Computer Science, Engineering and Technology

Vol: 3(2), June 2025

REST Publisher; ISSN: 2583-9179

Website: https://restpublisher.com/journals/cset/

DOI: https://doi.org/10.46632/cset/3/2/13

Optimizing Power Efficiency in Embedded AI: A CNN Performance Study on Custom SoC Architectures

Dheeraj Kumar Boddu

University of Maryland, Baltimore

Corresponding author Email: dheerajboddu97@gmail.com

Abstract: *Efficient power management is critical for embedded AI systems operating in constrained environments. This study investigates the power-performance trade-offs of deep learning models on a specialized heterogeneous processing platform. By leveraging dynamic workload distribution and frequency scaling techniques, I analyze the energy efficiency of multiple CNN architectures under varying computational loads. The findings provide key insights into optimizing inference performance while minimizing power consumption, offering practical guidance for deploying AI models on embedded hardware.*

1. INTRODUCTION

Embedded computing platforms are frequently deployed in environments where energy availability is limited, making energy-conscious operation vital to ensure both optimal performance and prolonged system usability. Within these platforms, power regulation is primarily governed by operating system-level mechanisms known as frequency scaling policies or governors. These policies dynamically manage energy usage by adjusting computational resources based on workload demands. As such, a comprehensive investigation is essential to determine the most effective strategies for energy utilization in embedded environments. Consequently, an in-depth examination of the interplay between energy efficiency and computational throughput across various system configurations is necessary to advance the understanding of energy-aware scheduling mechanisms.

This study conducts an empirical evaluation of the performance-energy balance of Convolutional Neural Network (CNN) models when executed on the Khadas VIM3, an edge computing board equipped with the Amlogic A311D SoC featuring a Heterogeneous Multi-Processing System-on-Chip (HMPSoC) architecture. The assessment involves five representative CNNs widely used for image classification tasks: AlexNet, GoogleNet, MobileNet, ResNet50, and SqueezeNet. These architectures are implemented using the ARM Compute Library (ARM-CL) and executed on the ARM-based HMPSoC of the Khadas board. The investigation focuses on understanding how configurable energy optimization parameters influence the runtime behavior and power usage of these models.

The embedded SoC in the Khadas VIM3 integrates a sixcore asymmetric processor cluster using ARM's big.LITTLE architecture, consisting of two high-performance 'big' cores and four power-efficient 'LITTLE' cores, alongside a dualcore Mali G52 MP4 Graphics Processing Unit (GPU). The upper operating frequencies for the big cores, LITTLE cores, and GPU are set at 2.2 GHz, 1.8 GHz, and 0.8 GHz, respectively. The system is supported by 4 GB of LPDDR4 RAM. The software environment includes Android version 9.0 with Linux kernel 4.9, with ARM-CL version 21.02 deployed for neural network execution.

To facilitate fine-grained parallelism, a customized researchgrade variant of ARM-CL, termed Pipe-All, is employed. This framework introduces a pipeline-based methodology that allows simultaneous inference workloads to be dispatched across the CPU clusters and GPU cores. Pipe-All exposes configurable parameters that permit task

partitioning among the computational units, thereby offering control over workload scheduling. The frequency scaling governor introduced in this analysis utilizes the aforementioned workload-balancing feature as a mechanism for dynamic power adjustment. Furthermore, the implementation incorporates Dynamic Voltage and Frequency Scaling (DVFS), a hardware-level technique that modulates processor voltage and clock speed at runtime. DVFS empowers the power manager to align computational power with operational demands, thereby minimizing unnecessary energy expenditure without compromising task performance.

2. LITERATURE REVIEW

The proliferation of artificial intelligence (AI) in embedded systems has necessitated the integration of energy-efficient processing mechanisms to sustain real-time inference under resource constraints. Several studies have explored the balance between computational performance and energy utilization in edge devices executing deep learning workloads. These investigations primarily examine the architectural features of system-on-chip (SoC) designs and the impact of power-aware scheduling on model execution efficiency.

