

IZeroDNAS: innovative differentiable NAS using zero-cost metrics and SuperNet initialization

Van-Tin Luu, Tao-Ting Chen and Khanh-Phong Bui

Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan

Vu-Hoang Tran

*Faculty of Electrical and Electronics Engineering,
Ho Chi Minh City University of Technology and Engineering,
Ho Chi Minh City, Vietnam*

Yung-Hui Li

Hon Hai Research Institute, Taipei City, Taiwan, and

Ching-Chun Huang

Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan

Abstract

This paper improves the stability and hardware-efficiency of zero-cost differentiable neural architecture search (NAS) for object detection, addressing two weaknesses of prior work: zero-cost rankings sensitive to SuperNet initialization and neglected edge-device deployment constraints. The authors propose IZeroDNAS, which integrates the NASWOT zero-cost metric into differentiable NAS. The authors introduce a SuperNet warm-up initialization, plus a depthwise-accelerated variant, to stabilize the rankings and a hardware-aware loss combining latency-aware and depth-aware terms built from a device look-up table. The authors evaluate on the ScaledYOLOv4 and YOLOv7 search spaces using Pascal VOC and COCO and deploy on Qualcomm RB5, Jetson TX2 and Jetson AGX Orin. Warm-up initialization reduces the mean average precision variance across random seeds from 5.687 to 0.0025 while raising accuracy. IZeroDNAS discovers shallower, wider models that are smaller and faster than manually designed detectors and the baseline ZeroDNAS, completing a search in under 5 h on a single RTX 2080, three times faster than comparable methods. The warm-up's stabilizing

© Van-Tin Luu, Tao-Ting Chen, Khanh-Phong Bui, Vu-Hoang Tran, Yung-Hui Li and Ching-Chun Huang. Published by Emerald Publishing Limited. This article is published under the Creative Commons Attribution (CC BY 4.0) licence. Anyone may reproduce, distribute, translate and create derivative works of this article (for both commercial and non-commercial purposes), subject to full attribution to the original publication and authors. The full terms of this licence may be seen at <http://creativecommons.org/licences/by/4.0/>

Funding: This work was financially supported in part (project number: 112UA10019) by the Co-creation Platform of the Industry Academia Innovation School, NYCU, under the framework of the National Key Fields Industry-University Cooperation and Skilled Personnel Training Act, from the Ministry of Education (MOE) and industry partners in Taiwan. It also supported in part by the National Science and Technology Council, Taiwan, under Grant NSTC-112-2221-E-A49-089-MY3, Grant NSTC-110-2221-E-A49-066-MY3, Grant NSTC-111-2634-F-A49-010, Grant NSTC-112-2425-H-A49-001 and in part by the Higher Education Sprout Project of the National Yang Ming Chiao Tung University and the MOE, Taiwan. The authors also would like to express their gratitude for the support from Hon Hai Research Institute (HHRI).



effect is shown empirically rather than proven theoretically and a strictly fair comparison with proprietary methods such as YOLO-NAS is infeasible due to their undisclosed search spaces. Future work targets broader zero-cost proxies and more accurate device-performance predictors. IZeroDNAS is the first method to jointly tackle ranking instability and hardware-awareness in zero-cost differentiable NAS for object detection. Its warm-up and depthwise-accelerated variant offer a simple, modular, resource-efficient way to stabilize zero-cost metrics in large-scale SuperNets, benefiting practitioners deploying detectors on edge devices.

Keywords Neural architecture search (NAS), Object detection, Edge devices, Zero-cost proxies

Paper type Research paper

1. Introduction

With the rapid advancement of deep learning, an increasing number of neural network architectures have been developed to meet the growing demand for both high performance and computational efficiency. In many real-world applications, such as autonomous driving and real-time object tracking, models are required not only to achieve high accuracy but also to satisfy strict hardware constraints, including low latency, limited memory footprint and reduced energy consumption. To address these challenges, neural architecture search (NAS) has emerged as a promising paradigm for automatically discovering network architectures that are tailored to specific tasks and deployment conditions.

By systematically exploring the design space, NAS relieves researchers from the burden of manual architecture engineering. In practice, particularly for edge computing and embedded AI systems, the discovered architectures must be compact and hardware-aware – optimized not only for accuracy but also for resource metrics such as FLOPs, latency and model size. Despite its potential, NAS remains a highly computationally expensive process. Traditional sampling-based NAS methods – including reinforcement learning-based, evolutionary and differentiable approaches (Liu *et al.*, 2019; Real *et al.*, 2019; Zoph and Le, 2016) – often require hundreds or even thousands of GPU days to converge to a final architecture. These methods typically treat NAS as a black-box optimization over a discrete and combinatorial space, necessitating the evaluation of a large number of sampled architectures and leading to significant inefficiency.

SuperNet-based Differentiable NAS (DNAS) methods (Guo *et al.*, 2020a; Liu *et al.*, 2019; Yu *et al.*, 2020) have been introduced to alleviate the computational burden of conventional NAS. By consolidating all candidate architectures into a unified SuperNet framework and relaxing the discrete search space into a continuous one, these approaches enable gradient-based optimization. Consequently, they substantially accelerate the search process compared to purely sampling-based methods. Nevertheless, even with DNAS, training a large SuperNet remains time-consuming, particularly for large-scale tasks such as object detection.

Recent works have explored zero-cost proxies (Abdelfattah *et al.*, 2021; Li *et al.*, 2023; Mellor *et al.*, 2021) as a complementary approach to further reduce search time. These proxies estimate a model's performance without requiring training and, when combined with sampling-based strategies like evolutionary algorithms, can efficiently rank candidate architectures. However, most existing metrics were originally designed for simpler classification models or relatively small search spaces (e.g. networks with fewer than 10 M parameters) and do not generalize well to the complex architectures typical of object detection tasks.

To overcome these limitations, we present an efficient search framework for large-scale object detection models, namely, ZeroDNAS and its improved variant IZeroDNAS. ZeroDNAS integrates zero-cost metrics directly into the differentiable NAS optimization process, eliminating the need to train candidate models during the search. This enables differentiable NAS to be applied to macro-architecture search spaces while significantly reducing computational cost. Despite these advantages, Table 5 shows that zero-cost

rankings are highly sensitive to SuperNet initialization. Variations in initial weights produce inconsistent rankings, leading to unstable search behavior and degraded performance. Moreover, the original ZeroDNAS does not account for deployment constraints such as inference latency and model size, limiting its suitability for edge devices. These issues are summarized in the upper section of Figure 1.

We address these challenges with IZeroDNAS, an enhanced version of ZeroDNAS that incorporates SuperNet warm-up initialization and hardware-aware loss functions, as illustrated in the lower section of Figure 1. Before architecture search, the SuperNet is trained for a few epochs to better approximate realistic performance behavior. This warm-up step substantially improves the consistency of zero-cost rankings across different random initializations, yielding more stable and robust search outcomes. Crucially, this phase does not aim to improve model accuracy but to stabilize ranking, which differentiates it from conventional pretraining and requires minimal additional computation. Once warm-up is completed, the search proceeds using the ZeroDNAS algorithm. Finally, hardware-aware loss terms – including depth-aware and latency-aware objectives – guide the search toward compact, fast and device-optimized architectures suitable for real-world deployment. Our contributions can be summarized as follows:

- We propose IZeroDNAS, a fast and robust NAS framework that integrates zero-cost proxies into differentiable NAS. This substantially reduces search time and memory usage, while enabling architectures tailored to device-specific constraints.
- We introduce SuperNet warm-up initialization to enhance the stability of zero-cost rankings in a resource-efficient manner. To balance stability and efficiency, we further provide two variants: a standard warm-up for robust accuracy and a depthwise-accelerated warm-up that significantly reduces initialization time.
- We incorporate hardware-aware loss terms to guide the search toward architectures with fewer parameters and faster inference on the target device.

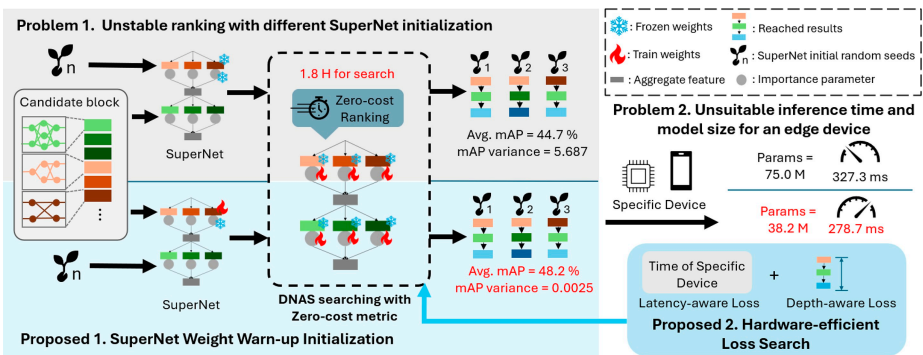


Figure 1. Problems and our proposed methods. The upper part illustrates two issues with ZeroDNAS (Poliakov *et al.*, 2023) and the lower part shows our proposed solutions to address these problems. *Problem 1:* the unstable zero-cost ranking due to different SuperNet initializations leads to varying search results and inconsistent performance, causing instability in the search algorithm. *Problem 2:* ZeroDNAS does not consider real-world deployment, resulting in models that are unsuitable for specific devices. *Proposed 1:* before the search stage, warm-up initialization trains the SuperNet weights for a few epochs to enhance search stability. *Proposed 2:* by incorporating hardware-aware loss, our method allows the search to identify models that are suitable for real-world applications

- Our framework is evaluated across diverse search spaces, including ScaledYOLOv4 (Wang *et al.*, 2021a) and YOLOv7 (Wang *et al.*, 2023), demonstrating superior performance compared to manually designed object detection models.

Relationship to prior work: This work extends our previous conference paper, ZeroDNAS (Poliakov *et al.*, 2023), which first explored the combination of differentiable NAS and zero-cost proxies for efficient architecture search. While ZeroDNAS significantly accelerated the search process, it did not address issues of ranking instability or deployment constraints. In this work, we address these limitations by proposing IZeroDNAS, which incorporates warm-up initialization and hardware-aware objectives. Furthermore, we provide new experimental analyses to demonstrate the robustness and generalization of the proposed method across search spaces and random seeds.

