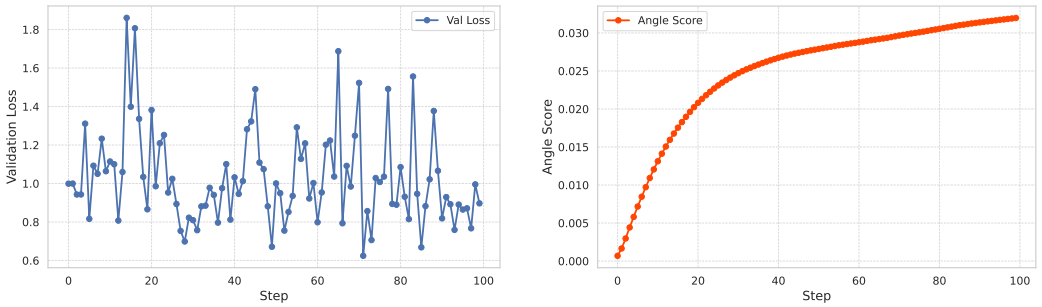
Semantic segmentation is a fundamental task in computer vision which focuses on pixel-level classification of images. In recent years, deep neural networks have achieved impressive success in semantic segmentation [1, 2, 3]. A critical factor behind this progress is the design of network architectures, which greatly impact overall performance. However, designing architectures that are both accurate and efficient often require substantial expertise and iterative experimentation.

To address this challenge, neural architecture search (NAS) has been proposed to automatically discover optimal architectures and reduce the need for manual effort. Current NAS methods [4, 5, 6, 7, 8, 9] typically embed the search space into a one-shot supernet and optimize it through weight sharing. Compared to traditional NAS methods, which usually require training thousands of networks, one-shot NAS significantly reduces the computational cost while having superior performance.

Nevertheless, the search space in NAS is enormous, especially for multi-branch segmentation networks (e.g., FasterSeg [8], with a search space of 10^{58}). Such space makes it difficult for the supernet's parameters to adapt to different subnets (paths) [10], leading to gradient oscillations [11].

*Equal contribution.
†Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).



(a) Validation loss (b) Angle score

Figure 1: Comparison of different performance proxies on a fixed path during the initial stage of supernet training. (a) Validation loss. (b) Angle score.

Additionally, the simultaneous presence of suboptimal and promising paths in the supernet further complicates the optimization process, as weak paths interfere with the highly shared weights [12].

To avoid these issues, some methods [13, 14] introduce a multi-path sampler, which samples multiple paths simultaneously during supernet training and filters out poor-performing ones, prioritizing training on paths with higher quality. These approaches solely rely on the validation loss to determine whether a path is promising. For example, GreedyNAS [13] computes the loss on only 2% of the validation set and use it as an evaluation metric. However, the limited sample size leads to high variance in performance estimates, making this metric unreliable—especially during the early stages of training. As shown in Fig. 1a, we randomly sampled a path during supernet training and observed significant fluctuations in its validation loss in the early stage, a pattern consistently found across other paths. While using a larger portion of the validation set could help reduce this variance, it would also incur substantial computational costs. Furthermore, relying solely on validation loss fails to capture a network's full potential from multiple perspectives. As suggested by recent study [15], the design of effective proxies should account for three key factors: expressiveness, generalization performance, and convergence.

In this paper, we have proposed corresponding proxies to quantify these three factors. For convergence, inspired by *ease-of-convergence hypothesis*[3], we adopt the angle score [17, 18], a metric that measures the distance between initialized and trained weights, to indicate the convergence behavior of a given path. As shown in Fig. 1b, this metric provides a more stable signal than validation loss, exhibiting a steady increase throughout training. For expressiveness estimation, an accuracy predictor is employed to generate relevant scores. Finally, leveraging insights from information bottleneck (IB) theory [19], we design a multi-scale IB (MS-IB) objective function tailored for segmentation models to assess their generalization capacities.

However, all three proxies cannot be uniformly utilized throughout supernet training, as their importance and reliability evolve across different training phases. For instance, the angle score tends stabilizes in the mid-to-late stages, as most paths have already converged by then, making it less discriminative for evaluating path quality. Similarly, the accuracy predictor used for expressiveness estimation must be trained on representative path samples. Nevertheless, in early training stages, the supernet produces unstable and noisy paths, which lead to unreliable training data for the predictor. As a result, accurate expressiveness estimation only becomes feasible once the supernet has reached a more stable phase. (Experimental evidence can be found in Appendix).

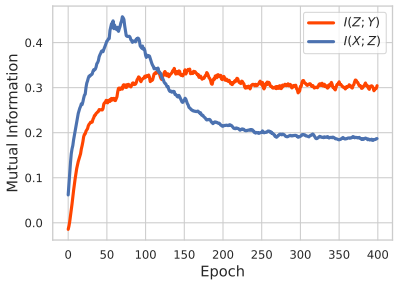


Figure 2: Changes in mutual information during supernet training.

Moreover, according to the IB theory [19], a well-generalized model should compress the input while preserving only the task-relevant features necessary for accurate prediction. To characterize this process, we monitor the mutual information between the input and intermediate representations (i.e., feature maps), denoted as $I(X;Z)$, and between the representations and the output, denoted

[3]A strong correlation exists between rapidly converging architectures and high-performing ones [16].

as $I(Z;Y)$. As shown in Fig. 2, $I(X;Z)$ first increases and then declines during early training, suggesting that the model first retains input information to fit the data, and subsequently compresses irrelevant features to improve generalization. At this point, the model has not started compressing redundant information, making MS-IB a relatively less accurate proxy for evaluating generalization.

Building on these empirical observations, we present a novel dynamic path selection (DPS) strategy that progressively shifts focus across these proxies throughout different stages of supernet training. Specifically, our method emphasizes convergence in the first stage, expressiveness in the middle stage, and a balanced combination of expressiveness and generalization in the final stage. As an additional practical consideration, FLOPs is included as a model complexity proxy across all training stages.

To realize this multi-stage, multi-objective framework, DPS integrates three key components. First, path selection is based on Pareto optimality, identifying non-dominated paths that achieve favorable trade-offs among multiple objectives. Second, a dynamic stage scheduler adaptively transitions between stages based on improvement trends and dynamic thresholds, eliminating the need for manual scheduling. Finally, to prevent the path search from collapsing into a narrow search space, we propose a diversity-driven exploration strategy that encourages sampling of structurally diverse paths.

