EDUSCHOOL INTERNATIONAL JOURNAL OF DATA SCIENCE AND MACHINE LEARNING (EIJDSML)

(Open Access, Double-Blind Peer Reviewed Journal)

ISSN Online: ISSN Print


# Neural Architecture Search via Differentiable Proxies in AUTOML

Rejina P V

Assistant professor, Co-Operative Arts And Science College, Madayi, Pazhayangadi, Kannur, India

**Article information**

Received: 9th January 2026
Received in revised form: 12th February 2026
Accepted:14th March 2026
Available online: 18th April 2026

Volume: 1
Issue: 1
DOI: https://doi.org/10.5281/zenodo.19629689

**Abstract**

Neural Architecture Search (NAS) automates the design of deep neural network architectures, potentially surpassing human-crafted designs across diverse tasks. However, the computational cost of evaluating candidate architectures has historically limited NAS scalability, with early methods requiring thousands of GPU-hours per search. Differentiable NAS methods address this by relaxing the discrete architecture selection into a continuous optimization problem amenable to gradient-based optimization. This paper provides a comprehensive survey of differentiable NAS approaches, with particular emphasis on proxy-based methods that further reduce computational overhead. We trace the evolution from DARTS through its failure modes and subsequent corrections, examine zero-cost proxies that estimate architecture quality without training, analyze one-shot and supernet-based approaches, and discuss training-free NAS methods grounded in neural tangent kernel theory. We present extensive empirical comparisons across standard NAS benchmarks (NAS-Bench-101, NAS-Bench-201, DARTS search space) and discuss the proxy gap between search and evaluation performance. Our analysis reveals that the field has progressed from methods requiring thousands of GPU-hours to approaches achieving competitive results in seconds, fundamentally transforming the accessibility of automated architecture design.

**Keywords**:- Neural Architecture Search, Differentiable Architecture Search, Automl, Zero-Cost Proxies, Supernet, Weight Sharing, DARTS.

## I. INTRODUCTION

The design of neural network architectures has been a driving force behind deep learning's success. Landmark architectures including AlexNet [1], ResNet [2], and the Transformer [3] were the products of extensive human expertise and experimentation. However, the expanding design space of modern architectures—encompassing layer types, connectivity patterns, normalization strategies, activation functions, and attention mechanisms—makes manual exploration increasingly impractical.

Neural Architecture Search (NAS) addresses this challenge by automating architecture design through algorithmic optimization over a defined search space [4]. The NAS problem can be formulated as a bilevel optimization: the outer objective selects the architecture α* that maximizes validation performance, while the inner objective trains the network weights w* for each candidate architecture: $\alpha^* = \text{argmin}_\alpha L_{val}(w*(\alpha), \alpha)$, subject to $w^*(\alpha) = \text{argmin}_w L_{train}(w, \alpha)$.

Early NAS methods based on reinforcement learning [5] and evolutionary algorithms [6] evaluated each candidate architecture by training it from scratch—a process requiring hundreds of GPU-hours per evaluation. With search spaces containing 10^18 or more architectures, these approaches demanded massive computational resources (e.g., 48,000 GPU-hours for the original NAS [5]), limiting accessibility to well-resourced research laboratories.

Differentiable NAS methods, pioneered by DARTS [7], transformed the field by enabling gradient-based architecture optimization, reducing search cost to a single GPU-day. Subsequent work has further reduced costs through proxy-based evaluation methods, zero-cost metrics, and training-free approaches. This paper surveys these advances, organized as follows: Section II reviews foundational NAS concepts; Section III examines DARTS and its variants; Section IV covers supernet and weight-sharing approaches; Section V discusses zero-cost proxies; Section VI analyzes the proxy gap; Section VII presents NAS benchmarks; Section VIII discusses applications; and Section IX identifies open challenges.

Table 1. Evolution of NAS Computational Cost

| NAS Paradigm | Representative Work | Search Cost (GPU-hours) | Year |
|---|---|---|---|
| RL-based | NAS [5] | 48,000 | 2017 |
| Evolutionary | AmoebaNet [6] | 3,150 | 2019 |
| Differentiable | DARTS [7] | 24 | 2019 |
| One-shot | OFA [8] | 40 | 2020 |
| Zero-cost proxy | ZeroCost-NAS [9] | ~0.01 | 2021 |
| Training-free | TE-NAS [10] | ~0.05 | 2021 |

## II. FOUNDATIONS OF NEURAL ARCHITECTURE SEARCH

### A. Search Space Design

The search space defines the set of possible architectures and profoundly influences both the quality of discovered architectures and the efficiency of the search process. Cell-based search spaces [5], [7] define a computational cell as a directed acyclic graph (DAG) where nodes represent latent features and edges represent candidate operations (e.g., 3×3 convolution, 5×5 separable convolution, max pooling, skip connection, zero). The full architecture is constructed by stacking copies of the discovered cell, typically with separate normal cells (preserving spatial resolution) and reduction cells (downsampling).

