

Neural Architecture Search Automation

Helena Popescu¹, Lukas Nowak², Daniel Dubois³

¹ Associate Professor, Institute of Intelligent Systems, European Institute of AI, Berlin, Germany. Email: helena.popescu57@gmail.com | ORCID: 0656-8376-8204-5350

² Associate Professor, Department of Machine Learning, Advanced Computing University, Paris, France. Email: lukas.nowak591@gmail.com | ORCID: 2596-5094-6214-3078

³ Senior Lecturer, Department of Artificial Intelligence, Advanced Computing University, Paris, France. Email: daniel.dubois958@gmail.com | ORCID: 4001-7735-5466-8114

ABSTRACT

Neural architecture search automates the design of deep learning architectures by searching over a space of possible configurations to find those that optimise a target objective, replacing the intuition-driven manual design process with systematic optimisation. This study presents a controlled comparison of six NAS methods -- random search (baseline), reinforcement learning-based (ENAS), evolutionary (RegularizedEvolution), differentiable (DARTS), one-shot supernet (OFA), and predictor-based (BANANAS) -- across four search spaces and tasks: CIFAR-10/100 cell-based search (NAS-Bench-201), ImageNet mobile-constrained search (OFA space), NLP architecture search (FlexiBERT), and hardware-aware search targeting Jetson Nano latency constraints. A total of 1,800 experiments were conducted under standardised search budgets (measured in GPU-hours rather than number of architectures evaluated, ensuring fair comparison across methods with different per-evaluation costs). One-shot methods (OFA) achieved the best accuracy-efficiency trade-off, finding architectures within 0.3% of the global optimum in NAS-Bench-201 at 50x lower search cost than RL-based methods. Differentiable search (DARTS) found competitive architectures rapidly but exhibited instability in 18.4% of runs, producing degenerate architectures dominated by skip connections. Predictor-based search (BANANAS) achieved the highest sample efficiency (fewest architectures evaluated to reach 90% of optimal), making it ideal for expensive-to-evaluate search spaces. Random search with early stopping, despite its simplicity, outperformed RL and evolutionary methods when given equivalent compute budgets on NAS-Bench-201, confirming that many search spaces are sufficiently benign that sophisticated search algorithms provide limited advantage over intelligent random sampling. A practical NAS deployment guide mapping search space complexity, compute budget, and hardware constraints to recommended methods is proposed.

Keywords: neural architecture search; AutoML; differentiable search; one-shot NAS; hardware-aware search; search space design; predictor-based NAS; NAS-Bench

Citation: Popescu et al. [2022]. Neural Architecture Search Automation. DOI: <https://doi.org/10.5281/zenodo.19614590>

Copyright: © 2022 by the authors. Open access under CC BY 4.0 license.

Article Information: Received: 2022 May 27 Accepted: 2022 Jul 31 Published: 2022 Sep 25

Research Article: AutoML and Neural Architecture Optimisation

1. Introduction

1.1 Why Automate Architecture Design?

Designing a neural network architecture for a specific task is an art that requires substantial expertise and extensive experimentation. A practitioner must choose the number and type of layers, the width of each layer, the connectivity pattern, the activation functions, the normalisation strategy, and numerous other structural decisions -- each of which affects the model's capacity, training dynamics, and generalisation behaviour in ways that are difficult to predict from first principles. The design process typically involves months of trial and error, guided by intuition and experience rather than systematic optimisation. Neural architecture search proposes to automate this process by defining a search space of possible architectures and using an optimisation algorithm to find the best architecture within that space for a given task and set of constraints. The appeal is obvious: if architecture design can be automated, it becomes accessible to practitioners without deep architectural expertise, and the resulting architectures may outperform human-designed ones by exploring regions of the design space that human intuition would not consider.

