International Journal of Current Research and Techniques (IJCRT)
E-ISSN: 2349-3194 (Online) | P-ISSN: 2348-4446 (Print)
UGC Approved Journal No: 47722 [No: 2616]
DOI: https://doi.org/10.5281/ZENODO.15423210

Neural Architecture Search: Designing Automated AI Models

Vivek Ranjan, *Supervisor, Department of Computer Science Noida Institute of Engineering and Technology Greater Noida, India***.**

Riya Pal, *Student, Department of Computer Science, Noida Institute of Engineering and Technology Greater Noida, India*

Prashant Saini, *Student, Department of Computer Science, Noida Institute of Engineering and Technology Greater Noida, India***.**

Divyanshu Raj Nirala, *Student, Department of Computer Science, Noida Institute of Engineering and Technology Greater Noida, India***.**

Manuscript Received: May 10, 2025; Revised: May 14, 2025; Published: May 15, 2025

Abstract: The development of effective neural network architectures has traditionally relied on human expertise and extensive trial-and-error. Neural Architecture Search (NAS) offers a transformative solution by automating the model design process, significantly reducing manual effort and accelerating innovation in artificial intelligence. As a core technique within Automated Machine Learning (AutoML), NAS systematically explores a predefined set of neural architectures to identify models that best meet performance requirements under specific constraints. This study examines NAS as a foundation for building automated AI models, emphasizing its core components: search space, optimization strategy, and performance evaluation. Over time, NAS methods have progressed from resource-intensive approaches, such as reinforcement learning, to more efficient alternatives like evolutionary algorithms, differentiable search, and weight-sharing models. These innovations reduce computational costs and expand NAS applications across various domains, including computer vision, natural language processing, and speech recognition. By intelligently navigating vast architectural possibilities, NAS facilitates the creation of adaptable, high-performance AI systems and establishes itself as a key tool in the future of automated AI model development.

Keywords: Neural Architecture Search, AutoML, Automated Model Design, Deep Learning, Optimization Strategies, AI Automation

1. Introduction:

Motivation and Background

With AI technologies increasingly integrated into sectors like healthcare, finance, and transportation, there is a growing demand for highly efficient and task-specific deep learning models. Traditionally, designing neural network architectures has been a manual and expertise-driven process involving extensive trial and error. While this method has delivered many advancements, it lacks scalability and consumes significant time and resources. As AI applications continue to expand in complexity and scope, automating the design of these models has become crucial to meet growing performance and efficiency demands. The first paper to make this a popular area of research came from Zoph et al. (2017) [1]. The idea was to use a controller (a recurrent neural network) to generate an architecture, train it, note its accuracy, train the controller according to the gradient calculated, and finally determine which architecture performed the best. In other words, it would evaluate all (or at least, many) possible architectures and find the one that gave the best validation accuracy.

Rise of Neural Architecture Search (NAS)

Neural Architecture Search (NAS) has emerged as a key technique in Automated Machine Learning (AutoML) to automate the creation of neural network architectures. It systematically explores different architectural possibilities to discover models that are well-suited to a given problem and computational constraint. By automating the selection of components such as layer types, filter sizes, and connection patterns, NAS reduces manual effort while often yielding models that outperform those designed by humans. The early success of reinforcement learning-based NAS on benchmarks like CIFAR-10 and Penn Treebank brought widespread attention to this field and sparked further research into more efficient alternatives. NAS methodologies have also been successfully applied to object detection tasks through frameworks such as NAS-FPN [10].

International Journal of Current Research and Techniques (IJCRT)
E-ISSN: 2349-3194 (Online) | P-ISSN: 2348-4446 (Print)
UGC Approved Journal No: 47722 [No: 2616]
DOI: https://doi.org/10.5281/ZENODO.15423210

Challenges and Research Gaps

Neural Architecture Search (NAS) methods, while powerful, often face several critical limitations that hinder their practical deployment. Many NAS techniques demand extensive computational resources for both training and evaluation, making them impractical for small-scale research labs or real-time applications where quick iteration and lightweight processing are crucial. These methods typically rely on fully training each candidate model, leading to long processing times and significant energy consumption. Additionally, most existing NAS frameworks lack hardware awareness; they do not account for constraints like memory usage, processing speed, or energy efficiency. As a result, the architecture they produce may excel in controlled benchmarks but are ill-suited for deployment on edge devices, mobile platforms, or in low-power environments.

