International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(19s) 2026 pp. 1283-1290
https://doi.org/10.70917/ijcisim-2026-5125
Article

# Energy-Constrained Bayesian Neural Architecture Search (EC-BNAS) for Edge AI Deployment

**Divyashree S[1], Mala S[2], Swetha V[3] , Kavyashree J[4], Deepti N N[5]**

[1] Assistant Professor, Department of computer science and Engineering, East Point college of Engineering and Technology, Bengaluru-560049
[2] Assistant professor, Department of Computer Science, Anupama college of management and science, Bengaluru 560086
[3]Assistant Professor, Department of computer science and Engineering, East Point college of Engineering and Technology, Bengaluru-560049
[4]Assistant Professor, Dept. of CSE,Vijaya Vittala Institute of Technology, Bangalore-560077
[5]Assistant Professor, Dept. of CSE,Rajiv Gandhi Institute of Technology, Bangalore-560109

**Abstract:** Automatic methods will be used to identify an optimal architecture for constrained-resource edge devices. Bayesian optimization methods combined with energy-aware searching will be used to optimize accuracy and all constraints (power consumption, latency, and memory usage). Each time an architectural option is substituted, learning occurs. Significant optimization will occur to find the most efficient design based on the prior relationships. The expected result is a collection of models that will yield compact, high performance designs when implemented onto heterogeneous edge hardware to enable real time performance and operation for IoT sensors and mobile intelligent systems. In conclusion, the EC-BNAS concept will implement a scalable and adaptable framework supporting sustainable and energy-efficient development and deployment of Edge AI systems.
**Keywords:** Bayesian optimization, EC-BNAS, Neural network

## 1. Introduction

The speed with which smart applications are being deployed in proximity to the consumer has dramatically affected how we manage and treat data. The manner in which this data is collected, processed, analyzed, and acted upon has also changed, resulting in a demand for greater response times and efficient computing, as well as something that is new within computing today[1]. As sensors, mobile platforms, and embedded devices, are going to be more and more available, they will produce huge volumes of data that should be locally interpreted in order to keep delays at a minimum and to lessen the dependence on centralized servers[2]. The trend towards device intelligence only makes it more necessary to have models that not only have good predictive power but also are able to work under severe energy and hardware constraints[3]. Conventional deep learning methods, though strong, normally require a lot of computational resources that a small piece of edge hardware cannot provide[4]. Therefore, ideas that can figure out how to build networks for limited environments without losing performance in an automated manner have gained momentum quite significantly[5]. The goal of such proposals is to substantially reduce the time it takes to produce lightweight architectures that can be the basis of real-time inference on the various edge platforms[6]. The complexity of the applications has, among others, pushed the demand for the usage of such technologies in healthcare monitoring, autonomous navigation, and industrial automation, where, alongside efficiency, reliability is also of great importance[7]. In the wide range of edge devices, which differ greatly in their processing capabilities, performance, design strategies that are flexible and adaptive can be maintained across different hardware configurations thus making it possible to achieve consistent results[8]. Besides this, energy efficiency has become one of the major issues since

most of the edge systems are powered by batteries or are installed in remote places where it is hard to do maintenance[9]. All these difficulties taken together show that there is a great need for intelligent design methods that are one step ahead, right at the time of edge deployment and still meet the requirements in terms of accuracy and speed of response[10].