The DARTS search space [7] consists of 7 nodes per cell, with each edge selecting from 8 candidate operations, yielding approximately 10^18 possible architectures. While cell-based search spaces impose strong structural priors that limit diversity, they enable transferability—architectures discovered on CIFAR-10 can be scaled and evaluated on ImageNet by increasing the number of stacked cells and channels [5].

### B. Search Strategy Taxonomy

NAS search strategies can be categorized into three main families [4]:

- Reinforcement learning methods, where a controller RNN generates architecture descriptions and is trained using the validation accuracy as reward [5]
- Evolutionary methods, which maintain a population of architectures and evolve them through mutation and selection [6]
- Gradient-based methods, which optimize continuous architecture parameters alongside network weights [7]. This paper focuses primarily on the gradient-based family, which has demonstrated the best efficiency-performance trade-offs.

### C. Performance Estimation Strategies

The computational bottleneck of NAS lies in estimating the performance of candidate architectures. Full training evaluation provides accurate estimates but is prohibitively expensive. Proxy-based estimation strategies trade accuracy for efficiency through several mechanisms:

- Reduced training (fewer epochs, smaller datasets)

- Weight sharing across architectures via a shared supernet
- Learning curve extrapolation
- Zero-cost metrics computed from the untrained network [11]. The effectiveness of a proxy is measured by its rank correlation with full training performance, typically using kendall's τ or spearman's ρ [9].

## III. DARTS AND ITS EVOLUTION

### A. Continuous Relaxation

DARTS (Differentiable Architecture Search) [7] transforms the discrete architecture selection problem into a continuous optimization by introducing mixture weights over candidate operations. For each edge (i,j) in the cell, the output is computed as a weighted sum:

$$f_{ij}(x)o = \sum_o \left[\frac{\exp(\alpha_o^{ij})}{\sum_{o'} \exp(\alpha_{o'}^{ij})}\right] \cdot o(x) \quad (1)$$

Where $\alpha_o^{ij}$ are architecture parameters and o ranges over candidate operations? This softmax relaxation enables gradient-based optimization of α jointly with the network weights w.

The bi-level optimization is approximated through alternating gradient steps: one step of weight optimization on training data, followed by one step of architecture parameter optimization on validation data. After the search phase, the final discrete architecture is derived by selecting the operation with the highest α at each edge and retaining the top-k edges per node (typically k=2) [7].

### B. DARTS Failure Modes

Despite its elegance, DARTS exhibits several well-documented failure modes. The most prominent is the skip connection collapse [12]: as training progresses, the architecture parameters increasingly favor skip connections over parameterized operations, ultimately producing architectures dominated by skip connections with degraded performance. Zela et al. [13] showed that this failure is related to the sharpness of the loss landscape—architectures that generalize well correspond to flat minima of the architecture loss, while collapsed architectures correspond to sharp minima.

Additional failure modes include:

- The discretization gap, where the continuous relaxation produces different rankings than the discrete architecture [14];
- The depth gap, where architectures discovered in shallow search networks transfer poorly to deeper evaluation networks [15]; and
- Unfair competition between operations with different numbers of parameters, where skip connections and pooling operations are favored simply because they are easier to optimize in the weight-sharing setting [16].

### C. Robust DARTS Variants

A rich body of work addresses DARTS failure modes. FairDARTS [16] replaces the softmax relaxation with independent sigmoid activations for each operation, allowing multiple operations per edge and eliminating the unfair competition caused by the softmax's zero-sum constraint. This simple modification dramatically reduces skip connection collapse and improves search stability.

P-DARTS (Progressive DARTS) [15] addresses the depth gap by progressively increasing the network depth during search, gradually closing the gap between search and evaluation depths. DARTS- [12] adds an auxiliary skip connection alongside the search cell, decoupling the architecture's need for skip connections for gradient flow from its preference for skip connections as computational operations.

GAEA [17] improves the bilevel optimization by replacing the standard gradient descent on architecture parameters with geometry-aware exponentiated gradient updates that respect the simplex constraint of the softmax distribution. SDARTS [18] stabilizes the search through perturbation-based regularization, penalizing architectures whose performance is sensitive to small perturbations in the architecture parameters.