Our main contributions are summarized as follows:

- We propose DPS, a selective training strategy for one-shot NAS in semantic segmentation, which dynamically shifts the selection focus across convergence, expressiveness, and generalization throughout supernet training. This stage-wise design enables comprehensive path evaluation and focuses optimization on high-quality paths in an adaptive manner.
- Three distinct proxies are proposed to evaluate path quality from multiple perspectives, aligning with the three stages of DPS. These metrics provide explicit feedback signals that guide the supernet training, helping to identify and optimize for promising paths during the search process.
- Extensive experiments demonstrate that DPS achieves state-of-the-art results within the same search space. The architectures discovered by DPS exhibit strong generalization, deliver superior performance, and maintain high efficiency.

2 Related Work and Background

In one-shot NAS, the architecture search space $\mathcal{A}$ is encoded into an over-parameterized supernet $\mathcal{N}(\mathcal{A}, W)$, where W denotes the set of shared weights across all candidate paths. The supernet is trained using a weight-sharing strategy and the corresponding training objective can be formulated as:

$$W_{\mathcal{A}} = \arg\min_{W} \mathcal{L}_{\text{train}}(\mathcal{N}(\mathcal{A}, W)), \tag{1}$$

where $\mathcal{L}_{\text{train}}$ is the loss function on the training set. Once the supernet has converged, the optimal path a^* can be obtained from the supernet:

$$a^* = \arg\max_{a \in \mathcal{A}} \text{Acc}_{\text{val}}(\mathcal{N}(a, w_a)), \tag{2}$$

where w_a is the weight inherits from $W_{\mathcal{A}}$ and Acc_{val} denotes the performance evaluated on the validation set.

As summarized in [20], one-shot NAS approaches mainly fall into two categories, distinguished by the way supernets are represented and optimized. The following provides an overview of each.

Gradient-based. Gradient-based NAS methods, such as DARTS [4], perform joint optimization of supernet parameters and weights through gradient descent within a differentiable search space. Due to their efficiency and end-to-end nature, most NAS approaches for semantic segmentation [7, 21, 22, 8, 9, 23] adopt this gradient-based paradigm. However, such joint optimization complicates the training of the supernet and inevitably introduces biases that may mislead the architecture search [6]. Moreover, these methods suffer from high GPU memory consumption [13] and struggle to incorporate different architecture constraints (e.g., FLOPs, latency and energy consumption) during the search process [6]. Instead, they often turn to relaxed regularization terms [24, 5], which offer limited efficacy in ensuring strict compliance with various constraints.

Sampling-based. Sampling-based NAS typically search on a discrete search space, and decouple the supernet training and architecture search into two separate stages. In the first stage, supernet was trained using different sampling strategies, such as uniform sampling [6, 25] or multi-path sampling [26, 27, 13, 14, 12] with different priorities. During this process, each sampled path is optimized individually, which alleviates the weight coupling issue observed in gradient-based NAS. In the second stage, typical search algorithms (e.g., evolutionary algorithms) are applied to identify the optimal path. Unlike gradient-based NAS, where hard constraints are difficult to enforce, sampling-based NAS can flexibly incorporate various resource constraints into different search algorithms.

3 Method

In this section, we will present our sampling-based NAS method named DPS. First, we begin with introducing different proxies to evaluating convergence, expressiveness, and generalization capacity of a candidate path. Then we discuss the path selection process in detail, which includes three main components: (1) a pareto optimality-based selection strategy, (2) a diversity-driven exploration mechanism, and (3) a dynamic stage scheduling approach for supernet training. Finally, the search pipeline based on evolution algorithm will be explained in short.

3.1 Proxies for Path Evaluation

Angle Score. To ensure a steady path selection in the early stage of supernet training, inspired by the *ease-of-convergence hypothesis*, we introduce an angle score [17, 18] to evaluate path convergence. Specifically, we compute the angle between the initial weights $\boldsymbol{w_0}$ and the trained weights $\boldsymbol{w}$ for path a:

$$\text{angle}(a) = \arccos\left(\frac{\langle \boldsymbol{V}(a, \boldsymbol{w_0}), \boldsymbol{V}(a, \boldsymbol{w})\rangle}{\|\boldsymbol{V}(a, \boldsymbol{w_0})\|_2 \cdot \|\boldsymbol{V}(a, \boldsymbol{w})\|_2}\right) \cdot \frac{1}{(1 + \lambda(t) \cdot n_a)^{\alpha}}, \tag{3}$$

where $\boldsymbol{V}(a, \boldsymbol{w_0})$ and $\boldsymbol{V}(a, \boldsymbol{w})$ are the concatenated weight vectors, obtained by flattening and stacking all weights along the path from input to output. However, in the early stages of training, certain paths may be sampled more frequently due to the stochastic nature of the search algorithm. This imbalance leads to overestimated angle scores for those paths—not because they converge better, but simply because they are updated more often. Such unfair comparisons can mislead the search process and favor paths with initialization advantages or those sampled more frequently due to randomness.

To mitigate this effect, we incorporate a sampling penalty term that decays with training steps, thereby reducing the impact of early-stage randomness and sampling imbalance on the convergence assessment. Here, n_a denotes the sum of sampling counts for all individual components (e.g., operations, connections) along path a, rather than the number of times the entire path is sampled—which is practically negligible due to the enormous search space (e.g., 10^{58}). The decay factor $\lambda(t)$ is a function of training step t, and it follows a linear decay schedule. The parameterα controls the overall penalty intensity.

Performance Predictor. To achieve accurate assessment of path expressiveness while avoiding the high computational cost of validation set evaluation, we train a performance predictor once the search process enters the second stage (e.g., 200th epoch). The following are the training details: (1) we begin by randomly sampling 1000 paths and measuring their actual accuracy through the current supernet. (2) we then split these samples into training and test sets at an 8:2 ratio. (3) Finally, we train a random forest regressor [28] on the training set and evaluate its performance on the test set. The regressor is configured with 100 decision trees and no maximum depth limitation.