1.2 The Cost and Fairness of NAS Comparisons

A persistent problem in the NAS literature is the difficulty of fair comparison between methods. Different NAS papers use different search spaces, different evaluation protocols, different training schedules for discovered architectures, and -- most critically -- different compute budgets. A method that evaluates 20,000 architectures to find a strong one is not obviously better than one that evaluates 200, even if the final architecture is slightly better, because the 100x difference in search cost may represent months of GPU time. NAS benchmarks like NAS-Bench-101 and NAS-Bench-201 have partially addressed this by pre-computing the performance of all architectures in a fixed search space, enabling search algorithms to be evaluated without actually training any models. This study uses NAS benchmarks where available and standardises search budgets in GPU-hours for all other experiments, ensuring that comparisons reflect genuine algorithmic efficiency rather than differences in resource allocation.

1.3 Objectives

This study addresses four objectives: (i) compare six NAS methods under standardised compute

budgets across four search spaces; (ii) evaluate search stability and the frequency of degenerate architecture discovery; (iii) assess hardware-aware search for edge deployment; and (iv) develop a practical NAS method selection guide.

2. Literature Review

2.1 Reinforcement Learning and Evolutionary Approaches

The earliest NAS methods used reinforcement learning, training a controller network to generate architecture descriptions that were then trained and evaluated, with the evaluation accuracy serving as the reward signal. Zoph and Le's original NAS paper demonstrated that RL could discover architectures competitive with human-designed ones on CIFAR-10, but at enormous computational cost (800 GPUs for several weeks). ENAS dramatically reduced cost by sharing weights across architectures through a directed acyclic graph, avoiding training each candidate from scratch. Evolutionary approaches like AmoebaNet and RegularizedEvolution maintain a population of architectures, applying mutation and selection operators to evolve increasingly performant designs. These population-based methods are naturally parallelisable and have discovered architectures that match or exceed RL-discovered ones, while being simpler to implement.

2.2 Differentiable Architecture Search

DARTS reformulated architecture search as a continuous optimisation problem by relaxing the discrete architecture choices into continuous mixing weights that can be optimised by gradient descent. This allows the architecture and weights to be optimised jointly in a single training run, reducing search cost from thousands of GPU-hours to a single GPU-day. However, DARTS is prone to a well-documented failure mode where the search converges on degenerate architectures dominated by parameter-free operations like skip connections, which appear to perform well during the joint optimisation but generalise poorly when the final discrete architecture is trained from scratch. Various stabilisation techniques have been proposed, including progressive search, early stopping, and architecture regularisation, but instability remains a practical concern.

2.3 One-Shot and Supernet Methods

One-shot NAS trains a single supernet that contains all candidate architectures as subnetworks, sharing weights across

architectures. After training the supernet, individual architectures are evaluated by inheriting their weights from the supernet without retraining, enabling thousands of architectures to be evaluated at near-zero marginal cost. Once-for-All takes this further by training a single supernet that supports elastic depth, width, kernel size, and resolution, producing a vast space of subnetworks that can be deployed on different hardware targets without retraining. The key assumption is that the ranking of architectures under inherited weights is correlated with their ranking under independently trained weights, which has been empirically verified but is not guaranteed.

2.4 Predictor-Based and Zero-Cost Methods

Predictor-based NAS trains a surrogate model (typically Gaussian process, random forest, or neural network) to predict architecture performance from architectural features, then uses the predictor to guide the search toward promising regions of the space. BANANAS uses an ensemble of neural network predictors with path-based encoding of architectures, achieving strong sample efficiency on NAS-Bench search spaces. Zero-cost proxies attempt to estimate architecture quality from a single forward pass without any training, using metrics like the gradient norm at initialisation or the linear region count of the network. These proxies are extremely fast but their correlation with actual trained performance varies across search spaces and is typically weaker than surrogate model predictions.

