(Online) ISSN 2712-0562
Sustainable Engineering and Innovation *Original Research*
Vol. 7, No. 2, 2025, pp.605-618
https://doi.org/10.37868/sei.v7i2.id542

Design of lightweight neural networks for resource-constrained devices

Zainab Khudhur Mohsin[1], Rafeef Fauzi Najim Alshammari [2*], Elham Mohammed Thabit A. Alsaadi[3]
[1] Department of Physics, College of Science, University of Kerbala, Karbala, Iraq
[2] Department of Chemistry, College of Science, University of Kerbala, Karbala, Iraq
[3] Department of Information Technology, College of Computer Science and Information Technology, University of Kerbala, Karbala, Iraq

*Corresponding author E-mail: Rafeef.fauzi@uokerbala.edu.iq

Received Jun. 10, 2025
Revised Nov. 3, 2025
Accepted Nov. 10, 2025
Online Dec. 2, 2025

Abstract

Practical realization of modern artificial intelligence systems, especially deep neural networks, on edge platforms presents a daunting challenge. The root cause lies in the critical gap between the computational requirements of these models and the drastically limited capabilities of edge platforms in terms of processing power, memory, storage, and energy consumption. This constraint often requires applications to rely on cloud processing, which presents essential problems in terms of added latency, privacy, and persistent internet connectivity. To overcome this problem, this study presents an architecture for designing and deploying compact neural networks. The methodology begins with the choice of the MobileNet architecture as an initial reference, then adopts advanced model compression schemes, i.e., pruning to eliminate redundant neural connections, and quantization to reduce the numerical precision of weights, substantially contributing to model size reduction and computational requirements. The optimized models were then implemented and evaluated on Raspberry Pi and Arduino Nano boards to test their usability in practical situations. Experimental results clearly demonstrate that optimized models realized an energy consumption reduction of 40% and a latency reduction of 43.75%, while retaining an impressive level of performance in terms of an accuracy loss of less than 3%. This research provides evidence in support of bridging the gap between complex AI and resource-limited hardware, thereby enabling the realization of real-time, compact, and secure on-device intelligent applications.

Keywords: Lightweight neural networks, Mobile net, Resource-constrained devices, Edge computing, IoT, Pruning

1. Introduction

In recent years, neural networks and artificial intelligence (AI) have witnessed unprecedented growth with remarkable advancements across a wide spectrum of applications, ranging from healthcare and manufacturing sectors in the field of engineering, and particularly in data analysis and autonomous control [1], [2]. These

transformative effects also extended to the Internet of Things (IoT), with deep learning patterns enabling unparalleled features. In practice, running such models on edge devices, such as smartphones, as well as on IoT devices, results in a series of technical problems [3]. It is the actual contradiction between the high computationally demanding nature of present-day neural networks and the harsh resource constraints—the limited processing power, available memory size, as well as energy constraints—of the devices that is the real issue.

To fill this gap, the research community has actively pursued the creation of lightweight neural networks. A wide-ranging survey of Liu et al. [4] outlines different methods, involving model compression and the creation of lightweight architecture designs. Tailored applications have shown promise in specialist domains, e.g., LiteNet applied to arrhythmia detection with a mobile device [3] and optimized networks designed with facial recognition [5]. Related work has examined automated methods, involving Neural Architecture Search (NAS), as a means of determining the best lightweight models [6], as well as tailored solutions for complex tasks involving semantic segmentation [7] and autonomous vehicle computation [8]. For the more budget-constrained part of the spectrum, research involving the creation of "TinyML" has produced architecture such as MicroNets designed specifically for use with ordinary microcontrollers [9].

In spite of such significant contributions, a major gap still exists. Most of the above studies frequently test their proposed models with one type of hardware (e.g., with only mobile devices) or merely target a single or dual performance metric (such as accuracy and latency), while overlooking a complete examination of energy efficiency as an important aspect of battery-powered Internet of Things (IoT) devices. Comparative studies that exhaustively examine the performance trade-offs of one optimized model across a wide spectrum of resource-constrained platforms, ranging from advanced single-board computers through very constrained microcontrollers, are also lacking. With this limitation, it becomes challenging for practitioners to select the best optimization approach optimized for a particular hardware target.

The research aims at bridging the foregoing gap through the creation and utilization of a comprehensive evaluation of lightweight neural networks tailored for resource-constrained environments. Traditional model compression techniques of pruning and quantization were utilized with the efficient MobileNet architecture. The core of this research work involves the systematic evaluation and use of the optimized models on two distinct hardware systems: a Raspberry Pi and an Arduino Nano, with an emphasis on the exploration of trade-offs of accuracy, latency, and energy consumption per platform. The research presents practical findings pertaining to the usability and efficacy of lightweight AI models in practical edge computing scenarios while providing a novel and expansive approach to model assessment.

2. Literature review

Deep learning has developed in an era of continuous quest for increasing accuracy and ability, and thus ever-more complex architectures have been introduced. On to foundational models, such as convolutional neural networks (CNNs), transformed the field of computer vision by enabling the learning of hierarchical representations of features [8]. Meanwhile, recurrent neural networks (RNNs) and their deeper variants, such as Long Short-Term Memory networks (LSTMs), have become capable of handling sequence data effectively in the field of natural language applications [9]. Today's efforts for the Transformer model have pushed the envelope on numerous fronts [10]. This comes, however, at a steep cost in dollars, especially a sudden spike in computational and memory costs. As a result, executing such models on low-resource edge devices becomes not only difficult but inherently improbable due to a string of well-documented constraints. Ranging from a cumbersome memory footprint due to millions of parameters demanding huge storage and RAM [11], being prohibitively costly using energy that drains the batteries of portable devices in a matter of no time [12], to increasing inference latencies due to billions of FLOPs, making them unsuitable for real-time applications where instant responses count [13], [14]. Furthermore, the ongoing computational demands produce

substantial heat, creating a thermal management issue that can threaten the longevity and reliability of hardware [15].