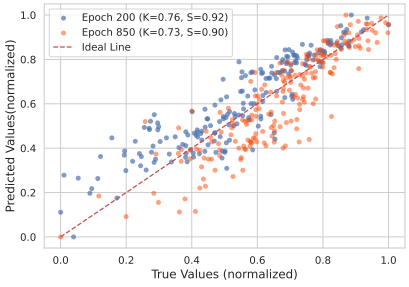


Figure 3: Rank correlation between the predicted values and the true values.

To evaluate the reliability of the performance predictor, we report Kendall's τ and Spearman's ρ computed on the test set. Furthermore, to ensure that the regressor maintains strong predictive capability in later stages, we randomly sample 200 paths at the 850th training epoch of the supernet

and measure their actual accuracy. The previously trained regressor (from the 200th epoch) is then used to predict their performance. As shown in Fig. 3, the random forest regressor achieves high rank correlation in both early and late stages (e.g., K = 0.76 / S = 0.92 at epoch 200; K = 0.73 / S = 0.90 at epoch 850), indicating consistent and reliable performance throughout the search process.

Information Bottleneck. Generalization is a fundamental property of deep neural networks (DNNs), as it determines how well a model performs on unseen data. According to the IB theory [19], DNNs achieve strong generalization by progressively compressing the input information and discarding task-irrelevant redundancy. This compression encourages the learning of compact and informative representations, which are less prone to overfitting and more likely to generalize well. To explicitly model this trade-off between information compression and task relevance preservation, the IB framework introduces a formal objective function:

$$\mathcal{L}_{IB}[p(z|x)] = I(Z;Y) - \beta I(X;Z), \tag{4}$$

where $I(X;Z)$ measures the mutual information between the input X and the learned representation Z, and $I(Z;Y)$ captures the informativeness of Z with respect to the target output Y. The parameter β controls the trade-off between compression and prediction. Mutual information $I(X;Y)$ can be defined as the Kullback-Leibler divergence between the joint distribution $p_{X,Y}$ and the product of marginal distributions $p_X \otimes p_Y$:

$$I(X;Y) = D_{KL}(p_{X,Y} \| p_X \otimes p_Y) = \mathbb{E}_{p_{X,Y}}[\log \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)}]. \tag{5}$$

This theoretical foundation motivates us to leverage the IB principle as a measure of generalization in our DPS framework. Specifically, by calculating the mutual information between input images X and feature maps Z, as well as between feature maps Z and output label Y, we construct an IB-based proxy that can select paths with better generalization capabilities. However, it is impossible to acquire exact distribution of X, Y and Z.

Inspired by [29, 30], we approximate $\mathcal{L}_{IB}[p(z|x)]$ by estimating a lower bound of $I(X;Z)$ and an upper bound of $I(Z;Y)$. This choice aligns with the IB objective: the lower bound on $I(X;Z)$ avoids overestimating compression and losing informative input features, while the upper bound on $I(Z;Y)$ prevents overestimating task relevance, ensuring sufficient predictive information is retained. As mentioned in [29], KL divergence has a dual representation known as the Donsker-Varadhan representation:

$$D_{KL}(P\|Q) = \sup_{T:\Omega\to\mathbb{R}} \left(\mathbb{E}_P[T] - \log(\mathbb{E}_Q[e^T])\right), \tag{6}$$

where T is any function from the space Ω to $\mathbb{R}$ such that the expectations are finite. To make the problem tractable, we parametrize the function T using a neural network, denoted as T_θ. Thus, $I(Z;Y)$ can be estimated as:

$$I(Z;Y) \geq \mathbb{E}_{p_{Z,Y}}[T_\theta] - \log(\mathbb{E}_{p_Z \otimes p_Y}[e^{T_\theta}]). \tag{7}$$

The network parameter θ are optimized by maximizing the objective function using gradient ascent (see Appendix A for details).

Having estimated the dependency between feature maps and labels, we now turn to the second term of the IB objective function. Since the direct computation of mutual information is generally intractable due to the unknown joint distribution $p_{X,Z}$, we adopt a variational approach known as variational Contrastive Log-ratio Upper Bound (vCLUB) [30], which provides a tractable upper bound of the mutual information. Specifically, it introduces a variational distribution $q_\phi(z|x)$ to approximate the true conditional distribution $p(z|x)$, and derives the following inequality:

$$I(X;Z) \leq \mathbb{E}_{p_{X,Z}}[\log q_\phi(z|x)] - \mathbb{E}_{p_X \otimes p_Z}[\log q_\phi(z|x)]. \tag{8}$$

In this formulation, the first expectation is taken over the true joint distribution $p_{X,Z}$, while the second is over the product of marginals $p_X \otimes p_Z$. The variational estimator $q_\phi(z|x)$ is implemented as a neural network parameterized by ϕ, and is trained to tighten the bound by maximizing the right-hand side of Eq. 8 (see Appendix A for details).

Multi-Scale Information Bottleneck (MS-IB). Based on Eq. (7) and Eq. (8), we implement an effective estimation of the Information Bottleneck (IB) objective. Nevertheless, this estimation is performed on a single-scale feature representation, often extracted from the final layer before the segmentation head.

For dense prediction tasks such as semantic segmentation, relying solely on the final representation Z may be suboptimal. This task requires capturing both fine-grained local details and high-level global context. Modern segmentation architectures [31, 32, 2, 33] typically exploit multi-scale feature maps extracted from different stages of the backbone network. Only using the final representation Z will neglects the potential redundancy or relevance encoded in other features with different scales.

To address this limitation, we propose the MS-IB, which extends the IB estimation to multiple stages of the segmentation network:

$$\mathcal{L}_{\text{MS-IB}} = \sum_{s=1}^{S} \left(I(Z_s; Y) - \beta I(X; Z_s) \right), \tag{9}$$

where S denotes the total number of stages considered. This formulation enables a more comprehensive evaluation of the generalization capacity of different paths.

3.2 Dynamic Path Selection

Path Selection with Pareto Optimality. We formulate path selection as a multi-objective optimization problem and adopt Pareto optimality to identify non-dominated candidates. At every selection step (every k iterations), we randomly sample m candidate paths from the supernet and evaluate them across multiple objectives—including expressiveness, convergence, and generalization ability—using proxies defined in 3.1. Computational cost (FLOPs) is also considered as an additional constraint. We subsequently determine the set of non-dominated paths among the candidates. A path a_i is said to *dominate* another path a_j if and only if:

$$\begin{aligned} &\forall x \in c, e, g, f, \quad x_{a_i} \leq x_{a_j}, \\ &\exists x \in c, e, g, f, \quad x_{a_i} < x_{a_j}, \end{aligned} \tag{10}$$

where c, e, g, and f denote the proxies for convergence, expressiveness, generalization, and negative FLOPs, respectively. The set of non-dominated paths forms the Pareto front, representing architectures that achieve optimal trade-offs among the competing objectives. Note that in Eq. (10), the metric f is used consistently throughout the entire supernet training process, while the other metrics are activated only during their respective stages.