2. Related works

In this section, we comprehensively review prior works related to NAS, with an emphasis on its application to object detection. We begin with a discussion of the evolution of NAS techniques, from sampling-based and differentiable approaches to recent advancements in zero-cost proxies. Next, we explore existing NAS strategies for object detection, highlighting their limitations in practical deployment scenarios. Finally, we examine manually designed object detectors that offer valuable insights into efficient and accurate architectural design, which we leverage to construct a more effective NAS search space.

2.1 Neural architecture search

NAS has emerged as a powerful approach to automate neural network design. Early methods, including NAS-RL (Zoph and Le, 2016) and Aging Evolution (Real *et al.*, 2019), primarily relied on sampling-based strategies that repeatedly train and evaluate candidate architectures. While effective in discovering competitive models, these approaches require enormous computational resources, often reaching thousands of GPU days, which limits their practical applicability.

To alleviate this burden, weight-sharing techniques such as ENAS (Pham *et al.*, 2018), SPOS (Guo *et al.*, 2020a) and BigNAS (Yu *et al.*, 2020) introduced the concept of a SuperNet, where multiple architectures are embedded and share parameters during training. These methods significantly reduce the training cost by avoiding redundant training of individual models. Nevertheless, due to their reliance on discrete sampling, they often struggle to capture fine-grained architectural trade-offs and may converge to suboptimal solutions.

To further improve efficiency and search quality, differentiable NAS (DNAS) approaches such as DARTS (Liu *et al.*, 2019) and FBNet (Wu *et al.*, 2019) relaxed the search space from discrete to continuous domains. This enabled the use of gradient-based optimization to simultaneously tune both architecture and operation weights. While these methods offer improved convergence speed and search quality, they demand high GPU memory and often suffer from overfitting or performance collapse when scaled to more complex tasks beyond image classification.

Recently, zero-cost proxies have emerged as a compelling alternative to reduce the computational demands of NAS. These methods aim to estimate the relative performance of architectures without any training or with minimal computation. For instance, NASWOT (Mellor *et al.*, 2021), SynFlow (Abdelfattah *et al.*, 2021), ZiCo (Li *et al.*, 2023) and SpeedyNAS (Ru *et al.*, 2021) introduced analytical or gradient-based metrics to quickly rank candidate architectures. While these proxies have shown promise on classification benchmarks, they often lack generalization when applied to large-scale or complex tasks

such as object detection. Furthermore, they are known to be highly sensitive to model initialization and network depth, which undermines their stability and robustness in real-world NAS pipelines.

2.2 NAS for object detection

Designing NAS pipelines for object detection presents a unique set of challenges, primarily due to the increased complexity and computational requirements of detection networks. Unlike classification tasks that involve a single prediction head, object detectors consist of multi-stage pipelines including backbone, feature pyramid networks (FPN) and detection heads. This makes the search space significantly larger and more entangled.

Early efforts in NAS for object detection, such as Auto-FPN (Xu *et al.*, 2019), NAS-FPN (Ghiasi *et al.*, 2019) and NAS-FCOS (Wang *et al.*, 2020a), concentrated on searching the FPN structure. These works demonstrated performance gains but often resulted in irregular and overly complex topologies that hinder real-time deployment, particularly on resource-constrained devices.

To address the fragmented nature of these search results, subsequent methods like SM-NAS (Yao *et al.*, 2020) and Hit-Detector (Guo *et al.*, 2020b) proposed a joint search of backbone and FPN components. This holistic view led to more coherent architectures and improved accuracy, but at the expense of increased GPU memory usage and search time, which remain prohibitive for many practitioners.

EAutoDet (Wang *et al.*, 2022b) improved upon this by constraining the SuperNet size and optimizing differentiable parameters under a tighter computational budget. While this approach achieved a better trade-off between performance and efficiency, it still required expensive hardware (e.g. NVIDIA V100 GPUs) and multi-day search periods.

Recent approaches have explored integrating zero-cost proxies into object detection NAS frameworks. For instance, MAEDet (Sun *et al.*, 2022) proposed a memory-aware proxy to guide early-stage pruning. Zero-DNAS (Poliakov *et al.*, 2023) introduced a constraint-aware NAS framework that leverages zero-cost metrics and latency prediction to efficiently search for deployment-friendly object detectors. The authors demonstrated significant speedups in the search process while maintaining competitive accuracy (Poliakov *et al.*, 2023). However, these proxy-based NAS methods often inherit the instability issues discussed earlier and may fail to generalize across different detection frameworks or deployment platforms. Moreover, many of these works ignore hardware-awareness during optimization, leading to architectures that are theoretically efficient but practically unoptimized for real devices.

2.3 Manually designed detectors as NAS guidance

While NAS methods strive to automate architecture design, many of the most effective object detectors in recent years – such as YOLOv7, PP-YOLOE and RT-DETR – have been manually crafted through expert knowledge and extensive empirical validation. These models illustrate several valuable design principles, including scale-aware representations, early fusion strategies and re-parameterized convolution blocks, which have proven effective across diverse tasks and platforms.

Despite their handcrafted nature, these models consistently outperform many NAS-based detectors in terms of accuracy-efficiency trade-offs, particularly on edge devices. Thus, they offer a strong prior for constructing NAS search spaces that are both expressive and hardware-friendly. Unfortunately, many prior NAS methods neglect these insights, resulting in either overly large or inefficient architectures.

In our work, we address these limitations by incorporating inductive biases from manually designed models into our NAS search space. Moreover, we propose a robust NAS

framework that includes SuperNet warm-up initialization to stabilize zero-cost metrics and a hardware-aware loss function that aligns the search process with deployment requirements. Details of our proposed approach are elaborated in Section 5.

3. Preliminary

The classical NAS problem is defined as:

$$\min_{\alpha \in A} \min_{w_\alpha} L(\alpha, w_\alpha), \quad (1)$$

where A is the search space and α is an architecture sampled from A . The goal is to identify an architecture α that achieves the best performance when trained with weights w_α .

Differentiable NAS (DNAS) (Wu *et al.*, 2019) reformulates NAS as a continuous optimization problem by introducing trainable importance parameters θ . Once optimized, these parameters are used to compute architecture probabilities p via the Gumbel-Softmax (Jang *et al.*, 2017), which indicates the selection probability of each candidate. Specifically, the architecture probability $p_{i,j}$ for the j -th candidate in the i -th stage is given by:

$$\begin{aligned} P_{\theta_i}(b_i = b_{i,j}) &= p_{i,j} = \text{GumbelSoftmax}(\theta_i) \\ &= \frac{\exp((\theta_{i,j} + g_{i,j})/t)}{\sum_j \exp((\theta_{i,j} + g_{i,j})/t)}, \end{aligned} \quad (2)$$

where θ_i denotes the importance array of i -th searchable stage and $\theta_{i,j}$ is the importance parameter of j -th choice in θ_i . $g_{i,j}$ represents the noise from a Gumbel distribution and the temperature t adjusts the softness of the output probability.

An architecture α can then be sampled according to these probabilities, with the sampling probability expressed as:

$$P_\theta(\alpha) = \prod_i P_{\theta_i}(b_i = b_{i,j}), \quad (3)$$

where $b_{i,j}$ is j -th candidate block at i -th searchable stage and b_i is a block of i -th stage that sampled from a candidate set $\{b_{i,j}\}$.

The objective of DNAS is to simultaneously optimize the importance parameters θ and the SuperNet weights w to sample high-performing architectures, formalized as:

$$\min_{\theta} \min_w E_{\alpha \sim P_\theta} L(\alpha, w). \quad (4)$$

Thus, the primary bottleneck in DNAS lies in training both the SuperNet weights w and the architecture parameters p , leading to a highly time-consuming process. Moreover, training all candidate blocks in parallel requires substantial memory, especially as the search space grows.

4. ZeroDNAS algorithm

To address the computational challenges in DNAS, including high training time and memory consumption, ZeroDNAS (Poliakov *et al.*, 2023) decouples the training of SuperNet weights and architecture parameters by incorporating zero-cost proxies. It optimizes only the importance parameters, leveraging the NASWOT (Mellor *et al.*, 2021) metric for architecture performance estimation.

4.1 NASWOT metric

The key idea of NASWOT is that better architectures tend to produce more diverse feature maps for different input images. It evaluates the similarity between activation outputs of a randomly initialized network across different inputs within a mini-batch. First, activation patterns of k input samples are encoded as binary codes. Next, a correlation matrix K is formed based on the Hamming distances among these codes. Finally, the WOT score is computed as the logarithm of the determinant of K :

$$WOTmetric = \log|K|. \quad (5)$$

where a higher WOT score indicates better expected architecture performance.

Unlike the original NASWOT, which computes scores for the entire architecture, ZeroDNAS evaluates the WOT metric individually for each candidate to align with the architecture parameters. A WOT score map $M_{WOT} \in \mathbb{R}^{|O_i| \times S}$ is constructed for the search, where $|O_i|$ is the number of candidates in the i -th stage and S is the total number of searchable stages.

4.2 Architecture searching

To discover an optimal architecture that balances accuracy and model complexity, ZeroDNAS measures the FLOPs of each candidate block, storing them in an FLOPs map $M_{flops} \in \mathbb{R}^{|O_i| \times S}$. The importance parameters θ are then updated based on both M_{WOT} and M_{flops} , reflecting the trade-off between candidate quality and computational cost. The expected zero-cost score and FLOPs of the SuperNet are computed using the architecture probabilities p from [equation \(2\)](#):

$$\mathbb{E}[WOT] = \sum_i \sum_j M_{WOTi,j} \times p_{i,j}, \quad (6)$$

$$\mathbb{E}[flops] = \sum_i \sum_j M_{flopsi,j} \times p_{i,j}, \quad (7)$$

where $M_{WOTi,j}$ and $M_{flopsi,j}$ correspond to the zero-cost score and FLOPs of the j -th candidate block at the i -th stage.

Finally, to define the objective for optimizing the importance parameter θ under the constraint of target FLOPs, ZeroDNAS incorporates the zero-cost loss L_{zero_cost} and FLOPs loss L_{flops} for the search. The FLOPs loss is measured by the mean square error between the achieved and target FLOPs:

$$L_{zero_cost} = -\mathbb{E}[WOT] \times \lambda, \quad (8)$$

$$L_{flops} = (\mathbb{E}[flops] - T_{flops})^2 \times \phi, \quad (9)$$

$$L_{overall} = L_{zero_cost} + L_{flops}, \quad (10)$$

where λ and ϕ control the relative importance of zero-cost performance and computational cost.