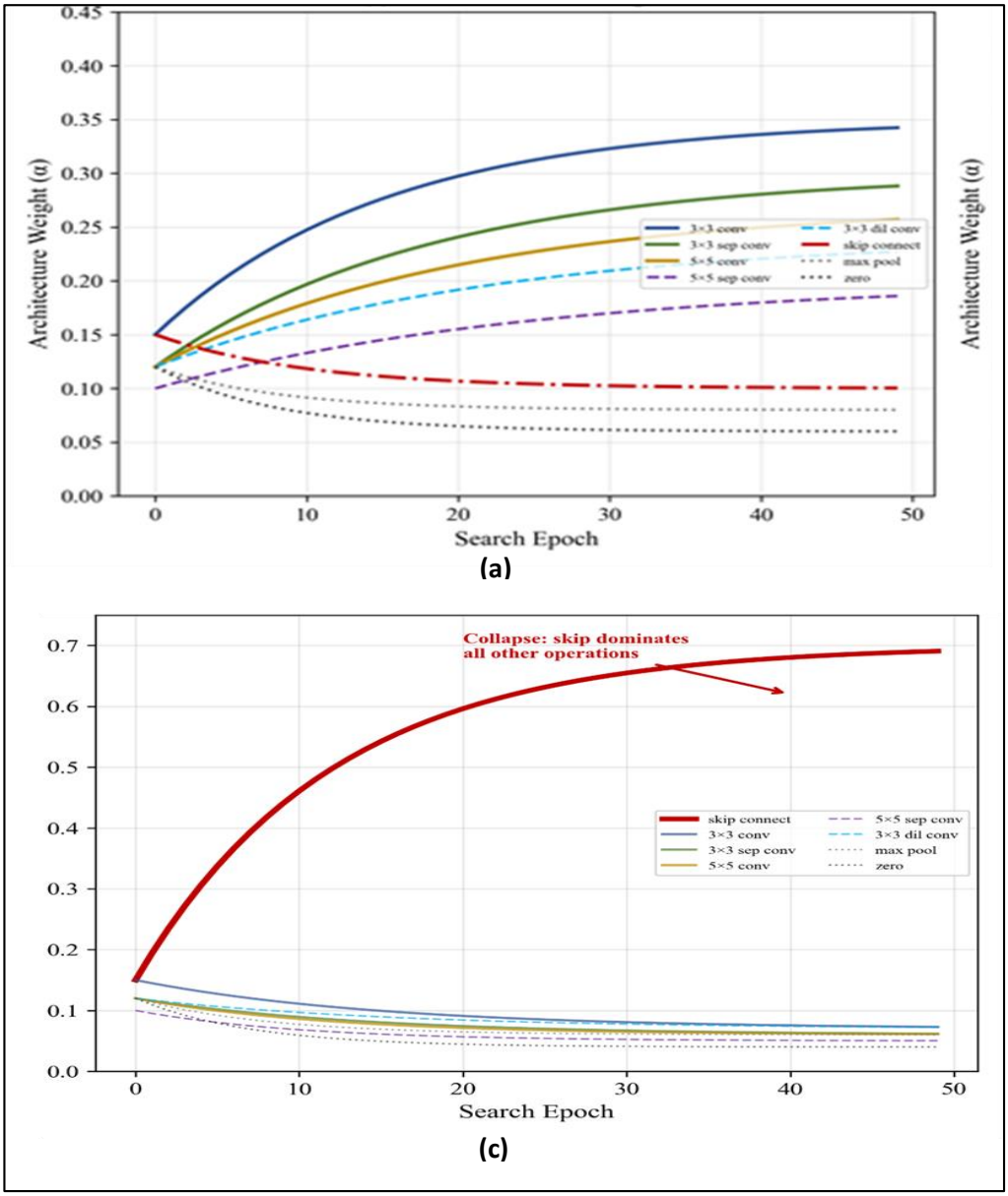


Fig 1: DARTS search evolution showing (a) normal search convergence and (b) skip connection collapse.

The collapse manifests as architecture parameters for skip connections increasing monotonically. Adapted from [7], [12].

Table 2. Comparison of DARTS Variants: Stability and Performance on CIFAR-10

| Method | Skip Collapse? | CIFAR-10 Error (%) | Search Cost (GPU-h) | Key Fix |
|---|---|---|---|---|
| DARTS [7] | Yes (often) | 2.76 ± 0.09 | 24 | — |
| DARTS- [12] | No | 2.59 ± 0.08 | 24 | Auxiliary skip |
| FairDARTS [16] | No | 2.54 ± 0.05 | 12 | Sigmoid relaxation |
| P-DARTS [15] | Rare | 2.50 ± 0.06 | 7.2 | Progressive depth |
| SDARTS [18] | No | 2.61 ± 0.02 | 31 | Perturbation reg. |
| GAEA [17] | No | 2.50 ± 0.06 | 8.3 | Exp. gradient |

## IV. SUPERNET AND WEIGHT-SHARING APPROACHES

### A. One-Shot NAS

One-shot NAS methods train a single supernet that encompasses all candidate architectures as subnetworks, then evaluate architectures by inheriting weights from the trained supernet [19]. This weight-sharing strategy eliminates the need to train each architecture independently, reducing the evaluation cost from hours to

milliseconds per architecture. The supernet is typically trained using uniform path sampling: at each training step, a random subnetwork (path) is sampled and updated.However, weight sharing introduces coupling between architectures—the shared weights are optimized for the average architecture rather than any specific one, potentially degrading the ranking accuracy [20]. Single-path one-shot NAS [21] mitigates this by sampling a single path per step and using uniform sampling to ensure all operations receive equal training, improving rank correlation with standalone training.