Our objective is to select the top-k paths from the Pareto front. If the number of non-dominated candidates is fewer than k, we supplement the selection with additional high-quality paths, chosen based on a composite score calculated by summing their ranks across all proxies.

Diversity-driven Exploration. To prevent the path selection from collapsing into narrow and suboptimal regions of the search space, we introduce a diversity-driven exploration strategy that promotes structural diversity and balances exploitation with exploration.

Let $\mathcal{S} = \{a_1, a_2, \ldots, a_m\}$ be the set of m candidate paths sampled at each training iteration, and let $\mathbf{h}_i$ denotes the discrete structural encoding of path a_i, we define the *pairwise structure distance* between two paths a_i and a_j using a weighted Hamming distance:

$$\text{Dist}(a_i, a_j) = \sum_{l=1}^{L} \omega_l \cdot \mathbb{I}[\mathbf{h}_i^{(l)} \neq \mathbf{h}_j^{(l)}], \tag{11}$$

where L is the maximum encoding length (with padding applied as needed), ω_l is a weight for the l-th position and $\mathbb{I}[\cdot]$ represent the indicator function. Then we estimate a diversity score of each path by computing its average *pairwise structure distance* within the sampled set:

$$\text{diversity}(a_i) = \frac{1}{m-1} \sum_{j=1}^{K} \text{Dist}(s_i, s_j), \tag{12}$$

Finally, the path with the highest diversity score is considered for replacement. If its score exceeds a predefined threshold η, it replaces the worst-performing path in the top-k set to maintain diversity while preserving performance.

Dynamic Stage Scheduling. To maximize the utility of each proxy and facilitate effective, adaptive path selection, we propose a dynamic stage scheduling mechanism that prioritizes different proxy

metrics at different training stages. Specifically, the mechanism focuses on convergence in the first stage, expressiveness in the middle stage, and both expressiveness and generalization in the final stage.

Unlike methods that enforce rigid transitions between training phases, our approach dynamically adjusts the selection focus based on learning dynamics. Stage transitions are determined by monitoring the improvement trend of the current main metric over a sliding window of size t_{win}. A transition is triggered when the average improvement drops below a dynamic threshold:

$$\tau = \epsilon \cdot \text{std}(\text{history}[-t_{win}:]), \tag{13}$$

and remains non-negative. Here, $\text{history}[-t_{win}:]$ denotes the most recent t_{win} values of the current metric, and ϵ is a sensitivity coefficient that controls the responsiveness of the threshold to historical variation. Note that, in stage II, the performance predictor is used solely for path ranking and does not provide estimates of relative improvement over time. Therefore, we instead use the training loss to assess the improvement trend and determine whether a stage transition should occur.

This approach dynamically adjusts the threshold based on the volatility of the metric: high variance delays path switching to allow for further optimization, while low variance enables timely transitions when performance improvements plateau.

The details of overall dynamic path selection strategy is given in the Appendix B.

3.3 Evolution Search with Elite Population.

To better initialize the population in the evolutionary algorithm and improve search efficiency, we maintain an elite population $\mathcal{M}$ after entering final stage of the training process. Specifically, in each iteration, the top-k paths are added to this elite population. If the population exceeds the predefined size limit $|\mathcal{M}|$, the oldest individuals are removed to maintain a fixed capacity. This strategy ensures that high-quality paths are preserved and reused during the search process, thereby accelerating the discovery of optimal paths. The algorithm details of evolution search refer to Appendix C.

4 Experiments

4.1 Experimental Settings

Datasets. We conduct experiments on three widely used semantic segmentation datasets: Cityscapes[34], CamVid [35], and BDD100K[36]. Cityscapes contains 2,975 training, 500 validation, and 1,525 test images with a resolution of 1024 × 2048, annotated with 19 semantic classes. CamVid consists of 367 training, 101 validation, and 233 test images at 720 × 960 resolution, labeled with 11 categories. BDD100K provides 7,000 training and 1,000 validation images (resolution: 720 × 1280), also annotated with 19 semantic classes. All models are trained without external data such as ImageNet [37].

Search Space. For a fair comparison, we adopt the same search space from SqueezeNAS [21] and FasterSeg [8]. The first search space use inverted bottleneck [38] as the basic building block and have five different factors, including kernel size, expand ratio, dilation ratio, groups and network depth. The second search space supports multi-branch architecture design. It is based on an L-layer cell framework, where each branch operates at a distinct resolution scale. Within each branch, every cell has five different operation choices. The searchable downsample rates are set to $s \in \{8, 16, 32\}$, allowing flexible control of input resolution and feature hierarchy.

Implementation Details. The pipeline of our method is divided into three stages: (1) Supernet training with dynamic path selection. (2) Evolution search with elite population. (3) Network retraining. In concrete, once the supernet is fully converged, we will perform an evolution search on it to obtain the optimal path. For more details on the training and search settings, please refer to Appendix C and D.

4.2 Ablation Studies

Searching on same search space. To demonstrate the effectiveness of DPS, we first compared it with other one-shot NAS method using the same search space, including samlping-based, gradient-based

Table 1: Comparison of one-shot NAS methods on Cityscapes validation set and CamVid test set, categorized into sampling-based (S), gradient-based (G), and training-free (F) approaches; GFLOPs is measured using an input size of 1024 × 2048.