5. IZeroDNAS

Although ZeroDNAS (Poliakov *et al.*, 2023) addressed the search time challenges by using a zero-cost proxy, we found that its results are sensitive to different SuperNet initializations and are not well-suited for real-world device applications, as illustrated in Figure 1. To overcome these limitations, we propose IZeroDNAS, which builds upon ZeroDNAS (Poliakov *et al.*, 2023), incorporating SuperNet warm-up initialization and hardware-aware loss. The whole IZeroDNAS process is illustrated in Figure 2. For a fair comparison of our improvements, we use the same ScaledYOLOv4 search space as ZeroDNAS. In addition, we designed a YOLOv7 search space to accommodate future applications.

5.1 Search space

We consider two primary search spaces in our experiments: the ScaledYOLOv4 search space and a newly designed YOLOv7 search space, reflecting recent architectural advancements and practical application scenarios. The detailed configurations of both search spaces are described below.

5.1.1 ScaledYOLOv4 search space. Our SuperNet integrates all candidate architectures into a single structure, accelerating the search process and increasing the likelihood of discovering optimal designs. Unlike early cell-based NAS methods such as DARTS (Liu *et al.*, 2019) and ENAS (Pham *et al.*, 2018), which often produce fragmented architectures, we adopt a block-wise search approach inspired by FBNet (Wu *et al.*, 2019), which generalizes better to practical applications.

The ScaledYOLOv4 search space builds on the manually designed model (Wang *et al.*, 2021a), which leverages cross stage partial (CSP) blocks (Wang *et al.*, 2020b) to improve gradient flow and reduce computational cost. Each stage corresponds to a CSP block, parameterized by d , the number of Darknet bottleneck modules and g , the channel reduction ratio. As shown in Figure 3(a), the backbone and neck each contain four searchable stages (green blocks). Backbone stages use $d \in 2, 4, 6, 8$ and $g \in 0.25, 0.50, 0.75$, while neck stages use $d \in 0, 2, 4, 6$ with the same g values, yielding 12 candidate options per stage and a total of $12^8 \approx 4.3 \times 10^8$ architectures.

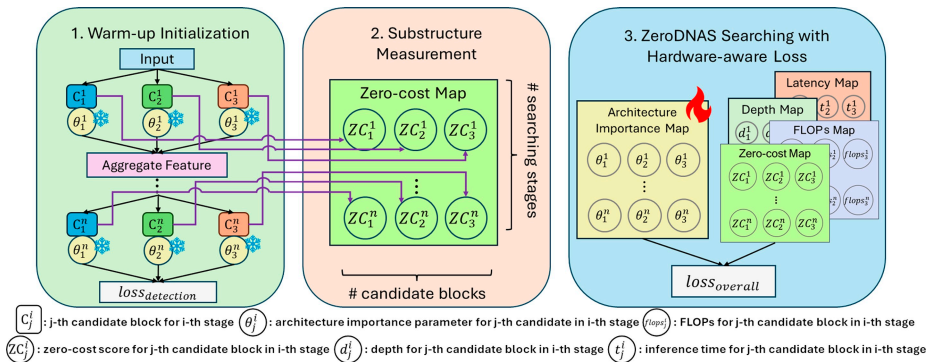


Figure 2. Search process of IZeroDNAS. First, the SuperNet weights are warmed up over a few epochs while keeping the importance parameters fixed. Next, for each candidate block in the SuperNet, the zero-cost score and other model complexity metrics (e.g. FLOPs) are measured to create a “zero-cost map” along with other model complexity maps. Finally, these maps are used to calculate objective losses, allowing the architecture to be determined through gradient descent optimization

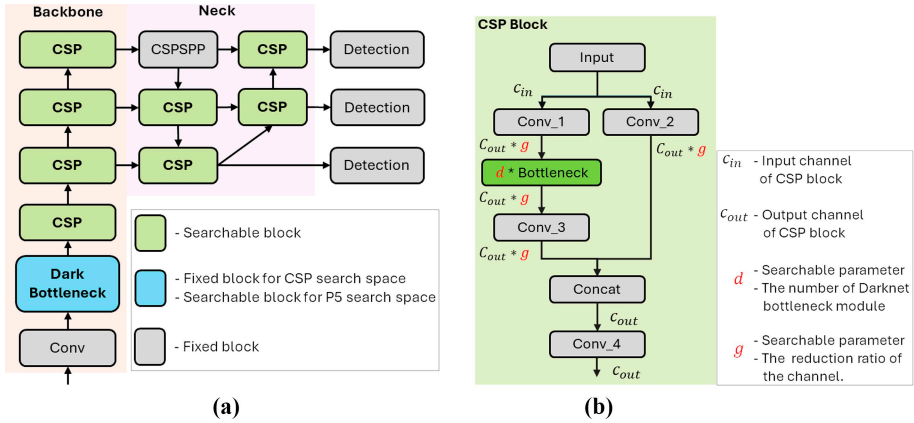


Figure 3. (a) The macro-architecture search space of ScaledYOLOv4. The search process targets CSP blocks in both the backbone and neck. Searchable blocks are shown in green, fixed blocks in gray and the blue block is only searchable in the YOLOv4-P5 space. (b) The CSP block's searchable structure includes two key parameters: d (number of bottleneck modules) and g (channel reduction ratio)

This design is extended to the ScaledYOLOv4-P5 variant by including nine searchable blocks (green and blue in Figure 3(a)). Backbone stages use $d \in 1, 3, 7, 15$, neck stages $d \in 0, 3, 6, 9$, while g remains unchanged, resulting in $12^9 \approx 5.1 \times 10^9$ possible architectures. Figure 3(b) illustrates the CSP block structure, where d and g control both computation and feature representation.

5.1.2 YOLOv7 search space. To capture recent design advances, we define a YOLOv7-based search space using ELAN and E-ELAN blocks (Wang et al., 2022a, 2023). ELAN balances long and short gradient paths to improve feature reuse, while E-ELAN increases network cardinality for enhanced gradient diversity.

The YOLOv7 SuperNet contains eight searchable stages (green blocks in Figure 4(a)), each defined by a channel scaling factor $g \in 0.75, 1.00, 1.25$ and a binary connection pattern specifying which intermediate layers are concatenated to the final output, as shown in Figure 4(b). For full YOLOv7, the five-layer pattern gives $2^5 \times 3 = 96$ options per stage, resulting in $(32 \times 3)^8 \approx 7.2 \times 10^{15}$ total architectures. For YOLOv7-tiny, the four-layer pattern yields $2^4 \times 3 = 48$ options per stage, totaling $(16 \times 3)^8 \approx 2.8 \times 10^{13}$, suitable for edge scenarios with strict latency and memory constraints.

Overall, this unified block-wise search space allows IZeroDNAS to efficiently explore diverse architectures across both ScaledYOLOv4 and YOLOv7, while accounting for feature representation, depth and computational requirements.

5.2 SuperNet warm-up initialization

To improve the stability of zero-cost proxy evaluations, we propose two variants of SuperNet warm-up initialization, which are described in the following subsections.

5.2.1 Conventional initialization. Zero-cost proxies offer a time-efficient method for evaluating a model without the need for training. However, they are highly sensitive to the model's initial state when applied to large models. This sensitivity can lead to significant variations in ranking results, resulting in searched architectures with inconsistent performance. As shown in Table 5, this sensitivity to initialization can produce unstable

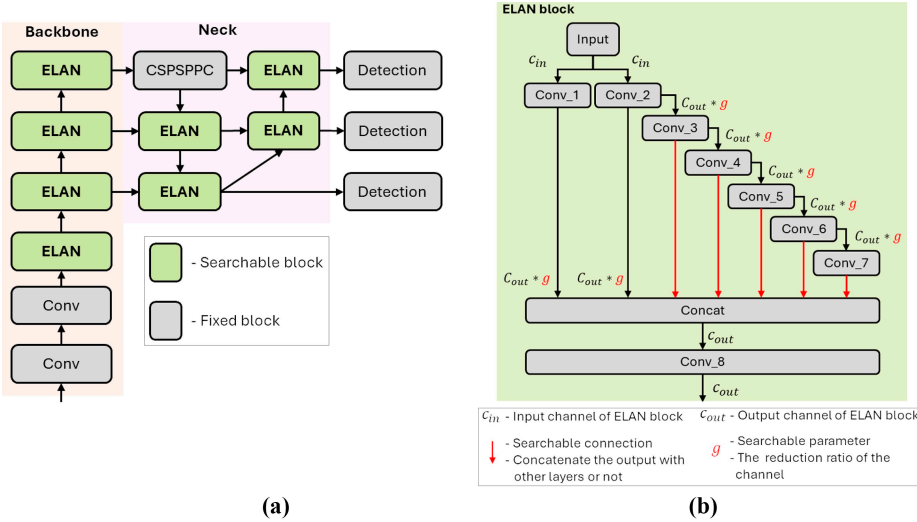


Figure 4. (a) The macro-architecture search space of YOLOv7. The search process targets ELAN blocks in both the backbone and neck. Searchable blocks are shown in green and fixed blocks in gray. (b) The searchable structure of the ELAN block includes two key parameters: g , which controls the channel reduction ratio and red connections indicating whether the output of a convolution layer is concatenated with others in the final aggregation

results. To mitigate this, we introduce a SuperNet warm-up initialization before the search stage. In this phase, the randomly initialized SuperNet weights are trained for a few epochs with fixed importance parameters (θ). This process stabilizes the network's feature representations, improving the reliability of zero-cost proxies and producing more consistent architecture rankings. Importantly, this warm-up requires only a small number of epochs, as it is solely intended to stabilize the ranking rather than optimize full model accuracy. In contrast, traditional SuperNet training typically requires 50–150 epochs to achieve comparable accuracy, making our approach highly time-efficient.

We also observe that the Gumbel softmax temperature t in equation (2) significantly affects ranking robustness. Early differentiable NAS (DNAS) algorithms used a high temperature during burn-in to ensure sufficient gradient signals for all candidate blocks. However, such high temperature effectively transforms the SuperNet into a structure akin to InceptionNet (Szegedy et al., 2015), leading to feature confusion among candidate blocks and inaccurate zero-cost rankings. Through empirical analysis presented in Section 6.3, we find that using a lower temperature substantially enhances the robustness of zero-cost proxies for evaluating model performance.