Research in [1] introduced DeepX, an inference engine that applies model partitioning and runtime optimization to reduce energy consumption during mobile inference. Similarly, the work presented in [2] emphasized the role of model pruning and quantization in decreasing the energy footprint of deep neural networks (DNNs), showing significant reductions in both memory access and computation overhead. These techniques have laid the groundwork for optimizing inference pipelines on low-power hardware

In the context of embedded platforms, dynamic voltage and frequency scaling (DVFS) has emerged as a founda- tional strategy for adaptive energy control. The work in [3] demonstrated how DVFS, when applied selectively to cores based on their workload intensity, can yield substantial energy savings with minimal performance degradation. Additionally, governor policies such as 'ondemand' and 'performance' have been explored in [4] to understand their influence on runtime behavior across different processor configurations.

Heterogeneous computing platforms, particularly those employing ARM's big.LITTLE architecture, have attracted significant attention due to their potential for fine-grained power management. The study in [5] analyzed task migration strategies and their energy-performance trade-offs, highlighting that workloads with varying compute density could benefit from core-specific dispatching. Complementary research in [6] provided an extensive overview of power management in heterogeneous multicore systems, discussing thermal constraints, core utilization patterns, and scheduling algorithms.

From a model-centric perspective, works such as [7] and [8] proposed lightweight CNN architectures explicitly designed for resource-constrained deployments. These networks, while exhibiting reduced parameter counts, maintain competitive accuracy on standard classification benchmarks and have proven suitable for real-time applications on low-power hardware. In particular, MobileNet's depthwise separable convolution and SqueezeNet's fire modules exemplify architecture-level optimizations geared toward minimizing inference latency and energy consumption.

Furthermore, frameworks like the ARM Compute Library (ARM-CL) and its variants have facilitated the benchmarking and deployment of CNNs on embedded hardware. Prior work in [9] utilized ARM-CL to evaluate CNNs on different ARMbased platforms, revealing performance bottlenecks and identifying scenarios where GPU acceleration could offer energyefficient speedups. Enhancements to these frameworks, such as the Pipe-All methodology used in this study, introduce greater flexibility in workload partitioning, enabling more granular control over execution pipelines.

Despite these advancements, limited research has systematically evaluated the interplay between frequency scaling, CNN architecture choice, and core affinity on a unified heterogeneous platform like the Khadas VIM3. This gap motivates the present study, which aims to contribute to the growing body of work on intelligent scheduling and power-aware deep learning by providing empirical insights into how architectural and software-level controls influence runtime energy dynamics.

3. METHOD

This section outlines the experimental methodology employed to analyze the interplay between power efficiency and computational performance for several advanced Convolutional Neural Network (CNN) architectures on an embedded hardware platform. The evaluation encompasses five prominent CNN models widely adopted for image classification applications: AlexNet, GoogleNet, MobileNet, ResNet50, and SqueezeNet. These neural models were executed using the ARM Compute Library (ARM-CL) within the Amlogic A311D-based Heterogeneous Multi-Processor Systemon-Chip (HMPSoC) integrated into the Khadas VIM3 embedded board.

The utilized SoC integrates a six-core asymmetric CPU architecture conforming to ARM's big.LITTLE design paradigm. Specifically, the cluster consists of four highperformance Cortex-A73 cores (referred to as the "big" cluster) and two energy-efficient Cortex-A53 cores (termed the "LITTLE" cluster). In addition, the platform includes a dualcore Mali-G52 MP4 GPU. The upper bounds of operational frequency are defined as 2.2 GHz for the high-performance CPUs, 1.8 GHz for the low-power CPUs, and 0.8 GHz for the GPU. This hardware is supported by a 4 GB LPDDR4 DRAM module. The system environment was based on Android version 9.0, utilizing the Linux kernel version 4.9, and the experiments were conducted using ARM-CL version 21.02.

To enable simultaneous processing across heterogeneous compute units, a modified variant of the standard ARM Compute Library, named Pipe-All [10], was employed. This specialized framework introduces pipeline-based execution, facilitating concurrent inference on both CPU clusters and the GPU. Pipe-All exposes tunable parameters that allow redistribution of computational load among the big cores, LITTLE cores, and GPU, providing a mechanism for controlled workload partitioning. These capabilities were leveraged as energy modulation controls by the frequency scaling governor evaluated in this work.