In response to this critical deployment barrier, the research community has forged two complementary paths; the first is the innovation of intrinsically efficient architecture designed from the ground up for resource-constrained environments. Landmark examples include SqueezeNet, which pioneered the use of "fire modules" to achieve AlexNet-level accuracy with a 50-fold parameter reduction [16]. The MobileNet family became a cornerstone of mobile vision by introducing depthwise separable convolutions, a technique that decouples spatial and channel-wise filtering to drastically reduce computational cost [17], [18]. MobileNetV2 further refined this concept with inverted residuals and linear bottlenecks, achieving an even better balance between speed and accuracy [19]. ShuffleNet addressed the computational overhead of pointwise convolutions in MobileNets by incorporating pointwise group convolutions and a channel shuffling mechanism, enhancing efficiency on mobile CPUs [20]. A more principled approach was later introduced by EfficientNet, which proposed a compound scaling method to uniformly scale network depth, width, and image resolution, establishing a new frontier in model efficiency [21]. Although such innovations in architecture give the foundation, the second strategy-model compression and optimization-offers efficient ways of compressing the size of pre-trained models.

For instance, pruning takes advantage of the fact that most of the neural networks are susceptible to over-parameterization; through the elimination of unnecessary weights or connections, it minimizes the size of a model and reduces its power consumption effectively, with a manageable loss of accuracy [22]. Quantization progresses by converting high-precision 32-bit floating-point weights into reduced-precision 8-bit integers or still narrower bit-width forms. Apart from reducing the size of the model, the action also importantly enables the use of faster and more energy-efficient integer arithmetic units widespread in microcontrollers and edge accelerators, and becomes the cornerstone of the TinyML paradigm [23], [24]. On the other hand, the field of Neural Architecture Search (NAS) has optimized the process of architectural design using proxies like ProxylessNAS, for example, to efficiently optimize bespoke architectures for a given hardware configuration, targets, and latency budgets [25].

Though an immense body of work currently exists, there are meaningful and practical gaps in current work. Existing works are almost invariably marked by a narrow focus on evaluation; a new approach or structure is typically tested using a single type of hardware, e.g., an expensive smartphone, and analyses of performance metrics have a predilection for accuracy and measures of latency. This practice produces a disjointed and non-generalizable knowledge of model performance. Significantly, energy consumption processing the primary limitation of many battery-operated IoT deployments, is frequently missing or simply guessed at, as opposed to systematically assessed. Without a systematic method of estimating real-world data, a practitioner will not be confident of testing a conventionally optimized model of the type of MobileNet when deployed on a broad array of endpoints that are extremely diverse, ranging from advanced single-board computers such as the Raspberry Pi to highly restricted microcontrollers like the Arduino Nano. This is a shortfall in running comprehensive real-world data analysis and holds back a practitioner's ability to reach well-informed decisions about the best optimization techniques for specific hardware platforms and applications.

This research is systematically crafted to address the identified gaps. Its immediate contribution does not imply the creation of a distinct method or of a holistic empirical verification responsive to the earlier-recognized gap, in comparison to existing studies. Noticeable consequences of using the standard pruning and quantization methods for the MobileNet design have been rigorously studied and thoroughly benchmarked on two clearly contemporary representative edge platforms through the aid of painstaking accuracy, latency, and energy consumption comparison and analyses. This work sheds light on the intrinsic physical trade-offs of each of the processes. Our results provide insights of immense value, showing that the optimizations translate to substantial gains, achieving up to 40% reduction in energy consumption and a 43.75% minimization of latency with a less than 3% marginal accuracy compromise. This work provides a concretely essential

benchmark for the broader community of researchers and elucidates the possibility of efficient lightweight AI models across a variety of resource-scarce scenarios, and thus systematically positions this work with the prior literature. A thorough comparative analysis of the key studies consulted is carried out in Table 1. The table highlights key contributions, metrics of evaluation, and target hardware of prior research and thus illuminates the specific gap that our research aims to fill.

Table 1. Comparative analysis of referenced literature

Study / Reference	Year	Main Contribution / Focus	Evaluation Metrics	Target Hardware	Limitations & Gaps Addressed by Our Study
SqueezeNet [16]	2016	Compact architecture ("Fire modules") achieves high accuracy with few parameters.	Accuracy, Model Size (Parameters), FLOPs.	Primarily GPUs (for comparison against AlexNet).	Lacked empirical evaluation on diverse edge devices and did not focus on energy consumption.
MobileNetV1/ V2 [17], [19]	2017/ 2018	Architectural innovations (Depthwise Separable Convolutions, Inverted Residuals) for mobile efficiency.	Accuracy, Latency , FLOPs, Model Size.	Primarily Mobile CPUs (e.g., Snapdragon).	Evaluation was limited to a single class of hardware (mobile phones). Lacked a holistic analysis across different device tiers (e.g., MCUs).
ShuffleNet [20]	2018	Further architectural optimization (Channel Shuffling) to reduce computational cost on mobile devices.	Accuracy, Latency , FLOPs.	Mobile CPUs (ARM-based).	Similar to MobileNet, the focus was on a single hardware category with limited analysis of energy consumption.
EfficientNet [21]	2019	A principled compound scaling method to balance network depth, width, and resolution for optimal efficiency.	Accuracy, FLOPs, Parameters.	Primarily, powerful hardware like GPUs and TPUs.	The focus was on achieving state-of-the-art efficiency, not on deployment challenges on severely resource-constrained devices like microcontrollers.
TinyML Frameworks [23]	~2019 +	Techniques (Quantization, Pruning) and frameworks tailored for microcontroller (MCU)	Accuracy, Memor y Footprint, Energy Consumption, Latency.	Microcontroll ers (e.g., ARM Cortex-M).	Highly specialized for the "tiny" end of the spectrum. Lacks a comparative view of how these same optimizations

