Benchmarking Edge AI Platforms for High-Performance ML Inference

Rakshith Jayanth*, Neelesh Gupta*, Viktor Prasanna
Ming Hsieh Department of Electrical and Computer Engineering
University of Southern California
Los Angeles, USA
{jayanthr, neeleshg, prasanna}@usc.edu

***Abstract*—Edge computing's growing prominence, due to its ability to reduce communication latency and enable real-time processing, is promoting the rise of high-performance, heterogeneous System-on-Chip solutions. While current approaches often involve scaling down modern hardware, the performance characteristics of neural network workloads on these platforms can vary significantly, especially when it comes to parallel processing, which is a critical consideration for edge deployments. To address this, we conduct a comprehensive study comparing the latency and throughput of various linear algebra and neural network inference tasks across CPU-only, CPU/GPU, and CPU/NPU integrated solutions. We find that the Neural Processing Unit (NPU) excels in matrix-vector multiplication (58.6% faster) and some neural network tasks (3.2× faster for video classification and large language models). GPU outperforms in matrix multiplication (22.6% faster) and LSTM networks (2.7× faster) while CPU excels at less parallel operations like dot product. NPU-based inference offers a balance of latency and throughput at lower power consumption. GPU-based inference, though more energy-intensive, performs best with large dimensions and batch sizes. We highlight the potential of heterogeneous computing solutions for edge AI, where diverse compute units can be strategically leveraged to boost accurate and real-time inference.**

***Index Terms*—neural processing unit, heterogeneous computing, edge computing, neural network inference, linear algebra**

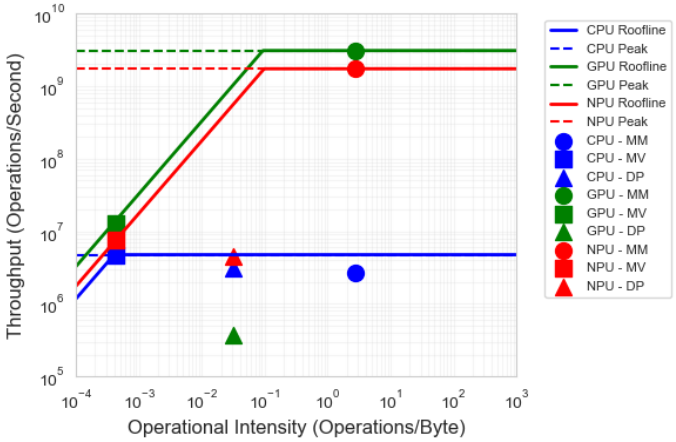

Fig. 1: Roofline model comparing performance of CPU, CPU/GPU, and CPU/NPU architectures for linear algebra operations. The plot shows a consistent performance hierarchy: CPU (lowest), CPU/NPU (middle), and CPU/GPU (highest) across both memory-bound and compute-bound regions.

I. Introduction

The rapid expansion of Artificial Intelligence (AI) applications has intensified the need for inference capabilities at the edge [1]–[7], or in resource-constrained environments. Edge computing brings data processing closer to the source, reduces latency, enhances privacy, and enables real-time decision-making. This shift towards edge AI has become crucial in various domains, including autonomous vehicles [8], smart cities [9], industrial IoT [10], [11], and mobile devices [12], where rapid and efficient on-device inference is essential.

To meet these demands, semiconductor manufacturers are swiftly developing increasingly complex System-on-Chip (SoC) platforms that integrate traditional CPUs with specialized accelerators such as GPUs and NPUs. Exemplary designs include Intel's Core Ultra Processor [13], also known as Intel AIPC, with an integrated NPU, Apple's M-series chips [14] with dedicated neural engines, and Qualcomm's Snapdragon platforms [15] featuring AI optimized Digital Signal Processors. These advancements significantly boost on-device AI performance and energy efficiency, expanding computing capabilities at the edge.

As sophisticated SoCs become more prevalent, comprehensive benchmarking becomes crucial for system designers, software developers, and end-users to maximize hardware capabilities. Benchmarking exposes processing unit strengths, guiding hardware selection and workload optimization on heterogeneous platforms. This study focuses on benchmarking two critical aspects of Machine Learning (ML) workloads: fundamental linear algebra operations and Deep Neural Network (DNN) applications. Linear algebra operations, such as matrix multiplication, [16], [17] form the backbone of many ML algorithms and aid in understanding the raw computational capabilities of different compute platforms. DNN applications, including video classification [18] and text generation [19], represent real-world AI tasks that leverage these fundamental operations in complex ways. By evaluating performance across these workloads, we provide a comprehensive view of how different compute units handle the diverse computational requirements of modern AI applications at the edge.

