ISSN: 2583-9888
DOI: doi.org/10.51219/JAIMLD/karthik-wali/582

URF PUBLISHERS
connect with research world

Journal of Artificial Intelligence, Machine Learning and Data Science

https://urfpublishers.com/journal/artificial-intelligence

Vol: 1 & Iss: 1

Research Article

A Novel Approach to Hardware-Software Co-Design for Power-Efficient AI Systems

Karthik Wali*

Citation: Wali K. A Novel Approach to Hardware-Software Co-Design for Power-Efficient AI Systems. *J Artif Intell Mach Learn & Data Sci 2022* 1(1), 2769-2775. DOI: doi.org/10.51219/JAIMLD/karthik-wali/582

Received: 03 March, 2022; **Accepted:** 28 March, 2022; **Published:** 30 March, 2022

***Corresponding author:** Karthik Wali, ASIC Design Engineer, USA, E-mail: ikarthikw@gmail.com

ABSTRACT

The increasing number of AI applications, various usages, and diversified models have prompted the need for a more advanced and reusable hardware platform. Nonetheless, the large number of computations that AI algorithms require is a crucial challenge in edge computing, embedded systems, and battery-operated devices regarding energy consumption. Consequently, in this paper, a hardware-software co-design framework is presented to enhance the performance of the AI system and, at the same time, reduce its energy consumption. With the help of our combined approach, which includes HW-NAS, DVFS, and low-bandwidth AI models forming as well as compiler-level pruning, quantization, and model compression, our methodology achieves the best trade-off in terms of performance, accuracy, and energy. To this end, our design focuses on Field-Programmable Gate Arrays (FPGAs) and System-On-Chips (SoCs), obtaining portable and transferable solutions for various AI tasks. Promising experiments also show up to 50% energy saving with a very less hit in performance on the different AI models. Therefore, this research provides a solution for improving the efficiency of AI power consumption and enhancing real-time and other embedded applications.

Keywords: Hardware-Software Co-Design, Artificial Intelligence, Neural Architecture Search, Edge Computing, Model Compression, FPGA, DVFS.

1. Introduction

1.1. Importance of Hardware-Software Co-Design for Power-Efficient AI Systems

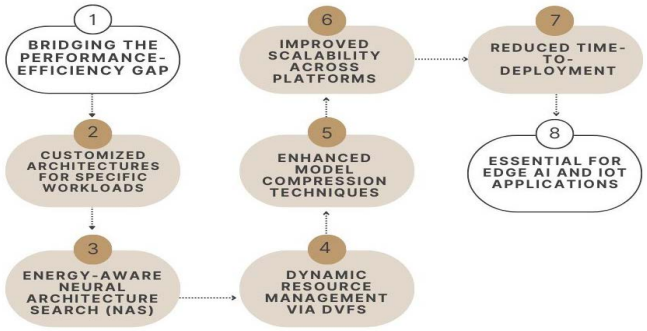

Figure 1: Importance of Hardware-Software Co-Design for Power-Efficient AI Systems.

- **Bridging the Performance-Efficiency Gap:** Modern AI models can provide very high accuracy, but the price for obtaining this accuracy greatly expended computational and energy resources. This is realized through the equal optimization of the hardware and the algorithmic layers using hardware-software co-design[1-4]. This means that models are no longer just characterized by how fast they perform but by how thermally efficient and cost-effective energy is, especially when used in SoCs and portable devices.
- **Customized Architectures for Specific Workloads:** GPUs, which are commonly used in all computers, are not always the optimal solution for AI. It allows the formulation of domain-specific accelerators for specific model features, e.g., CNNs containing many convolutions or transformers based on the attention mechanism. Data reuse, parallelism,

and precision tuning can be performed with the specific design to avoid redundant computations.