5.2.2 Depthwise-accelerated initialization. While the conventional warm-up improves ranking stability, it still introduces a nonnegligible computational overhead. To reduce this cost, we propose a depthwise-accelerated warm-up initialization. The core idea is to train a lightweight approximation of the standard convolutional filters using depth-wise separable convolutions. Specifically, instead of training the standard convolutional filters $W_{std} \in \mathbb{R}^{C_{out} \times C_{in} \times k \times k}$ directly, we decompose them into a depth-wise filter $W_{dw} \in \mathbb{R}^{C_{in} \times 1 \times k \times k}$ and a point-wise filter $W_{pw} \in \mathbb{R}^{C_{out} \times C_{in} \times 1 \times 1}$. The standard convolution can then be approximated by:

$$W_{\text{std}}[i, j, u, v] = W_{\text{pw}}[i, j, 0, 0] \cdot W_{\text{dw}}[j, 0, u, v] \quad (11)$$

where i and j index the output and input channels and (u, v) index the spatial kernel positions. This decomposition substantially reduces both the number of parameters and the computational cost during warm-up training, while the merged standard filters are restored during inference to maintain equivalent performance. An overview of this scheme is illustrated in Figure 5.

In summary, the depthwise-accelerated variant provides a computationally efficient alternative to conventional warm-up, preserving the benefits of stabilizing zero-cost rankings while significantly reducing initialization time.

5.3 Hardware-aware

In real-world applications, such as automated vehicles, fast inference on specific edge devices is crucial. While FLOPs are commonly used as a constraint to indicate efficiency, fewer FLOPs do not always translate to faster inference on a given device. Instead, measured inference time is the most accurate performance indicator. Moreover, for models with similar FLOPs, we observed that shallow, wide models tend to have faster inference times than deeper, narrower ones, as shown in Table 15 and discussed in Section 6.8. To ensure optimal performance on specific devices, we incorporate both inference time and model depth into our optimization process through latency-aware loss (L_{latency}) and depth-aware loss (L_{depth}), collectively referred to as hardware-aware loss.

To obtain the real inference time on a specific device, a straightforward approach would be to measure the inference time of each architecture directly on the target device during the search process. However, this is both time-consuming and impractical. Instead, we use a prebuilt look-up table for each candidate block, which contains the real inference times measured on the target device. By leveraging this look-up table, we can estimate the overall inference time of different architectures on the target device based on the candidate blocks' real inference time:

$$\mathbb{E}[\text{Latency}] = \sum_i \sum_j \text{latency}_{i,j} \times p_{i,j}, \quad (12)$$

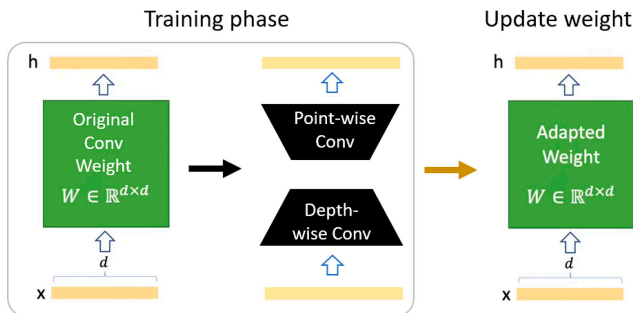


Figure 5. Overview of the proposed depth-wise separable convolution warm-up training scheme and standard convolution inference

$$L_{\text{latency}} = \mathbb{E}[\text{Latency}] \times \beta, \quad (13)$$

where $\text{latency}_{i,j}$ is the time of the j -th candidate block at the i -th stage in the look-up table. β is the hyperparameter to balance the importance of the latency-aware loss L_{latency} .

To find shallower models, which typically offer faster speeds, we integrate depth-aware loss to regulate the model depth:

$$\mathbb{E}[\text{Depth}] = \sum_i \sum_j \text{depth}_{i,j} \times p_{i,j}, \quad (14)$$

$$L_{\text{depth}} = \mathbb{E}[\text{Depth}] \times \omega, \quad (15)$$

where $\text{depth}_{i,j}$ is the depth of the j -th candidate block at the i -th stage. ω is the hyperparameter to balance the importance of the depth-aware loss L_{depth} .

Finally, combined with [equation \(10\)](#), the overall loss function of IZeroDNAS is:

$$L_{\text{overall_device}} = L_{\text{overall}} + L_{\text{latency}} + L_{\text{depth}}. \quad (16)$$

6. Experiment

This section presents the results of our IZeroDNAS process applied to ScaledYOLOv4 and YOLOv7. We use mean average precision (mAP) as the accuracy metric, reporting both mAP (IOU ranging from 50% to 90%) and AP50 (IOU of 50%) as evaluation measures. During the warm-up initialization, we trained the SuperNet weights using a batch size of eight and a resolution of 416x416 for 20 epochs on the Pascal VOC 2007 data set. We used the Stochastic gradient descent optimizer with an initial learning rate of 0.01, gradually reduced to 0.002 using a cosine annealing scheduler. For the search stage, we switched to the Adam optimizer, setting hyperparameters $\lambda = 0.03$ for L_{WOT} , $\phi = 0.005$ for L_{FLOPs} , $\beta = 0.002$ for L_{latency} and $\omega = 0.001$ for L_{depth} in the overall loss function, based on our experience. The Gumbel temperature was gradually decreased from 5 to 0.1 during this phase. For evaluating the model's real-world performance, we used the Qualcomm Robotics RB5 platform as our target device and measured the inference time (ms) for one input image to assess the model's inference speed. Before applying our IZeroDNAS, we prebuilt a look-up table of real inference speeds for the RB5 device, which was used during the search process. To further validate the hardware-aware loss under different platforms, the resulting architectures were also tested on NVIDIA Jetson TX2 and Jetson AGX Orin devices.

6.1 Main results

To evaluate our IZeroDNAS, we assess three key aspects: performance compared to manually designed models, performance compared to NAS-based methods within the same search space and performance compared to NAS-based methods in different search spaces.

Manually designed models: Within the ScaledYOLOv4 search space, we compare models searched by IZeroDNAS with manually designed models ([Wang et al., 2021a](#)) and our base method, ZeroDNAS ([Poliakov et al., 2023](#)), on the Pascal VOC data set. As shown in [Table 1](#), IZeroDNAS not only outperforms the manually designed models in metrics such as mAP and FLOPs – demonstrating its ability to discover high-performance models with fewer parameters but also achieves the fastest inference time on edge devices, which is highly meaningful for real-world applications. Further discussion can be found in 6.2.

Table 1. Comparison under the same Scaled-YOLOv4 (Wang *et al.*, 2021a) search space on the Pascal VOC data set (evaluated on Qualcomm RB5). Models searched by IZeroDNAS achieve competitive accuracy with smaller size than manually-designed models and the base method ZeroDNAS, while maintaining similar FLOPs. Italic values indicate the best results. “speed-up” denotes the accelerated variant, while “normal” refers to the standard version

Model	Target FLOPs	FLOPs	Parameters	mAP (%)	AP50 (%)	Inference time (ms)
YOLOv4-P5 (Wang <i>et al.</i> , 2021a) (baseline)	–	69.9G	70.3M	47.5	71.6	313.1
ZeroDNAS-P5	70.0G	70.8G	75.0M	44.7	68.4	327.2
IZeroDNAS-P5 (normal)	70.0G	68.9G	38.2M	48.2	72.3	278.7
IZeroDNAS-P5 (speed-up)	70.0G	68.5G	43.9M	47.8	72.0	279.7
YOLOv4-CSP (Wang <i>et al.</i> , 2021a) (baseline)	–	52.7G	52.5M	46.9	71.7	256.0
ZeroDNAS-CSP	50.0G	50.9G	58.1M	46.6	71.5	222.2
IZeroDNAS-CSP (normal)	50.0G	48.6G	47.0M	47.1	71.4	206.2
IZeroDNAS-CSP (speed-up)	50.0G	49.5G	59.2m	47.0	71.5	222.6

NAS-based methods in the same search space: We compare our searched models with other NAS methods within the same ScaledYOLOv4 search space on the Pascal VOC data set. As shown in Table 2, IZeroDNAS identifies models with superior performance while maintaining fast search times. Although our search time is slightly longer than ZeroDNAS, we achieve higher accuracy (47.1% mAP) and fewer parameters, demonstrating improved search stability and efficiency. To ensure fairness, we reevaluate the previous NAS methods – Cream (Peng et al., 2020), AttentiveNAS (Wang et al., 2021b) and our earlier ZeroDNAS – under an additional efficiency loss constraint. These methods were originally optimized only for FLOPs, which correlate with computational complexity but do not directly reflect inference latency. Incorporating efficiency loss changes the optimization objective from accuracy-only to a dual objective that simultaneously encourages shallower, faster architectures. Consequently, a slight drop in mAP is expected, as reducing network depth improves latency at the expense of representational capacity. In contrast, our IZeroDNAS variants explicitly optimize with both depth loss and latency loss, yielding a more faithful and effective search objective. This design enables a superior accuracy–latency trade-off, as evidenced in Table 2, where IZeroDNAS consistently outperforms prior NAS approaches in both metrics.

NAS-based methods in different design spaces: To evaluate generalization across search spaces, we compare our models with recent NAS methods designed on different search spaces using the COCO data set, a standard benchmark for object detection. Table 3 shows that IZeroDNAS can discover competitive architectures in under 5 h on a single RTX 2080 GPU, which is three times faster than MAE-DET (Sun et al., 2022), a state-of-the-art zero-cost NAS method. While the warm-up initialization slightly increases search time compared to ZeroDNAS (Poliakov et al., 2023), IZeroDNAS maintains excellent computational efficiency with lower FLOPs and demonstrates strong generalization across data sets.

YOLOv7 search space: Finally, we extend our evaluation to the YOLOv7 search space, comparing our searched models against manually designed models and ZeroDNAS. As summarized in Table 4, the warm-up initialization facilitates highly effective searches, achieving 50.1% mAP for YOLOv7 and 33.8% mAP for YOLOv7-tiny. In terms of AP50, we obtain 74.3% for YOLOv7 and 58.7% for YOLOv7-tiny. Across both the ScaledYOLOv4 and YOLOv7 search spaces, these results underscore the robustness and stability of IZeroDNAS in discovering high-performing models across different YOLO architectures, highlighting its versatility for a wide range of object detection tasks.