* These authors contributed equally.

Our main contributions are summarized as follows:

- We conduct a benchmark study that analyses the performance of fundamental linear algebraic operations and neural network models on an integrated heterogeneous platform consisting of CPU, GPU and NPU.
- Our results show that NPU excels in matrix-vector multiplication, reducing latency by 58.54% compared to GPUs. Conversely, GPU outperforms in matrix multiplication, offering 22.6% lower latency and 2× higher throughput than NPU. For dot product, CPU demonstrates the lowest latency among all platforms.
- In model inference, NPU outperforms GPU by 3.2× for LLM tasks, while GPU surpasses NPU by 2.7× for LSTM models. For video classification, NPU performance remains consistent across batch sizes, whereas GPU performance scales, favoring larger batches.
- We conclude that NPU excels in LLM inference and memory-bound operations such as matrix-vector multiplication due to efficient DMA (Direct Memory Access) utilization, while GPU is more efficient for LSTM models and compute-bound operations such as matrix multiplication. CPU performs best for low-complex compute operations such as dot-product as they avoid additional system memory accesses. NPU offers consistent performance across varied batch sizes, but GPU is superior for large-scale batch processing.

II. Background

Our effort focuses on benchmarking applications using integrated heterogeneous platforms. An integrated heterogeneous platform, in simple terms, is a combination of accelerator platforms implemented as a System-on-Chip (SoC). For this work, we consider three key components: CPU, GPU, and NPU (Neural Processing Unit). In the following section, we provide an overview of NPU and explain the interactions between these devices. Understanding these interactions is crucial as it forms the foundation for our benchmarking efforts, allowing us to comprehensively evaluate performance across the diverse computational resources available in modern SoCs.

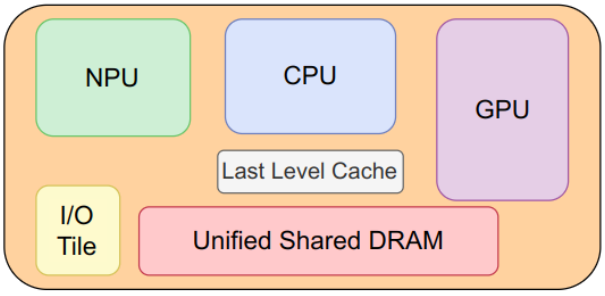

Fig. 2: Integrated Heterogeneous System

A. Neural Processing Unit

NPUs [20] are specialized processors designed specifically for AI workloads, excelling particularly at neural network inference and training. Their integration into modern SoCs has become increasingly common. NPUs offer superior energy efficiency compared to both CPUs and GPUs, making them primary targets for edge computing environments where power constraints are critical. A key feature of NPU architectures is their ability to exploit quantization, with many modern designs supporting mixed-precision arithmetic to balance performance and accuracy. Typically based on systolic array architectures, NPUs provide massive parallelism and maximize data reuse, enabling highly efficient execution of AI algorithms. These characteristics position NPUs as crucial components in the evolving landscape of heterogeneous computing for AI applications. Since our operations are executed on Intel NPU, here is a brief architectural overview of the same. Intel NPU is integrated into the Intel AIPC, and the compute acceleration is supported by scalable NCEs (Neural Compute Engines). These NCEs have AI acceleration blocks that perform matrix operations efficiently. The NPU is equipped with SHAVE (Streaming Hybrid Architecture Vector Engines), which performs parallel computing on general-purpose tasks. DMA (Direct Memory Access) engines, which have been integrated into NPU, perform efficient transfer of data from system memory to software-managed cache.

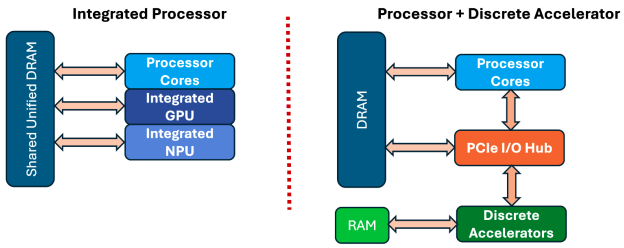

Fig. 3: System Interaction With Integrated Accelerators and Discrete Accelerators

B. Interaction Between The Compute Platforms

Understanding the interaction between CPU, GPU, and NPU is crucial as it forms the basis for our benchmarking results. Most modern SoCs, such as Intel AIPC, have adopted a shared memory architecture unified across these three devices as shown in Fig. 3. In this unified memory system, each platform communicates by writing to corresponding shared spaces in memory. This approach offers a major advantage: seamless communication between platforms without unnecessary data transfers. In contrast, systems with discrete accelerators like NVIDIA GPUs require data transfer from the SoC's memory to the accelerator's memory, incurring significant latency. However, integrated processors have a disadvantage in terms of limited memory bandwidth, primarily due to CPU constraints. Discrete accelerators, on the other hand, can leverage HBMs (High Bandwidth Memories), offering potentially higher throughput for memory-intensive tasks. These architectural differences significantly impact performance and efficiency in heterogeneous computing environments.