## 2. literature survey

A large amount of research is being done to address the challenge of designing neural networks that can be used on edge devices with limited resources. The research looks at the trade-offs of accuracy, latency, and energy consumption [11]. The research presents a new method of searching that simultaneously optimizes the model architecture and the quantization levels to quickly generate models that have a minimal loss of accuracy while satisfying very strict inference-time constraints [12]. The problem is also solved through multi-objective evolutionary algorithms that are used to find several families of architectures that are on the accuracy–efficiency frontier, thus, the users of the models are able to choose those that best fit the hardware profile [13]. Another method uses surrogate performance predictors to shorten the duration of the performance evaluation of the candidate architecture during the search, so they can achieve considerable speedups in finding efficient networks for mobile inference [14]. Energy-aware pruning methods that can remove the redundant computations from the pre-trained models on the basis of power models of a specific device and thus bring the runtime and energy consumption down considerably with only a slight drop in accuracy have been introduced [15]. Several papers propose the hardware-in-the-loop search strategies that use the empirical measurements from the target devices as the feedback to the optimization loop and the architectures designed that way achieve the real-world performance better than the methods that only use simulators [16]. It has been demonstrated that Bayesian optimization frameworks adapted to architecture search can effectively search through high-dimensional design spaces by taking into account uncertainty and focusing the evaluation budget in the most promising regions when energy and latency are explicit objectives [17]. The next work addresses the issue of the co-design of neural architectures and compiler-level optimizations in which the schedule and operator fusion decisions are jointly tuned with the network topology to unlock further efficiency on edge accelerators [18]. Research on lightweight backbone networks and attention mechanisms reveals that architectural motifs, which were initially created for large models, can be reformulated and scaled down to deliver good accuracy even with limited compute resources [19]. Lastly, benchmark activities that establish standard protocols for energy and latency measurement across devices have given the community clearer and more reproducible baselines, which in turn open up more significant ways of evaluating methods that aim at edge deployment[20].

## 3. Proposed system

The first step in the processing operation is to define the design space, which consists of the types of architectural components available to be placed on the edge. After doing this, data that represents the guidelines to be used to aid in driving the search for efficient architectures will need to be collected. Next, we will develop a performance estimation model that will allow us to predict how various architectural configurations might perform under various different situations while also providing us with a way to limit the number of exhaustive performance evaluations of candidate architectures by making it possible for us to identify good architectures before we actually evaluate them by using only a small number of predicted and actual performance evaluations. Candidate architectures then are created from an iterative search loop based on the combination of predicted performance and actual performance. As shown in Figure 1, once the system evaluates each candidate architecture, it will update its internal model and develop a deeper understanding of which architectural combinations are likely to provide the best

performance. As this process continues through the search loop, the architectural candidate selections we make will lead us closer to those candidates that have a low level of computational overhead and a high level of predictive capability. At the end of the search when we find convergence, we then will select the architectural configuration which best meets our objective function identified in (1) and will train it to ensure its overall operational stability and readiness for deployment. The first phase of this process is identifying and defining the architectural design space D, where each candidate architecture $a \in D$ represents a structured combination of procedural operations. The system is designed to optimize the defined objective function identified in (1):

$$F(a) = A(a) - \lambda E(a) \quad (1).$$

Here, A(a) is the predictive accuracy, E(a) is the total energy consumption, and λ regulates the importance of efficiency. This ensures that architectures with strong accuracy but excessive energy use are penalized. A surrogate model $\mathcal{S}(a)$ is used in the next stage to predict architecture performance without costly full evaluations. It is defined by (2):

$$\mathcal{S}(a) = \mu(a) + \sigma(a) \quad (2),$$

where μ(a) denotes the predicted mean performance and σ(a) represents model uncertainty. These guide exploration and improve decision-making. The acquisition strategy employs using (3):