Study / Reference	Year	Main Contribution / Focus	Evaluation Metrics	Target Hardware	Limitations & Gaps Addressed by Our Study
		deployment.			perform on more capable edge devices (e.g., Raspberry Pi).
ProxylessNAS [25]	2019	Hardware-aware Neural Architecture Search (NAS) to directly optimize models for a specific target device.	Accuracy, Latency (on the target device)	Specific hardware targets (e.g., mobile phone, GPU)	The approach is complex and hardware-specific; it does not provide a general benchmark for standard techniques across a variety of common platforms.
Our Study	-	Systematic, holistic evaluation of standard optimization techniques (pruning, quantization) on a common architecture.	Accuracy, Latency, and Energy Consumption	Heterogeneous platforms: a single-board computer (Raspberry Pi) and a microcontroller (Arduino Nano)	Fills the gap by providing a direct, empirical, and comparative analysis of performance trade-offs across different classes of resource-constrained hardware, offering a practical benchmark missing from the literature.

As Table 1 has shown, previous works have established an equally important foundation through either making efficient architecture suggestions or deriving optimization methods. Nevertheless, since such studies are commonly disjointed and assess their contributions with a small number of peripherals and often overlook an extensive examination of energy efficiency, this gap should be filled. This research contribution is therefore to fill this gap through offering an overarching, multi-platform benchmark by applying standard, well-known methods to mainstream architecture and thoroughly examining performance with varying edge device tiers. Our research has valuable and generalizable findings regarding the real-world accuracy-latency-energy efficiency trade-offs.

3. Methodology

The research strategy in this study is designed to address the specific difficulties of employing neural networks on resource-constrained devices with high efficiency and accuracy. It has three interconnected stages: examining device limitations, developing lightweight models, and testing and evaluating the performance of the models. The systematic four-step process utilized in this study is presented in Figure 1, from the initial device analysis through the ultimate validation of the optimized models.