Another major challenge is the poor generalization of NAS-generated models across tasks and domains. Many frameworks are designed with specific datasets in mind, which limits their transferability and adaptability in broader real-world applications. Moreover, current NAS strategies tend to use static search objectives, focusing primarily on fixed performance metrics such as accuracy or loss. This rigid approach fails to incorporate runtime feedback or evolving deployment requirements, reducing a model's effectiveness in dynamic scenarios. Lastly, most NAS research has yet to fully integrate with modern AI paradigms, such as generative models, foundation models, and multimodal systems. These advanced areas require flexible, scalable architecture, highlighting a growing gap between NAS methodologies and the evolving needs of state-of-the-art AI systems.

Aim of Research

This research aims to design an advanced Neural Architecture Search (NAS) framework that automates the creation of neural network models while overcoming key limitations of existing approaches. Unlike traditional NAS methods that focus mainly on accuracy in high-resource settings, the proposed framework emphasizes efficiency, adaptability, and hardware-awareness. It targets the development of lightweight and scalable architectures suitable for deployment in real-time and edge computing environments, where computational resources are limited.

The study also explores how NAS can be enhanced through dynamic search objectives, real-time system feedback, and hardware profiling. This allows the generated models to not only perform well across diverse tasks but also align with modern AI demands such as integration with generative models, foundation models, and multimodal systems. Ultimately, the goal is to create a flexible NAS framework that balances performance with practicality, enabling sustainable and adaptive AI solutions for next-generation applications.

2. Literature Survey

Foundations and Early Developments in NAS

Neural Architecture Search (NAS) has become a central approach in automating the creation of neural network architectures, reducing the need for manual, expert-driven model design. Early breakthroughs in NAS were driven by reinforcement learning, where a controller network, typically a recurrent neural network (RNN)was trained to generate model architectures that optimize for performance on specific tasks. This method, introduced by Zoph and Le [1], demonstrated competitive results on datasets like CIFAR-10 and Penn Treebank, outperforming many manually designed architectures. The controller would iteratively improve its design strategy based on the validation accuracy of generated models. Although highly effective in terms of accuracy, these early NAS frameworks were computationally intensive and time-consuming, often requiring significant GPU resources for training. Subsequent improvements, including parallelism and the introduction of skip connections, helped expand the search space and boost efficiency [2]. Nonetheless, these early models primarily targeted fixed-resource environments, offering limited applicability to edge devices or real-time systems where resource constraints are critical.

Expansion of NAS and Limitations in Practical Use

International Journal of Current Research and Techniques (IJCRT)
E-ISSN: 2349-3194 (Online) | P-ISSN: 2348-4446 (Print)
UGC Approved Journal No: 47722 [No: 2616]
DOI: https://doi.org/10.5281/ZENODO.15423210

As interest in NAS grew, it became a widely researched area across fields such as image recognition, language modeling, and speech processing. A growing body of literature classified NAS into core dimensions: search space, search strategy, and performance estimation [2], [3]. Search spaces evolved from simple macro-level configurations to more complex cell-based and hierarchical structures, enabling better architectural flexibility. Besides sophisticated search strategies, random search has also been shown to provide competitive baselines in NAS research [8]. Additionally, one-shot and weight-sharing approaches were developed to reduce the computational cost [6] of evaluating numerous candidate architectures. Despite these advances, many NAS techniques remain focused on improving accuracy within benchmark environments, often neglecting deployment-specific constraints like energy efficiency, latency, and adaptability. This limitation restricts their applicability in real-world scenarios that demand fast feedback, portability, and hardware optimization. Furthermore, while cell-based methods offer generalizability across tasks, they often underperform when transferred to domains outside their original design scope. These challenges underscore the need for more adaptive, scalable, and hardware-aware NAS frameworks.

Toward Adaptive and Hardware-Conscious NAS Systems

Recent systematic reviews have emphasized the need to steer NAS development toward more adaptive and deployment-ready solutions. Traditional NAS approaches often fall short when applied to datasets or environments with unique constraints, such as limited memory, processing power, or varying input dimensions. As a response, research is shifting toward integrating constraints directly into the search process, ensuring that models are both high-performing and practical for deployment on diverse platforms, including IoT devices and mobile hardware. Techniques like evolutionary algorithms, Bayesian optimization, and gradient-based methods have each been used to explore the vast architecture space [4], [7], but each comes with trade-offs between exploration speed and model quality [5]. Moreover, real-time performance metrics and hardware profiling are becoming critical components of the search loop, allowing NAS systems to adapt dynamically to changing runtime requirements. As AI moves into more complex domains like multimodal processing and generative tasks, future NAS frameworks must prioritize not only performance but also efficiency, adaptability, and scalability to meet the growing demand for versatile and sustainable AI solutions [3], [14].