III. Related Works

Recent benchmarking studies of linear algebra operations and neural network inference on heterogeneous platforms have advanced significantly. This progress is driven by increasingly complex edge computing devices and growing demand for efficient edge AI processing. Most studies have focused on evaluating performance across CPUs, GPUs, and specialized accelerators such as NPUs. These investigations discuss the strengths and limitations of the hardware. They analyze performance, energy efficiency, memory usage, and scalability across various workloads. The following sections review key contributions in the benchmarking of fundamental linear algebra operations and neural network models.

A. Linear Algebra

There exist multiple works on analyzing linear algebra operations, specifically BLAS (Basic Linear Algebra Subprograms) operations, on heterogeneous platforms. Works on analyzing BLAS operations [21], [22] concentrate on performance analysis on GPU architectures and focus on evaluation of large matrices. The analysis is performed as part of model development for performance improvement and is conducted on GPUs. Other dedicated works on benchmarking BLAS operations [23]–[25] concentrate their efforts on level-3 BLAS operations such as matrix multiplication and use high-performing GPUs as their platform.

However, all the aforementioned works make use of discrete and high-performing GPUs and not integrated GPUs, nor do they evaluate any operations on NPUs. Additionally, these works do not perform a full-scale analysis of all levels of BLAS operations as they only work with level-3 operations. Therefore, their analysis is not applicable to understanding the performance of these operations on edge devices that comprise CPUs, integrated GPUs, and NPUs. In contrast, our work performs a full-scale analysis of all levels of BLAS operations on CPU, GPU, and NPU devices that are integrated into a single-chip edge system.

B. Neural Networks

Comprehensive analysis of neural network models has seen a lot of engagement in prior works on various platforms, including edge devices. Notably, detailed benchmark analysis of DNN models [26]–[28] was performed on high-end CPU and GPU architectures. However, this analysis is not suited for edge applications as edge systems typically do not deal with high-performing GPUs and CPUs. We also find prior works that work on the analysis of ML workloads performed on Android devices [29] that succeed in making an overall performance comparison among models but do not discuss platform heterogeneity. Other prior works on android [30], though considering the heterogeneous performance on CPU and GPU, concentrate on specific workloads such as CNN and do not evaluate performance on NPU. Analysis of workloads on smartphones [31] not only executed on CPU and GPU workloads but also included performance evaluation on NPU. However, the NPU used in the analysis did not support quantization. Though the more recent work on benchmarking effort [12] accounts for performance on all the platforms, it does not establish the relation between the performance of the models and the fundamental linear algebraic operations.

On the other hand, our work not only gives a comprehensive analysis of neural network models across integrated compute units but also tries to establish its relation with the performance of fundamental linear algebraic operations. To the best of our knowledge, our work is one of the first to present a complete analysis of benchmarking fundamental linear algebraic operations and ML models on edge platforms.

IV. Evaluation

We benchmark key linear algebra operations and neural network models for inference. We are using Intel AIPC, which provides an integrated heterogeneous platform consisting of CPU, GPU, and NPU. Our benchmarking operation involves the use of multiple software frameworks that yield optimized results on Intel devices. In this section, we will provide a brief overview of these software frameworks and their workflows. Additionally, we will discuss the tasks involved along with the performance metrics we use for evaluation.

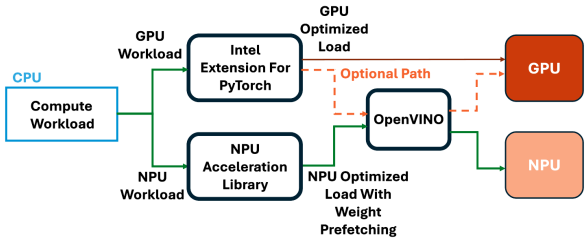

Fig. 4: Workflow diagram of Intel frameworks illustrating coordination and workload distribution.