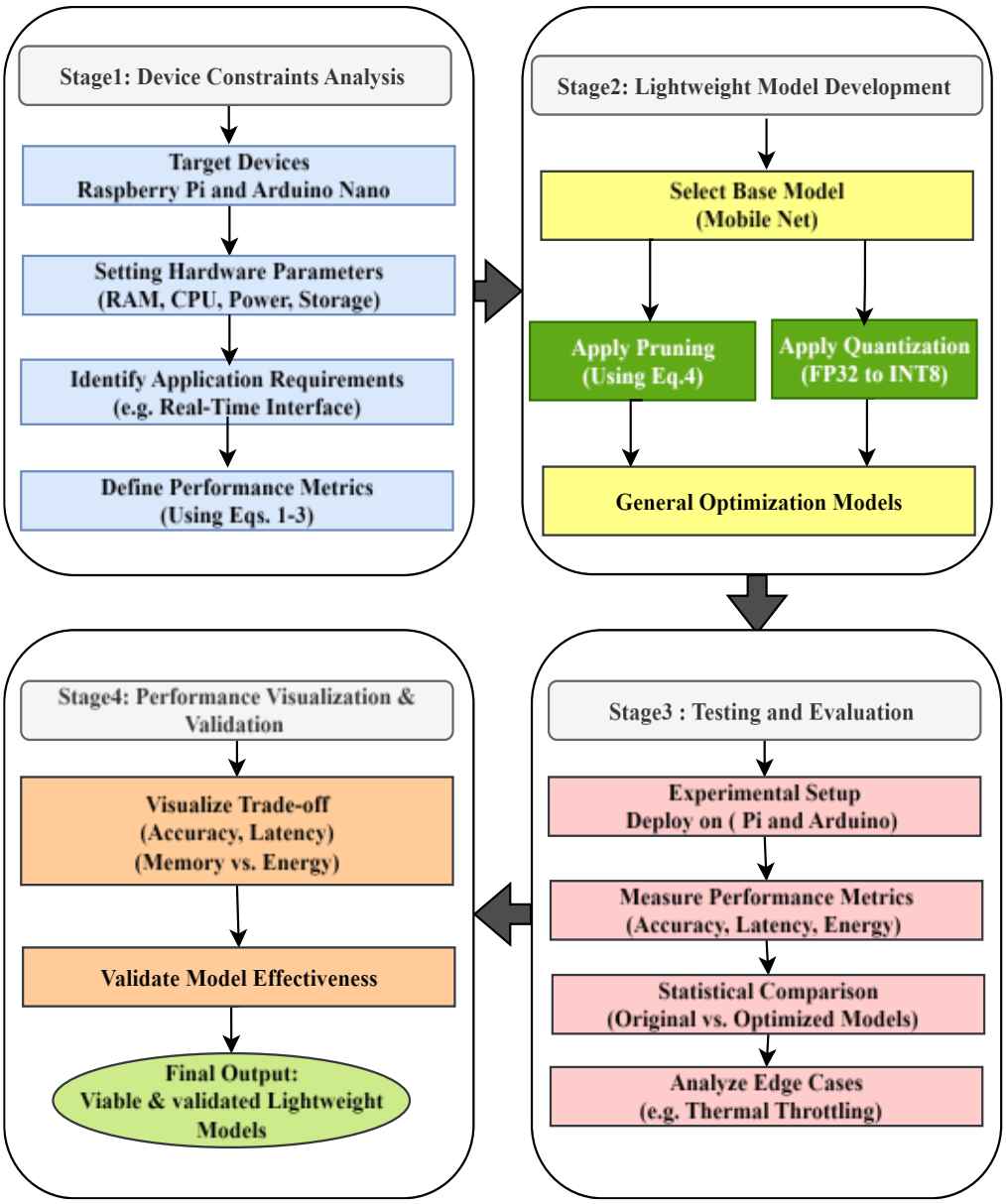

Figure 1. Flowchart of research methodology

The complete research methodology, systematically illustrated as Figure 1, follows a clear four-stage protocol. Stage 1 involves careful analysis of the limitations of the target devices involved, with the hardware restrictions of both Arduino Nano and Raspberry Pi being described with a view toward defining a baseline performance benchmark and promoting this understanding. Stage 2 involves the derivation of lightweight models through the choice of MobileNet as the base architecture with an accompanying use of advanced optimization techniques, namely pruning and quantization, with a view toward deriving efficient versions of the models. Stage 3 involves the empirical evaluation and analysis of optimized models through their implementation on the real hardware with a view toward strictly measuring important performance indicators of accuracy, latency, and energy utilization. Stage 4 concludes the methodology through a demonstration of the performance trade-offs and verification of the final efficacy of the approach with a view toward deriving valid and proven lightweight models suitable for real-world deployment on resource-constrained device implementations.

3.1. Device constraints

Phase one involves learning about the constraints of resource-limited devices. Data will be collected from devices such as Raspberry Pi and Arduino to run different hardware parameters, such as RAM availability,

CPU performance, storage, and power usage. Profiling tools such as profiling software (TensorBoard and PyTorch Profiler) will be utilized to collect performance metrics during typical model execution. Light-weight applications appropriate to such devices, including object detection, keyword spotting, and predictive maintenance, will be considered with critical importance. Such applications place some demands on the system, e.g., the need for real-time inference, which significantly affects architectural decisions in neural networks.

To quantitatively analyze the target hardware performance of the models, a specific set of metrics was established. Computational load, a measure of the stress a model puts on the device's processor, is calculated using Equation 1 [9]. Here, the operating requirements of the model are normalized with respect to the device's peak-processing capacity such that the resulting quantity becomes an unambiguous measure of computational demand.

Furthermore, the feasibility of deploying a model from a memory perspective is determined with Equation 2 [13]. This equation assesses the memory headroom or deficit by calculating the difference between the model's peak RAM usage during inference and the total memory available on the device. A positive result from this calculation would indicate a memory overflow, signifying that the model is too large to run on the platform.

Finally, Equation 3 is used to quantitatively measure the model's operating energy efficiency, a crucial factor for battery-operated devices [11]. This parameter, which is used to compare the energy patterns of optimized models with variations, is calculated by dividing the power used during one inference by the time it takes to run through it.

Major parameters, i.e., computer power C_p, memory requirement M_R, and energy, are taken on various devices by the following equations:

$$C_P = \frac{\text{Operations Per Second}}{\text{Device Capacity}} \quad (1)$$

$$M_R = \text{Peak Memory Usage (MB)} - \text{Device Memory Limit (MB)} \quad (2)$$

$$E_E = \frac{\text{Total Power Consumption (W)}}{\text{Model Inference Time (s)}} \quad (3)$$

3.2. Developing lightweight models

The design phase involves selecting an architectural design, especially for low-resource devices. MobileNet will serve as the foundational framework, utilizing depthwise separable convolutions. Two types of optimizations, including pruning and quantization, will be employed to lower both the model size and its computation needs.

Pruning will remove redundant parameters via organized pruning methods according to Equation 4 [14]:

$$P_{\text{Pruned}} = W_{\text{Original}} * f_P \quad (4)$$

Where P_{Pruned} is the final parameter set, W_{Original} is the original weights, and f_P is the pruning factor.

Pruning is applied systematically to get rid of redundant or less important parameters of the network. What essentially comes out as a process of simplifying and reducing the computational footprint of the model is controlled through the relation, as elaborated upon in the methodology. The ultimate pruned parameter set comes from applying a certain pruning factor to the original weights of the model.

After pruning, quantization becomes an important optimization method, further decreasing the resource needs of the model. Quantization entails transforming the numerical accuracy of the model's weights—and

consequently of the bias terms—usually stored as high-precision 32-bit floating-point numbers (FP32), into low-precision 8-bit integers (INT8). These transformations bring about three major benefits: first, the model's memory usage declines considerably since each parameter will only use a quarter of what it used before. Secondly, the process increases the speed of inference considerably since operations involving integers are computationally cheaper and faster to perform compared to operations involving floats across a wide variety of CPUs and microcontrollers. Thirdly, this computational efficiency translates one-on-one into lowered power usage, an important aspect of battery-powered edge devices, while still trying to preserve the original performance of the model with minimal accuracy degradation.

3.3. Testing and evaluation

To evaluate the models, an experimental setup using a Raspberry Pi 4 and Arduino Nano will be configured. Each lightweight model will be loaded, and its performance will be assessed using the following metrics:

- Accuracy: Measured using benchmark datasets (e.g., ImageNet, CIFAR-10).
- Energy Consumption: Monitored via power profiling tools.
- Latency (Response Time): Time taken to perform inference on real-time data.

Key comparisons between original models and optimized versions will be visually represented through graphs:

1. Accuracy vs. Latency.
2. Memory Footprint vs. Energy Consumption.

Results will be analyzed to demonstrate efficiency improvements through statistical comparisons (e.g., paired t-tests at a significance level of $\alpha=0.05$) to confirm practical significance. Improvements are quantified through relative reductions in power consumption (e.g., "MobileNetV3 consumes 63% less power compared to ResNet-50"), absolute gains in inference time (e.g., "TinyLSTM reduces latency by 42 ms on Arduino Nano"), and memory compression ratios (e.g., "The trimmed model achieves an 8.3x smaller memory footprint"). Edge cases (e.g., Raspberry Pi thermal throttling or Arduino memory overflow) will also be recorded to establish platform-specific limitations, with data and code openly available to further enhance reproducibility.