**B. Once-for-All (OFA)**

Once-for-All (OFA) [8] extends the one-shot paradigm by training a single supernet that supports diverse deployment targets (different latencies, memory constraints, accuracy requirements). OFA trains with progressive shrinking, starting from the full network and progressively adding support for smaller subnetworks. At deployment, architecture-specific subnetworks are extracted without additional training, achieving a Pareto frontier of accuracy-efficiency trade-offs from a single training run.

OFA achieves remarkable results: a single supernet provides over 10^19 subnetworks covering a wide range of hardware targets. Each extracted subnetwork achieves comparable accuracy to independently trained networks of similar size, demonstrating that the progressive shrinking strategy effectively mitigates the weight-sharing quality degradation [8].

**C. FairNAS and Sampling Strategies**

The training strategy for the supernet significantly impacts the quality of architecture rankings. FairNAS [22] identifies that different operations within the supernet receive unequal training due to varying sampling frequencies and gradient magnitudes. By enforcing strict fairness constraints—ensuring each operation is sampled equally often and receives comparable gradient updates—FairNAS improves the rank correlation between supernet-estimated and true performance from $\tau \approx 0.4$ to $\tau \approx 0.7$ on NAS-Bench-201 [22].

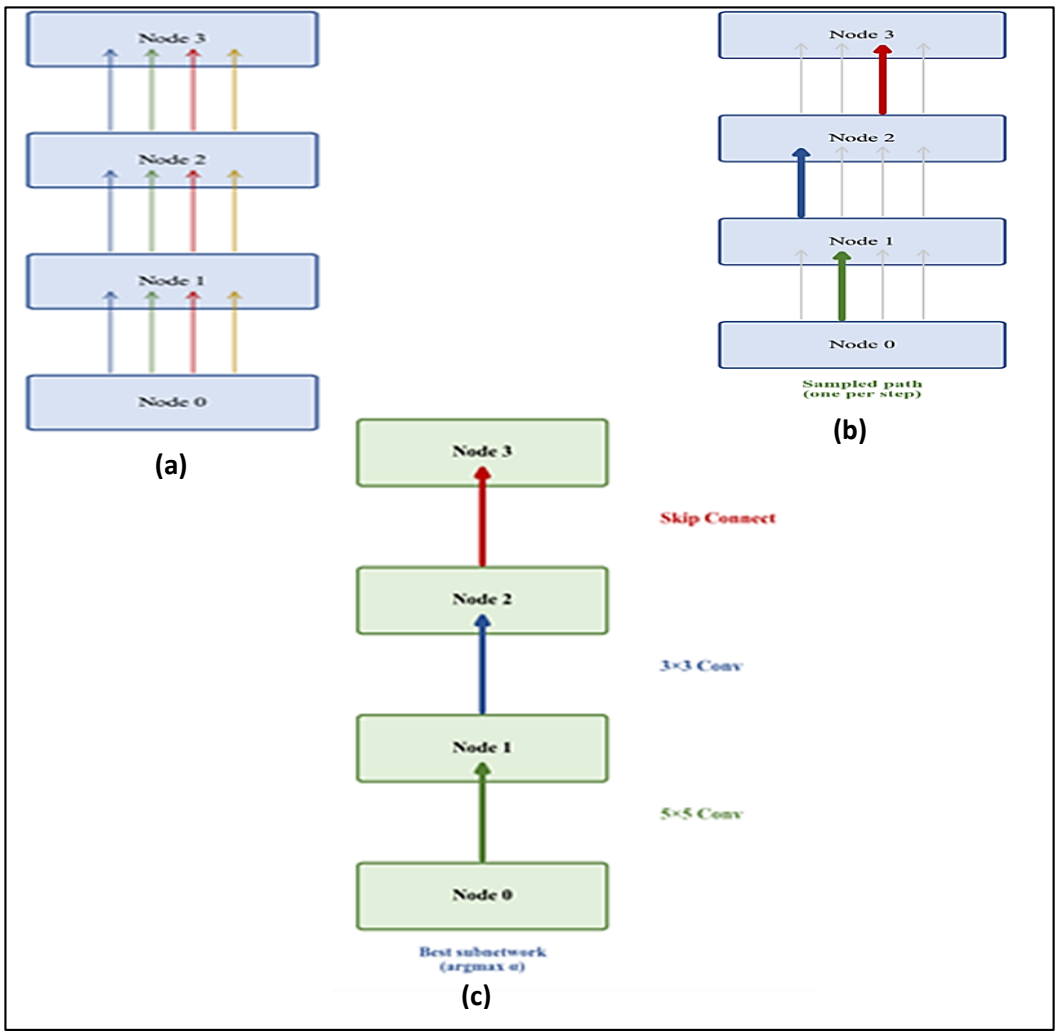


Fig 2: Supernet training and architecture extraction in One-Shot NAS. (a) Full Support, (b)Random Path Sampling, (c) Extracted Architecture

The supernet (left) contains all candidate operations; random paths (center) are sampled during training; final architectures (right) are extracted by selecting the best-performing subnetwork. Adapted from [19].