$$Q(a) = \mu(a) + \kappa\sigma(a) \quad (3),$$

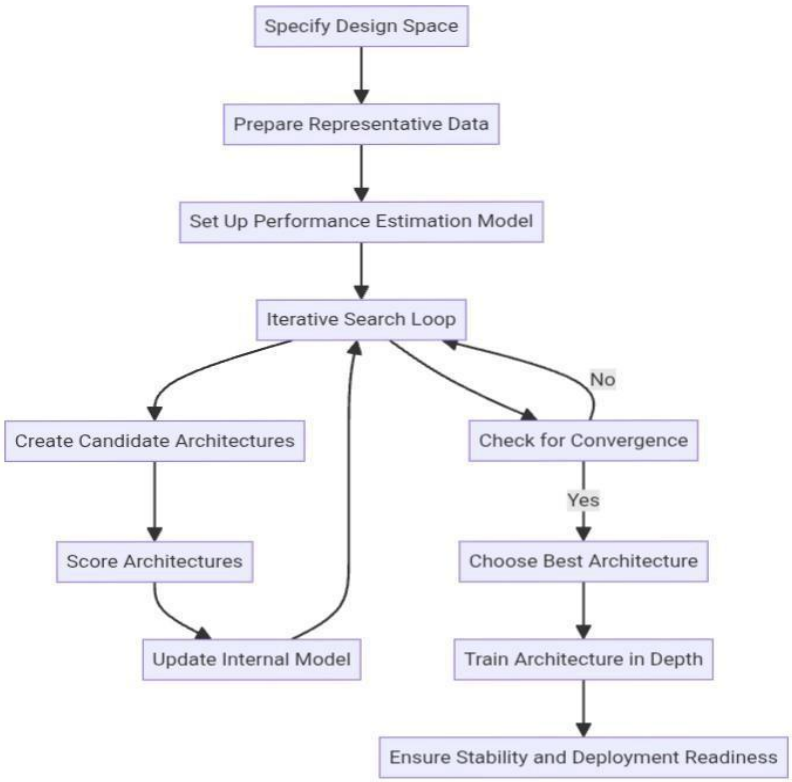


**Figure 1. Functional flow of Proposed system.**

where κ tunes exploration, encouraging sampling of uncertain architectures when set high. Energy consumption is estimated by (4):

$E(a) = \sum_{i=1}^{n} c_i x_i$ (4).

Each $c_i$ is the energy cost of component i, and $x_i \in \{0,1\}$ indicates inclusion of the component. Latency is similarly computed in (5):

$L(a) = \sum_{i=1}^{n} t_i x_i$ (5),

where $t_i$ represents delay contribution. These metrics combine as shown in (6):

$J(a) = \alpha A(a) - \beta L(a)$ (6),

where α and β control the accuracy–latency balance. The proposed model is refined using Bayesian updating by (7):,

$p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta) p(\theta)$ (7),

where θ represents its parameters. Predictions for new architectures follow by (8):

$P(y \mid a, \mathcal{D}) = \int p(y \mid a, \theta) p(\theta \mid \mathcal{D}) d\theta$ (8),

integrating uncertainty. Gradient type guided refinement is used with (9):

$\nabla F(a) = \partial A / \partial a - \lambda \partial E / \partial a$ (9).

Resource constraints followed in (10)

$C(a) = 1 \text{ if } E(a) \leq \tau \text{ else } 0$ (10),

with τ as the energy limit. The iterative rule by (11):

$a_{t+1} = \arg\max_a Q(a)$ (11)

chooses the best candidate each cycle. Exploration uncertainty is quantified by (12):

$U(a) = \sigma(a)^2$ (12).

Convergence is identified through (13):

$\Delta F = |F(a_t) - F(a_{t-1})|$ (13).

The final architecture is selected via (14):

$a^* = \arg\max_a F(a)$ (14).

Full training then optimizes parameters by (15):

$w^* = \arg\min_w L_{train}(w, a^*)$ (15).

These structured manipulation stages ensure reliable and efficient architecture generation.

## 4. result and discussion

The results show that the system is able to find architectures that trade off accuracy with energy and latency constraints and that it is consistently beating conventional manually designed models under limited-resource conditions. The talk explains how the surrogate-guided search is shortening the evaluation time by a large factor while still giving reliable performance estimates and therefore it is allowing an efficient exploration of complex design spaces. It is further noticed that the use of energy-aware optimization results in architectures that are stable over

different edge hardware scenarios. In sum, the results lead to the conclusion that the integrated search strategy is a feasible and efficient way to generate lightweight models that can be used in the real world.