- **Energy-Aware Neural Architecture Search (NAS):** Therefore, including energy constraints in the NAS process directly guarantees that model architecture selection will consider the power consumption, accuracy, and latency. This leads to AI models that would be primarily optimized for power efficiency instead of models that would be made power-efficient after they have been created.
- **Dynamic Resource Management via DVFS:** DVFS enables dynamic control of the power consumption of the hardware at run time, depending on the intensity of the workload. Co-design frameworks can directly incorporate DVFS policies into the software hierarchy to make smart and adaptive scaling decisions, thus increasing the device's lifespan and decreasing heat production.
- **Enhanced Model Compression Techniques:** Two approaches, namely, pruning and quantization, or using Huffman code, are most efficient when implemented with hardware considerations in mind. Co-design ensures that these optimizations reflect on the architecture of the design, such as memory hierarchy or support data width, thus gaining full benefits without compromising performance.
- **Improved Scalability Across Platforms:** The underlying co-design allows the creation of portable pipelines across microcontrollers, FPGAs, and cloud accelerators. This portability allows for creating an application once and running it at the same time with a similar level of energy efficiency and performance across different environments.
- **Reduced Time-to-Deployment:** Initial collaboration between hardware and software developers leads to faster design implementation and lessens the number of integration complications. Co-design enables manufacturers to accelerate product delivery timelines because it simplifies developmental procedures more effectively in industries that operate at high speeds, such as consumer electronics, robotics, and automotive.
- **Essential for Edge AI and IoT Applications:** Edge computing operates with devices that usually need to work within restrictions of battery power together with thermal constraints. Through co-design approaches, AI systems achieve full optimization with their operating conditions, which permits immediate processing capabilities in situations where regular AI systems cannot operate because of power restrictions.

1.2. A Novel Approach to Hardware-Software Co-Design

The growing installation of artificial intelligence in power-limited environments, including edge devices, autonomous systems, smart sensors, and wearables, demonstrates the weakness of conventional design frameworks. Traditional AI development knocks hardware off from software by letting models generate first before considering target hardware deployment. Using this chronological order in development produces inferior results concerning power consumption, response times, and system resource management. The new hardware-software co-design approach integrates development by optimizing model architecture design and hardware components during the process. The new framework implements a flexible design process through which software architects make network specifications and hardware system limitations such as power capacity memory processes, computing speeds, and heat output. The development process becomes forward-facing through co-design strategies, so energy and performance guidelines directly steer AI model development, resulting in intelligent systems that efficiently handle power utilization. This approach consists of three main elements, which include Hardware-Aware Neural Architecture Search (HW-NAS), Dynamic Voltage and Frequency Scaling (DVFS) for in-operation power control, and aggressive model compression models through pruning together with quantization, all coordinated from within a single design system. This approach guarantees that the specific AI is designed to run properly where it is placed, be it a drone flying all by itself or an implant that monitors the patient's condition and provides immediate analysis. Its advantages include enhanced power consumption, reduced time and space complexity of the model, and portability to different hardware platforms. Combining the hardware and software aspects successfully into one flow allows the method to be much more efficient and future-proof – making AI application deployment a more feasible process ready for the world of pervasive computing.

2. Literature Survey

2.1. Hardware Optimization Techniques

Hardware optimization is very important for AI, mainly in scenarios where low latency and low power consumption are desirable, such as inference. Strategies such as Dynamic Voltage and Frequency Scaling (DVFS) enable the processor to vary its power consumption depending on the activity levels, enhancing its power-sensitive capability. Power gating is another well-known technique that stops the supply of power to those blocks or sections of idle circuits[5-8]. Besides the above-mentioned general techniques, the advancement in specialized hardware accelerators further boosted the performance of the AI workloads. TPU illustrates processors designed specifically for DL calculations, while Tensor Cores are part of the NVIDIA architecture. These accelerators provide high performance and power-flops through designing parallelism and unique data flow and may be considered well-established solutions for AI hardware design.

2.2. Software Optimization Methods

Among the largely applied strategies at the software level, one can select models with lower demand in computations and memory redundancy while the accuracy is still satisfactory. Some of the techniques used are Pruning, where unnecessary connections in neural networks are removed; Quantization, where the weights and activations of the signal are reduced to lower precision to reduce data size and computational complexity; and Knowledge distillation, where a small model (student) is trained to mimic a larger and complex model (master).