Method	Type	GFlops	mIoU (%)	
			Cityscapes	CamVid
SqueezeNAS Search Space				
Random Search	F	17.32	69.8	70.7
Uniform Sampling [6]	S	17.10	71.6	71.8
Loss Based [13]	S	12.57	72.7	73.1
SqueezeNAS [21]	G	10.86	72.4	73.2
Ours	S	10.96	**74.5**	**73.8**
FasterSeg Search Space				
Random Search	F	31.48	69.7	67.9
Uniform Sampling [6]	S	31.77	71.2	69.9
Loss Based [13]	S	30.64	70.9	71.8
FasterSeg [8]	G	28.20	73.1	71.1
SasWOT [39]	F	29.34	71.3	64.3
Ours	S	29.59	**73.2**	**72.1**

and training-free methods. Note that 'Loss Based' refers to using the validation loss as an evaluation metric to filter out suboptimal paths, as done in GreedyNAS [13]. As shown in Table 1, our DPS achieves the best performance on both search spaces. On the SqueezeNAS search space, our method outperforms the original gradient-based SqueezeNAS by 2.1% on Cityscapes and 0.6% on CamVid, while maintaining nearly the same computational cost (FLOPs). For the FasterSeg search space, [8] employed knowledge distillation to further boost accuracy, whereas our approach achieves slightly higher accuracy without relying on extra training strategies. These results demonstrate that DPS achieves an excellent trade-off between accuracy and computational efficiency, thereby showcasing its effectiveness in architecture search.

Effect of Different Proxies. To analyze the effect of different proxies, we conduct ablation studies on each component using various proxy combinations. The results are summarized in Table 2. We observe that increasing the number of proxies significantly improves the network performance. Specifically, the combination involving all three proxies achieves the best result, yielding an mIoU of 74.5%. In contrast, relying on a single proxy results in noticeably inferior performance. These findings highlight the importance of evaluating paths from multiple perspectives. By incorporating more proxies into the path selection process, our method is able to capture complementary signals during training, enabling a more comprehensive assessment of path quality. This, in turn, enhances the effectiveness of the architecture search and leads to better-performing models.

Table 2: Comparison of different proxy combinations on Cityscapes dataset.

Proxies			Stage	GFlops	mIoU (%)
c	e	g			
✓			1	14.33	72.8
	✓		2	13.82	72.6
		✓	3	11.91	72.8
✓	✓		1→2	12.51	73.8
✓		✓	1→3	12.12	73.4
	✓	✓	2→3	11.28	73.1
✓	✓	✓	1→2→3	10.96	**74.5**

Effect of Diversity-driven Exploration and Dynamic Stage Scheduler. We further study the impact of our proposed diversity-driven exploration (DDE) and dynamic stage scheduler (DSS). As summarized in Table 3, both strategies contribute to performance improvement, with DSS showing a more significant effect. This indicates that adaptive stage switching is critical for ensuring the reliability of the evaluation metrics. Using inappropriate fixed-stage switching strategies can greatly reduce metric reliability, potentially leading to suboptimal path selection.

Effect of MS-IB. To investigate the impact of MS-IB on model generalization, we design two different experiments. First, we evaluate the model's robustness against various common corruptions and perturbations following the experimental setup in [40]. Specifically, we construct a benchmark containing 16 types of algorithmically generated corruptions across four categories: noise, blur, weather, and digital. Under this setting, we compare the model's performance using single-scale

Table 3: Ablation study of diversity-driven exploration and dynamic stage scheduler.

DDE	DSS	GFlops	mIoU
		10.82	72.8
✓		11.47	73.4
	✓	10.31	73.8
✓	✓	10.96	**74.5**

Table 4: Ablation study of different IB configurations in transfer learning.

Method	GFlops	mIoU (%)	
		CamVid	BDD100K
None	4.12	67.7	43.5
Single-scale	3.72	**70.2**	44.9
Multi-scale	3.61	69.7	**45.2**

Table 5: Comparison of model performance under various corruptions across IB configurations.

Method	Blur				Noise				Digital				Weather				Avg
	Motion	Defoc	Glass	Gauss	Gauss	Impul	Shot	Speck	Bright	Contr	Satur	JPEG	Snow	Spatt	Fog	Frost	
None	63.1	61.3	58.0	65.1	3.4	4.3	4.1	18.7	57.9	49.3	49.4	34.7	**15.6**	51.1	32.2	15.0	36.4
Single-scale	64.9	63.3	**58.8**	66.5	**4.9**	7.4	5.0	17.9	58.3	50.8	**53.9**	**38.8**	14.9	50.4	31.5	**18.4**	37.9
Multi-scale	**65.0**	**63.7**	58.6	**66.9**	4.8	**9.9**	**6.3**	**25.9**	**61.1**	**53.4**	50.7	37.9	14.4	**52.9**	**35.3**	15.3	**38.9**

Table 6: Comparison of different NAS methods on Cityscapes. * denotes the model is reduced for acc-efficiency trade-offs, implemented by [41].

Method	Flops	Resolution	mIoU (%)	
			Val	Test
Auto-DeepLab* [7]	27.29	512 × 1024	71.2	-
HR-NAS-A [41]	1.91	512 × 1024	74.2	-
CAS [22]	-	768 × 1536	71.6	70.5
GAS [42]	-	769 × 1537	-	71.8
FasterSeg [8]	28.20	1024 × 2048	73.1	71.5
SqueezeNAS-Large [21]	10.86	1024 × 2048	72.4	-
SasWOT [39]	29.34	1024 × 2048	71.3	69.8
Ours	10.96	1024 × 2048	**74.5**	**73.5**

IB, multi-scale IB, and without the IB module. As presented in Table 5, the multi-scale IB module outperforms the baseline in the majority of corruption types, yielding an average mIoU gain of 2.5%. Meanwhile, the single-scale IB also demonstrates moderate improvements in robustness, with particularly strong performance observed on digital corruptions.

In the second experiment, we evaluate cross-dataset generalization by directly transferring Cityscapes-pretrained models to CamVid[35] and BDD100K [36], fine-tuning only the segmentation head while keeping the backbone frozen. As shown in Table 4, both single-scale and multi-scale IB strategies improve generalization over the baseline. The single-scale IB achieves the highest mIoU on CamVid (70.2%), slightly outperforming the multi-scale version (69.7%). This variant computes the IB based solely on the final feature before the segmentation head, whereas the multi-scale IB aggregates IB scores from multiple feature resolutions (e.g., 1/8, 1/16 of input size), aiming to capture a broader information flow. However, the modest performance drop with multi-scale IB on CamVid suggests that not all feature levels contribute equally to generalization. Since CamVid contains low-resolution images with simpler scenes, additional features may introduce redundancy rather than useful signal. These results indicate that future work could explore adaptive weighting of multi-scale features, emphasizing high-level ones near the output while maintaining lower-level regularization, to better balance informativeness and redundancy.