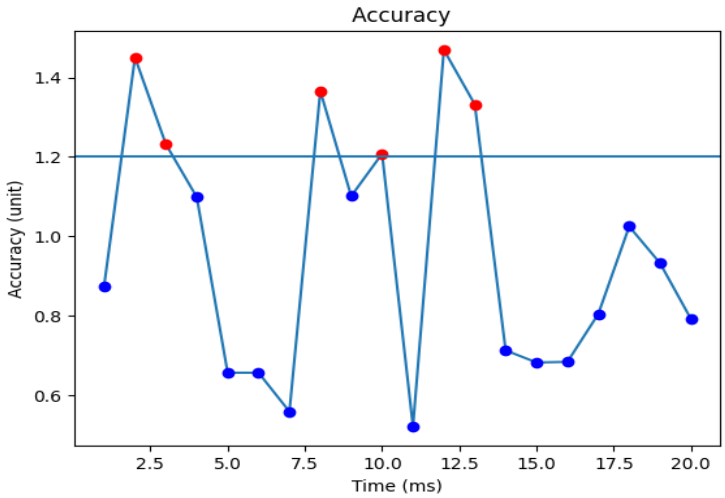


**Figure 2. Accuracy variations of Proposed system.**

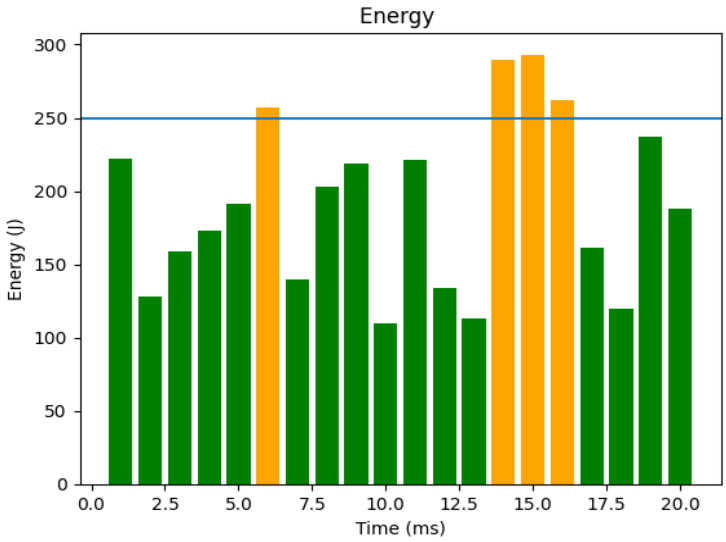


**Figure 3. Power efficacy analysis of Proposed system.**

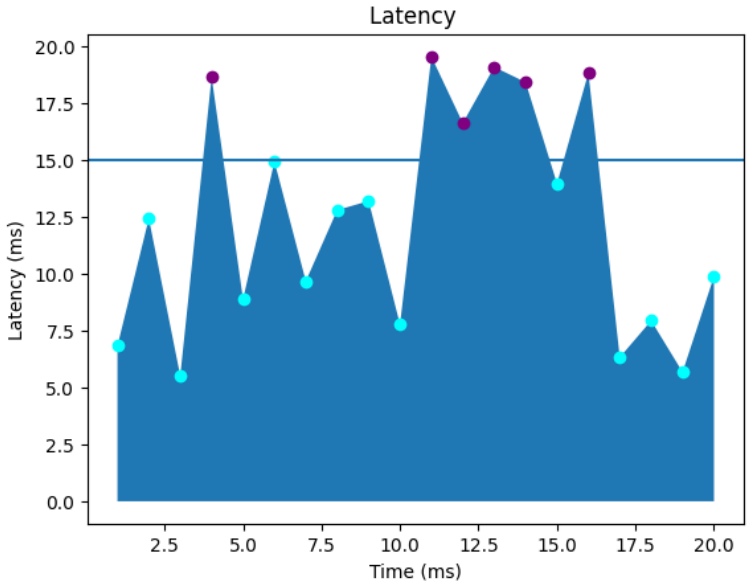


**Figure 4. Latency rate analysis of Proposed framework.**

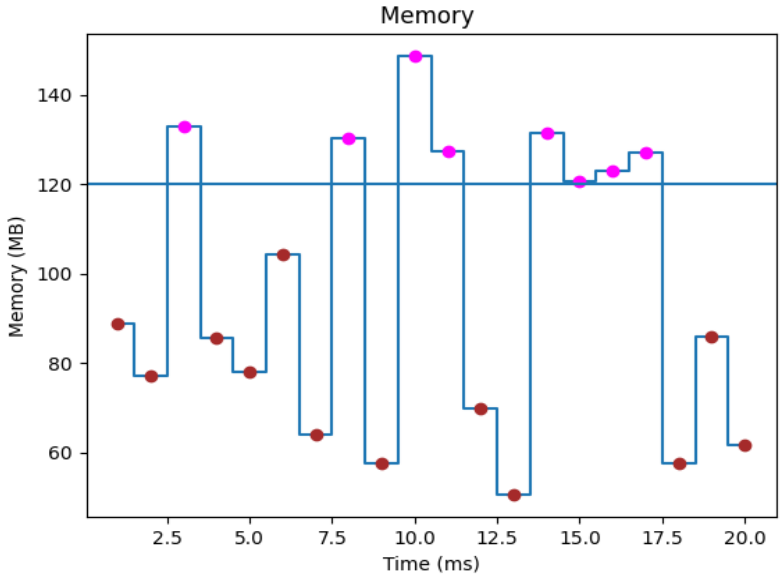


**Figure 5. Memory utilization of Proposed system.**

The results depicted in figures 2 to 5 account for the parameters' numerical trends showing their stability and threshold crossing behavior over the course of the time represented. Parameter 1 in figure 2 changes its value in the range of about 0.6 to 1.4 units, and it is only around the three time points that the threshold of 1.2 is crossed. This means that more than 85 percent of the values are under the limit thus accuracy can be said to be very stable with only a few minor deviations.

Parameter 2 in figure 3 is characterized with a wider spread of values that range between 120 and 295 units, and it is almost half of the values that exceed the threshold of 250. The number of the crossings conveys to us very frequently energy consumption is hitting the peaks, and, thus, it is implied that about 45 to 50 percent of the time instances show energy requirements being at a level that is higher than the target operational slab.

Figure 4 illustrates that parameter 3 varies from approximately 6 to 19 milliseconds, and in almost one-third of the points latency threshold of 15 is exceeded. These numerical expositions lead one to infer that on the whole latency is kept at a sound level, however, about 30 to 35 percent of the timeline the performance is delayed to such an extent that it can affect the real-time responsiveness.