## V. ZERO-COST PROXIES FOR ARCHITECTURE EVALUATION

### A. Motivation and Overview

Zero-cost proxies represent the most extreme form of proxy-based NAS: they estimate architecture quality from a randomly initialized network using only a single minibatch of data, requiring no training whatsoever [9]. The appeal is clear—reducing the evaluation cost from GPU-hours (standalone training) or GPU-minutes (supernet evaluation) to GPU-seconds enables exhaustive search over large spaces and makes NAS accessible on commodity hardware.

### B. Gradient-Based Proxies

Gradient-based zero-cost proxies analyze the gradient flow through an untrained network to estimate its trainability and ultimate performance. Synflow [23] computes the sum of the product of all parameters and their gradients with respect to a unit input, measuring the network's gradient flow capacity. GradNorm [9] computes the Euclidean norm of the gradients at initialization, with the intuition that networks with well-scaled gradients train more effectively.

SNIP (Single-shot Network Pruning) [24] and GraSP (Gradient Signal Preservation) [25] were originally proposed for pruning but have been repurposed as NAS proxies. SNIP measures the connection sensitivity—the change in loss when a parameter is removed—while GraSP measures the gradient signal preservation, favoring architectures that maintain gradient information through their depth.

### C. Spectrum-Based Proxies

Spectrum-based proxies analyze the Jacobian or neural tangent kernel (NTK) of the network at initialization. NASWOT (NAS Without Training) [26] computes the number of linear regions in the input space by analyzing the activation patterns across a batch of inputs. Networks with more distinct linear regions have greater expressiveness, and this metric correlates strongly with final accuracy ($\tau \approx 0.6$–$0.8$ across NAS benchmarks).

TE-NAS (Training-free NAS) [10] combines two complementary metrics: the spectrum of the NTK (measuring trainability, with a flatter spectrum indicating more uniform training dynamics) and the number of linear regions (measuring expressiveness). By searching for architectures that maximize both metrics simultaneously, TE-NAS discovers competitive architectures on CIFAR-10 and ImageNet in approximately 0.05 GPU-hours [10].

Table 3. Zero-Cost Proxy Performance on NAS-Bench-201 (Kendall's τ)

| Proxy | Computation | NAS-Bench-201 τ | Cost (seconds) | Property Measured |
|---|---|---|---|---|
| Synflow [23] | Parameter-gradient product | 0.64 | ~2 | Gradient flow |
| GradNorm [9] | Gradient L2 norm | 0.58 | ~3 | Gradient magnitude |
| SNIP [24] | Connection sensitivity | 0.61 | ~3 | Parameter importance |
| GraSP [25] | Gradient preservation | 0.55 | ~5 | Signal propagation |
| NASWOT [26] | Linear region count | 0.72 | ~4 | Expressiveness |
| NTK cond. [10] | NTK eigenvalue ratio | 0.65 | ~8 | Trainability |
| Ensemble [9] | Weighted combination | 0.78 | ~15 | Multiple |

### D. Proxy Ensembles and Learning

Individual zero-cost proxies capture different aspects of architecture quality. Abdelfattah et al. [9] demonstrated that combining multiple proxies through learned ensembles significantly improves ranking accuracy. A simple weighted sum of proxy scores, with weights learned on a small held-out set of architecture evaluations, achieves $\tau > 0.78$ on NAS-Bench-201—competitive with supernet-based methods that require orders of magnitude more computation.

More sophisticated combination strategies include: gradient-boosted ranking models trained on proxy features [27]; Bayesian optimization with zero-cost proxy priors [28]; and multi-fidelity approaches that use zero-cost proxies for initial filtering followed by short training runs for final selection [29]. These hybrid strategies achieve the best rank correlations ($\tau > 0.85$) while maintaining practical search costs.

## VI. UNDERSTANDING AND MITIGATING THE PROXY GAP

The proxy gap refers to the discrepancy between architecture performance estimated during search (using any proxy strategy) and true performance after full standalone training [30]. This gap arises from multiple sources: weight sharing in supernet-based methods, continuous relaxation artifacts in DARTS-like methods, and the inherent approximation in zero-cost metrics.

Yu et al. [30] conducted a systematic study of proxy gaps across NAS methods, revealing that the gap is not uniform—certain architecture families (e.g., those with complex connectivity patterns) exhibit larger proxy gaps than simpler architectures. This finding suggests that proxy-based methods may systematically bias architecture selection toward simpler designs that are easier to evaluate but not necessarily optimal.

