MicroNAS: Zero-Shot Neural Architecture Search for MCUs

Ye Qiao, Haocheng Xu, Yifan Zhang, Sitao Huang
Department of Electrical Engineering and Computer Science
University of California, Irvine, Irvine, California, USA
{yeq6, haochx5, yifanz58, sitaoh}@uci.edu

***Abstract*—Neural architecture search (NAS) effectively discovers new convolutional neural network (CNN) architectures, particularly for accuracy optimization. However, prior approaches often require resource-intensive training on super networks or extensive architecture evaluations, limiting practical applications. To address these challenges, we propose MicroNAS, a hardware-aware zero-shot NAS framework designed for microcontroller units (MCUs) in edge computing. MicroNAS considers target hardware optimality during the search, utilizing specialized performance indicators to identify optimal neural architectures without heavy computational costs. Compared to previous works, MicroNAS achieves up to $1104\times$ improvement in search efficiency and discovers models with over $3.23\times$ faster MCU inference while maintaining similar accuracy.**

I. Introduction

Manually designing high-accuracy or computationally efficient convolutional neural network (CNN) topologies in computer vision, speech recognition, and object detection demands significant amount of time, expertise, and resources. These models are often too large for deployment on edge devices like microcontroller units (MCUs) due to resource limitations. To automate model design within constraints, neural architecture search (NAS) is essential [1]. While NAS has been evolving, most approaches still face challenges in time-consuming training and evaluation [2]. In this context, we introduce MicroNAS, an efficient zero-shot NAS framework tailored for MCUs. MicroNAS enables effective architecture search for edge devices without excessive computational costs, facilitating practical NAS deployment in edge computing.

The main contributions of this work are: **MicroNAS Framework:** We introduce MicroNAS, a novel zero-shot NAS framework for optimal CNN architectures in MCU-based inference. **Hybrid Objective Function:** Our proposed objective function, combining neural tangent kernel spectrum, linear region count, and hardware proxies, significantly enhances NAS quality for MCUs. **Hardware-Aware Pruning-Based Search Algorithm:** We propose an innovative pruning-based search algorithm to improve search efficiency under resource constraints. **Analysis Results:** Using trainless proxies on the NAS-Bench-201 space, we achieve up to $1104\times$ improvement in search efficiency compared to prior works, discovering models with over $3.23\times$ faster MCU inference while maintaining similar accuracy. This work marks a substantial stride toward efficient NAS solutions for edge devices.

II. Proposed MicroNAS Framework

To eliminate the need for training and evaluation during the architecture search process, we employ key indicators that capture the trainability [3], expressivity [4], and hardware performance of neural networks. We incorporate multiple zero-cost proxies and generalize the search to discover the optimal CNN cell structure for MCUs. A cell-based search space defines each architecture as a directed acyclic graph (DAG), with nodes representing feature maps and edges corresponding to operations as shown in Fig. 1.

A. Performance Indicators

1) Spectrum of Neural Tangent Kernel: The neural tangent kernel (NTK) is a mathematical construct used to analyze the behavior and properties of neural networks [5]. Indeed, the trainability of a neural network, which refers to its convergence and generalization ability, is a crucial aspect of zero-shot NAS. The NTK's condition number relies on a single mini-batch, and the chosen batch size affects NTK spectrum consistency, influencing search outcomes. Investigating Kendall-τ correlation against logarithmic batch size scales, Fig. 2b reveals an optimal batch size range of 16 to 32. Increasing beyond 32 to 128 doesn't notably alter Kendall-τ correlation but significantly escalates search costs. Our experiments empirically evaluate and adopt a batch size of 32 for optimal search.

2) Linear Region Count: In our study, we assess the expressivity of a simple CNN with each layer containing a single convolutional operator followed by the ReLU activation function. The ReLU's piecewise linearity allows the network's input space to be divided into distinct linear regions (LR) [6]. Each LR is associated with a set of affine parameters, and the network's expressivity is determined by the number of these linear regions it can separate.

B. Hardware Indicators

The hardware-aware search process focuses on low-power edge microcontroller units, emphasizing effective management of computational costs, processing latency, and memory usage. We prioritize inference latency as it directly influences the system's end-to-end processing time in real world. To take inference latency into account, we integrate floating-point operations (FLOPs) estimation (F) and hardware latency modeling (L) into the architecture search for sampled models. Our approach includes tunable weight factors for precise control over the contributions of F and L during the search.

1) The Number of Floating-Point Operations (FLOPs): FLOPs count is a crucial indicator of deep learning model complexity, reflecting computing time in the target environment regardless of hardware specialization. Our estimation considers various layer operations. While our experiments