In conjunction with task redistribution, Dynamic Voltage and Frequency Scaling (DVFS) was utilized as a secondary power tuning mechanism. DVFS dynamically adjusts both the voltage supply and operating frequency of processor cores at runtime, thereby optimizing energy use relative to application demands without significantly compromising response latency. The integrated governor dynamically harnessed both Pipe-All's task scheduling features and DVFS capabilities to examine the power-performance trade-offs under various configurations.

The hardware setup followed standardized interfacing procedures. A host computer operating on Ubuntu 20.04 was connected to the Khadas VIM3 board via a USB-A interface. Additional peripherals, such as power meters and serial consoles, were configured as per development documentation to enable accurate data capture during experimentation. Power and latency metrics were recorded across multiple configurations. For each CNN model, minimum, maximum, and median power values (in watts) were collected along with the mean inference latency per frame. The number of frames processed on the LITTLE cluster, big cluster, and GPU were set to 80, 100, and 100, respectively. Each processing unit was designated as the primary stage in the software pipeline during its corresponding run. To explore the frequency-scaling impact, the maximum operating frequency of both the LITTLE and big clusters was incrementally adjusted based on the board's permitted frequency spectrum. These measurements enabled the creation of visual representations — including graphs and charts — to highlight trends in energy consumption versus computational speed.

The results derived from this systematic approach present generalized conclusions on the responsiveness of power and performance metrics to software-controlled efficiency parameters within the embedded board. Such observations are valuable for shaping governor algorithms and designing energyaware execution policies suitable for resource-constrained environments.

To extend the investigation, an additional test was performed to evaluate the significance of pipeline stage ordering and its influence on overall energy efficiency and inference latency. In this case, the frequencies were statically set at 1.704 GHz for the Cortex-A73 cores and 1.2 GHz for the Cortex-A53 cores. The aim was to hold frequency variables constant while analyzing how execution order affects performance. This study was structured in stages: initially, each core was evaluated in isolation; subsequently, paired combinations were assessed; finally, all three compute resources were tested in nine distinct permutations. This multi-stage exploration aimed to extract actionable insights into optimal

workload partitioning strategies and the contribution of individual CNN components to total execution time across configurations.

4. RESULT

The experimental findings were interpreted and depicted through a series of graphical illustrations as depicted in Figure 1, a direct correlation can be observed between power usage and execution delay for the evaluated CNN models. Specifically, models such as ResNet50 and Squeeze Net follow a trend where increased computational throughput results in greater power draw.

In contrast, Figure 2 portrays an alternative relationship between latency and energy consumption across a more limited range of operational intervals. Despite the reduction in range, the observed behavior aligns consistently with the general trend reported across other figures in this dataset.