4.3 Comparison to state of the art methods.

In this experiment, we compare our method with existing state-of-the-art NAS approaches on the Cityscapes[34] datasets. As shown in Table 6, DPS achieves superior performance. Specifically, it obtains the highest mIoU on the Cityscapes validation and test sets, reaching 74.5% and 73.5%, respectively.

5 Conclusion

We propose DPS, a dynamic path selection strategy for one-shot NAS in semantic segmentation. By integrating stage-wise evaluation based on convergence, expressiveness, and generalization, DPS

adaptively focuses supernet training on high-quality paths, enabling targeted optimization. To align with these three stages, we introduce three complementary proxies that evaluate path quality from corresponding perspectives and guide selection through explicit feedback. Extensive experiments validate its effectiveness in identifying efficient models with strong generalization and performance.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB3303800, in part by the Natural Science Foundation of Hunan Province under Grant 2024JJ3013, in part by the National Natural Science Foundation of China under Grant 62425305, in part by the Science and Technology Innovation Program of Hunan Province under Grant 2023RC1048.

References

[1] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 801–818, September 2018.

[2] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *Proc. Adv. Neural Inf. Process. Syst.*, volume 34, pages 12077–12090, 2021.

[3] Jiacong Xu, Zixiang Xiong, and Shankar P. Bhattacharyya. Pidnet: A real-time semantic segmentation network inspired by pid controllers. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 19529–19539, June 2023.

[4] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *Proc. Int. Conf. Learn. Represent.*, 2019.

[5] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 10734–10742, June 2019.

[6] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 544–560, 2020.

[7] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L. Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 82–92, June 2019.

[8] Wuyang Chen, Xinyu Gong, Xianming Liu, Qian Zhang, Yuan Li, and Zhangyang Wang. Fasterseg: Searching for faster real-time semantic segmentation. In *Proc. Int. Conf. Learn. Represent.*, 2020.

[9] Xiong Zhang, Hongmin Xu, Hong Mo, Jianchao Tan, Cheng Yang, Lei Wang, and Wenqi Ren. Dcnas: Densely connected neural architecture search for semantic image segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 13956–13967, June 2021.

[10] Haibin Wang, Ce Ge, Hesen Chen, and Xiuyu Sun. Prenas: Preferred one-shot learning towards efficient neural architecture search. In *Proc. Int. Conf. Mach. Learn.*, pages 35642–35654, 2023.

[11] Chengyue Gong and Dilin Wang. Nasvit: Neural architecture search for efficient vision transformers with gradient conflict-aware supernet training. In *Proc. Int. Conf. Mach. Learn.*, 2022.

[12] Tao Huang, Shan You, Fei Wang, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Greedynasv2: Greedier search with a greedy path filter. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 11902–11911, June 2022.

[13] Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. Greedynas: Towards fast one-shot nas with greedy supernet. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 1999–2008, June 2020.

[14] Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. Attentivenas: Improving neural architecture search via attentive sampling. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 6418–6427, June 2021.

[15] Guihong Li, Duc Hoang, Kartikeya Bhardwaj, Ming Lin, Zhangyang Wang, and Radu Marculescu. Zero-shot neural architecture search: Challenges, solutions, and opportunities. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(12):7618–7635, 2024.

[16] Yao Shu, Wei Wang, and Shaofeng Cai. Understanding architectures learnt by cell-based neural architecture search. In *Proc. Int. Conf. Learn. Represent.*, 2020.

[17] Yiming Hu, Yuding Liang, Zichao Guo, Ruosi Wan, Xiangyu Zhang, Yichen Wei, Qingyi Gu, and Jian Sun. Angle-based search space shrinking for neural architecture search. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 119–134, 2020.

[18] Xuanyang Zhang, Pengfei Hou, Xiangyu Zhang, and Jian Sun. Neural architecture search with random labels. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 10907–10916, June 2021.

[19] Tishby, Naftali and Zaslavsky, Noga. Deep learning and the information bottleneck principle. In *Proc. IEEE Inf. Theory Workshop*, pages 1–5, 2015.

[20] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20(55):1–21, 2019.

[21] Albert Shaw, Daniel Hunter, Forrest Iandola, and Sammy Sidhu. SqueezeNAS: Fast neural architecture search for faster semantic segmentation. In *Proc. IEEE Int. Conf. Comput. Vis. Neural Architects Workshop*, 2019.

[22] Yiheng Zhang, Zhaofan Qiu, Jingen Liu, Ting Yao, Dong Liu, and Tao Mei. Customizable architecture search for semantic segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 11641–11650, June 2019.

[23] Yanyu Li, Changdi Yang, Pu Zhao, Geng Yuan, Wei Niu, Jiexiong Guan, Hao Tang, Minghai Qin, Qing Jin, Bin Ren, Xue Lin, and Yanzhi Wang. Towards real-time segmentation on the edge. In *Proc. AAAI Conf. Artif. Intell.*, volume 37, pages 1468–1476, Jun. 2023.

[24] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *Proc. Int. Conf. Learn. Represent.*, 2019.

[25] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 12270–12280, October 2021.

[26] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 12239–12248, October 2021.

[27] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 702–717, 2020.

[28] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[29] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *Proc. Int. Conf. Mach. Learn.*, pages 531–540, 2018.

[30] Pengyu Cheng, Weituo Hao, Shuyang Dai, Jiachang Liu, Zhe Gan, and Lawrence Carin. Club: A contrastive log-ratio upper bound of mutual information. In *Proc. Int. Conf. Mach. Learn.*, pages 1779–1788, 2020.

[31] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 2881–2890, July 2017.

[32] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 418–434, September 2018.

[33] Meng-Hao Guo, Cheng-Ze Lu, Qibin Hou, Zhengning Liu, Ming-Ming Cheng, and Shi-min Hu. Segnext: Rethinking convolutional attention design for semantic segmentation. In *Proc. Adv. Neural Inf. Process. Syst.*, volume 35, pages 1140–1156, 2022.

[34] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3213–3223, June 2016.

[35] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 44–57, 2008.

[36] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 2636–2645, June 2020.

[37] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 248–255, 2009.

[38] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4510–4520, June 2018.