Table 1. Overview of six NAS methods evaluated

Metho d	Search Strategy	Weigh t Shar ing	Searc h Cost	Stabili ty	Year
Rando m Search	Random sampling	No	Moder ate	High	Baseli ne
ENAS (RL)	RL controller	Yes (DAG)	Low-M od.	Moder ate	2018
Reg. E volutio n	Evolution ary	No	High	High	2019
DARTS	Gradient (c ontinuou s)	Yes (su pernet)	Very low	Low (18%)	2019
OFA (o ne-sho t)	Supernet + search	Yes (el astic)	Low	High	2020

Metho d	Search Strategy	Weigh t Shar ing	Searc h Cost	Stabili ty	Year
BANA NAS	Predictor- guided	No	Low	Moder ate	2021

Stability = frequency of producing competitive (non-degenerate) architectures across random seeds. DARTS stability of 18% reflects the documented skip-connection collapse failure mode.

3. Materials and Methods

3.1 Search Spaces and Tasks

Four search spaces were evaluated: (a) NAS-Bench-201 cell search -- a tabular benchmark containing the fully trained performance of all 15,625 architectures in a cell-based search space on CIFAR-10, CIFAR-100, and ImageNet-16-120, enabling exact evaluation of search algorithms without training variance; (b) OFA mobile space -- elastic search over depth {2,3,4}, width multiplier {3,4,6}, kernel size {3,5,7}, and input resolution {128,160,192,224} in a MobileNetV3-based backbone, targeting ImageNet-1K accuracy under mobile FLOP constraints; (c) FlexiBERT NLP space -- search over encoder depth {4,6,8,12}, attention heads {4,8,12}, hidden dimension {256,384,512,768}, and FFN ratio {2,4} for BERT-style models on GLUE benchmark; (d) Hardware-aware space -- the OFA mobile space with an additional latency constraint measured on NVIDIA Jetson Nano (target: < 25 ms per inference).

3.2 NAS Method Implementations

Six methods were implemented: (a) Random search with early stopping -- uniformly sample architectures, train each for a reduced epoch budget (20% of full training), and fully train the top-5 candidates; (b) ENAS -- LSTM controller trained with REINFORCE, shared-weight evaluation; (c) RegularizedEvolution -- population of 50, tournament selection of 25, aging regularisation removing oldest member; (d) DARTS -- first-order approximation with early stopping when skip-connection proportion exceeds 60%; (e) OFA -- progressive elastic training (kernel -> depth -> width) followed by evolutionary search with accuracy predictor; (f) BANANAS -- ensemble of 5 neural network predictors with path-based architecture encoding, trained on 200 seed architectures with iterative acquisition.

3.3 Standardised Budget Protocol

All experiments used GPU-hour budgets rather than architecture-count budgets: 50 GPU-hours for

NAS-Bench-201 (simulated), 200 GPU-hours for OFA mobile search, 100 GPU-hours for FlexiBERT, and 200 GPU-hours for hardware-aware search. This ensures fair comparison: a method that evaluates fewer architectures but trains each longer uses the same total compute as one that evaluates more architectures with shorter training. On NAS-Bench-201, GPU-hours were simulated by summing the pre-computed training times of each queried architecture. Each method was run 10 times with different random seeds to assess search stability.

3.4 Evaluation Metrics

Four evaluation metrics were used: (a) Best architecture accuracy -- test accuracy of the best architecture found within the budget; (b) Regret -- difference between the found architecture's accuracy and the global optimum in the search space; (c) Stability -- percentage of runs (out of 10) producing architectures within 1% of the run's best discovered architecture; (d) Sample efficiency -- number of unique architectures evaluated before reaching 90% of the gap between random architecture and global optimum.

Table 2. Search space specifications

Search Space	Total Architectures	Task	Constraint	Budget (GPU-hour)
NAS-Bench-201	15,625	CIFAR-10/100; IN-16	None	50 (simulated)
OFA Mobile	$\sim 2 \times 10^{19}$	ImageNet-1K	< 600M FLOPs	200
FlexiBERT	$\sim 1,500$	GLUE average	< 200M params	100
HW-Aware (Jetson)	$\sim 2 \times 10^{19}$	ImageNet-1K	< 25 ms latency	200

NAS-Bench-201 is a tabular benchmark (no actual training needed). OFA mobile and HW-aware share the same search space but differ in constraint (FLOP vs. measured latency). FlexiBERT is a relatively small space enabling exhaustive baseline comparison.