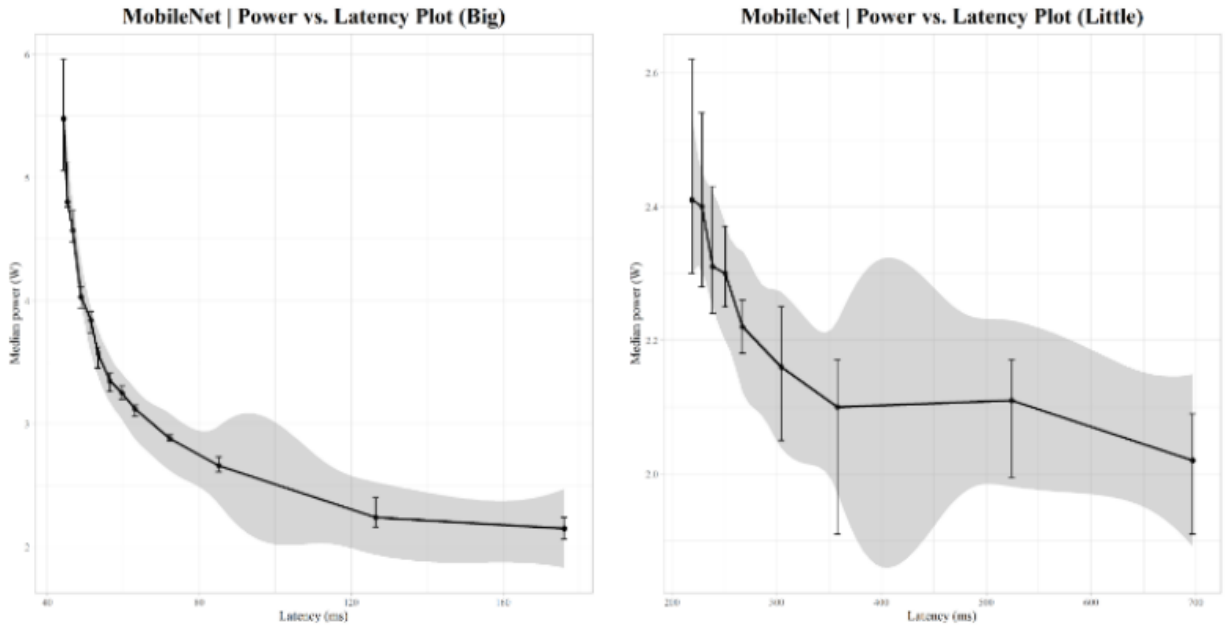

FIGURE 1. Mobile Net on the Big core

Figures 2 and 3 examine how distributing CNN computation segments across diverse processing elements affects system latency and responsiveness. The performance contrast is particularly significant when assessing execution on the high-power CPU cluster versus the low-power one. In most scenarios, excluding AlexNet, operating the CNNs on reduced frequency settings of the Big CPU results in lower latency than running them at higher performance settings on the Little CPU.

Notably, Figure 2 includes an anomalous deviation near the midpoint of the performance curve, an effect that is mirrored in the frequency response of GoogleNet at the same operational parameters.

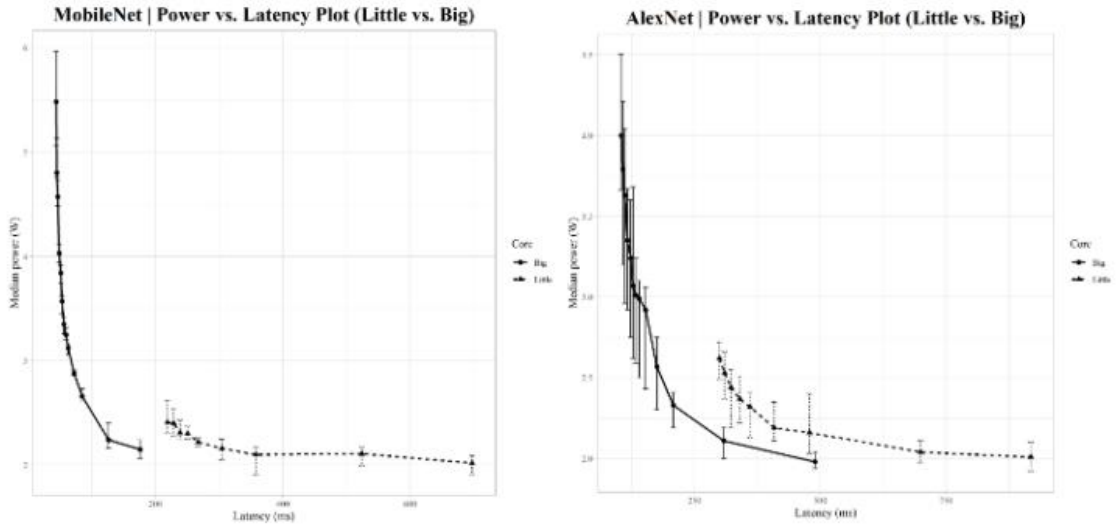

FIGURE 2. MobileNet on both CPUs

Figure 3 further quantifies computational efficiency by analyzing the throughput-to-power ratio across different architectures. This evaluation metric, expressed as η = (Frames/sec)/ Watts, provides insight into power efficiency. Given the relationship Frames/sec = 1/ Latency, it follows that:

$$\eta = 1 / (\text{Latency} \times \text{Power})$$

This enables a comparative analysis of performance per watt across execution units. It reveals that while the GPU maintains a stable efficiency baseline, the Big CPU outperforms it in scenarios demanding higher responsiveness.