[39] Chendi Zhu, Lujun Li, Yuli Wu, and Zhengxing Sun. Saswot: Real-time semantic segmentation architecture search without training. In *Proc. AAAI Conf. Artif. Intell.*, volume 38, pages 7722–7730, Mar. 2024.

[40] Christoph Kamann and Carsten Rother. Benchmarking the robustness of semantic segmentation models. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 462–483, June 2020.

[41] Mingyu Ding, Xiaochen Lian, Linjie Yang, Peng Wang, Xiaojie Jin, Zhiwu Lu, and Ping Luo. Hr-nas: Searching efficient high-resolution neural architectures with lightweight transformers. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 2982–2992, June 2021.

[42] Peiwen Lin, Peng Sun, Guangliang Cheng, Sirui Xie, Xi Li, and Jianping Shi. Graph-guided architecture search for real-time semantic segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4203–4212, 2020.

A Training Details of Mutual Information Estimator

Before entering the final stage of supernet training, we optimize two mutual information estimators to separately compute the two terms in MS-IB, which are used for evaluating path generalization. Once training reaches the final stage, these estimators are fixed and switched to evaluation mode. The detailed training procedures are shown in Algorithm 1 and Algorithm 2.

Algorithm 1 The optimization algorithm of T_θ.

Input: Neural network T_θ, training set $\mathcal{D}_{train} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, number of iterations T required to enter Stage III
Output: Optimized neural network T_θ

for $t = 1$ to T **do**
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}_{train}$:
  $\mathcal{B} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_b, y_b)\}$
  Compute Z by the sampled path a:
  $z_1, z_1, \ldots, z_b = a(x_1, x_2, \ldots, x_b)$
  Sample $\overline{Z}$ from marginal distribution of Z:
  $\overline{z_1}, \overline{z_2}, \ldots, \overline{z_b} \sim p_Z$
  Compute the value of $I(Z;Y)$ by Eq.7:
  $\mathcal{V} = \frac{1}{b}\sum_{i=1}^{b} T_\theta(\overline{z_i}, y_i) - \log(\frac{1}{b}\sum_{i=1}^{b} e^{T_\theta(\overline{z_i}, y_i)})$
  Update the parameters θ using gradient ascent:
  $\theta \leftarrow \theta + \frac{\partial \mathcal{V}}{\partial \theta}$
end for

Algorithm 2 The optimization algorithm of q_ϕ.

Input: Variational distribution $q_\phi(z|x)$, training set $\mathcal{D}_{train} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, number of iterations T required to enter Stage III
Output: Optimized neural network q_ϕ

for $t = 1$ to T **do**
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}_{train}$:
  $\mathcal{B} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_b, y_b)\}$
  Compute Z by the sampled path a:
  $z_1, z_2, \ldots, z_b = a(x_1, x_2, \ldots, x_b)$
  Sample $\overline{Z}$ from marginal distribution of Z:
  $\overline{z_1}, \overline{z_2}, \ldots, \overline{z_b} \sim p_Z$
  Compute the positive term:
  $L_{\text{pos}} = \frac{1}{b}\sum_{i=1}^{b} \log q_\phi(z_i|x_i)$
  Compute the negative term:
  $L_{\text{neg}} = \frac{1}{b^2}\sum_{i=1}^{b}\sum_{j=1}^{b} \log q_\phi(\overline{z_j}|x_i)$
  Compute the vCLUB estimate:
  $I_{\text{vCLUB}} = L_{\text{pos}} - L_{\text{neg}}$
  Update the parameters ϕ using gradient ascent:
  $\phi \leftarrow \phi + \frac{\partial I_{\text{vCLUB}}}{\partial \phi}$
end for

B Dynamic Path Selection

Details of dynamic path selection are presented below. We summarize the key settings as follows: $t_{win} = 20$, $m = 10$, $k = 5$, $\epsilon_1 = 2e-5$, $\epsilon_2 = 4e-5$.

Algorithm 3 Dynamic Path Selection (DPS)

Input: Supernet $\mathcal{N}$ with weights W, training set $\mathcal{D}_{train}$, performance predictor f_{pred}, max iteration T, window size t_{win}, diversity threshold η, sensitivity coefficients ϵ_1, ϵ_2

Initialize stage counters: $s = 1$
Track historical performance: history $\leftarrow$ []
for $t = 1$ to T **do**
  if $t \mod 5 == 1$ **then**
    Initialize score list: Scores $\leftarrow$ []
    Sample m paths from search space: $\mathcal{P} = \{a_1, ..., a_{10}\}$
    for each path $a_i \in \mathcal{P}$ **do**
      Compute FLOPs score: $\text{Score}_f(a_i) = -\text{FLOPs}(a_i)$
      if $s == 1$ **then**
        Compute convergence score: $\text{Score}_c(a_i) = \text{AngleScore}(a_i)$
      else if $s == 2$ **then**
        Compute expressiveness score: $\text{Score}_e = f_{pred}(a_i)$
      else
        Compute expressiveness score: $\text{Score}_e = f_{pred}(a_i)$
        Compute generalization score: $\text{Score}_g = \text{MS-IB}(a_i)$
      end if
      Scores $\leftarrow$ Scores $\cup \{(a_i, \text{Score}_f, \text{Score}_{c/e/g})\}$
    end for
    $\mathcal{P}_{top_k} \leftarrow \text{NonDominatedSort}(\text{Scores}, k)$
    if $|\mathcal{P}_{top_k}| < k$ **then**
      Supplement $\mathcal{P}_{top_k}$ with paths having lowest aggregated rank across all metrics
    end if
    Apply diversity-driven exploration:
    for each $a_i \in \mathcal{P} \setminus \mathcal{P}_{top_k}$ **do**
      if $\text{diversity}(a_i) > \delta$ **then**
        Identify $a_{\text{worst}} = \arg\max_{a \in \mathcal{P}_{top_k}} \text{RankSum}(a)$
        Replace a_{worst} in $\mathcal{P}_{top_k}$ with a_i
        break
      end if
    end for
    Train selected paths $\mathcal{P}_{top_k}$
    Update history:
    if $s == 1$ **then**
      history $\leftarrow$ history $\cup \left\{ \frac{1}{|\mathcal{P}|} \sum_{a_i \in \mathcal{P}} \text{AngleScore}(a_i) \right\}$
    else if $s == 2$ **then**
      history $\leftarrow$ history $\cup \left\{ \frac{1}{|\mathcal{P}_{top_k}|} \sum_{a_i \in \mathcal{P}_{top_k}} \text{TrainLoss}(a_i) \right\}$
    end if
    Update stage s based on learning dynamics:
    Extract recent window: $\mathcal{W} \leftarrow \text{history}[-t_{win}:]$
    Compute average improvement: $\Delta = \frac{1}{t_{win}-1} \sum_{i=1}^{t_{win}-1} (\mathcal{W}[i] - \mathcal{W}[i-1])$
    if $\Delta < \tau$ **then**
      $s \leftarrow s + 1$
    end if
  end if
end for

C Evolution Search with Elite Population

Details of the evolutionary search are presented below. We summarize the key settings as follows: $\mathcal{M} = 1,500$, $G = 20$, $N = 50$, $p_m = 0.2$, $p_s = 0.1$, $k = 10$.