6.2 Ablation study

Impact of proposed methods: To validate the key contributions of our algorithm – stability improvement and real-world device applicability – we compare the average mAP and variance across four SuperNet initializations (random seeds), along with the actual inference time on the target device. As shown in Table 5, the previous work, ZeroDNAS (Poliakov et al., 2023), is highly sensitive to different SuperNet initializations, resulting in a large mAP variance of 5.687 and an overall mAP of only 44.7%. In addition, its inference time on the target edge device is slower than that of the manually designed model. In contrast, with our proposed warm-up initialization, IZeroDNAS achieves much more consistent performance, with a significantly reduced mAP variance of 0.0025, enhancing the stability of the search algorithm while also improving overall accuracy to 48% mAP. Furthermore, the hardware-aware loss, which accounts for real-world inference, directs the search toward smaller models that typically achieve faster inference on the target device. By incorporating these two methods, IZeroDNAS consistently discovers the best candidate architecture, achieving

Table 2. Comparison with NAS-based methods on the Pascal VOC data set (evaluated on Qualcomm RB5). The results demonstrate that our search method is faster and produces more accurate and efficient architectures than other NAS algorithms on a specific device, owing to the proposed hardware-aware NAS design. The architectures discovered by IZeroDNAS also perform better or competitively compared with previous NAS-based methods. It is worth noting that the architectures identified by NAS-based methods do not necessarily outperform YOLOv4-CSP (Guo *et al.*, 2020b). This is because all of the NAS-based approaches in the table incorporate hardware-aware loss terms to optimize inference latency, thus making a deliberate trade-off between accuracy and efficiency. In contrast, YOLOv4-CSP (Guo *et al.*, 2020b) focuses solely on maximizing detection accuracy without considering latency constraints

Model	FLOPS	Parameters	mAP(%)	AP50(%)	Search time	Latency (ms)
YOLOv4-CSP (Wang <i>et al.</i> , 2021a) (baseline)	52.7G	52.5M	46.9	71.7	–	256
Cream (Peng <i>et al.</i> , 2020)	52.7G	44.9M	45.5	69.0	108 h	246
AttentiveNAS (Wang <i>et al.</i> , 2021b)	40.7G	40.0M	45.7	69.5	104 h	232
FBNet (Wu <i>et al.</i> , 2019)	–	–	–	–	Out of memory	–
ZeroDNAS-CSP	50.9G	58.1M	46.6	71.5	1.8h	222.2
IZeroDNAS-CSP (normal)	48.6G	47.0M	47.1	71.4	6.2 h	205.2
IZeroDNAS-CSP (speed-up)	50.1G	62.6M	47.0	71.5	4.5h	222.6

Table 3. Comparison with NAS-based methods on the COCO minval data set. Our IZeroDNAS approach shows advantages by requiring less powerful GPUs and shorter search times compared to other methods. In addition, the models obtained through IZeroDNAS achieve competitive performance in terms of accuracy

Model	Size	mAP (%)	AP50 (%)	FLOPs	GPU	Overall memory	Search time
NAS-FCOS (Wang <i>et al.</i> , 2020a)	1088 × 800	39.8	–	189.6G	v100 (32GB)	256GB	672 h
SM-NAS (Yao <i>et al.</i> , 2020) ^a	800 × 600	40.1	58.2	23.8G	v100 (32GB)	> 256GB	4512 h
Hit-detector (Guo <i>et al.</i> , 2020)	800 × 1200	41.4	62.4	272G	v100 (32GB)	256GB	> 100 H
OPANAS (Liang <i>et al.</i> , 2021)	1333 × 800	39.6	–	197Gg	v100 (32GB)	128 GB	96 h
MAE-DET-M (Sun <i>et al.</i> , 2022) ^b	1333 × 800	42.0	–	89.9G	v100 (32GB)	256GB	14.4h
YOLOv4-CSP (Wang <i>et al.</i> , 2021a) (baseline)	416 × 416	41.6	61.2	52.7G	–	–	–
ZeroDNAS-CSP	416 × 416	41.4	60.9	50.9G	2080 (11GB)	6GB	1.8 h
IZeroDNAS-CSP (normal)	416 × 416	42.2	61.7	48.6G	2080 (11GB)	9GB	6.2 h
IZeroDNAS-CSP (speed-up)	416 × 416	41.3	61.1	49.8G	2080 (11GB)	9GB	4.5 h

Note(s): ^aThe origin paper only provide the backbone FLOPs for the SM-NAS-E2 model (not include the FLOPs of FPN, RPN, Cascade detection head). ^bThe origin paper which only provide the backbone FLOPs for the MAE-DET-M (RetinaNet) model

Table 4. Comparison under the same YOLOv7 search space on the Pascal VOC data set. Models searched by IZeroDNAS without hardware-aware loss have higher accuracy in mAP and competitive performance in AP50 than manually-designed models and our based method, ZeroDNAS, with a target model complexity matching the FLOPs of the manual designs

Model	Size	Target FLOPs	FLOPs	Parameters	mAP(%)	AP50(%)
YOLOv7 (baseline)	640 × 640	–	105.0 G	37.3M	49.9	73.9
ZeroDNAS-v7	640 × 640	104.0 G	101.3 G	34.2M	43.4	68.9
IZeroDNAS-v7 (without hardware-aware loss)	640 × 640	104.0 G	104.3 G	40.4M	50.1	74.3
YOLOv7-tiny (baseline)	640 × 640	–	13.3 G	6.0M	33.4	58.9
ZeroDNAS-v7-tiny	640 × 640	12.5 G	11.9 G	5.5M	33.4	58.8
IZeroDNAS-v7-tiny (without hardware-aware loss)	640 × 640	12.5 G	12.8 G	6.4M	33.8	58.7

Table 5. Ablation study of each component of our proposed methods on the Pascal VOC data set (evaluated on Qualcomm RB5). The warm-up column is divided into normal and speed-up submodes

Setting	Warm-up		Hardware-aware loss	Target FLOPs	FLOPs	Parameters	mAP(%)	mAP variance	Inference time (ms)
	Normal	Speed-up							
YOLOv4-P5 (Wang et al., 2021a)	–	–	–	–	69.9 G	70.3 M	47.5	–	313.1
(baseline)									
ZeroDNAS-P5	–	–	–	70 G	70.8 G	75.0 M	44.7	5.687	327.2
IZeroDNAS-P5	–	–	V	70 G	68.7 G	39.5 M	48.0	0.047	273.9
	V	–	–	70 G	70.2 G	70.0 M	48.1	0.0025	302.4
	V	–	V	70 G	68.9 G	38.2 M	48.2	0.0025	278.7
	–	V	–	70 g	70.7 g	61.7 m	47.9	0.6	310.3
	–	V	V	70 g	68.5 g	43.9 m	47.8	0.0256	279.7

both superior accuracy and a compact model with optimized inference speed, making it well-suited for real-world applications.

Influence of each loss: To further investigate the impact of each loss component on the search process, we conducted an ablation study of the IZeroDNAS loss function, as shown in Table 6. In this table, the target constraint is set at 70 G FLOPs, similar to the manually designed YOLOv4-P5 model. As previously mentioned, the zero-cost loss serves as a critical metric for IZeroDNAS in guiding the model search. However, without additional loss constraints, the search process becomes uncontrolled, potentially leading to suboptimal models that waste computational resources. By incorporating FLOPs loss, the search can better align with the target constraint.

Next, we focus on the proposed hardware-aware losses, including depth-aware loss and latency-aware loss. Depth-aware loss helps identify models that outperform manually designed ones while also providing slightly faster inference on the target device. On the other hand, latency-aware loss focuses on real-world inference time by using the prebuilt look-up table. This approach leads to the discovery of smaller models with faster inference times on the target device, though at the expense of model accuracy. Fast inference but low accuracy is not ideal. Therefore, IZeroDNAS combines all these losses to achieve an optimal balance between accuracy and device performance under the target constraint.

6.3 Stability analysis

Warm-up NAS: Our NAS-specific warm-up goes beyond the traditional goal of stabilizing learning rates. It is designed to improve architecture search stability by reducing fluctuations in SuperNet parameter initialization, Gumbel sampling probabilities and zero-cost proxy rankings – factors critical for zero-cost NAS. While this increases the nominal runtime to 6.2 h, it markedly enhances search stability and the reliability of sampled architectures. As shown in Table 7, IZeroDNAS consistently achieves higher mAP across different random seeds compared to ZeroDNAS. Although ZeroDNAS has a lower nominal search time (1.8 h), its instability often requires multiple trials (effectively $1.8 \times N$ trials) to find high-performing architectures. In contrast, the greater stability of IZeroDNAS allows a single search to reliably identify top architectures, highlighting the essential role of NAS-specific warm-up in improving both search efficiency and performance.

WOT stability during warm-up initialization: We analyze the rankings of candidate blocks to assess the stability of WOT throughout the SuperNet warm-up initialization phase. We measure the ranking correlation between any two WOT maps (M_{WOT}) across different random seeds using Kendall's Tau (Kendall, 1938). Kendall's Tau values close to 1 indicate strong positive correlation, while values near -1 reflect strong negative correlation. To reduce the sensitivity of zero-cost rankings to SuperNet initialization, we aim for higher positive correlations between rankings across different random initializations. As shown in Figure 6, the warm-up phase gradually increases this ranking correlation, leading to a more stable WOT metric. In addition, Table 5 demonstrates that the improved stability during warm-up benefits the performance of searched results in IZeroDNAS compared to the base ZeroDNAS method.