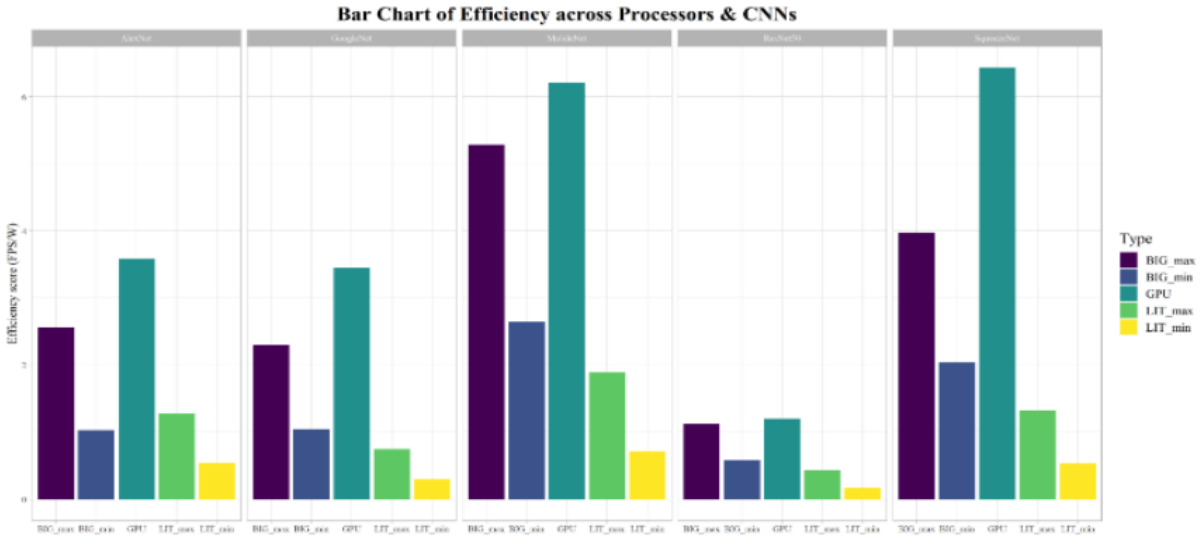

FIGURE 3. Efficiency Scores of the 5 CNNs

The secondary analysis explored how execution order influences both power consumption and computational latency when CNN components are assigned sequentially to heterogeneous processing elements. A phenomenon termed "pipeline imbalance" or "execution congestion" was frequently observed. In configurations where a weaker processing

unit precedes a more capable one, the faster unit often remains idle while awaiting input, resulting in reduced utilization and performance penalties. Conversely, when stronger components lead the pipeline, the imbalance persists as the subsequent slower stages become overloaded, producing an increase in cycle times and inefficient power usage.

Further exploration revealed that individual segments of CNNs, differing in computational intensity, significantly influence total processing time when reassigned to different cores. When an entire network was executed on a single processor, the sequence of other inactive units did not affect latency. However, when workloads were shared among two or three processing units, altering the processing sequence noticeably impacted inference duration. Specifically, placing the A53 (Little) cores before the others led to a substantial delay in completion. Meanwhile, swapping the order between the GPU and Big CPU had a minimal yet measurable influence in some cases.