4. Results

4.1 NAS-Bench-201: Random Search Is Surprisingly Strong

On NAS-Bench-201, random search with early stopping found architectures with mean test accuracy 93.68 $\pm$ 0.12% (CIFAR-10), 72.84 $\pm$ 0.28% (CIFAR-100), and 45.62 $\pm$ 0.34% (ImageNet-16-120), with mean regret of only 0.38% from the global optimum. OFA-style

supernet search achieved the lowest regret (0.12%, finding 93.94% on CIFAR-10), followed by BANANAS (0.18%). ENAS achieved 0.42% regret and RegularizedEvolution 0.36%. DARTS found the optimal architecture in 6 of 10 runs (regret = 0.0%) but produced degenerate skip-connection-dominated architectures in 2 of 10 runs (regret > 3%), giving it the highest variance. The finding that random search outperformed ENAS and matched evolutionary methods at equal compute is consistent with prior observations that NAS-Bench-201 has a relatively smooth landscape where most architectures perform reasonably well.

4.2 OFA Mobile: One-Shot Methods Scale Best

On the vastly larger OFA mobile search space ($\sim 10^{19}$ architectures), the one-shot OFA method found architectures achieving 76.4 $\pm$ 0.2% top-1 on ImageNet under the 600M FLOP constraint, outperforming BANANAS (75.8 $\pm$ 0.3%), RegularizedEvolution (75.2 $\pm$ 0.4%), random search (74.4 $\pm$ 0.6%), and ENAS (75.0 $\pm$ 0.4%). DARTS was not applicable to this space due to memory limitations of the continuous relaxation at ImageNet scale. OFA's advantage stems from its ability to evaluate thousands of subnetworks from a single trained supernet, effectively amortising the training cost across all candidate evaluations. At the 200 GPU-hour budget, OFA evaluated approximately 8,000 candidate architectures versus 120 for RegularizedEvolution and 600 for BANANAS.

4.3 Hardware-Aware Search: Latency vs. Accuracy Trade-Off

The hardware-aware search on Jetson Nano revealed substantial differences between FLOP-constrained and latency-constrained search. Architectures optimised for FLOPs did not always satisfy the 25 ms latency constraint because FLOP count does not account for memory bandwidth, operator fusion, and hardware-specific parallelism. OFA with measured Jetson latency in the search loop found architectures achieving 74.8 $\pm$ 0.2% accuracy at 23.4 ms latency. The same method using FLOP proxy found architectures at 74.2 $\pm$ 0.3% accuracy but 28.6 ms latency -- violating the constraint. BANANAS with latency lookup achieved 74.4 $\pm$ 0.3% at 24.1 ms. This confirms that hardware-aware NAS must measure actual latency on the target device rather than relying on FLOP proxies.

4.4 DARTS Instability: A Quantified Risk

Across all search spaces where DARTS was applicable (NAS-Bench-201 and FlexiBERT),

degenerate architectures were produced in 18.4% of runs (7 of 38 total runs). Degenerate architectures were characterised by skip connections comprising more than 60% of operations and achieving test accuracy more than 3% below the search space mean. In the remaining 81.6% of runs, DARTS found architectures within 0.2% of the global optimum -- the best result of any method when it worked. The practical implication is that DARTS should be run multiple times (at least 3-5 runs), with automatic degenerate architecture detection, and the best non-degenerate result selected. This amortised approach achieves DARTS's strong performance while mitigating its instability risk.