Parameter convergence effects: We observe that parameter variance consistently decreases across seeds (Figure 7), suggesting that more stable parameters produce more consistent features, which reduces WOT variance and improves ranking correlation. Unlike prior zero-cost proxies (Abdelfattah et al., 2021; Li et al., 2023; Mellor et al., 2021), which rely solely on random initialization, our NAS-specific warm-up provides a simple yet effective guideline to stabilize zero-cost metrics in larger models. Moreover, IZeroDNAS

Table 6. Ablation study of loss functions in IZeroDNAS on Pascal VOC under the YOLOv4-P5 search space (evaluated on Qualcomm RB5). FLOPs loss improves constraint alignment, latency-aware loss reduces size and speed with some accuracy trade-off and combining all losses balances accuracy with real-world efficiency

Model	FLOPs loss		Depth-aware		Latency-aware		Target FLOPs	FLOPs	Parameters	mAP (%)	AP50 (%)	Inference time (ms)
		loss		loss		loss						
YOLOv4-P5 (Wang <i>et al.</i> , 2021a) (baseline)	–	–	–	–	–	–	–	69.9 G	70.3M	47.5	71.6	313.1
IZeroDNAS-P5	–	–	–	–	–	–	70 G	60.2 G	53.9 M	47.6	72.1	258.5
	V	–	–	–	–	–	70 G	70.2 G	70.0 M	48.1	72.2	302.4
	V	V	–	–	–	–	70 G	69.4M	69.4M	47.9	72.4	293.9
	V	–	–	–	V	V	70 G	68.2 G	43.5 M	46.9	72.2	275.3
	V	V	–	V	V	V	70 G	68.9 G	38.2 M	48.2	72.3	278.7

Table 7. Comparison of mAP (%) and search time for different seeds and settings

Setting	Seed	mAP (%)	Search time
YOLOv4-CSP (baseline)	–	47.01	–
ZeroDNAS (without pretraining)	1st seed	46.60	1.8 h
	2nd seed	47.06	
	3rd seed	47.18	
	4th seed	46.65	
	Average	46.87	
IZeroDNAS (with pretraining)	1st seed	47.14	6.2 h
	2nd seed	47.32	
	3rd seed	47.24	
	4th seed	47.19	
	Average	47.19	

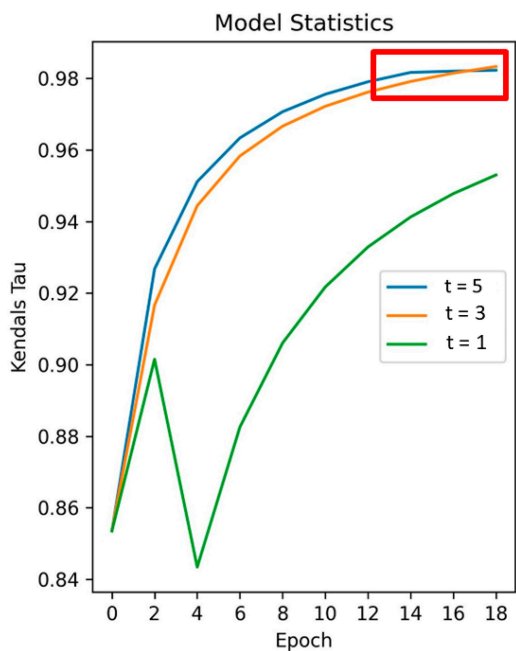


Figure 6. The influence of the Gumbel temperatures on the WOT ranking correlation during the warm-up initialization. Higher temperatures increase the correlation and start converging around 12 epochs, as indicated by the red box

can serve as a modular replacement for zero-cost proxies, facilitating seamless integration with future improvements.

Practical impact on detection performance: To further assess the practical impact of warm-up, we examined how additional warm-up epochs affect actual detection performance. Figure 8 shows that as warm-up progresses, AP50 not only becomes more stable but also improves slightly, converging around 20 epochs. The variance of AP50 across the four

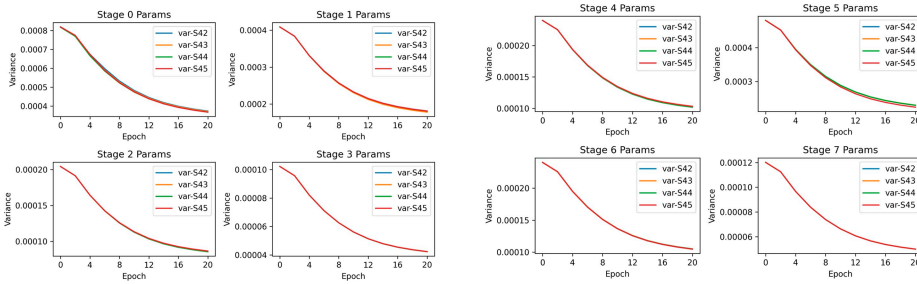


Figure 7. Parameter variance across seeds during SuperNet warm-up. Lower variance indicates more stable parameters and features, contributing to improved WOT stability

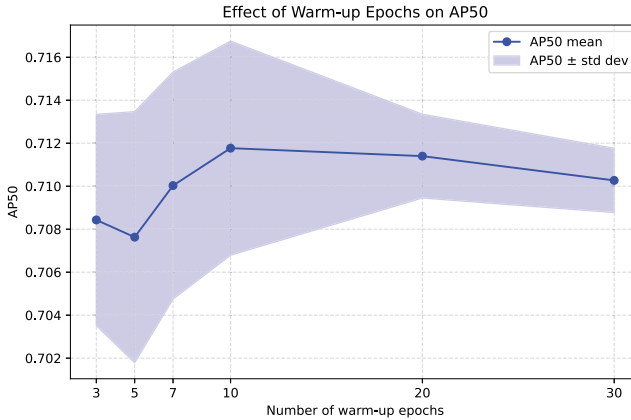


Figure 8. Effect of warm-up epochs on AP50. AP50 stabilizes around 20 epochs, demonstrating that sufficient warm-up improves both WOT stability and detection performance

random seeds decreases as well, indicating that sufficient warm-up enhances both WOT stability and detection reliability.

Consistency across initializations: Beyond correlation analysis, we further investigate the consistency of architecture selection under different SuperNet initializations. As shown in Figure 9, each SuperNet i represents the i -th randomly initialized model. Without warm-up (left), different initializations favor disjoint candidate groups (e.g. SuperNet 3 favors candidates 9–12, while SuperNet 2 prefers 5–8). After warm-up (right), all SuperNet i consistently converge to selecting candidates within the range 5–8, indicating significantly improved search stability. While a formal mathematical proof of this property remains open, our empirical results suggest that warm-up serves as an effective and practical means of stabilizing zero-cost metrics in large-scale SuperNets.

Effect of Gumbel temperature: Besides warm-up effects, the choice of the Gumbel temperature hyperparameter (t in 2) also significantly influences ranking correlation during initialization. As mentioned in 3, temperature t controls the softness of the output probabilities, thereby affecting architecture selection. As seen in Figure 6 and Table 8, both the searched results and the effect of warm-up vary with different temperature settings. A

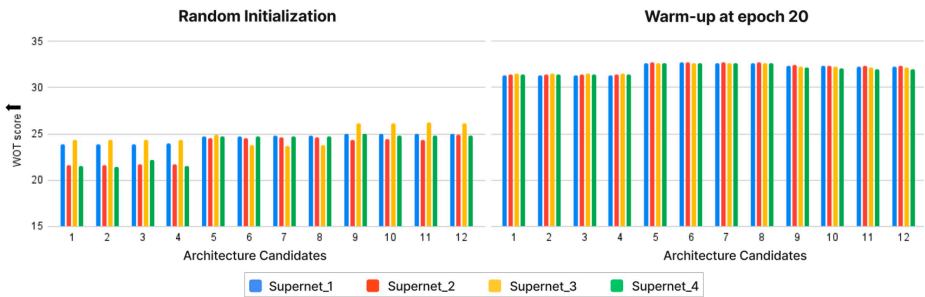


Figure 9. Selection scores (Y-axis) vs model candidates (X-axis) without (left) and with (right) the warm-up step. A higher WoT score suggests a greater likelihood of selection

lower temperature ($t = 1$) leads to overall instability, suggesting difficulties in correctly ranking candidate blocks and requiring longer training to improve ranking correlation. In contrast, higher temperatures ($t = 3$ and $t = 5$) enable faster convergence of ranking correlation by providing stronger supervisory signals for candidate blocks. However, as shown in Table 8, an excessively high temperature ($t = 5$) may result in inaccurate ranking and degraded performance. Therefore, in our experiments, setting $t = 3$ during the SuperNet warm-up initialization yields the most stable and optimal results.

6.4 Zero-cost proxy selection

In this section, we explain why NASWOT was chosen as our zero-cost proxy for the experiments. Each zero-cost proxy is grounded in a different underlying principle: SNIP (Lee et al., 2019) focuses on parameter sensitivity, NASWOT (Mellor et al., 2021) emphasizes feature similarity and ZiCo (Li et al., 2023) considers the influence of training gradient properties on convergence. These distinct design philosophies lead to varying estimation capabilities across different task models. To identify the most suitable proxy for YOLO-series object detection, we applied several zero-cost proxies in combination with an evolutionary algorithm, following the approaches outlined in their respective papers.

As shown in Table 9, NASWOT achieves overall competitive performance compared to manually designed models and outperforms the other zero-cost proxies. Based on these results, we selected NASWOT as the zero-cost proxy for our experiments. Nevertheless, it is important to note that while NASWOT can identify models with competitive accuracy, it tends to produce larger models with approximately 58 M parameters, which is less suitable for edge device deployment. In contrast, our IZeroDNAS approach not only discovers smaller models with only 47 M parameters but also achieves a higher mAP of 47.1%, surpassing the baseline.

Table 8. The performance of searched results with different Gumbel temperatures during the warm-up initialization. Each temperature is tested four times in the ScaledYOLOv4 search space. Italic values are the best results

Temperature(t)	Worse model mAP (%)	Average model mAP (%)
$t = 5$	46.96	47.14
$t = 3$	47.09	47.19
$t = 1$	46.86	47.04

Table 9. Comparison with zero-cost proxies on the pascal VOC data set. All zero-cost proxies are integrated with an evolutionary algorithm to search within the YOLOv4-CSP search space. Among these proxies, NASWOT proves to be the most effective in providing a more accurate estimate of object detection models compared to other zero-cost proxies. By combining NASWOT with DNAS, IZeroDNAS is able to search for high-performing, small size models

Model	FLOPS	Parameters	mAP (%)	AP50 (%)
YOLOv4-CSP (Wang <i>et al.</i> , 2021a) (baseline)	52.7G	52.5M	46.9	71.7
SNIP (Lee <i>et al.</i> , 2019)	48.1G	71.2M	44.9	70.5
Synflow (Tanaka <i>et al.</i> , 2020)	39.7G	43.2M	46.3	70.8
ZiCo (Li <i>et al.</i> , 2023)	49.5G	62.1M	47.0	70.1
NASWOT (Mellor <i>et al.</i> , 2021)	49.4G	57.9m	47.0	71.5
ZeroDNAS-CSP	50.9G	58.1M	46.6	71.5
IZeroDNAS-CSP (normal)	48.6G	47.0M	47.1	71.4
IZeroDNAS-CSP (speed-up)	49.5G	59.2M	47.0	71.5

6.5 Hardware-aware loss generalization

To evaluate the generalizability of our hardware-aware loss, we extended experiments to two embedded platforms, Jetson TX2 and Jetson AGX Orin, with results summarized in Tables 10 and 11.