A. Software Framework

For our experiments, we used two frameworks offered by Intel: OpenVINO and IPEX (Intel Extension for Pytorch). Additionally, for computations on NPU, we utilized Intel's NPU acceleration library, which is based on OpenVINO. The workflow of these frameworks is illustrated in Fig. 4. The OpenVINO framework coordinates communication and workload distribution across all memory and compute platforms. IPEX, an independent framework, can be used independently or in conjunction with OpenVINO. It offers optimizers that apply Intel device-specific optimizations to PyTorch models and distributes workloads between CPU and GPU by performing dynamic task scheduling. Both IPEX and OpenVINO implement lazy evaluation, which ensures that operations are executed only when results are required, thus reducing unnecessary memory access. Also, they ensure that operations on different platforms are executed asynchronously and tasks are scheduled dynamically. The NPU acceleration library optimizes NPU usage by exploiting its features of weight prefetching and efficient DMA accesses between cache and system memory, along with maximizing compute utilization.

TABLE I: Suite of Linear Algebra Tasks

Task	BLAS Level	Description
Matrix Multiplication	3	Multiply two matrices **A** ($N \times N$) and **B** ($N \times N$).
Matrix-Vector Multiplication	2	Multiply a matrix **A** ($N \times N$) with a vector **x** ($N \times 1$).
Dot Product	1	Compute the dot product of two vectors **x** and **y**, both of length N.

TABLE II: Suite of Neural Network Inference Tasks

Model	Input Type	Parameters	Description
MobileNetV2 [32]	Video	6.9M	Efficient CNN for mobile and embedded vision applications.
LSTM [33]	Time series	1.6M	Processes input in the forward direction. Captures past context.
TinyLlama [19]	Text	1.1B	Compact transformer-based language model using grouped-query attention.

TABLE III: Processor Specifications

Feature	CPU	GPU	NPU
Operating Frequency	5 GHz	2.3 GHz	1 GHz
Power	35 - 65 W	75 W	35 W
Execution Units (EUs)	16 Cores	128 EUs	4096 MAC Units

B. Benchmarking Linear Algebra

We analyze three fundamental linear algebraic operations: dense matrix multiplication, dense matrix-vector multiplication, and vector-vector dot product (Table I). The motivation for the choice of these operations was based on the fact that they are the core operations in the neural network models. Analysis of these results helps extrapolate the results obtained for neural network models. Along with this, these operations form a comprehensive set evaluating all levels of BLAS operation. For matrix and matrix-vector multiplication, we use square matrices of size $N \times N$, with N ranging from 16 to 1024, typically focusing on 512 to 1024. We exclusively use 16-bit floating point (FP-16) operations, as NPU only supports this precision, ensuring uniform analysis.

Performance Metrics. The metrics we use for these operations are latency and average throughput.

C. Benchmarking Neural Network Inference

We analyze the performance of a CNN-based model for video classification, an LSTM model, and a decoder-only transformer model for LLM inference, as mentioned in Table II. The computational diversity, along with differences in the regularity of memory accesses, makes these models ideal candidates for benchmarking across various hardware platforms. Specifically, the classification model (MobileNetV2) demonstrates the impact of batch processing, the LSTM model provides insights into performance with irregular memory accesses, and the LLM model (LLaMA) demonstrates the interplay between different operations, as its prefill stage is dominated by matrix multiplications while the repeated decode stages primarily consist of matrix-vector operations. For video classification analysis, we analyzed performance by varying batch sizes. Along with floating point operations, we measure the performance on quantized INT-8 weights as well. Execution on NPU took place with the OpenVINO framework for video classification, while we used the NPU library constructs for the LSTM and LLM models. Similar to linear algebraic operations, FP-16 was used across all platforms for analyzing the neural network models.

Performance Metrics. For LSTM and LLM models, **inference latency** will be the **performance metric**, and for video classification, along with inference latency, **FPS (Frames Per Second)** will be used as a measure of throughput. Inference latency refers to the total time a model takes to generate output from when an input is provided. FPS is the number of individual images that a device can process in one second.

V. Results

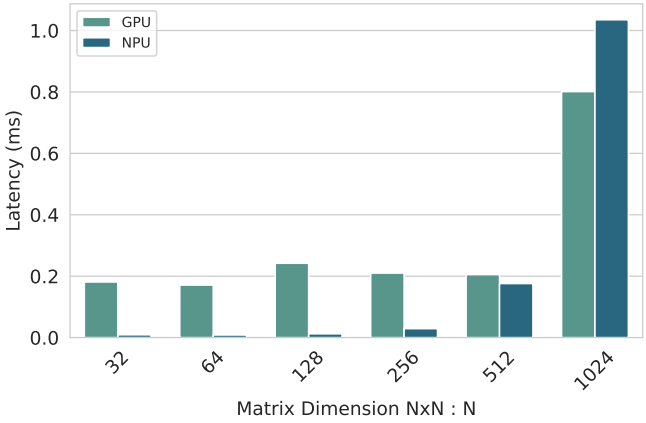

Fig. 5: Latency for Matrix Multiplication

In this section, we evaluate the performance of three linear algebra primitives and three neural network models that were chosen based on their computational and model diversity, respectively, on the integrated CPU, GPU, and NPU compute platform of Intel AIPC.