Algorithm 4 Evolution Search

Input: Trained supernet $\mathcal{S}$, search space $\mathcal{A}$, elite population $\mathcal{M}$, number of generations G, population size N, mutation probability p_m, crossover probability p_c, validation set $\mathcal{D}_{val}$
Output: Optimal Architecture α^*

```
Initialize population: P ← RandomSample(M, N)
for t = 1 to G do
  P_parent ← Select top k subnets from P by mIoU
  P_child ← ∅
  if Random() < p_c then
    P_child ← P_child ∪ Crossover(P_parent)
  end if
  if Random() < p_m then
    P_child ← P_child ∪ Mutation(P_parent)
  end if
  if |P_child| < N then
    Sample N − |P_child| subnets from M and add to P_child
  end if
  P ← P_child
end for
α* ← arg max_{α∈P} mIoU(S(α), D_val)
return α*
```

D Implementation Details

Supernet Training. In this stage, we train the supernet on Cityscapes[34] dataset. We employ the stochastic gradient descent (SGD) optimizer with an initial learning rate of 0.01, momentum of 0.9, and weight decay of 0.0005. The learning rate is adjusted using a polynomial decay policy with a power of 0.9. During training, we apply standard data augmentation techniques including random cropping, random scaling, and horizontal flipping. All models are trained for 850 epochs with a total batch size of 12 across two RTX 3090 GPUs. In addition, the OHEM loss is adopted to enhance model performance.

Network Retraining. For network retraining, we adopt the same training strategy used for the supernet. In the case of transfer learning, the learning rate is set to 0.007, and the models are trained for 500 epochs on CamVid [35] and 200 epochs on BDD100K [36], with all other settings remaining unchanged.

E Experiments on Proxy Analysis

To examine the behavior of the convergence proxy, we tracked the angle score averaged over 10 randomly sampled paths at different training epochs. As reported in Table 7, the score increases rapidly in the early stage (0.65 at epoch 100) and gradually saturates after epoch 400 (around 1.23 at epoch 800). This confirms that the angle score becomes less discriminative in the mid-to-late stages.

To assess the reliability of the performance predictor across training, we measured its correlation with the ground-truth accuracy at different epochs. Table 8 reports Kendall's τ and Spearman's ρ correlations. The results indicate that the predictor is relatively noisy in the early stage (e.g., τ=0.53 at epoch 50), but its accuracy improves steadily and stabilizes after around 200 epochs (τ=0.76, ρ=0.92). This trend supports our design choice of activating the predictor only from Stage II onward, when the supernet has reached a more stable phase.

Table 7: Angle score across epochs.

Epoch	Angle Score (avg. 10 paths)
100	0.65
200	0.99
400	1.20
600	1.22
800	1.23

Table 8: Rank correlations between predicted and true accuracies across epochs.

Epoch	τ (Kendall)	ρ (Spearman)
50	0.53	0.72
100	0.63	0.82
200	0.76	0.92
400	0.75	0.92

F Searched Architectures

Table 9: Searched architecture using DPS on the SqueezeNAS search space. k, g, e, d denote kernel size, groups, expansion ratio, and dilation ratio, respectively.

Block	Operator	C_{out}	Down
1	k3_g2_e1_d1	16	2
2	k5_g1_e3_d1	24	4
3	k5_g2_e1_d1	24	4
4	k5_g1_e3_d1	24	4
5	k3_g1_e1_d1	24	4
6	k5_g1_e3_d1	32	8
7	k3_g1_e3_d2	32	8
8	k3_g1_e1_d1	32	8
9	k5_g1_e3_d1	32	8
10	k5_g1_e6_d1	64	16
11	k5_g1_e6_d1	64	16
12	k5_g1_e3_d1	64	16
13	Identity	64	16
14	k3_g1_e6_d1	96	16
15	k3_g2_e1_d1	96	16
16	k3_g1_e3_d1	96	16
17	k3_g1_e3_d2	96	16
18	k3_g1_e1_d1	160	16
19	k3_g1_e1_d2	160	16
20	k5_g1_e3_d1	160	16
21	k3_g1_e3_d2	160	16
22	k3_g1_e1_d2	160	16

Table 10: Searched architecture using DPS on the FasterSeg search space. Left: cells for branch with final downsample rate of 16. Right: cells for branch with final downsample rate of 32.

Cell	Operator	Exp.	C_{out}	Down
1	zoomed conv.	12	96	8
2	zoomed conv.	6	48	8
3	zoomed conv.	10	80	8
4	conv.	4	32	8
5	zoomed conv.	4	64	16
6	zoomed conv.	4	64	16
7	zoomed conv.	8	128	16

Cell	Operator	Exp.	C_{out}	Down
1	zoomed conv.	12	96	8
2	conv.	6	96	16
3	zoomed conv. ×2	4	128	32
4	zoomed conv. ×2	4	128	32
5	zoomed conv.	12	384	32
6	zoomed conv.	4	128	32

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The contributions and scope are clearly presented in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitation is discussed in sec 4.2, effect of MS-IB.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: This paper does not contain any mathematical proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The experiment settings and implementation details are described in Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: See supplemental material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experiment settings and implementation details are described in Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not report error bars, as prior literature on the same benchmarks typically does not include them. However, we conduct extensive experiments on three benchmarks to support the robustness of our method.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computer resources we used are listed in Appendix D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We follow the rules carefully.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited all the public datasets we used.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: We only use LLM for writing and editing.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.