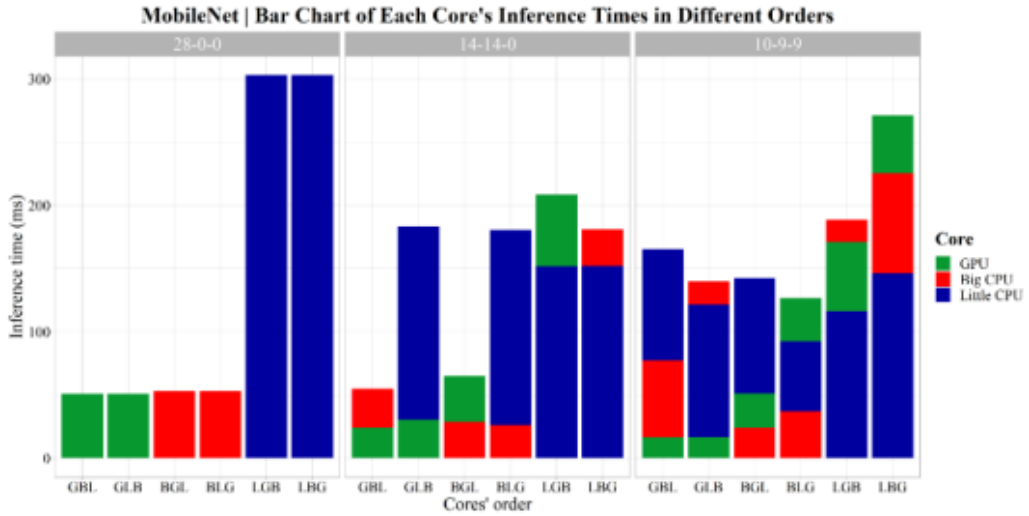

FIGURE 5. Inference time of different orders of MobileNet

From a throughput standpoint, equally partitioning computational blocks between the GPU and Big CPU generally enabled the latter to match the former in terms of execution time. However, such benefits were not uniform across models; for instance, MobileNet failed to maintain consistent latency reduction despite an equal division of workload. This inconsistency indicates that the intrinsic distribution of operations in individual CNN segments can substantially impact execution timing, motivating further investigation into how partitioning strategies should account for computational density in specific network layers. This is exemplified in Figure 6, which depicts the variability in efficiency scores across multiple execution sequences for GoogleNet.

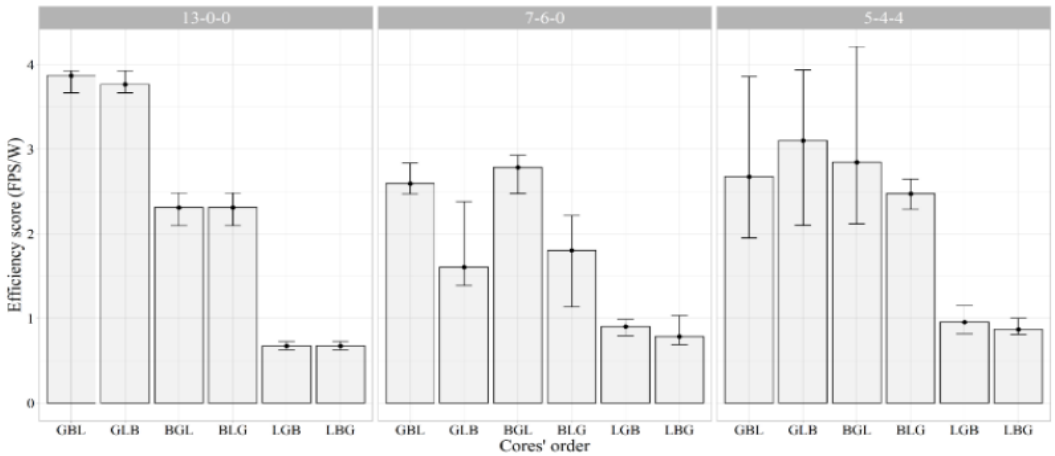

FIGURE 6. Efficiency Scores of different orders of GoogleNet

5. CONCLUSION AND DISCUSSION

The comprehensive assessment of convolutional neural networks (CNNs) on the Khadas VIM3 embedded platform underscored a distinct interdependence between computational efficiency and energy consumption. Elevation in the operating frequency of the Big cluster cores corresponded to a noticeable enhancement in frames rendered per second (FPS) and a reduction in latency per frame. However, this improvement in throughput incurred a proportional increase in power draw. This phenomenon was consistently observed in both AlexNet and GoogleNet, though GoogleNet exhibited greater energy demand and superior responsiveness relative to AlexNet.

Analysis revealed that peak frequency configurations exerted a more substantial influence on energy usage compared to intermediate or lower frequency states. This observation implies that the dynamic voltage and frequency scaling (DVFS) mechanisms and other internal power regulation features embedded in the board's control firmware exert significant authority over power consumption at elevated performance states, whereas their effectiveness diminishes in low-power modes.

A notable discovery included the emergence of inflection points in power draw at specific frequency thresholds, particularly visible on the Big core plots. This inflection was especially prominent in GoogleNet, suggesting the activation of microarchitectural features such as enhanced instruction pipelines or advanced branch prediction units within the Cortex-A73 architecture, which may not be present in the Cortex-A53 cores. These architectural optimizations likely exploit frequency-sensitive execution paths, although direct evidence correlating CNN workloads to specific microarchitectural triggers remains inconclusive based on existing literature.