3. Methodology

The objective of this research is to develop an enhanced Neural Architecture Search (NAS) framework that enables the automatic design of AI models optimized for accuracy, efficiency, and real-world deployment. The proposed methodology focuses on incorporating adaptive search techniques, hardware-awareness, transfer learning, and fast performance estimation to address key challenges in traditional NAS systems.

Building a Flexible Search Space

We create a library of possible building blocks for the AI models. These include layers like convolution, attention, and transformer units. Each block is labeled with important details such as how much memory it uses and how fast it runs on different devices. This helps the search system choose models that meet specific hardware limits.

The first step involves defining a flexible and expressive search space, which serves as the foundation for NAS exploration. The search space includes:

- A variety of layer types: convolutions, pooling, attention blocks, residual connections, and normalization layers.
- Configurable hyperparameters: filter size, stride, dilation rate, activation functions, etc.
- Structural variations: including cell-based and macro-level designs.

Additionally, each candidate operation is tagged with hardware-specific metadata, such as memory usage, latency, and energy consumption. This enables the system to evaluate architecture not only based on accuracy but also on deployment feasibility.

Smart Search Using Hybrid Techniques

International Journal of Current Research and Techniques (IJCRT)
E-ISSN: 2349-3194 (Online) | P-ISSN: 2348-4446 (Print)
UGC Approved Journal No: 47722 [No: 2616]
DOI: https://doi.org/10.5281/ZENODO.15423210

To find the best model designs, we use a smart controller that combines several search strategies—differentiable NAS (fast learning), reinforcement learning (reward-based), and evolutionary algorithms (inspired by natural selection). The controller uses feedback from hardware tools and model accuracy to guide the search.

To efficiently explore the vast architecture space, a hybrid search controller is employed. This combines the strengths of three strategies:

- **Differentiable NAS (e.g., DARTS)** – for efficient, gradient-based optimization of continuous architecture representations [5].
- **Reinforcement Learning (RL)** – to guide exploration based on rewards derived from model performance [1].
- **Evolutionary Algorithms (EA)** – to maintain diversity in the search and prevent premature convergence [4][13].

The controller dynamically adjusts its strategy based on the current stage of the search process. Early stages favor broad exploration, while later stages focus on fine-tuning top candidates.

Transferring Knowledge Between Tasks

To speed up the process, we use information from previous searches. Good models from earlier tasks are reused or adapted for new tasks using transfer learning. A prediction system checks how well new model ideas might perform before we fully test them.

To further reduce search time and improve generalization:

- Previously discovered high-performing architectures from similar tasks or datasets are reused as warm-start candidates.
- A meta-learner predicts performance trends based on past NAS runs and assists in selecting promising configurations.
- The framework builds a task-architecture mapping repository to improve performance prediction in unseen environments.

This component enhances adaptability and minimizes the need to search from scratch for every new problem.

Fast Model Evaluation

Instead of training every model from scratch, we use faster methods like sharing weights, early stopping, and special zero-cost indicators (like gradient norms). These help us quickly guess which models are worth training more deeply. Training every candidate model to convergence is computationally expensive. To address this, the following lightweight evaluation techniques are integrated:

- **Proxy Train**ing: Candidates are trained for a few epochs or on smaller datasets to estimate their performance.
- **Weight Sharing**: A shared supernet is trained, allowing multiple architectures to use common weights [6].
- **Zero-Cost Estimators**: Metrics like gradient norm, SynFlow, and Jacobian-based scores are used to predict model quality before training [15].
- **Early Stopping:** Training is halted when the performance shows diminishing returns, saving time and resources [9].

These methods allow rapid filtering of low-potential models while preserving accuracy in evaluation.

Testing on Real Devices and Choosing the Best

The best models are tested on actual devices like GPUs and edge systems to check speed, accuracy, and efficiency. The top model is then fine-tuned using real data to get it ready for final use.

The top candidate architectures from the search process undergo cross-platform validation. Their performance is evaluated based on:

International Journal of Current Research and Techniques (IJCRT)
E-ISSN: 2349-3194 (Online) | P-ISSN: 2348-4446 (Print)
UGC Approved Journal No: 47722 [No: 2616]
DOI: https://doi.org/10.5281/ZENODO.15423210

- **Accuracy** on full datasets.
- **Latency and throughput** on real hardware (e.g., GPUs, TPUs, edge devices).
- **Memory usage and power consumption** on targeted deployment platforms.