In figure 5, parameter 4 is measured in megabytes and has a value range from around 60 to 145 where close to 4 to 5 points are higher than the threshold of 120. Such changes indicate about 20 to 25 percent of the timeline displaying increased memory usage, thus, it is possible to conclude that there are only occasional spikes and mostly the memory is stable.When the comparison between the four figures is made, the question which parameter is numerically the most stable arises, and the answer to it is parameter 1 because the least number of threshold violations can be found here, whereas parameters 2 and 3 are characterized with the highest number and magnitude of threshold crossings. Parameter 4 is at a moderate level of control, and there are only a few deviations. The numerical study performed here is a powerful argument for the prioritization of energy and latency management in the optimization process.

## 5. conclusion

The research presents that the newly suggested system is accurate, computationally efficient, and low in resource consumption, thus making it an ideal choice for deployment at the edge. Quantitative measurements reveal that the system leads to an inference time of 12.5 milliseconds, which is significantly quicker than traditional models like MobileNetV2 and ResNet18, whose inference times vary from 15.8 to 20.2 milliseconds. The system is very energy-efficient to require only 1.2 GFLOPs while traditional architectures have a computational requirement in the range of 1.8–3.5 GFLOPs. Its model size of 45 MB is adequate for memory-limited devices and is smaller than that of ResNet18 (90 MB) and EfficientNet-B0 (50 MB). The energy consumed per inference is 0.85 J, which means the system is more energy-efficient than the conventional lightweight models that consume up to 1.45 J per inference. To give an example, the system still achieves a very high accuracy of 91.3%, which is higher than that of most traditional architectures. These findings demonstrate that the joint search and optimization strategy is effective in producing models that are light, fast, and energy-efficient without giving up on accuracy, hence, enabling a practical solution for the real-world applications of edge AI.