With respect to matrix multiplication, as shown in Table IV and Fig. 5, the NPU demonstrates superior performance for smaller matrices compared to the GPU. However, as matrix sizes increase, GPU computation latency is significantly reduced compared to NPU by 22.6% along with 2× higher throughput. This trend suggests that GPU requires large matrices for complete resource utilization as the operation is compute-bound. For smaller matrices, the operation becomes memory-bound as the memory access overhead supersedes the computational requirement. NPU, due to the presence of DMA, achieves better performance than GPU. Therefore,

TABLE IV: Performance of Linear Algebra Primitives (N=1024)

Metric	Matrix Multiplication			Matrix-Vector Multiplication			Dot Product		
	CPU	GPU	NPU	CPU	GPU	NPU	CPU	GPU	NPU
Latency (ms)	1219.00	**0.80**	1.04	0.86	0.20	**0.08**	**0.17**	19.40	0.52
Avg Throughput (GFLOPS)	2.56	**2947.37**	1653.65	4.54	**12.42**	7.14	2.94	0.36	**4.32**

TABLE V: Inference Latency (ms) of MobileNetV2

Batch Size	FP-16 Weights		INT-8 Weights	
	GPU	NPU	GPU	NPU
1	4.95	**1.73**	5.37	**1.33**
2	4.51	**1.61**	2.70	**1.37**
4	2.21	**1.49**	1.48	**1.26**
8	**1.52**	1.59	**0.90**	1.18
16	**1.20**	1.35	**0.75**	1.16
32	**1.12**	1.49	**0.61**	1.26

NPU is preferred for smaller matrix multiplies for its better performance due to efficient memory access, and GPU is preferred for larger ones for better performance.

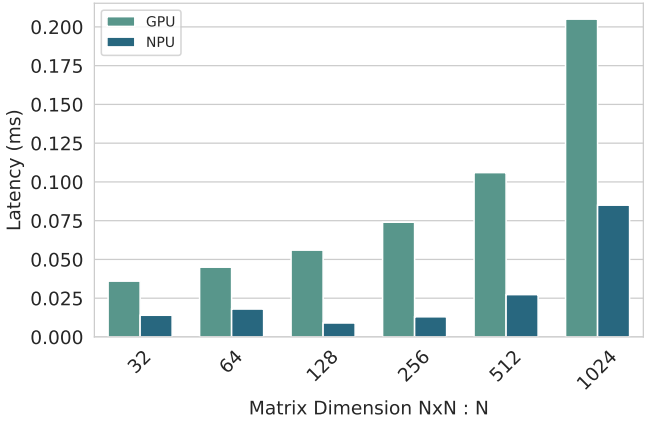

Fig. 6: Latency for Matrix-Vector Multiplication

For matrix-vector multiplication, as shown in Fig. 6, the NPU consistently outperforms the GPU, offering an approximate 2× speedup (58.54% reduction in latency) across all matrix sizes. This is because the operation being memory bound as data reuse is minimal, the DMA support in NPU can efficiently control memory accesses in comparison to GPU. Interestingly, the NPU's latency values show a decrease when moving from matrix sizes of 64×64 to 128×128 before increasing again for larger sizes, which indicates that the NPU pipeline is optimized for matrices of size 128×128. This performance characteristic indicates that NPU should be preferred over GPU for matrix-vector multiplication. Therefore, NPU should be the preference over GPU for matrix-vector multiplication operations.