979-8-3503-4859-0/DATE24

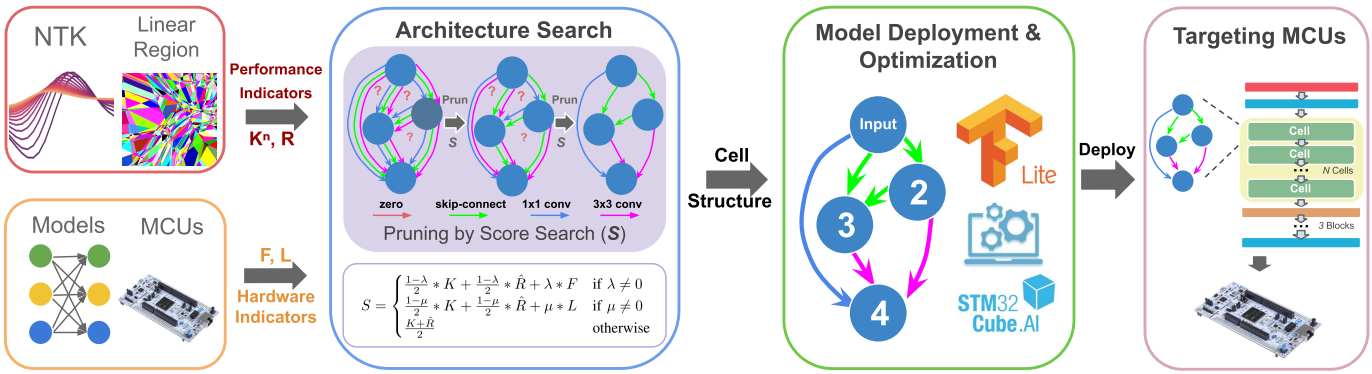
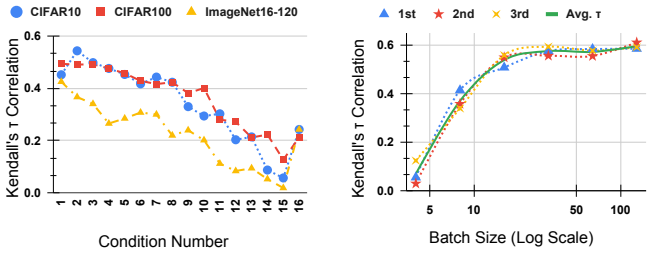


Fig. 1: Overview of Proposed MicroNAS Workflow



(a) Kendell's τ vs. K_i (b) Kendall-τ vs. Batch Size

Fig. 2: Kendall-τ vs Batch Size, Condition Number K_i

show a positive correlation between FLOPs count and model accuracy, it's important to note that FLOPs alone don't represent absolute accuracy or real-world hardware performance due to redundancy and topological differences in convolutional neural networks. Therefore, we introduce estimated model latency as an additional hardware indicator for a more comprehensive model assessment.

2) Estimated Latency: To address this, we developed a custom latency estimator to accurately model inference latency on the target MCUs based on provided cell structures. The approach involves profiling each operation individually within the search space and generating a reference lookup table. Specific details of the secondary stage of the model structure, including the number of cells and input/output channels for each cell, are gathered. Finally, constant hardware latency overhead is profiled and incorporated into the overall latency estimation. Our latency model was validated as accurate, reliable, and simple in further experiments with the integration of our hardware-aware pruning-based search.

TABLE I: Results on CIFAR-10

NAS Frameworks	FLOPs (M)	Params (M)	Speedup	Search Time	ACC
μNAS [2]	-	0.014	-	552	86.49
TE-NAS [3]	188.66	1.317	1	0.43	93.78
Ours	**51.04**	**0.372**	**3.23×**	**0.43**	**93.88**

III. Experiment and Results Analysis

We evaluate our MicroNAS on NAS-Bench-201 [7], and test our search results on a STM32 NUCLEO-F746ZG board.

In the search, MicroNAS adapts FLOPs and latency indicator weights, consistently discovering highly efficient models across various constraints with minimal performance degradation. Even without hardware constraints, our baseline result surpasses state-of-the-art TE-NAS [3]. Our hardware-aware strategy provides a latency advantage of 1.59× to 3.23× with negligible performance trade-offs. Notably, our latency-guided search outperforms TE-NAS [3], halving inference time on the target MCU. Compared to μNAS [2], another NAS for edge hardware, MicroNAS finds significantly better-performing models with approximately 1104× faster in search time (reported in GPU hours) and 6.2% better performance. The latency-guided search demonstrates superior and more balanced performance than the FLOPs-guided search, attributed to MCU-specific bias in our latency modeling. Both approaches are valuable for MCU platforms, with fine-grained latency estimation resulting in superior search outcomes.

IV. Conclusion

This paper introduces MicroNAS, a framework that combines neural network analysis and targets MCU hardware constraints for identifying optimal architectures with minimal performance degradation and low training and evaluation costs. The proposed MCU latency estimation model accurately predicts real-world inference latency on MCUs and has potential applicability to other edge devices. MicroNAS achieves a substantial search efficiency improvement, surpassing previous works by 1104×, discovering models with more than 3.23× faster MCU inference while maintaining comparable accuracy. Future experiments will incorporate peak memory usage modeling of MCUs to guide the search and enhance the MicroNAS framework further.

References

[1] J. Lin, W.-M. Chen, J. Cohn, C. Gan, and S. Han, "MCUNet: Tiny deep learning on iot devices," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[2] E. Liberis, L. Dudziak, and N. D. Lane, "μNAS: Constrained Neural Architecture Search for Microcontrollers," in *Proceedings of the 1st Workshop on Machine Learning and Systems*, ser. EuroMLSys '21, 2021.

[3] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on ImageNet in four GPU hours: A theoretically inspired perspective," *arXiv preprint arXiv:2102.11535*, 2021.

[4] M. Lin, P. Wang, Z. Sun, H. Chen, X. Sun, Q. Qian, H. Li, and R. Jin, "Zen-nas: A zero-shot nas for high-performance deep image recognition," 2021.

[5] L. Xiao, J. Pennington, and S. Schoenholz, "Disentangling trainability and generalization in deep neural networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 462–10 472.

[6] H. Xiong, L. Huang, M. Yu, L. Liu, F. Zhu, and L. Shao, "On the number of linear regions of convolutional neural networks," 2020.

[7] X. Dong and Y. Yang, "Nas-bench-201: Extending the scope of reproducible neural architecture search," in *International Conference on Learning Representations (ICLR)*, 2020.