Across both devices, the IZeroDNAS variants consistently achieve a favorable balance between detection accuracy and inference efficiency. In particular, IZeroDNAS-P5 and IZeroDNAS-CSP exhibit substantially lower latency compared to the ScaledYOLOv4 baselines and the original ZeroDNAS models. On the Jetson TX2, IZeroDNAS reduces inference time while maintaining or slightly improving mAP, whereas on the Jetson AGX Orin, it attains the lowest latency among all evaluated architectures. These findings demonstrate that the proposed hardware-aware loss effectively generalizes across heterogeneous platforms, enabling architectures that are simultaneously high-performing and low-latency, which is critical for real-time deployment scenarios.

6.6 Shallow and wide architectures discovered by IZeroDNAS

To ensure a fair analysis, we analyze the results obtained from IZeroDNAS under complex constraints. We observe that IZeroDNAS tends to favor shallower models compared to manually designed ones, achieving better accuracy. The following discussion will explore IZeroDNAS tendencies within the search spaces of ScaledYOLOv4 (Wang *et al.*, 2021a) and YOLOv7 (Wang *et al.*, 2023).

Table 10. Performance comparison of various models on Jetson TX2. The results show that IZeroDNAS achieves higher accuracy and lower latency while maintaining similar computational cost

Model	Target FLOPS	FLOPS	Parameters (M)	mAP(%)	AP50(%)	Latency (ms)
YOLO-P5	–	69.9	70.3	47.5	71.6	225.762
ZeroDNAS-P5	70	70.0	79.339	42.98	66.79	212.428
IZeroDNAS-P5	70	68.8	93.545	47.78	71.64	203.083
YOLOv4-CSP	–	52.7	52.5	46.9	71.7	148.679
ZeroDNAS-CSP	50	49.6	54.625	44.05	69.7	138.364
IZeroDNAS-CSP	50	48.8	41.304	45.8	71.22	134.076

Table 11. Performance comparison of various models on Jetson AGX Orin. IZeroDNAS demonstrates superior accuracy and the lowest latency among all evaluated architectures

Model	Target FLOPS	FLOPS	Parameters (M)	mAP(%)	AP50(%)	Latency (ms)
YOLO-P5	–	69.9	70.3	47.5	71.6	20.036
ZeroDNAS-P5	70	70.0	79.339	42.98	66.79	15.187
IZeroDNAS-P5	70	68.7	66.750	47.74	71.55	14.754
YOLOv4-CSP	–	52.7	52.5	46.9	71.7	17.366
ZeroDNAS-CSP-416	50	49.6	54.625	44.05	69.7	12.796
IZeroDNAS-CSP	50	49.6	54.625	46.61	71.32	11.994

ScaledYOLOv4 search space: To achieve models of varying complexity, manually designed ScaledYOLOv4 models (Wang et al., 2021a) use a scaling approach that linearly increases the network’s depth and channel count. However, we have observed that IZeroDNAS tends to find more accurate models with shallow and wider architectures, characterized by lower depth and more channels. As shown in Table 12, manually designed YOLOv4-CSP has 30 Darknet bottleneck modules, which indicates the model depth within our searchable blocks in total, whereas our IZeroDNAS-CSP has just 20 Darknet bottleneck modules. Similarly, YOLOv4-P5 has 52 bottleneck modules, which is substantially deeper than the 27 modules in our IZeroDNAS-P5.

ScaledYOLOv4 search space: As shown in Table 13, the v7-IZeroDNAS model without hardware-aware loss tends to select connection paths from shallower layers, as indicated by the lower values of connection choices compared to the manually designed model. This suggests that v7-IZeroDNAS reduces the overall model depth, consistent with the patterns observed in the ScaledYOLOv4 search space. In contrast, the v7-IZeroDNAS-tiny model without hardware-aware loss maintains a depth similar to YOLOv7-tiny, likely because the tiny version is already underfitting and does not benefit from further depth reduction. We also analyze the shortest and longest gradient paths. In our search space, these paths are calculated by summing the smallest and largest connection choices, respectively. While Wang et al. (2022a) highlighted the importance of tuning both shortest and longest gradient paths, they did not provide precise guidelines for models of different scales. In our experiments, the searched models tend to increase the length of the shortest gradient path compared to manually designed models, which may provide useful insights for gradient path selection.

6.7 Comparison with other YOLO-based NAS methods

To provide a more comprehensive evaluation, we include a comparison with other YOLO-based NAS methods, such as YOLO-NAS. Because the details of YOLO-NAS’s search space, strategy and implementation are not publicly available, we cannot exactly reproduce its search process for a fully fair comparison. Attempts to apply our IZeroDNAS framework to the YOLO-NAS search space were not feasible due to the lack of official documentation or detailed description of the search space.

Instead, we use the officially released YOLO-NAS models (YOLO-NAS-S, YOLO-NAS-M and YOLO-NAS-L) as a reference and fine-tune them on the VOC data set. The results are reported in Table 14 alongside our IZeroDNAS architectures. Although the original search spaces differ, making the comparison not strictly fair, these results serve as a practical performance reference. As shown, IZeroDNAS achieves slightly lower accuracy than the fine-tuned YOLO-NAS models, but it consistently provides substantially faster

Table 12. Network architecture of YOLOv4-Series model and our searched model on the Pascal VOC data set. Each YOLOv4-CSP has eight stages of the searched block and YOLOv4-P5 has nine stages of the searched block. Both “IZeroDNAS-CSP” and “IZeroDNAS-P5” are shallower and wider than manually designed models. With smaller model size, IZeroDNAS can search results with higher or competitive accuracy

Model	FLOPs	Parameters	mAP (%)	AP50 (%)	Network architecture	
					Channel Ratio g	Depth d
YOLOv4-P5 (Wang <i>et al.</i> , 2021a) (baseline)	69.9G	70.3M	47.5	71.6	0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5	1,2,15,15,7,3,3,3,3
IZeroDNAS-P5	68.6G	40.0M	48.3	72.4	0.5,0.75,0.5,0.5,0.5,0.75,0.75,0.5,0.25	7,7,7,3,1,3,6,3,0
YOLOv4-CSP (Wang <i>et al.</i> , 2021a) (baseline)	52.7G	52.5M	46.9	71.7	0.5,0.5,0.5,0.5,0.5,0.5,0.5	2,8,8,4,2,2,2,2
IZeroDNAS-CSP	48.6G	47.0M	47.1	71.4	0.75,0.5,0.5,0.75,0.5,0.75,0.25,0.5	4,2,2,2,4,2,4,0

Table 13. Network architecture of YOLOv7-Series model and our searched model on the pascal VOC data set. Both YOLOv7 and YOLOv7-tiny have eight stages of searched block. “v7-IZeroDNAS” is shallower than manually designed models with higher accuracy on the PascalVOC data set

Model	FLOPs	Parameters	mAP (%)	Channel ration g	Network architecture		Gradient path	
					Connection choice	Longest	Shortest	
YOLOv7 (Wang et al., 2023) (baseline)	105.0G	37.3M	49.92	1.00, 1.00, 1.00, 1.00 1.00, 1.00, 1.00, 1.00	[2,4],[2,4],[2,4],[2,4] [1,2,3,4],[1,2,3,4], [1,2,3,4],[1,2,3,4]	32	12	
IZeroDNAS-v7 (without hardware-aware loss)	104.3G	40.4M	50.10	0.75, 0.50, 0.50, 0.50 0.50, 0.50, 0.75, 0.75	[3],[3,4],[2,3],[2,3] [2,3],[1,2,3],[2,3,4]	29	15	
YOLOv7-tiny (Wang et al., 2023) (baseline)	13.3G	6.0M	33.45	1.00, 1.00, 1.00, 1.00 1.00, 1.00, 1.00, 1.00	[1,2],[1,2],[1,2],[1,2] [1,2],[1,2],[1,2],[1,2]	16	8	
IZeroDNAS-v7-tiny (without hardware-aware loss)	12.8G	6.4M	33.83	0.75, 0.75, 1.00, 1.00 1.25, 1.25, 1.25, 1.25	[1,2],[1],[1,2],[1,2] [1,2],[2,3],[1,2]	16	9	

Table 14. Comparison of IZeroDNAS variants with fine-tuned YOLO-NAS models on the VOC data set

Method	mAP 0.5(%)	mAP(%)	Latency (ms)
IZeroDNAS-CSP	71.4	47.1	10.6
IZeroDNAS-P5	72.3	48.2	15.3
IZeroDNAS-v7	74.3	50.1	18.9
YOLO-NAS-S	78.97	52.83	22.45
YOLO-NAS-M	77.95	53.49	24.93
YOLO-NAS-L	<i>79.08</i>	<i>54.82</i>	25.81

inference speeds. This demonstrates the effectiveness of our hardware-aware design in achieving a favorable trade-off between accuracy and efficiency, making IZeroDNAS particularly suitable for deployment scenarios with strict latency requirements.

6.8 Depth-aware efficiency in architecture search

As discussed in Section 6.6, IZeroDNAS often favors shallower architectures to achieve higher accuracy. Beyond accuracy, inference time is another crucial factor for edge deployment, where computational resources are limited. Thus, identifying architectures that can deliver faster inference under the same FLOPs constraint becomes an important objective. To illustrate this, we compare ResNet152 (He *et al.*, 2015) and Wide ResNet50_2 (Zagoruyko and Komodakis, 2016), two architectures with similar FLOPs but markedly different depths. Table 15 reports their inference times on both an Nvidia 3090ti GPU and the Qualcomm Robotics RB5 platform, our experimental target device. Despite their comparable FLOPs, ResNet152, with its 152 layers, exhibits substantially slower inference than Wide ResNet50_2, which has only 50 layers but a wider design. The results highlight that model depth plays a critical role in practical efficiency, with shallower-and-wider networks offering significant speed advantages without compromising FLOPs. Motivated by this observation, we incorporate a *depth-aware loss* into our hardware-aware loss formulation. This additional term guides the search toward architectures that not only achieve accuracy improvements, as seen in Section 6.6, but also exhibit faster inference on real devices. As shown in Table 15, this strategy is consistent with empirical trends, enabling IZeroDNAS to better balance accuracy, efficiency and real-world deployment performance.