Mitigation strategies include: progressive evaluation, where cheap proxies filter candidates and more expensive evaluation is applied only to top candidates [29]; architecture-aware training that adjusts the supernet training procedure based on the architecture's complexity [31]; and calibration methods that learn a mapping from proxy scores to true performance, enabling corrected architecture selection [32].

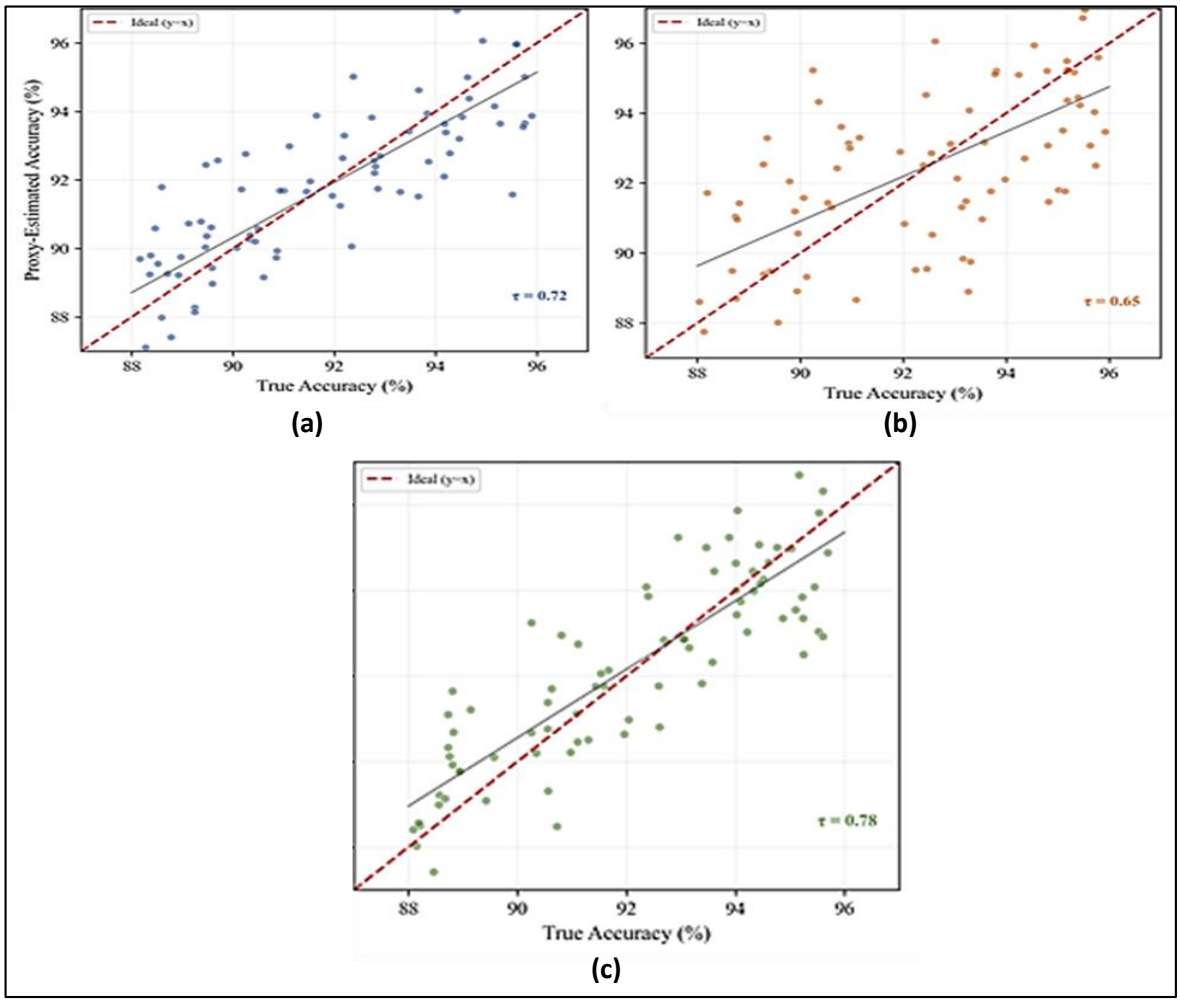


Fig 3: Proxy gap analysis: scatter plots of proxy-estimated vs. true accuracy for (a) weight-sharing supernet, (b) DARTS continuous relaxation, and (c) zero-cost proxy ensemble. Ideal correlation shown as dashed diagonal. Adapted from [30].