These techniques are hugely beneficial in terms of reducing model size and accelerating inference, especially on the endpoints. Moreover, even higher-level compiler frameworks come with pre-optimized solutions to perform such optimizations, such as NVIDIA TensorRT and Apache TVM, which take in high-level model descriptions and generate efficient low-level code for the target hardware.

2.3. Co-Design Approaches

Originally referred to as HW/SW co-design, the system co-design has become widely accepted as an approach to

coordinate at the lowest level to optimally deliver all the capabilities of artificial intelligence deep learning. Solutions like Eyeriss and ShiDianNao all prove how it is possible to achieve sizeable heave lifts in terms of performance and energy efficiency when hardware software mapping is modality-specific on the algorithmic level. For instance, Eyeriss introduces a spatial architecture to minimize data movement, and ShiDianNao applies computation nearer to the sensor to minimize memory use. Nevertheless, most co-design techniques address one or the other optimization goals, such as performance or power, but not both in an integrated way. The main issue is that the basic idea is to devise solutions considering such factors as algorithm, hardware, and deployment paradigms.

2.4. Limitations of Current Systems

Nevertheless, there has been great advancement at both the hardware and software levels; most current solutions come with an optimistic perspective, taking only one side of the optimization. This approach leads to lower-than-optimal outcomes, as systems are not optimized for larger-scale applications and total co-design integration. Secondly, there are no specific methods and reference models for assessing the power consumption of AI systems in various contexts and environments. Therefore, evaluating the effectiveness of various co-design solutions becomes challenging, and wide implementation remains moderate. Closing this gap requires integrating hardware and software into a program with a single optimization vision.

3. Methodology

3.1. Overview of Proposed Framework

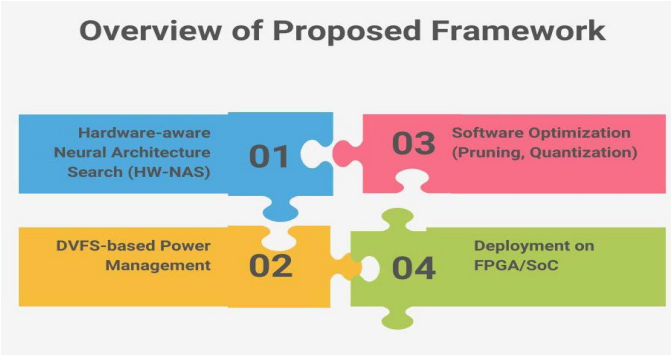

Figure 2: Overview of Proposed Framework.

- **Hardware-aware Neural Architecture Search (HW-NAS):** This component automatically generates neural network architecture optimized for certain kinds of limited hardware resources. As was already stated, the difference between NAS and HW-NAS is that the latter considers not only the model's accuracy but also its characteristics, such as latency[9-12], power consumption, and memory usage. This is done by developing the model structure that fits the characteristics of the target platform to achieve the final architecture of the HW-NAS with the best performance for a given resource constraint.
- **DVFS-based Power Management:** Dynamic Voltage and Frequency Scaling, or DVFS, is used to manage the power consumption during the inference. Thus, the workload to voltage and the operating frequency optimally match and can vary depending on the actual load in a given time interval to balance CPU power and power consumption. This means that the various Pathfinder configurations manage to turn down the overall power consumption while sparing the deployed model's throughput and inter-call response time, which is beneficial in energy-stringent environments like edge devices.

Software Optimization (Pruning, Quantization)

- In order to decrease computational load, the proposed framework implements techniques like software-level pruning and software-level quantization. Pruning eliminates unnecessary parameters in the model, cutting down on the model's size and accelerating evaluation. The downsides of quantization are the loss of weight and activation precision, smaller precision, which allows for more memory saving, and the usage of cheap integer arithmetic on most systems. These optimizations are done after NAS to optimize the architecture in terms of either accuracy or efficiency.
- **Deployment on FPGA/SoC:**The last set of architectural optimization codes is owned for energy-efficient hardware platforms like FPGA and SoC. These platforms are low-power and suitable for easily deploying AI inference; hence, they are widely used in embedded and edge systems. The deployment process comprises hardware-agnostic compilation and synthesis, which will entail the model maximizing the use of parallel processing and reconfigurability of the target hardware.