7. Conclusion

We introduce IZeroDNAS to improve the search stability of differentiable NAS with zero-cost proxies and to optimize searched models for edge devices in terms of size and inference

Table 15. Inference time of ResNet152 and wide ResNet50_2 on the NVIDIA 3090ti and qualcomm RB5. With similar FLOPs, wide ResNet50_2 is shallower than ResNet152 but has a faster inference time. *Italic values are the best results*

Model	FLOPs	Parameters	Inference time	
			3090ti (ms)	RB5 (ms)
ResNet152 (He <i>et al.</i> , 2015) (Deeper model)	11.5 G	60.19M	0.022	138.3
Wide ResNet50_2 (Zagoruyko and Komodakis, 2016) (Shallower model)	11.4 G	68.88M	<i>0.013</i>	<i>113.8</i>

time. To mitigate the sensitivity of initialization, we propose a SuperNet warm-up scheme that enhances ranking correlation and stability, making the search more robust. To further accelerate this process, we present a depthwise-accelerated initialization, which reduces training cost by leveraging depthwise separable convolutions during warm-up while maintaining equivalent performance at inference. In addition, hardware-aware loss guides the search toward compact, fast and accurate models. Experiments demonstrate that IZeroDNAS surpasses state-of-the-art NAS methods and manually designed models in both accuracy and real-device efficiency. Future work will explore broader zero-cost proxies for stability and more accurate predictors of device performance.

Authors' contribution

Van-Tin Luu and Tao-Ting Chen contributed equally to this work.

References

- Abdelfattah, M.S., Mehrotra, A., Dudziak, L. and Lane, N.D. (2021), "Zero-cost proxies for lightweight NAS", ICLR, available at: <https://openreview.net/forum?id=0cmMMY8J5q>
- Ghiasi, G., Lin, T.-Y. and Le, Q.V. (2019), "NAS-FPN: learning scalable feature pyramid architecture for object detection", *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7029-7038, doi: [10.1109/CVPR.2019.00720](https://doi.org/10.1109/CVPR.2019.00720).
- Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., and Sun, J. (2020a), "Single path one-shot neural architecture search with uniform sampling", in Vedaldi, A., Bischof, H., Brox, T. and Frahm, J. (Eds), *ECCV, Lecture Notes in Computer Science*, Springer, Vol. 12361, pp. 544-560, doi: [10.1007/978-3-030-58517-4_32](https://doi.org/10.1007/978-3-030-58517-4_32).
- Guo, J., Han, K., Wang, Y., Zhang, C., Yang, Z., Wu, H., Chen, X. and Xu, C. (2020b), "Hit-detector: hierarchical trinity architecture search for object detection", *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11402-11411.
- He, K., Zhang, X., Ren, S. and Sun, J. (2015), "Deep residual learning for image recognition", arXiv: 1512.03385 [cs.CV], available at: <https://arxiv.org/abs/1512.03385>
- Jang, E., Gu, S. and Poole, B. (2017), "Categorical reparameterization with Gumbel-Softmax", ICLR, available at: <https://openreview.net/forum?id=kE3y85ee>
- Kendall, M.G. (1938), "A new measure of rank correlation", *Biometrika*, issn: 00063444, Vol. 30 Nos 1-2, pp. 81-93, available at: www.jstor.org/stable/2332226 (accessed 29 January 2024).
- Lee, N., Ajanthan, T. and Torr, P. (2019), "Snip: single-shot network pruning based on connection sensitivity", *International Conference on Learning Representations*, available at: <https://openreview.net/forum?id=B1VZqjAcYX>
- Li, G., Yang, Y., Bhardwaj, K. and Marculescu, R. (2023), "ZiCo: Zeroshot NAS via inverse coefficient of variation on gradients", ICLR, available at: <https://openreview.net/forum?id=rwo-ls5GqGn>
- Liang, T., Wang, Y., Tang, Z., Hu, G. and Ling, H. (2021), "OPANAS: one-shot path aggregation network architecture search for object detection", *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10190-10198, doi: [10.1109/CVPR46437.2021.01006](https://doi.org/10.1109/CVPR46437.2021.01006).
- Liu, H., Simonyan, K. and Yang, Y. (2019), "DARTS: differentiable architecture search", ICLR, available at: <https://openreview.net/forum?id=1eYHoC5FX>
- Mellor, J., Turner, J., Storkey, A., and Crowley, E.J. (2021), "Neural architecture search without training", in Meila, M. and Zhang, T., (Eds), *ICML, Proceedings of Machine Learning Research*, PMLR, Vol. 139, pp. 7588-7598, available at: <https://proceedings.mlr.press/v139/mellor21a.html>
- Peng, H., Du, H., Yu, H., Li, Q., Liao, J. and Fu, J. (2020), "Cream of the crop: distilling prioritized paths for one-shot neural architecture search", *Advances in Neural Information Processing Systems*, Vol. 33.

- Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. (2018), "Efficient neural architecture search via parameters sharing", in Dy, J. and Krause, A. (Eds), *ICML, Proceedings of Machine Learning Research*, PMLR, Vol. 80, pp. 4095-4104, available at: <https://proceedings.mlr.press/v80/pham18a.html>
- Poliakov, E., Hung, W.-J. and Huang, C.-C. (2023), "Efficient constraint-aware neural architecture search for object detection", *2023 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 733-740, doi: [10.1109/APSIPAASC58517.2023.10317340](https://doi.org/10.1109/APSIPAASC58517.2023.10317340).
- Real, E., Aggarwal, A., Huang, Y. and Le, Q.V. (2019), "Regularized evolution for image classifier architecture search", *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33 No. 1, pp. 4780-4789, available at: <https://ojs.aaai.org/index.php/AAAI/article/view/4405>, doi: [10.1609/aaai.v33i01.33014780](https://doi.org/10.1609/aaai.v33i01.33014780).
- Ru, B., Lyle, C., Schut, L., Fil, M., der Wilk, V M., and Gal, Y. (2021), "Speedy performance estimation for neural architecture search", in Beygelzimer, A., Dauphin, Y., Liang, P. and Vaughan, J. W. (Eds), *Advances in Neural Information Processing Systems*, available at: <https://openreview.net/forum?id=8V2hZW0d2aS>
- Sun, Z., Lin, M., Sun, X., Tan, Z., Li, H. and Jin, R. (2022), "MAE-DET: revisiting maximum entropy principle in Zero-Shot NAS for efficient object detection", *ICML*, pp. 20810-20826, available at: <https://proceedings.mlr.press/v162/sun22c.html>
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. (2015), "Going deeper with convolutions", *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-9, doi: [10.1109/CVPR.2015.7298594](https://doi.org/10.1109/CVPR.2015.7298594).
- Tanaka, H., Kunin, D., Yamins, D.L., and Ganguli, S. (2020), "Pruning neural networks without any data by iteratively conserving synaptic flow", in Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. and Lin, H. (Eds), *Advances in Neural Information Processing Systems*, Curran Associates, Inc., Vol. 33, pp. 6377-6389, available at: https://proceedings.neurips.cc/paper_files/paper/2020/file/46a4378f835dc8040c8057beb6a2da52-Paper.pdf
- Wang, C.-Y., Bochkovskiy, A. and Liao, H.-Y.M. (2021a), "Scaled-YOLOv4: scaling cross stage partial network", *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13024-13033, doi: [10.1109/CVPR46437.2021.01283](https://doi.org/10.1109/CVPR46437.2021.01283).
- Wang, C.-Y., Bochkovskiy, A. and Liao, H.-Y.M. (2023), "YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors", *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7464-7475, doi: [10.1109/CVPR52729.2023.00721](https://doi.org/10.1109/CVPR52729.2023.00721).
- Wang, C., Liao, H.M. and Yeh, I. (2022a), "Designing network design strategies through gradient path analysis", arXiv:2211.04800 [cs.CV].
- Wang, D., Li, M., Gong, C. and Chandra, V. (2021b), "AttentiveNAS: improving neural architecture search via attentive sampling", *CVPR*, pp. 6418-6427.
- Wang, X., Lin, J., Zhao, J., Yang, X. and Yan, J. (2022b), "EAutoDet: efficient architecture search for object detection", *ECCV*, doi: [10.1007/978-3-031-20044-1_38](https://doi.org/10.1007/978-3-031-20044-1_38).
- Wang, N., Gao, Y., Chen, H., Wang, P., Tian, Z., Shen, C. and Zhang, Y. (2020a), "NAS-FCOS: fast neural architecture search for object detection", *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11940-11948, doi: [10.1109/CVPR42600.2020.01196](https://doi.org/10.1109/CVPR42600.2020.01196).
- Wang, C.-Y., Mark, H.-Y.L., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W. and Yeh, I.-H. (2020b), "CSPNet: a new backbone that can enhance learning capability of CNN", *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1571-1580, doi: [10.1109/CVPRW50498.2020.00203](https://doi.org/10.1109/CVPRW50498.2020.00203).
- Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y. and Keutzer, K. (2019), "FBNet: hardware-aware efficient ConvNet design via differentiable neural architecture search", *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10726-10734, doi: [10.1109/CVPR.2019.01099](https://doi.org/10.1109/CVPR.2019.01099).

- Xu, A., Yao, A., Li, A., Liang, A. and Zhang, A. (2019), "Auto-FPN: automatic network architecture adaptation for object detection beyond classification", *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6648-6657, doi: [10.1109/ICCV.2019.00675](https://doi.org/10.1109/ICCV.2019.00675).
- Yao, L., Xu, H., Zhang, W., Liang, X. and Li, Z. (2020), "SM-NAS: structural-to-modular neural architecture search for object detection", *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34 No. April, pp. 12661-12668, doi: [10.1609/aaai.v34i07.6958](https://doi.org/10.1609/aaai.v34i07.6958).
- Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P., Tan, M., Huang, T., Song, X., Pang, R. and Le, Q. (2020), "BigNAS: Scaling up neural architecture search with big single-stage models", in Vedaldi A., Bischof H., Brox T. and Frahm J. (Eds), *ECCV, Lecture Notes in Computer Science*, Springer, Vol. 12352, pp. 702-717, doi: [10.1007/978-3-030-58571-6_41](https://doi.org/10.1007/978-3-030-58571-6_41).
- Zagoruyko, S. and Komodakis, N. (2016), "Wide residual networks", *BMVC*.
- Zoph, B. and Le, Q.V. (2016), "Neural architecture search with reinforcement learning", *CoRR*, abs/1611.01578, arXiv:1611.01578, available at: <http://arxiv.org/abs/1611.01578>

Corresponding author

Ching-Chun Huang can be contacted at: chingchun@nycu.edu.tw