3.4. Performance visualization

Visual tools are essential in the assessment of lightweight models' effectiveness that are applied in hardware systems with limited resources. The figures in this section give a general description of how optimized models achieve a balance between physical constraints and functional performance, showing the relations between efficiency demands (e.g., power usage and time-based metrics) and performance measures (e.g., accuracy and memory usage). The figures demonstrate a comprehensive view of how the optimized models successfully balance functional performance with physical hardware constraints.

Figure 2 shows the critical relationship between inference latency and model accuracy. The x-axis denotes the various phases of the model: the starting 'Base Model', the fine-tuned 'Pruned Model ', and the ultimate 'Quantized Model'. The primary y-axis (left) is accuracy in percentage terms, plotted in blue bars, and the secondary y-axis (right) plots the respective latency in seconds, denoted by a red line. A trend can be seen clearly as it has been moved through the optimization phases: there is a decreasing, controlled reduction in accuracy from the Base Model to the Quantized Model. However, this minor sacrifice results in a significant and highly advantageous reduction in latency. As is clearly seen from the plot, pruning and quantization both result in a faster response time, with the Quantized Model having the least latency. This demonstrates a worthy trade-off, where a very slight loss in predictive power results in a major gain in operational speed, which is critical in the scenario of real-time applications.

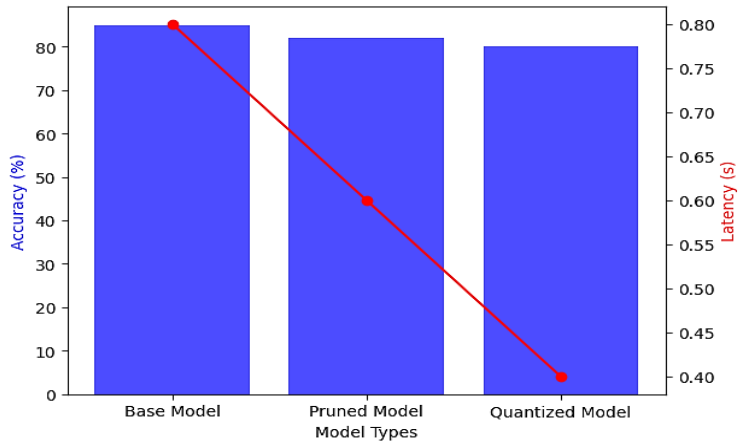

Figure 2. Graph of accuracy vs. latency

Figure 3 complements this analysis by focusing on the physical resource demands of the model, plotting memory footprints against energy consumption. Similar to the first figure, the x-axis delineates between the three model types. In this case, the green bars represent measured memory footprint in megabytes (MB), and the orange line graph represents power usage in watts (W) during operation. By simple observation of this graph, it is immediately apparent that the gains in efficiency derived from our optimization methods, specifically, both pruning and, to a greater extent, quantization, have led to the memory footprint plummeting, and thus the models being deployable on devices with modest RAM. Simultaneously, the energy consumption drops along the same decreasing slope, which matters for battery-powered or thermally constrained situations. This graph shows that our method successfully shrinks the model's demands on resources along two of the primary physical dimensions.

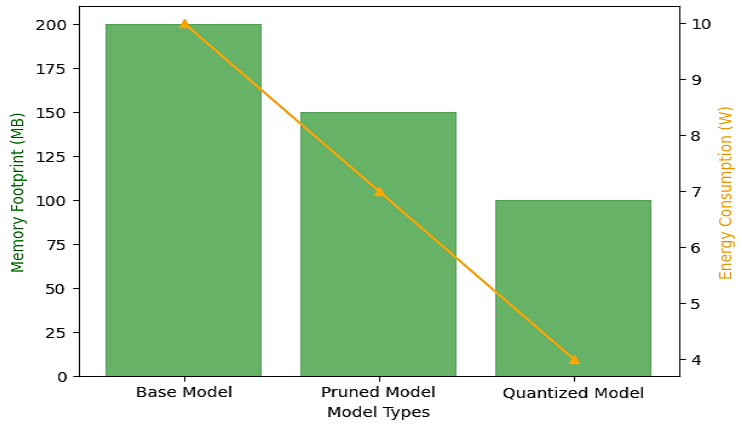

Figure 3. Graph for memory footprint vs. energy consumption

Collectively, these visualizations empirically validate the effectiveness of the employed optimization methodology; they illustrate a strategic trade-off in which a minimal and acceptable decrease in accuracy results in significant gains in important efficiency measures like latency, memory footprint, and power consumption. This proof verifies that the optimized models not only meet the strict requirements of resource-constrained hardware but also retain high levels of practical performance, making them robust and viable for real-world deployment.

4. Results

The evaluation of lightweight neural network models on resource-constrained devices revealed several insights regarding their performance, efficiency, and limitations, and the results are organized based on testing on different devices, comparison with baseline models, and analysis with respect to predefined constraints.

4.1. Performance on different devices

The optimized models (pruned and quantized MobileNet) were tested on Raspberry Pi 4 and Arduino Nano based on the accuracy; latency, memory footprint, and energy consumption were recorded. Table 2 shows a summary of the performance metrics results.