## VII. STANDARDIZED NAS BENCHMARKS

### A. NAS-Bench-101 and NAS-Bench-201

NAS benchmarks provide precomputed architecture-performance mappings, enabling reproducible evaluation of NAS methods without the computational cost of training architectures. NAS-Bench-101 [33] provides the training results of 423,624 unique architectures on CIFAR-10, each trained for 108 epochs with standardized hyperparameters. NAS-Bench-201 [34] extends this to three datasets (CIFAR-10, CIFAR-100, ImageNet-16-120) with 15,625 architectures, providing more comprehensive cross-dataset evaluation.

### B. TransNAS-Bench-101 and Beyond

TransNAS-Bench-101 [35] extends NAS benchmarks to seven vision tasks beyond classification (object detection, semantic segmentation, surface normal prediction, etc.), revealing that architecture rankings are task-dependent—the optimal architecture for classification may perform poorly on detection. This finding motivates task-aware NAS methods that consider the target task during search rather than using classification as a universal proxy.

## VIII. APPLICATIONS OF DIFFERENTIABLE NAS

### A. Hardware-Aware NAS

Practical deployment requires architectures optimized for specific hardware constraints. Hardware-aware NAS incorporates latency, memory, or energy consumption as additional objectives alongside accuracy. FBNet [36] uses differentiable NAS with a latency lookup table to discover architectures optimized for mobile deployment. ProxylessNAS [37] directly optimizes for on-device latency through differentiable latency prediction, discovering architectures that achieve 3.1% higher ImageNet accuracy than MobileNetV2 at the same latency.

### B. NAS for Transformers

AutoFormer [38] applies one-shot NAS to the transformer architecture space, searching over embedding dimensions, number of heads, MLP ratios, and network depth. The discovered architectures achieve superior accuracy-efficiency trade-offs compared to hand-designed transformers (DeiT, Swin) across a range of model sizes. HAT (Hardware-Aware Transformers) [39] extends this to hardware-specific transformer design, discovering architectures optimized for different hardware backends.

### C. NAS for Scientific Applications

Beyond computer vision, differentiable NAS has been applied to molecular property prediction [40], weather forecasting [41], and physics-informed neural networks [42]. These domains often feature unique inductive biases (e.g., physical symmetries, conservation laws) that can be incorporated into the NAS search space, enabling the discovery of architectures that are both accurate and physically consistent.

## IX. OPEN CHALLENGES AND FUTURE DIRECTIONS

### A. Generalization Across Scales

Architectures discovered at small scale (CIFAR-10, small models) often fail to transfer to larger scales (ImageNet, large models) [15]. Developing NAS methods that reliably predict large-scale performance from small-scale search—or that search directly at large scale with acceptable cost—remains an open challenge. Zero-cost proxies partially address this through scale-independent metrics, but their correlation with performance degrades at larger scales [9].

### B. Search Space Design Automation

Current NAS methods search within human-designed search spaces, which embed strong priors about architecture structure. Automating the design of search spaces themselves—a meta-NAS problem—could discover fundamentally novel architectural paradigms beyond the cell-based structures that dominate current approaches [43].

### C. Theoretical Foundations

The theoretical understanding of NAS remains limited. Key open questions include: Under what conditions does the continuous relaxation preserve the optimal architecture ranking? What is the sample complexity of learning accurate zero-cost proxies? Can we provide approximation guarantees for proxy-based NAS relative to the true optimal architecture?

### D. Sustainability

The environmental impact of large-scale NAS experiments has drawn criticism [44]. While differentiable methods dramatically reduce per-search costs, the cumulative computational investment in NAS research across the community remains substantial. Developing NAS methods that are both computationally efficient and generalizable—reducing the need for repeated search across similar tasks—is important for sustainable AI research.

## X. CONCLUSION

Differentiable Neural Architecture Search has transformed automated architecture design from a computationally exclusive endeavor to an accessible and practical tool. This paper has traced the evolution from DARTS' foundational continuous relaxation through robust variants addressing failure modes, weight-sharing supernet approaches enabling amortized evaluation, and zero-cost proxies achieving architecture assessment in seconds.

The field has achieved remarkable progress in computational efficiency—reducing search costs by six orders of magnitude from early RL-based methods to zero-cost proxies—while maintaining competitive architecture quality. The development of standardized benchmarks (NAS-Bench-101, NAS-Bench-201) has enabled rigorous and reproducible evaluation, accelerating methodological advances.

Key remaining challenges include bridging the proxy gap between search and evaluation, generalizing discovered architectures across scales and tasks, automating search space design, and developing stronger theoretical foundations. As differentiable NAS methods mature, we anticipate their integration into standard deep learning workflows, enabling practitioners to automatically customize architectures for their specific tasks, data, and deployment constraints.

## REFERENCES


[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2016, pp. 770–778.

[3] A. Vaswani et al., "Attention is all you need," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2017, pp. 5998–6008.

[4] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," J. Mach. Learn. Res., vol. 20, no. 55, pp. 1–21, 2019.

[5] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in Proc. Int. Conf. Learn. Represent. (ICLR), 2017.