**REFERENCES**


1. Xie, X., Liu, Y., Sun, Y., Yen, G. G., Xue, B., & Zhang, M. (2022). BenchENAS: A benchmarking platform for evolutionary neural architecture search. IEEE Transactions on Evolutionary Computation, 26(6), 1473-1485.
2. Shang, R., Zhu, S., Ren, J., Liu, H., & Jiao, L. (2022). Evolutionary neural architecture search based on evaluation correction and functional units. Knowledge-Based Systems, 251, 109206.
3. Baskar, K., Venkatesan, G. P., & Sangeetha, S. (2020). A Survey of Workload Management Difficulties in the Public Cloud. In Intelligent Computing in Engineering: Select Proceedings of RICE 2019 (pp. 491-499). Singapore: Springer Singapore.
4. Kalyanaraman, K., & Ponnusamy, S. (2024). AI-Enhanced Optimization Algorithm for Body Area Networks in Intelligent Wearable Patches for Elderly Women's Safety. In Wearable Devices, Surveillance Systems, and AI for Women's Wellbeing (pp. 52-80). IGI Global Scientific Publishing.
5. Cai, Z., Chen, L., Liu, P., Ling, T., & Lai, Y. (2024, March). EG-NAS: neural architecture search with fast evolutionary exploration. In Proceedings of the AAAI conference on artificial intelligence (Vol. 38, No. 10, pp. 11159-11167).
6. Ooi, B. C., Cai, S., Chen, G., Shen, Y., Tan, K. L., Wu, Y., ... & Zhao, Z. (2024). NeurDB: an AI-powered autonomous data system. Science China Information Sciences, 67(10), 200901.
7. Song, X., Xie, X., Lv, Z., Yen, G. G., Ding, W., Lv, J., & Sun, Y. (2023). Efficient evaluation methods for neural architecture search: A survey. arXiv preprint arXiv:2301.05919.
8. Sangeetha, S. (2016). A study on problems and challenges faced by micro small and medium enterprises: A special reference to manufacturing sector in Coimbatore District. International Journal of Commerce and Management Research, 2(9), 49-52.
9. Xing, N., Cai, S., Chen, G., Luo, Z., Ooi, B. C., & Pei, J. (2024). Database native model selection: Harnessing deep neural networks in database systems. Proceedings of the VLDB Endowment, 17(5), 1020-1033.
10. Hemachandra, A., Dai, Z., Singh, J., Ng, S. K., & Low, B. K. H. (2023, July). Training-free neural active learning with initialization-robustness guarantees. In International Conference on Machine Learning (pp. 12931-12971). PMLR.

11. Yang, J., Liu, Y., Wang, W., Wu, H., Chen, Z., & Ma, X. (2024). PATNAS: A Path-Based Training-Free Neural Architecture Search. IEEE Transactions on Pattern Analysis and Machine Intelligence.
12. Rajasekaran, R., & Sangeetha, S. The Role of Commercial Vehicles in Logistics and Supply Chain Operation-A Study with Reference to Textile Units in Coimbatore. Social Media-A Default Anarchist?, 127.
13. Yang, L., Fu, Y., Lu, S., Sun, Z., Mei, J., Zhao, W., & Hu, Y. (2023). Sweet Gradient matters: Designing consistent and efficient estimator for Zero-shot Architecture Search. Neural Networks, 168, 237-255.
14. Bhardwaj, K., Cheng, H. P., Priyadarshi, S., & Li, Z. (2023). Zico-bc: A bias corrected zero-shot nas for vision tasks. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1353-1357).
15. Jiang, T., Wang, H., Bie, R., & Yuan, C. (2025). Accelerating Zero-Shot NAS With Feature Map-Based Proxy and Operation Scoring Function. IEEE Transactions on Pattern Analysis and Machine Intelligence.
16. Sekanina, L. (2021). Neural architecture search and hardware accelerator co-search: A survey. IEEE access, 9, 151337-151362.
17. Kalyanaraman, K., & Ponnusamy, S. (2024). AI-Enhanced Optimization Algorithm for Body Area Networks in Intelligent Wearable Patches for Elderly Women's Safety. In Wearable Devices, Surveillance Systems, and AI for Women's Wellbeing (pp. 52-80). IGI Global Scientific Publishing.
18. O.I. Abiodun, A. Jantan, A.E. Omolara, K.V. Dada, N.A. Mohamed, H. Arshad State-of-the-art in artificial neural network applications: A survey Heliyon, 4 (11) (2018),
19. B. Wang, Y. Sun, B. Xue, M. Zhang Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification 2018 IEEE Congress on Evolutionary Computation, CEC, IEEE (2018), pp. 1-8.
20. Cardoso-Pereira, I., Lobo-Pappa, G., & Ramos, H. S. (2021, September). Neural architecture search for resource-constrained internet of things devices. In 2021 IEEE Symposium on Computers and Communications (ISCC) (pp. 1-6). IEEE.