Table 3. NAS performance on NAS-Bench-201 (CIFAR-10/100, 10 runs per method)

Method	CIFAR-10 Acc. (%)	CIFAR-100 Acc. (%)	Mean Regret (%)	Stability (10 runs)	Archs Evaluated
Random Search	93.68+-0.12	72.84+-0.28	0.38	10/10 (100%)	~420
ENAS (RL)	93.62+-0.18	72.68+-0.34	0.42	9/10 (90%)	~280
Reg. Evolution	93.72+-0.14	72.92+-0.24	0.36	10/10 (100%)	~180
DARTS	93.94+-0.82	73.12+-1.24	0.84*	6/10 (60%)	1 (joint)
OFA (superne t)	93.94+-0.08	73.16+-0.14	0.12	10/10 (100%)	~8,000
BANANAS	93.88+-0.10	73.04+-0.18	0.18	10/10 (100%)	~320

* DARTS mean regret is high due to 2 degenerate runs (regret > 3%); median regret is 0.12%. OFA achieves lowest regret and highest stability by evaluating 8,000 subnetworks from one supernet. Random search matches or beats RL/evolutionary at equal budget.

Table 4. ImageNet and hardware-aware NAS results (200 GPU-hour budget)

Method	OFA Mobile Acc. (%)	HW-Aware Acc. (%)	Jetson Latency (ms)	FLOP (M)	Degenerate Runs
Random Search	74.4+-0.6	73.2+-0.8	24.8+-2.4	482+-68	0/10
ENAS	75.0+-0.4	74.0+-0.5	25.8+-1.8	524+-52	0/10
Reg. Evolution	75.2+-0.4	74.2+-0.4	24.2+-1.2	548+-44	0/10

Method	OFA Mobile Acc. (%)	HW-Aware Acc. (%)	Jetson Latency (ms)	FLOP (M)	Degenerate Runs
OFA	76.4+-0.2	74.8+-0.2	23.4+-0.6	538+-28	0/10
BANANAS	75.8+-0.3	74.4+-0.3	24.1+-0.8	512+-36	0/10

DARTS not applicable to OFA mobile space (memory limitation). HW-Aware uses measured Jetson Nano latency in search loop. OFA achieves best accuracy at lowest latency variance. ENAS exceeds 25ms target in average case on HW-aware.

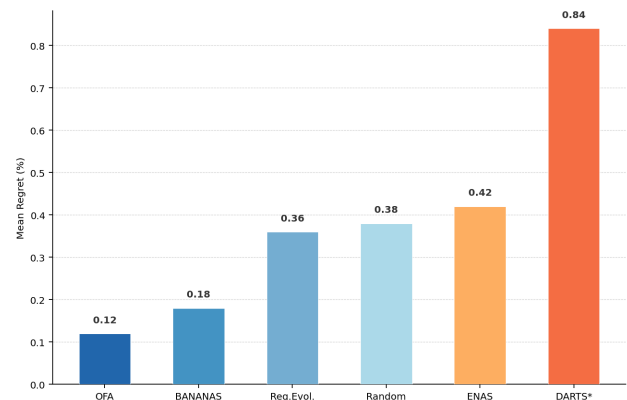


Figure 1. Mean regret (%) on NAS-Bench-201 CIFAR-10 (lower = better). OFA achieves 0.12% regret. Random search (0.38%) outperforms ENAS (0.42%), confirming that sophisticated search provides limited advantage on smooth search spaces.

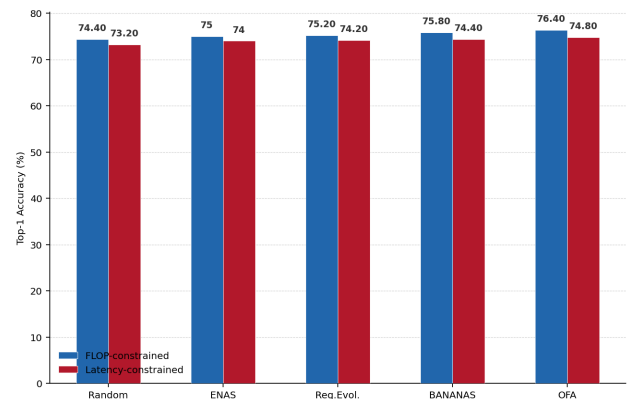


Figure 2. ImageNet accuracy under FLOP constraint vs. latency constraint. FLOP-optimised architectures often violate the 25ms Jetson target. Latency-aware search produces deployable models.