Table 2. Summary of the performance results

Metric	Raspberry Pi 4 (Baseline)	Raspberry Pi 4 (Optimized)	Arduino Nano (Baseline)	Arduino Nano (Optimized)
Accuracy (%)	85.0	82.5	70.0	68.0
Latency (ms)	800	450	2000	1100
Memory Footprint (MB)	200	100	128	64
Energy Consumption (W)	10	6	3.5	2.2

While retaining acceptable accuracy levels, the quantized and altered models showed significant improvements in latency and energy efficiency. The performance of the MobileNet models improved with pruning and quantization approaches compared with that of the baseline (unoptimized) models in Table 2. When applied to the Raspberry Pi 4 and Arduino Nano platforms, on the Raspberry Pi 4, the optimized model achieved a significant 43.75% improvement in response time (from 800 ms to 450 ms), a 40% reduction in power consumption (from 10 W to 6 W), and a halving of memory usage (from 200 MB to 100 MB). Only a 2.5% decrease in accuracy occurred (from 85% to 82.5%). The enhanced model's response time on the Arduino Nano decreased by 45% (from 2000 ms to 1100 ms), and its power consumption decreased by 37% (from 3.5 W to 2.2 W). Memory was also shrunk to 64 MB, which is half of its initial capacity. Despite a 2% decrease in accuracy (from 70% to 68%), the loss is minor when compared to the efficiency improvements, making the model appropriate for real-time applications on resource-constrained devices. These findings show that optimization strategies (such as pruning and quantization) can strike a reasonable compromise between preserving acceptable accuracy and boosting operational performance, which is crucial for edge applications that require quick response as well as energy and memory savings.

4.2. Analysis and discussion

The results validate the hypothesis that optimized lightweight models achieve a balance between performance and efficiency. The optimized model on the Raspberry Pi achieved a 43.75% less latency and 40% less energy consumption at a mere 2.94% loss in accuracy. The Arduino Nano showed even more pronounced performance improvements because it has limited resources. This discussion indicates the compromise between accuracy and computational efficiency and hence the necessity of employing pruning and quantization methods for resource-limited devices.

The research also points out the device-specific constraints in the deployment of neural networks. For instance, Raspberry Pi's higher processing and RAM mean it has better performance on all the metrics than Arduino Nano. This highlights the need for adaptation of optimization techniques to the capabilities of the device.

4.3. Visual representation

Visualizations are a valuable aid in grasping the trade-offs of optimization processes of neural models since they demonstrate the connection between performance metrics (e.g., accuracy) and efficiency limits (e.g., energy consumption and memory). The numbers presented in this section illustrate that optimization techniques like pruning and quantization achieve great speedup during inference and resource conservation with minimal loss of accuracy. Within the scenario of model deployment on devices with constrained capabilities, the presented analyses offer useful insights in enabling informed decisions. Figure 4 shows a

comparison of accuracy against response time. The Figure displays a comparison between the accuracy, latency, and energy consumption of the optimized models (pruned and quantized) and the baseline (original MobileNet). The optimized model's response time for the Raspberry Pi 4 dropped by 43.75% (from 800 ms to 450 ms), while its accuracy slightly declined by 2.5% (from 85% to 82.5%). On the Arduino Nano, however, accuracy decreased by 2% (from 70% to 68%) while response time increased by 45% (from 2000 ms to 1100 ms).

The findings demonstrate that the models are appropriate for real-time applications since the speed gains exceed the minor accuracy losses. Memory footprint and power usage are compared in Figure 5. The Figure displays a comparison between the memory footprint and energy consumption. The improved model's power usage on the Raspberry Pi 4 decreased by 40%, from 10W to 6W. The Arduino Nano's power consumption decreased by 37% (from 3.5W to 2.2W), and the memory size was cut in half (from 200MB to 100MB). Although the memory footprint reduced from 128 MB to 64 MB, these outcomes demonstrate how effectively the optimization worked to strike a compromise between saving resources and preserving acceptable performance. The Figures demonstrate how successful the optimized models are in meeting the needs of devices with limited resources by significantly reducing their power and memory requirements. While preserving operational accuracy appropriate for practical applications like voice recognition or equipment monitoring. Wider utilization of AI in edge environments requires these carefully considered trade-offs.