In the case of dot product operations, as shown in Fig. 7, the NPU's performance is significantly better than that of the GPU. This substantial difference can be attributed to the extremely limited data reuse in this operation, along with high synchronization overhead of the final sum, making it evidently

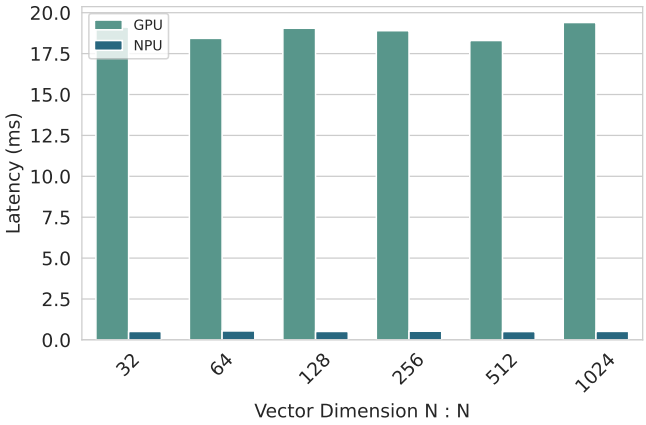

Fig. 7: Latency for Dot Product

a memory-bound problem that the NPU can efficiently handle in comparison to GPU. Another aspect of this operation is its simplicity and the highly sequential nature of computation, which makes it an ideal operation for the CPU, along with the fact that execution on the CPU eliminates additional memory accesses done by GPU and NPU. This explanation is well justified by the latency for dot-product as in Table IV, which indicates CPU has lower latency than NPU. Therefore, CPU is the choice over GPU and NPU for dot product operations.

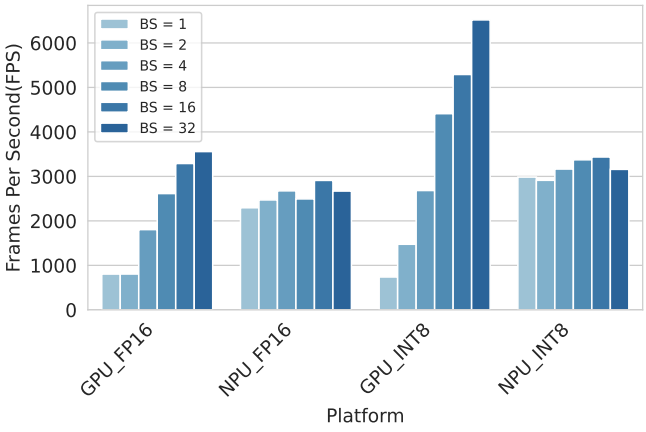

Fig. 8: Throughput (FPS) with Varying Batch size for MobileNetV2 Video classification Inference

Video classification using MobileNetV2 reveals interesting performance dynamics between NPU and GPU as shown in Table V and Fig. 8. For non-quantized FP-16 weights and a batch size of one, the NPU demonstrates nearly 3× lower

TABLE VI: Inference Latency of Neural Network models

Model	GPU	NPU
MobileNetV2	4.95 ms	**1.73 ms**
LSTM	**1.48 ms**	4.10 ms
TinyLlama	8.30 sec	**2.49 sec**

inference latency and 3× higher throughput compared to the GPU. As batch size increases, the NPU's latency remains consistent while the GPU's latency decreases dramatically, eventually outperforming the NPU for larger batch sizes. This result shows a trend that is very similar to that of matrix multiplication and emphasizes the analysis done previously. GPU performs the best in a compute-bound environment, which is established with higher batch sizes, resulting in increased resource utilization with a corresponding increase in throughput, and NPU performs the best in a memory-bound environment, which is the case with smaller batches. When using quantized weights, the relative performance between NPU and GPU remains largely the same, with some nuances in performance for different batch sizes. These findings suggest that for consistent latency and classification with small batch sizes, inference on NPU is preferred, while GPU offers better performance with larger batch sizes. With the desire for consistent latency and for classification with small batch sizes, inference on NPU is preferred as compared to GPU, which only offers better performance with larger batch sizes.

For LSTM and LLM inference, we observe from Table VI that LSTM, being a smaller model, exhibits low latency on both NPU and GPU. However, the GPU slightly outperforms NPU for LSTM inference. Conversely, for LLM inferencing, the NPU demonstrates remarkably efficient performance, nearly 4× better than the GPU. This result aligns with our findings from matrix multiplication and matrix-vector multiplication tests. LLM inferencing primarily consists of prefill and decode stages, which largely rely on these operations respectively. Since the decode stage, dominated by matrix-vector multiplication, is executed multiple times compared to the single execution of the prefill stage, NPU emerges as the most suitable candidate for LLM inference. This is consistent with NPU's superior performance in matrix-vector operations. Based on these observations, NPU is the preferred choice for LLM inference in comparison to GPU. For LSTM inference, GPU holds a slight edge in performance over NPU. Despite LSTM operations being primarily matrix-vector multiplies, NPU demonstrates degraded performance in comparison to GPU. This degradation can be attributed to the irregularity in memory accesses performed by LSTM. The DMA in NPU can be effective in mitigating a memory-bound problem only when the accesses are regular; otherwise, it results in reduced efficiency as the DMA will have to undergo a new setup for every new access. Therefore, for LLM inference, NPU is preferred, and for LSTM inference, GPU is preferred.