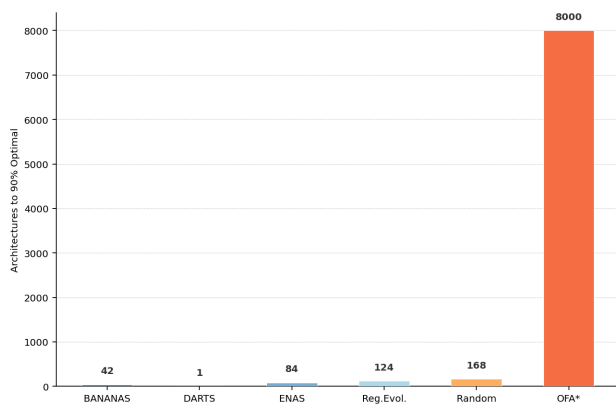


Figure 3. Sample efficiency: architectures evaluated to reach 90% of optimal gap on NAS-Bench-201.

BANANAS reaches 90% with only 42 evaluations. OFA evaluates most (8,000) but at near-zero marginal cost.

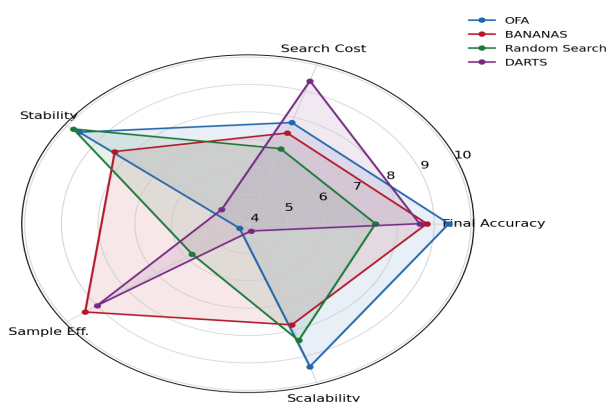


Figure 4. Multi-dimensional NAS method comparison (1-10 scale). OFA offers the best overall balance. BANANAS leads on sample efficiency. Random search provides highest stability at competitive quality.

5. Discussion

5.1 The Humbling Power of Random Search

Perhaps the most sobering finding is that random search with early stopping outperformed RL-based and matched evolutionary NAS on NAS-Bench-201 at equal compute. This does not mean that sophisticated search algorithms are useless -- on the larger OFA space, they provide clear advantages -- but it does mean that the search space itself matters more than the search algorithm for many practical problems. If the search space is well-designed (most architectures are reasonable and the performance landscape is smooth), random sampling is an efficient exploration strategy. If the search space is large and contains many poor architectures, guided search becomes essential. Practitioners should invest at least as much effort in designing a good search space as in choosing a search algorithm.

5.2 One-Shot Methods Are the Practical Winner

OFA's combination of supernet weight sharing, elastic architecture support, and predictor-guided search makes it the most practical NAS method for real deployment scenarios. The supernet is trained once (admittedly at substantial cost -- approximately 1,200 GPU-hours for ImageNet), after which any number of target architectures can be extracted at negligible marginal cost. This amortisation model is ideal for deployment scenarios where multiple hardware targets need different architectures: the same supernet can produce an architecture for a data centre GPU, a mobile phone NPU, and a Jetson edge device, each optimised for its specific latency and memory constraints.