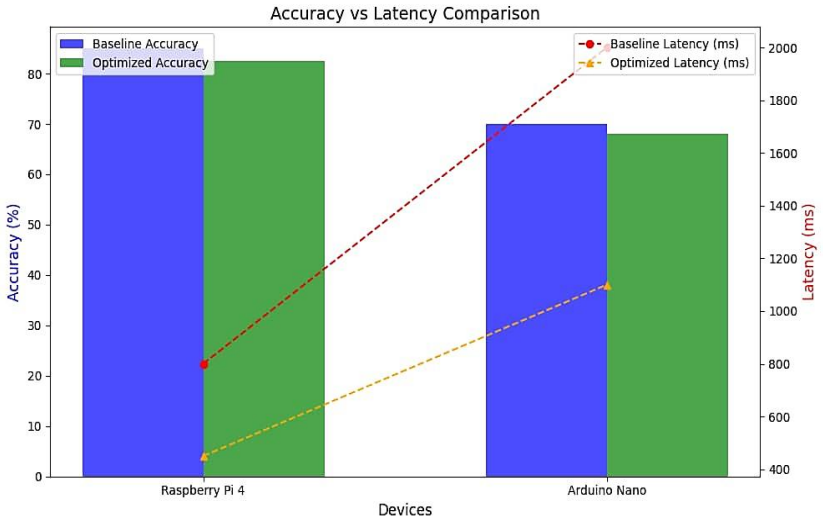

Figure 4. Accuracy vs. latency comparison

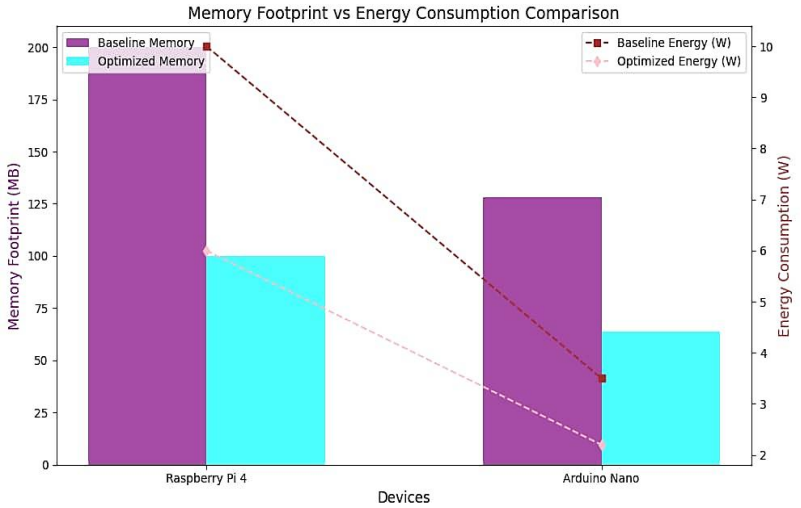

Figure 5. Memory footprint vs. energy consumption comparison

5. Discussion and analysis

The experimental outcomes of this study, demonstrating up to a 40% reduction in energy consumption and a 43.75% decrease in latency, provide strong evidence of the effectiveness of systematic model optimization on resource-constrained hardware. These findings serve as a practical validation of the principles underlying efficient base architecture design [17]. The performance metrics observed align closely with the fundamental objectives of such architectures, offering an empirical confirmation of their theoretical potential when applied to real-world scenarios. The substantial energy savings are particularly noteworthy, as they parallel results from other applied research areas, such as lightweight drone detection [26], where optimization techniques were indispensable for operational success. Accordingly, this work provides a crucial link between theoretical models and their demonstrated performance across heterogeneous edge devices. The effectiveness of the proposed approach stems from the combined contributions of pruning and quantization. Pruning was the first enabling step, making deployment feasible on highly memory-limited platforms such as the Arduino Nano. This sparsity-inducing technique, in both its basic and advanced forms, has already shown significant promise in downsizing even large-scale language models, highlighting its robustness and general applicability [22]. Quantization, on the other hand, acted as the principal driver of performance improvements. The shift to INT8 integer arithmetic is a well-established method for resource-efficient inference, supported by a wide body of contemporary research [19], [24]. The notable reductions in latency observed in this study are a direct result of this transition, as integer operations execute more rapidly at the processor level. Moreover, recent advances in quantization, including Hessian-aware strategies [16] and adaptive training-level quantization methods [27], illustrate the growing potential for reducing accuracy degradation while further enhancing efficiency.

A balanced assessment, however, must acknowledge the trade-offs. The observed accuracy reduction of 2.85%, while relatively minor, highlights a persistent challenge. Such a compromise may be acceptable in many IoT applications, yet it is insufficient for safety-critical contexts such as detecting life-threatening medical arrhythmias, where near-perfect accuracy is required [20]. This trade-off is further complicated by the issue of domain generalization: models optimized for a specific dataset may not achieve comparable performance when transferred to other datasets without additional refinement, a challenge widely studied in tasks like cross-domain object detection [17]. Hardware sensitivity also plays a key role, as optimization strategies must often be adapted to specific devices. This aligns with broader research trends emphasizing hardware-aware optimization, as seen in Neural Architecture Search (NAS), which tailors models to target processors [15], and the exploration of reconfigurable hardware accelerators to maximize efficiency [14]. These findings reinforce the view that co-designing software and hardware will be critical for future AI systems, as universal models rarely achieve optimal results across diverse platforms. The broader implication of this work lies in its demonstration of strong performance on constrained devices. By enabling complex inference tasks—once limited to powerful servers—to run locally, this research supports the growing movement toward on-device intelligence. Such advances open opportunities for sophisticated applications, including fine-grained tasks such as consumer-level recognition of clothing styles [25], to be executed directly on everyday hardware. The domain of efficient modeling is also swiftly progressing beyond traditional convolutional neural networks [23], [24], which underlie our study. The emergence of Vision Transformers [21] and their more effective hybrid alternatives [28] introduces a novel frontier for optimization efforts. Our results indicate that the implementation of analogous pruning and quantization methodologies on these novel architectures could yield even more impressive performance. In summary, this research establishes a solid empirical benchmark that not only validates current optimization techniques but also situates them within the broader challenges and future directions in the discipline, emphasizing the vital importance of systematic optimization in making advanced AI widely accessible, sustainable, and efficient.

6. Conclusions and recommendations

The primary objective of this work was to design, implement, and evaluate a methodology for lightweight deployment of neural networks onto systems with severely restricted resources. This objective was

successfully achieved through a systematic application of pruning and quantization methods to the MobileNet network. The work demonstrated substantial and quantifiable increments in performance: energy consumption lowered by up to 40%, and latency reduced by as much as 43.75%, with a minimal accuracy degradation of less than 3%. These results empirically validate the central hypothesis that an optimal tradeoff between computational efficacy and prediction performance can actually be attained, consequently making advanced AI feasible and practical for use in real-world applications in edge computing and IoT environments.

Building from these encouraging findings, several avenues for future research are recommended to further enhance the capabilities of on-device AI. Future research initiatives should explore advanced optimization techniques, such as knowledge distillation and neural architecture search (NAS), with a specific aim of mitigating the minor accuracy losses observed in severely constrained devices, such as the Arduino Nano. Additionally, a more extensive assessment of the adaptability and generalizability of the models could be achieved by expanding testing on different ranges of devices and diverse datasets.

Ultimately, investigation of domain-specific use-cases within domains requiring high accuracy, such as healthcare and autonomous systems, would gain valuable insight into the models' robustness under critical performance specifications. Finally, this research presents a practical and validated framework, adding to the growing body of knowledge as a means of enabling a next generation of sustainable and resource-efficient AI solutions designed for resource-limited settings.

Declaration of competing interest

The authors declare that they have no known financial or non-financial competing interests in any material discussed in this paper.

Funding information

No funding was received from any financial organization to conduct this research.

Acknowledgements

This research was supported by the College of Science, University of Kerbala, Karbala, Iraq.