Disparities among CNN models were also evident in terms of their resource utilization characteristics. For instance, MobileNet v2 consistently operated with reduced energy expenditure while delivering higher frame throughput when juxtaposed with AlexNet and GoogleNet. This trend suggests that MobileNet v2 is inherently optimized for low-power inference, making it a preferable candidate for energy-constrained environments on ARM-based heterogeneous architectures like the Khadas VIM3.

These experimental insights reinforce the significance of incorporating energy efficiency as a core consideration in neural network selection for edge deployments. The architectural diversity among CNNs translates to varied interactions with processing cores, indicating that optimal model deployment may depend not only on computational complexity but also on the targeted processor's microarchitectural traits.

Moreover, the compiled data offers a valuable foundation for identifying the most power-efficient neural networks for applications requiring a delicate balance between speed and energy conservation. Certain models demonstrate enhanced efficiency when executed on specific cores or frequencies, suggesting the potential for workload distribution strategies that allocate different layers or segments of CNNs to the most suitable processing unit in order to maximize performance per watt (F P S/P_{avg}), where P_{avg} is the average power consumption over time.

Furthermore, this study illustrates the importance of examining processor frequency scaling in conjunction with workload profiling, as alterations in clock rates can lead to significant shifts in energy-to-performance ratios. Consequently, adjusting CPU core frequency settings must be approached with a holistic understanding of model behavior to avoid inefficient configurations.

To broaden the applicability of these findings, future investigations may consider replicating this type of benchmarking on alternative platforms, including other ARM-based systems and x86 embedded devices. Such comparative studies could uncover generalized patterns or unique platform-specific behaviors that would assist in guiding model selection and deployment strategies across heterogeneous hardware ecosystems.

The visual plots and tabulated data presented throughout this study allow for direct comparison of CNN performance metrics as a function of processor frequency and power draw. These graphical representations make it possible to

detect which neural architectures provide optimal efficiency under constrained power budgets and which frequency settings are most conducive to maintaining a favorable trade-off between processing speed and energy utilization.

REFERENCES

[1]. N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, and F. Kawsar, "DeepX: A software accelerator for low-power deep learning inference on mobile devices," in Proc. 15th Int. Conf. Information Processing in Sensor Networks (IPSN), ACM, 2016.

[2]. S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," arXiv preprint arXiv:1510.00149, 2015.

[3]. A. Molnos, P. Pop, and P. Eles, "Energy-efficient DVFS scheduling for ´ multi-core real-time systems using reinforcement learning," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 36, no. 11, pp. 1758– 1771, 2017.

[4]. A. Pathak and M. Sharma, "Energy performance evaluation of dynamic frequency scaling in multicore processors," in Proc. Int. Conf. High Performance Computing and Simulation (HPCS), IEEE, pp. 488–494, 2012.

[5]. J. Choi, W. Park, Y. Choi, and S. Yoo, "Understanding the performance and energy efficiency of ARM big.LITTLE architecture for mobile applications," in Proc. Design Automation Conference (DAC), IEEE, pp. 1–6, 2013.

[6]. S. Mittal, "A survey of techniques for improving energy efficiency in embedded computing systems," Int. J. Comput. Aided Eng. Technol., vol. 8, no. 6, pp. 851–874, 2016

[7]. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[8]. 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 1MB model size," arXiv preprint arXiv:1602.07360, 2016.

[9]. M. Li, J. Yang, B. Wang, and J. Liu, "Performance analysis of convolutional neural networks on ARM-based embedded platforms using the ARM Compute Library," in Proc. IEEE 3rd ITNEC, vol. 1, pp. 392–396, 2019.

[10]. M. Aghapour, D. Sengupta, J. Kołodziej, and A. Majumdar, "Pipe-All: Fine-grained pipeline-based parallelism for deep learning on ARM-based embedded systems," in Proc. 22nd IEEE/ACM Int. Symp. Cluster, Cloud and Internet Computing (CCGrid), IEEE, pp. 284–293, 2022.