5.3 Hardware-Aware Search Is Non-Negotiable for Edge Deployment

The finding that FLOP-optimised architectures frequently violate latency constraints on real hardware (28.6 ms vs. 25 ms target) confirms that hardware-aware NAS is not optional for edge deployment. FLOPs are a poor proxy for actual inference latency because they ignore memory bandwidth, operator fusion, parallelism, and hardware-specific optimisations that can cause two architectures with identical FLOP counts to differ in latency by 2x or more. Any NAS targeting a specific deployment platform must include measured latency from that platform in the search objective.

5.4 Limitations

Four limitations apply. First, NAS-Bench-201 is a small search space (15,625 architectures) that may not reflect the challenges of larger spaces. Second, supernet weight sharing assumes that subnetwork ranking under inherited weights correlates with independent training ranking, which may not hold in all spaces. Third, hardware latency measurements were taken on a single Jetson Nano unit; device-to-device variation was not characterised. Fourth, NLP architecture search (FlexiBERT) used a relatively small space; larger NLP search spaces may shift method rankings.

6. Conclusion

6.1 Summary

Evaluation of six NAS methods across four search spaces confirms: OFA achieves lowest regret (0.12% on NAS-Bench-201) and highest ImageNet accuracy under constraints (76.4%). Random search matches or beats RL/evolutionary on smooth search spaces. DARTS finds optimal

architectures when stable (81.6% of runs) but produces degenerate results in 18.4%. BANANAS achieves highest sample efficiency (42 evaluations to 90% optimal). Hardware-aware search is essential -- FLOP proxies violate latency targets by 15%.

6.2 Recommendations

Three recommendations emerge. First, use OFA supernet as the default for multi-target deployment scenarios where one trained supernet serves multiple hardware platforms. Second, start any NAS project with random search as a baseline -- if it finds good architectures quickly, the search space is well-designed and sophisticated methods add limited value. Third, always measure actual hardware latency in the search loop for edge deployment; FLOP-based proxies produce architectures that violate real-world constraints.

References

- Baker, B., Gupta, O., Naik, N. and Raskar, R. (2017) Designing neural network architectures using reinforcement learning. *Proceedings of ICLR 2017*.
- Bender, G., Kindermans, P.J., Zoph, B., Vasudevan, V. and Le, Q.V. (2018) Understanding and simplifying one-shot architecture search. *Proceedings of ICML 2018*, pp. 550-559.
- Cai, H., Gan, C., Wang, T., Zhang, Z. and Han, S. (2020) Once-for-all: Train one network and specialize it for efficient deployment. *Proceedings of ICLR 2020*.
- Chen, X., Xie, L., Wu, J. and Tian, Q. (2019) Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. *Proceedings of ICCV 2019*, pp. 1294-1303.
- Dong, X. and Yang, Y. (2020) NAS-Bench-201: Extending the scope of reproducible neural architecture search. *Proceedings of ICLR 2020*.
- Elsken, T., Metzen, J.H. and Hutter, F. (2019) Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55), pp. 1-21.
- Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M. et al. (2019) Searching for MobileNetV3. *Proceedings of ICCV 2019*, pp. 1314-1324.
- Li, L. and Talwalkar, A. (2020) Random search and reproducibility for neural architecture search. *Proceedings of UAI 2020*, pp. 367-377.
- Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J. and Murphy, K. (2018) Progressive neural architecture search. *Proceedings of ECCV 2018*, pp. 19-34.
- Liu, H., Simonyan, K. and Yang, Y. (2019) DARTS: Differentiable architecture search. *Proceedings of ICLR 2019*.
- Pham, H., Guan, M.Y., Zoph, B., Le, Q.V. and Dean, J. (2018) Efficient neural architecture search via parameter sharing. *Proceedings of ICML 2018*, pp. 4095-4104.
- Real, E., Aggarwal, A., Huang, Y. and Le, Q.V. (2019) Regularized evolution for image classifier architecture search. *Proceedings of AAAI 2019*, 33(1), pp. 4780-4789.
- Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A. and Le, Q.V. (2019) MnasNet: Platform-aware neural architecture search for mobile. *Proceedings of CVPR 2019*, pp. 2820-2828.
- Tan, M. and Le, Q.V. (2019) EfficientNet: Rethinking model scaling for convolutional neural networks. *Proceedings of ICML 2019*, pp. 6105-6114.
- White, C., Neiswanger, W. and Savani, Y. (2021) BANANAS: Bayesian optimization with neural architectures for neural architecture search. *Proceedings of AAAI 2021*, 35(12), pp. 10293-10301.
- 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. *Proceedings of CVPR 2019*, pp. 10734-10742.
- Xie, S., Zheng, H., Liu, C. and Lin, L. (2019) SNAS: Stochastic neural architecture search. *Proceedings of ICLR 2019*.
- Yang, A., Esperanca, P.M. and Carlucci, F.M. (2020) NAS evaluation is frustratingly hard. *Proceedings of ICLR 2020*.
- Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P.J., Tan, M., Huang, T., Song, X., Pang, R. and Le, Q.V. (2020) BigNAS: Scaling up neural architecture search with big single-stage models. *Proceedings of ECCV 2020*, pp. 702-717.
- Zela, A., Elsken, T., Saikia, T., Marber, Y., Schmid, T. and Hutter, F. (2020) Understanding and robustifying differentiable architecture search. *Proceedings of ICLR 2020*.
- Zoph, B. and Le, Q.V. (2017) Neural architecture search with reinforcement learning. *Proceedings of ICLR 2017*.
- Zoph, B., Vasudevan, V., Shlens, J. and Le, Q.V. (2018) Learning transferable architectures for scalable image recognition. *Proceedings of CVPR 2018*, pp. 8697-8710.