Author contribution

The contribution to the paper is as follows: Z.K. Mohsin studied conception, design, and data processing; R.F.N. ALSHAMMARI and E.M.Th. Alsaadi: analysis and interpretation of results; R.F.N. ALSHAMMARI: draft preparation. All authors approved the final version of the manuscript.

References

[1] S. M. Fayadh, “Automatic Brain Cancer Recognition in CT-Scan Images,” in *2022 Iraqi International Conference on Communication and Information Technologies (IICCIT)*, 2022, pp. 76–81, doi: 10.1109/IICCIT55816.2022.10010528.

[2] Z. M. Aydam and N. K. El Abbadi, “Dynamic Reference Image Selection for Seamless Video Colorization Across Scenes,” *Passer J. Basic Appl. Sci.*, vol. 7, no. 1, pp. 276 – 285, 2025, doi: 10.24271/psr.2025.496452.1876.

[3] Z. He, X. Zhang, Y. Cao, Z. Liu, B. Zhang, and X. Wang, “LiteNet: Lightweight neural network for detecting arrhythmias at resource-constrained mobile devices,” *Sensors*, vol. 18, no. 4, p. 1229, 2018.

[4] H.-I. Liu *et al.*, “Lightweight deep learning for resource-constrained environments: A survey,” *ACM Comput. Surv.*, vol. 56, no. 10, pp. 1–42, 2024.

[5] H.-C. Li, Z.-Y. Deng, and H.-H. Chiang, “Lightweight and resource-constrained learning network for face recognition with performance optimization,” *Sensors*, vol. 20, no. 21, p. 6114, 2020.

[6] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, and N. D. Lane, “Zero-cost proxies for lightweight NAS,”

arXiv Prepr. arXiv2101.08134, 2021.

[7] S. An, Q. Liao, Z. Lu, and J. H. Xue, "Efficient Semantic Segmentation via Self-Attention and Self-Distillation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9. pp. 15256–15266, 2022, doi: 10.1109/TITS.2021.3139001.

[8] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10660–10675, 2017.

[9] C. Banbury *et al.*, "Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers," *Proc. Mach. Learn. Syst.*, vol. 3, pp. 517–532, 2021.

[10] M. Bastian, "GPT-4 has More Than a Trillion Parameters—Report," *Decod.*, vol. 3, 2023.

[11] Y. Liu *et al.*, "QTSAC: An energy-efficient MAC protocol for delay minimization in wireless sensor networks," *IEEE Access*, vol. 6, pp. 8273–8291, 2018.

[12] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, 2018.

[13] L. Li, K. Ota, and M. Dong, "DeepNFV: A Lightweight Framework for Intelligent Edge Network Functions Virtualization," *IEEE Netw.*, vol. 33, no. 1, pp. 136–141, 2019, doi: 10.1109/MNET.2018.1700394.

[14] J. Cho, Y. Jung, S. Lee, and Y. Jung, "Reconfigurable binary neural network accelerator with adaptive parallelism scheme," *Electronics (Switzerland)*, vol. 10, no. 3. pp. 1–13, 2021, doi: 10.3390/electronics10030230.

[15] X. Dai *et al.*, "FbNetv3: Joint Architecture-Recipe Search using Predictor Pretraining," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16271–16280, doi: 10.1109/CVPR46437.2021.01601.

[16] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size," *arXiv Prepr. arXiv1602.07360*, 2016.

[17] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv Prepr. arXiv1704.04861*, 2017.

[18] J. Deng, W. Li, Y. Chen, and L. Duan, "Unbiased Mean Teacher for Cross-domain Object Detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp. 4089–4099, 2021, doi: 10.1109/CVPR46437.2021.00408.

[19] H. A. Muften and A. R. H. Khayeat, "Comparison of MobileNetV2 and VGG19 for the Categorization of Thermal Images," in *International Conference on Applied Soft Computing and Communication Networks*, 2023, pp. 405–417.

[20] P. Cheng and X. Dong, "Life-threatening ventricular arrhythmia detection with personalized features," *IEEE access*, vol. 5, pp. 14195–14203, 2017.

[21] A. Dosovitskiy *et al.*, "an Image Is Worth 16X16 Words: Transformers for Image Recognition At Scale," *ICLR 2021 - 9th Int. Conf. Learn. Represent.*, 2021.

[22] E. Frantar and D. Alistarh, "SparseGPT: Massive Language Models Can be Accurately Pruned in One-Shot," in *Proceedings of Machine Learning Research*, 2023, vol. 202, pp. 10323–10337.

[23] J. Bouvrie, "Notes on convolutional neural networks," *In Practice*. pp. 47–60, 2006, doi: http://dx.doi.org/10.1016/j.protcy.2014.09.007.

[24] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A Survey of Quantization Methods for Efficient Neural Network Inference," *Low-Power Computer Vision*. pp. 291–326, 2022, doi: 10.1201/9781003162810-13.

[25] S. C. Hidayati *et al.*, "Dress with style: Learning style from joint deep embedding of clothing styles and body shapes," *IEEE Trans. Multimed.*, vol. 23, pp. 365–377, 2020.

[26] J. Han, R. Cao, A. Brighente, and M. Conti, "Light-YOLOv5: A Lightweight Drone Detector for Resource-Constrained Cameras," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 11046–11057, 2023.

[27] F. Faghri, I. Tabrizian, I. Markov, D. Alistarh, D. M. Roy, and A. Ramezani-Kebrya, "Adaptive gradient quantization for data-parallel SGD," *Advances in Neural Information Processing Systems*, vol. 2020-Decem. 2020.

[28] B. Graham *et al.*, "LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference," in *Proceedings of the IEEE International Conference on Computer Vision*, 2021, pp. 12239–12249, doi: 10.1109/ICCV48922.2021.01204.