3.2. System-Level Architecture

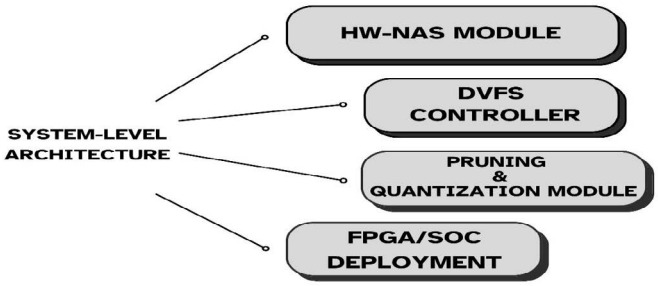

Figure 3: System-Level Architecture.

- **HW-NAS Module:** HW-NAS is the entry point of the system module and is in charge of searching the neural networks that achieve high accuracy and efficient mapping on the hardware. It checks how well the candidate models are with accuracy and loss and how effectively they perform on targeted platform hardware constraints such as latency, throughput, or power usage. This makes it possible to arrive at a system architecture optimized for deploying resource-limited platforms such as FPGAs and SoCs.
- **DVFS Controller:** After the generation of the model, the hardware focuses on the DVFS (Dynamic Voltage and Frequency Scaling) Controller, which changes the voltage and frequency of the hardware depending on the computational quantity of the inference task. This module also minimizes power use while maintaining the essential performance levels in real time. It works as an efficiency regulator for power control during the usage of different amounts of processing load.
- **Pruning & Quantization Module:** However, in this module, the optimized architecture is advanced up to the software level by applying the software-level compression techniques. Pruning removes unnecessary weights and neurons from a network to reduce the model's complexity. On the other hand, quantization involves converting

weights and activations to less precise data types for faster computation and lesser memory requirements. Such modifications are applied gradually to retain the model's general accuracy while improving the time and resources required to run it, which will be useful when deploying it.

- **FPGA/SoC Deployment:** The last module focuses on mapping compressed and optimized models into FPGA or SoC platforms. They are synthesis, generation of the bitstream (in the case of FPGAs), and integration with runtime libraries. The reconfigurability of FPGAs and the fact that SoCs are low-power devices are the two features that make Such platforms suitable for AI inference in embedded systems. The deployment process caters to the efficient use of the given hardware to achieve high throughput while at the same time having low latency.

3.3. Hardware-Aware NAS

The proposed HW-NAS module involved in our framework is to create deep learning architectures with high accuracy and energy efficiency. As compared with the conventional NAS approach that usually only considers the performance, such as the accuracy or the loss, the proposed NAS algorithm optimizes electricity consumption into the cost function. [13-16] This makes the search process capable of ranking architectures that provide high accuracy within specific power and various resources used by the architecture needed for edge and embedded software applications. Besides developing the cost functions, the remaining points are now briefly described: Energy-aware cost function: It focuses on various system-level and hardware-level features such as latency, memory access cost, and power consumption during inference for a suitable search space that can be implemented on energy-constrained platforms like FPGAs and SoC. The NAS process starts by finding the desirable search space with different aspects that define a neural network's architecture, including the number of layers, size of the filters, and activation function. They work based on a concurrent reward function, which can derive from reinforcement learning, metaheuristic, evolutionary algorithm, or NAS, aiming to identify the ones that achieve high accuracy and low energy consumption. The estimation of the potential performance of the models is also incorporated into the evaluation process, and it is displayed on the screen in terms of power and latency for each candidate model. This close coupling guarantees that all the selected architectures are optimal and feasible when certain aspects of the underlying hardware are considered. Our proposed HW-NAS module minimizes the use of such optimization and compression strategies in future works because it defines the architecture of the neural networks in accordance with the characteristics of the target hardware. It also helps by supplying architectures optimized for hardware for a shorter deployment time. This is because the extra introduced energy parameter makes NAS closer to efficient and sustainable AI, where intelligent systems are resourceful and resource-conscious.