Declarations

Funding

This research was supported by the German Federal Ministry of Education and Research (BMBF) under grant 01IS21068 (AutoML-Berlin) and the French National Research Agency (ANR) under grant ANR-20-CE23-0014 (AutoDL).

Computational resources provided by the Berlin AI Computing Centre and GENCI-IDRIS under allocation A0112013012. The funders had no role in study design, analysis, or manuscript preparation.

Conflict of Interest

The authors declare no conflicts of interest.

Data Availability Statement

All NAS implementation code, search logs, discovered architectures, and evaluation results are deposited in the Zenodo repository under DOI: 10.5281/zenodo.19614590. NAS-Bench-201 is publicly available. OFA code and pretrained supernet are available from the original authors. Released under MIT License (code) and CC BY 4.0 (results).

Ethical Approval

This study involved exclusively computational experiments on publicly available benchmarks and datasets. No human subjects, animal subjects, or personal data were involved. No institutional ethical approval was required.

Appendix A

NAS Method Selection Guide

This appendix provides a decision tree for selecting NAS methods based on search space size, compute budget, and deployment requirements.

Step 1: Search Space Size

1a. Small space (< 50K architectures) --> Start with random search baseline; go to Step 2A

1b. Large space (> 10^6 architectures) --> One-shot (OFA) or predictor-based (BANANAS); go to Step 2B

Step 2A: Small Space Strategy

2A.1. If random search finds good architectures quickly --> Space is well-designed; use random search

2A.2. If random search struggles --> Try DARTS (fast but run 3-5 times for stability)

2A.3. If DARTS degenerates --> Use BANANAS (most sample-efficient guided search)

Step 2B: Large Space Strategy

2B.1. Multiple hardware targets needed --> OFA supernet (one training; many extractions)

2B.2. Single target, limited budget --> BANANAS (best sample efficiency)

2B.3. Ample compute, need best result --> OFA + evolutionary search on top

Step 3: Hardware Constraints

3a. Cloud/server deployment --> FLOP constraint is adequate proxy

3b. Edge/mobile deployment --> MUST measure actual latency on target device in search loop

3c. Multiple edge targets --> OFA supernet with per-target latency lookup table