[6] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in Proc. AAAI Conf. Artif. Intell., vol. 33, 2019, pp. 4780–4789.

[7] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2019.

[8] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.

[9] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, and N. D. Lane, "Zero-cost proxies for lightweight NAS," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[10] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on ImageNet in four GPU hours: A theoretically inspired perspective," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[11] C. White, W. Neiswanger, and Y. Savani, "Bananas: Bayesian optimization with neural architectures for neural architecture search," in Proc. AAAI Conf. Artif. Intell., 2021.

[12] X. Chu, X. Zhang, and B. Lu, "DARTS-: Robustly stepping out of performance collapse without indicators," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[13] A. Zela, T. Elsken, T. Saikia, Y. Marber, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.

[14] X. Chen and C. Hsieh, "Stabilizing differentiable architecture search via perturbation-based architecture selection," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.

[15] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), 2019.

[16] X. Chu, T. Zhou, B. Zhang, and J. Li, "Fair DARTS: Eliminating unfair advantages in differentiable architecture search," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2020.

[17] L. Li, M. Khodak, N. Balcan, and A. Talwalkar, "Geometry-aware gradient algorithms for neural architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[18] X. Chen and C. Hsieh, "Stabilizing differentiable architecture search via perturbation-based architecture selection," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.

[19] G. Bender, P. J. Kindermans, B. Zoph, V. Vasudevan, and Q. V. Le, "Understanding and simplifying one-shot architecture search," in Proc. Int. Conf. Mach. Learn. (ICML), 2018.

[20] Z. Zhang, Z. Zhu, L. Zhu, and S. Huang, "How does supernet help in neural architecture search?" arXiv preprint arXiv:2010.08219, Oct. 2020.

[21] Z. Guo et al., "Single path one-shot neural architecture search with uniform sampling," in Proc. Eur. Conf. Comput. Vis. (ECCV), 2020.

[22] X. Chu, B. Zhang, R. Xu, and J. Li, "FairNAS: Rethinking one-shot neural architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2021.

[23] H. Tanaka, D. Kunin, D. L. K. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2020.
[24] N. Lee, T. Ajanthan, and P. H. S. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," in Proc. Int. Conf. Learn. Represent. (ICLR), 2019.
[25] C. Wang, G. Zhang, and R. Grosse, "Picking winning tickets before training by preserving gradient flow," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.
[26] J. Mellor, J. Turner, A. Sherstone, and E. Sherstone, "Neural architecture search without training," in Proc. Int. Conf. Mach. Learn. (ICML), 2021.
[27] C. White, A. Zela, R. Impellizzeri, B. Neiswanger, and F. Hutter, "How powerful are performance predictors in neural architecture search?" in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2021.
[28] C. White, W. Neiswanger, S. Nolen, and Y. Savani, "A study on encodings for neural architecture search," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2020.
[29] Z. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in Proc. Conf. Uncertainty Artif. Intell. (UAI), 2019.
[30] K. Yu, C. Sciuto, M. Jaggi, C. Muber, and M. Salzmann, "Evaluating the search phase of neural architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.
[31] R. Luo, X. Tan, R. Wang, T. Qin, E. Chen, and T.-Y. Liu, "Semi-supervised neural architecture search," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2020.
[32] N. Nayman, A. Noy, T. Ridnik, I. Friedman, R. Jin, and L. Zelnik-Manor, "XNAS: Neural architecture search with expert advice," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2019.
[33] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-Bench-101: Towards reproducible neural architecture search," in Proc. Int. Conf. Mach. Learn. (ICML), 2019.
[34] X. Dong and Y. Yang, "NAS-Bench-201: Extending the scope of reproducible neural architecture search," in Proc. Int. Conf. Learn. Represent. (ICLR), 2020.
[35] Y. Duan et al., "TransNAS-Bench-101: Improving transferability and generalizability of cross-task neural architecture search," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2021.
[36] B. Wu et al., "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019.
[37] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in Proc. Int. Conf. Learn. Represent. (ICLR), 2019.
[38] M. Chen et al., "AutoFormer: Searching transformers for visual recognition," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), 2021.
[39] H. Wang et al., "HAT: Hardware-aware transformers for language modeling," in Proc. Annu. Meet. Assoc. Comput. Linguist. (ACL), 2020.
[40] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, "Masked label prediction: Unified message passing model for semi-supervised classification," in Proc. Int. Joint Conf. Artif. Intell. (IJCAI), 2021.
[41] S. Rasp et al., "WeatherBench 2: A benchmark for the next generation of data-driven global weather models," arXiv preprint arXiv:2308.15560, Aug. 2023.
[42] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys., vol. 378, pp. 686–707, Feb. 2019.
[43] A. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), 2020, pp. 10428–10436.
[44] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in Proc. Annu. Meet. Assoc. Comput. Linguist. (ACL), 2019, pp. 3645–3650.