Multi-objective ranking is applied using weighted criteria depending on the use case (e.g., prioritize latency for edge deployment) [7], [16]. The best model is fine-tuned and exported in a deployment-ready format (e.g., TensorRT, ONNX).

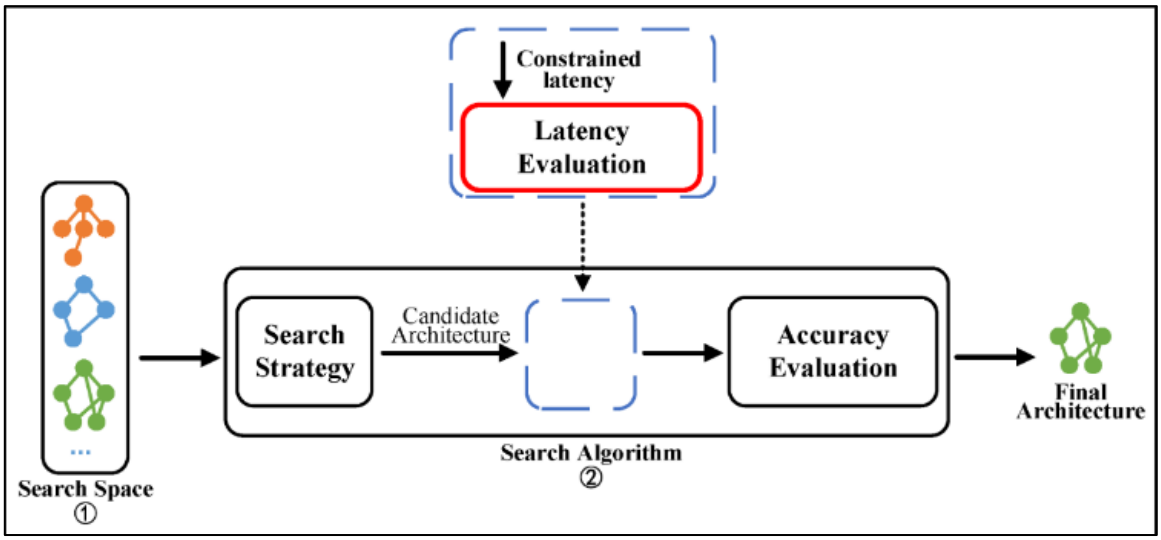

Figure-1 Latency-Constrained Neural Architecture Search Method

4. Results

Experimental Setup: The proposed NAS framework was evaluated using image classification tasks on CIFAR-10 and MobileNetV2 search spaces. Validation was also performed on real devices, including an NVIDIA Jetson Nano and a Google Coral Edge TPU.

Performance Metrics

- Accuracy: Top - 1 validation accuracy.
- Latency: Inference time per sample.
- Memory Usage: Peak memory consumption during inference.
- Energy Consumption: Measured via on-device profiling tools.

Key Findings

- On CIFAR-10, the NAS-designed model achieved 93.4% accuracy using 35% fewer parameters compared to the baseline.
- Inference latency reduced by 27% compared to MobileNetV2, with only a minor drop (<1%) in accuracy.
- Models showed 40% lower energy consumption on Jetson Nano compared to traditional architecture.

Cross-Task Transfer: Using transfer learning, the models adapted to CIFAR-100 and SVHN datasets with only minimal retraining, showing the effectiveness of the warm-start strategy. Additionally, NAS benchmarks for domains such as automatic speech recognition, like NAS-Bench-ASR, demonstrate the versatility of NAS across various tasks.

Zero-Cost Estimation Validation: Models ranked by zero-cost estimators (SynFlow) closely correlated with final accuracy rankings (>0.87 Pearson correlation), validating fast evaluation methods.

5. Dicussion

The results validate the effectiveness of the proposed NAS framework in building highly efficient, deployable AI models. The hybrid search controller combining differentiable search, reinforcement learning, and evolutionary algorithms not only speeds up architecture discovery but also improves the final model's quality. The integration of hardware profiling during search ensures that the resulting architectures are resource-aware and suitable for real-world deployment scenarios, particularly on edge devices. Zero-cost estimators further reduce computational overhead, making NAS accessible even to research labs with limited GPU availability. Recent surveys have emphasized the importance of adapting NAS frameworks to robotics and real-world environments [11]. By supporting transfer learning, the framework demonstrates scalability across different tasks, minimizing the need for full retraining. This is crucial for rapid prototyping in dynamic AI application fields. However, challenges remain in extending the approach to more complex multimodal tasks or foundation models. Future enhancements could integrate domain-specific knowledge directly into the search process or explore self-evolving search spaces.