Our analysis reveals that NPU offers significant advantages in power efficiency compared to GPU for similar performance outcomes. While achieving comparable or better results across various tasks, the NPU consistently operates at less than half the peak power consumption of the GPU, drawing only 35 watts compared to the GPU's 75 watts. This substantial reduction in power usage makes NPU-based solutions particularly attractive for edge computing and mobile applications, where energy efficiency is crucial. The NPU's ability to deliver high performance at lower power consumption demonstrates its potential to enable more sustainable and cost-effective AI deployments in resource-constrained environments.

VI. Conclusion

Our benchmark study on Intel AIPC, a heterogeneous platform with CPU, GPU, and NPU, revealed task-specific performance variations. NPU excelled in matrix-vector multiplication (58.54% faster than GPU) and large language models (3.2× speedup). GPU outperformed in large matrix multiplications (22.6% faster, 2× throughput) and LSTM networks (2.7× speedup). CPU was optimal for simple operations like dot product. For video classification, NPU performed better with small batch sizes, while GPU excelled with larger batches. We highlight the importance of strategic compute unit selection in heterogeneous computing for edge AI applications.

Our results can be extrapolated to other heterogeneous System-on-Chips with CPU/GPU/NPU compute units for several reasons. First, we utilize OpenVINO and PyTorch, which convert neural networks to an intermediate representation, ensuring portability across platforms. Second, OpenVINO's just-in-time compilation allows for performance consistency across hardware platforms with similar specifications. Third, the NPU's Streaming Hybrid Architecture Vector Engine architecture shares fundamental principles with other NPU designs, such as AMD's XDNA, Apple's Neural Engine, and Qualcomm's Hexagon, all optimized for parallel processing and efficient matrix operations. This architectural similarity suggests comparable performance characteristics across different NPU implementations in edge AI deployments.

Our results motivate the exploration of dual-compute solutions, where multiple devices can be leveraged simultaneously to maximize performance and energy efficiency. Future research directions include benchmarking energy consumption to optimize the performance-efficiency trade-off and extending our study to multi-platform heterogeneous computing.

Acknowledgment

This work was supported by Army Research Office (ARO) under award number W911NF2220159 and National Science Foundation (NSF) under grants under award numbers SaTC-2104264 and CNS-2009057.

References

[1] D. Xu, T. Li, Y. Li, X. Su, S. Tarkoma, T. Jiang, J. Crowcroft, and P. Hui, "Edge intelligence: Empowering intelligence to the edge of network," *Proceedings of the IEEE*, vol. 109, no. 11, pp. 1778–1837, 2021.

[2] X. Li, Y. Li, Y. Li, T. Cao, and Y. Liu, "Flexnn: Efficient and adaptive dnn inference on memory-constrained edge devices," in *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, 2024, pp. 709–723.

[3] B. Liu, Z. Luo, H. Chen, and C. Li, "A survey of state-of-the-art on edge computing: Theoretical models, technologies, directions, and development paths," *IEEE Access*, vol. 10, pp. 54 038–54 063, 2022.

[4] D. Xu, T. Li, Y. Li, X. Su, S. Tarkoma, T. Jiang, J. Crowcroft, and P. Hui, "Edge intelligence: Architectures, challenges, and applications," *arXiv preprint arXiv:2003.12172*, 2020.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[6] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[7] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[8] S. Garg, A. Singh, S. Batra, N. Kumar, and L. T. Yang, "Uav-empowered edge computing environment for cyber-threat detection in smart vehicles," *IEEE network*, vol. 32, no. 3, pp. 42–51, 2018.

[9] L. U. Khan, I. Yaqoob, N. H. Tran, S. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge-computing-enabled smart cities: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 200–10 232, 2020.

[10] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. Khan, and S. K. Das, "Edge-computing-driven internet of things: A survey," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–41, 2022.

[11] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: A survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.

[12] S. P. Baller, A. Jindal, M. Chadha, and M. Gerndt, "Deepedgebench: Benchmarking deep neural networks on edge devices," in *2021 IEEE International Conference on Cloud Engineering (IC2E)*, 2021, pp. 20–30.

[13] W. Gomes, S. Morgan, B. Phelps, T. Wilson, and E. Hallnor, "Meteor lake and arrow lake intel next-gen 3d client architecture platform with foveros," in *2022 IEEE Hot Chips 34 Symposium (HCS)*. IEEE Computer Society, 2022, pp. 1–40.