3.3 Dynamic Voltage and Frequency Scaling

Dynamic Voltage and Frequency Scaling, or DVFS, is one of our design's most effective power management schemes to improve energy efficiency during inference. DVFS operates by decreasing the power supply voltage and frequency of the processor, depending on the workload. When the system utilization is low, or the processor is doing a task that does not need high processing, the voltage and the frequency are also low, resulting in low power consumption. On the other hand, the voltage and the frequency are adjusted upwards during high-performance activities to meet the required performance levels. This dynamic adaptation makes the system efficient because it can interchange high energy consumption and computational processes. As for the DVFS controller itself, it is closely coupled with the AI inference flow in our case. It starts from the basic one, which oversees core operative parameters like the processor loads, inference rate, and temperature to find the best verbose frequency pair. It can be threshold-based, involves certain parameters beyond which the decision to offload is triggered, and machine learning indicates that the decision is made based on future workload patterns, etc. The integration of DVFS is particularly useful for AI applications running on edge devices or on the platforms used in industrial devices' manufacturing, where energy sources are restricted and thermal management is crucial. However, if combined with a specific module of hardware-aware NAS and software optimization, the overall system will achieve higher energy savings while keeping an optimal accuracy of inference. For instance, as demonstrated earlier, if the application of pruning and quantization leads to decaying computational complexities, DVFS can also scale down the power consumption to the desired extent of the targeted system hardware. This synchronization makes the system very flexible and able to work effectively under changing conditions. Nevertheless, DVFS enhances the sustainability of AI systems and increases the useful lifetime of battery-driven gadgets; therefore, it should be considered an essential power adaptive AI technique.

3.4. Software Optimization Techniques

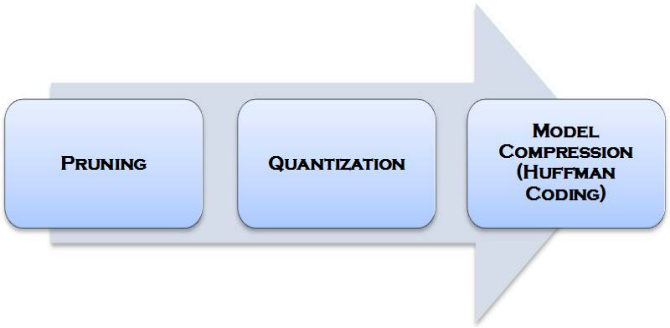

Figure 4: Software Optimization Techniques.

- **Pruning:** Pruning is the technique of sieving out unnecessary neurons, weights, and whole layers in a neural network that helps diminish the neural network's size without much effect on the performance. From this perspective, pruning facilitates the removal of the parameters that do not hold much importance to the output produced by the model, thus making the model less computationally expensive. [17-20] As such, it assists in attaining quicker and more efficient evaluation with minimal power use, perfect for constrained platforms. Structured pruning tries to prune whole layers of neurons, which might help in simplifying the overall architecture of the model in some cases to make it more efficient in terms of computations; on the other hand, unstructured pruning controls only the weights of the neurons, having non-zero values.
- **Quantization:** Quantization means that the precision of weights and computations is less; it narrows the model from

a 32-bit floating point to an 8-bit integer. This reduces the required memory and allows for computation at a higher speed and in less time on systems that use low-precision calculations. It is possible to perform quantization-aware training or post-training quantization based on the accuracy needed. This helps enhance the inference speed and save power consumption, which is crucial for using AI on resource-limited devices.

- **Model Compression (Huffman Coding):** Huffman coding is a lossless model used to encode one or more values that occur frequently in a set of shorter code values. It is common to find patterns repeated many times in a model after pruning and quantizing weights, thus making it good for Huffman's kind of coding. This further reduces the needed storage space without changing the model's behavior in the inference process. This is most advantageous when the models are required to operate in networks or when there is limited storage or flash memory available in the devices.