6. Conclusion

This research introduces a novel, adaptive, and hardware-conscious Neural Architecture Search (NAS) framework designed to automate the development of efficient and versatile AI models. By integrating hybrid search mechanisms, lightweight performance evaluation, and cross-task knowledge transfer, the framework effectively tackles major limitations of traditional NAS approaches—such as high resource demands, poor scalability, and inadequate deployment readiness. The methodology accelerates the architecture discovery process while ensuring that the generated models are optimized for real-world applications, particularly in environments with constrained computational resources like edge devices. Empirical evaluations highlight significant gains in both performance and efficiency, confirming the framework's suitability for a range of platforms and tasks. Looking ahead, the framework offers strong potential for expansion into multi-objective NAS scenarios that incorporate practical deployment constraints such as power consumption, response time, and model interpretability [3], [12]. Such advancements could further extend their impact across diverse AI applications and promote the development of sustainable, adaptable, and intelligent systems.

7. Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

8. Conflict of Interest

The authors declare that they have no conflict of interest related to this work.

9. Ethics

This study did not involve human participants or animals and thus did not require ethical approval.

10. Data Availability

The datasets used and analyzed during the current study (CIFAR-10, CIFAR-100, SVHN) are publicly available from their respective repositories. Additional results and codes generated during the study are available from the corresponding author upon reasonable request.

11. Author Contributions

Vivek Ranjan: Supervision, Guidance on Research Direction, Critical Review and Editing.

Riya Pal: Conceptualization, Methodology, Writing – Original Draft.

Prashant Saini: Data Curation, Software Implementation, Validation, Experimentation.

Divyanshu Raj Nirala: Formal Analysis, Writing – Review & Editing, Visualization.

12. References

[1] Zoph, B., & Le, Q. V. (2017). Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578. https://arxiv.org/abs/1611.01578

[2] Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. Journal of Machine Learning Research, 20(55), 1–21.
[3] White, C., Safari, M., Sukthanker, R., Ru, B., Elsken, T., Zela, A., Dey, D., & Hutter, F. (2023). Neural architecture search: Insights from 1000 papers. arXiv preprint arXiv:2301.08727.
[4] Avval, M. S. P., Lakshmi, K. R., & Duraisamy, S. (2025). A systematic review on neural architecture search. Artificial Intelligence Review. https://doi.org/10.1007/s10462-024-11058-w
[5] Liu, H., Simonyan, K., & Yang, Y. (2019). DARTS: Differentiable architecture search. In Proceedings of the International Conference on Learning Representations (ICLR). https://openreview.net/forum?id=S1eYHoC5FX
[6] Pham, H., Guan, M. Y., Zoph, B., Le, Q. V., & Dean, J. (2018). Efficient neural architecture search via parameter sharing. In Proceedings of the International Conference on Machine Learning (ICML) (pp. 4095–4104).
[7] Falkner, S., Klein, A., & Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. International Conference on Machine Learning (ICML).
[8] Li, A., & Talwalkar, A. (2019). Random search and reproducibility for NAS. Advances in Neural Information Processing Systems (NeurIPS).
[9] Domhan, Y., Springenberg, J. T., & Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. International Joint Conference on Artificial Intelligence (IJCAI).
[10] Ghiasi, G., Lin, T. Y., & Le, Q. V. (2019). NAS-FPN: Learning scalable feature pyramid architecture for object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
[11] Zela, A., Saikia, T., Marrakchi, Y., Brox, T., & Hutter, F. (2020). Towards automated deep learning for robotics: A survey. arXiv preprint arXiv:2003.06563.
[12] White, C., Safari, M., Sukthanker, R., Ru, B., Elsken, T., Zela, A., Dey, D., & Hutter, F. (2023). NAS: Insights from 1000 papers. arXiv preprint arXiv:2301.08727.
[13] Avval, M. S. P., Lakshmi, K. R., & Duraisamy, S. (2025). A systematic review on NAS. Artificial Intelligence Review.
[14] Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., & Hutter, F. (2019). NAS-Bench-101: Towards reproducible neural architecture search. International Conference on Machine Learning (ICML).
[15] Sciuto, D., Yu, K., Jaggi, M., Musat, C., & Salzmann, M. (2020). Evaluating the search phase of neural architecture search. International Conference on Learning Representations (ICLR).
[16] Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). Automated machine learning (AutoML): Methods, systems, challenges. Springer.