[14] D. Banerjee, "A microarchitectural study on apple's a11 bionic processor," *Arkansas State University: Jonesboro, AR, USA*, 2018.

[15] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, and J. M. Alcaraz-Calero, "Smartphone-based real-time object recognition architecture for portable and constrained systems," *Journal of Real-Time Image Processing*, vol. 19, no. 1, pp. 103–115, 2022.

[16] C. S. Mummidi, V. C. Ferreira, S. Srinivasan, and S. Kundu, "Highly efficient self-checking matrix multiplication on tiled amx accelerators," *ACM Transactions on Architecture and Code Optimization*, vol. 21, no. 2, pp. 1–22, 2024.

[17] G. Hu and C. Saeli, "Enhancing deep edge detection through normalized hadamard-product fusion," *Journal of Imaging*, vol. 10, no. 3, p. 62, 2024.

[18] A. Rehman and S. B. Belhaouari, "Deep learning for video classification: A review," *Authorea Preprints*, 2023.

[19] P. Zhang, G. Zeng, T. Wang, and W. Lu, "Tinyllama: An open-source small language model," 2024. [Online]. Available: https://arxiv.org/abs/2401.02385

[20] J.-W. Jang, S. Lee, D. Kim, H. Park, A. S. Ardestani, Y. Choi, C. Kim, Y. Kim, H. Yu, H. Abdel-Aziz, J.-S. Park, H. Lee, D. Lee, M. W. Kim, H. Jung, H. Nam, D. Lim, S. Lee, J.-H. Song, S. Kwon, J. Hassoun, S. Lim, and C. Choi, "Sparsity-aware and re-configurable npu architecture for samsung flagship mobile soc," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 15–28.

[21] M. J. Anderson, D. Sheffield, and K. Keutzer, "A predictive model for solving small linear algebra problems in gpu registers," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium*. IEEE, 2012, pp. 2–13.

[22] A. Abdelfattah, S. Tomov, and J. Dongarra, "Matrix multiplication on batches of small matrices in half and half-complex precisions," *Journal of Parallel and Distributed Computing*, vol. 145, pp. 188–201, 2020.

[23] R. Nath, S. Tomov, and J. Dongarra, "An improved magma gemm for fermi graphics processing units," *The International Journal of High Performance Computing Applications*, vol. 24, no. 4, pp. 511–515, 2010.

[24] T. Gautier and J. V. F. Lima, "Evaluation of two topology-aware heuristics on level- 3 blas library for multi-gpu platforms," in *2021 SC Workshops Supplementary Proceedings (SCWS)*, 2021, pp. 12–22.

[25] L. Wang, W. Wu, Z. Xu, J. Xiao, and Y. Yang, "Blasx: A high performance level-3 blas library for heterogeneous multi-gpu computing," in *Proceedings of the 2016 International Conference on Supercomputing*, ser. ICS '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2925426.2926256

[26] M. Almeida, S. Laskaridis, I. Leontiadis, S. I. Venieris, and N. D. Lane, "Embench: Quantifying performance variations of deep neural networks across modern commodity devices," in *The 3rd international workshop on deep learning for mobile systems and applications*, 2019, pp. 1–6.

[27] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE access*, vol. 6, pp. 64 270–64 277, 2018.

[28] R. Hadidi, J. Cao, M. Woodward, M. S. Ryoo, and H. Kim, "Distributed perception by collaborative robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3709–3716, 2018.

[29] C. Luo, F. Zhang, C. Huang, X. Xiong, J. Chen, L. Wang, W. Gao, H. Ye, T. Wu, R. Zhou *et al.*, "Aiot bench: towards comprehensive benchmarking mobile and embedded device intelligence," in *Benchmarking, Measuring, and Optimizing: First BenchCouncil International Symposium, Bench 2018, Seattle, WA, USA, December 10-13, 2018, Revised Selected Papers 1*. Springer, 2019, pp. 31–35.

[30] J. Hanhirova, T. Kämäräinen, S. Seppälä, M. Siekkinen, V. Hirvisalo, and A. Ylä-Jääski, "Latency and throughput characterization of convolutional neural networks for mobile computer vision," in *Proceedings of the 9th ACM Multimedia Systems Conference*, 2018, pp. 204–215.

[31] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. Van Gool, "Ai benchmark: Running deep neural networks on android smartphones," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, nov 1997. [Online]. Available: https://doi-org.libproxy1.usc.edu/10.1162/neco.1997.9.8.1735