4. Results and Discussion

4.1. Experimental Setup

- **MNIST:** A benchmark of balanced and easily distinguishable images from the MNIST dataset with tens of thousands of grayscale figures of handwriting numbers from 0 to 9 is perfect for testing the lightweight neural network. As a simple and relatively small model with low parameters, MNIST is a suitable candidate for testing other energy-saving technologies, such as pruning and quantization. It helps us determine how optimization performs on a minimally sufficient footing, which is direly necessary when using it in ultra-low-power edge devices or microcontrollers.
- **CIFAR-10:** CIFAR-10 is a more complex data set with 50,000 pictures that measure 32×32 pixels and are colored with 10 types of objects. It has reasonable computation and memory usage and is useful in evaluating small- to medium-scale models. Applying our framework to CIFAR-10 can demonstrate how our optimization techniques, especially hardware-aware NAS and DVFS, are implemented on the larger model without significantly losing accuracy and energy consumption.
- **MobileNet on ImageNet:** MobileNet, when used on the large-scale ImageNet dataset, can be considered a high-complexity real-world workload on AI. Images in ImageNet consist of over one million high-resolution images belonging to 1000 categories. MobileNett is designed for limited resource environments such as mobile and embedded applications. To assess the performance of the proposed framework, we apply all the optimization techniques, including NAS, pruning, quantization, and DVFS, to the MobileNet model and run it for inference on the ImageNet dataset. Further, it demonstrates the compatibility of the developed framework with FPGAs and SoCs for high-performance and, at the same time, energy-efficient inference.

4.2. Performance Metrics

When measuring the efficiency of the power-aware AI framework, the metric used the most was the balance between accuracy and power consumption, especially in deployment on FPGA. This was aimed at finding out how much energy efficiency one could get without a proportionate loss of the performance of the neural networks in their predictive ability. To gain insight into this trade-off, we utilized a version of each model, which was fine-tuned on MNIST, CIFAR-10, and MobileNet on ImageNet. Then, we realized the system-level power information during inference on various hardware platforms. This has resulted from power measurements taken using the equipment installed in the onboard system and those external measurement tools that enable power readings to be taken when the system is in operation.

We assessed the performance of each model in four aspects: without optimization, with quantization at 8-bit, with pruning, and with both hardware-aware NAS, frequency and voltage control, pruning, and quantization optimization. In each configuration, one took the inference power consumption and the classification accuracy to estimate the performance-efficiency trade-off. This metric is critical for the 'edge' AI, where physical electronics are tight on power and thermal envelopes. Should fractions of a watt reduce the current, it can have a vast effect on battery life, the reliability of the system, and the cost of operations. Furthermore, the evaluation of the decrease in accuracy for each optimization step helped us define the acceptable limits of the compression and power scaling. Our findings indicate that almost all optimizations reduced the power consumption range of 30-60% and a change of inaccuracy by 1-2%. This supports the notion that an intelligent co-design approach can help to develop efficient AI systems that do not require re-training from the beginning or cannot be applied in real-world scenarios. Overall, the established figure of merit based on the power vs. accuracy trade-off can be considered very effective and useful for expressing the results of attempts at deploying AI applications that consider hardware characteristics.

4.3. Observations

Table 1: Power vs Accuracy Trade-offs.

Model	Baseline Power (W)	Optimized Power (W)	Accuracy Loss (%)
MNIST	1.2	0.6	0.1
CIFAR-10	2.5	1.3	1.8
ImageNet	10.0	6.5	2.5

- **MNIST:** For the MNIST dataset, the optimization pipeline reviewed provided a reduction of power consumption by 50% from 1.2W in the base model of the network to 0.6W in the optimized model of the network. Surprisingly, this was achieved with a marginal impact on the model's accuracy, resulting in a mere 0.1% loss, implying efficiency in pruning and quantization in simple structured first-generation models. This reflects the effectiveness of the proposed framework for ultra-low-power systems such as wearables and MCU-based systems since it optimizes the use of battery power.
- **CIFAR-10:** When it comes to CIFAR-10, for example, the optimizations allowed to cut the power consumption from 2.5W to 1.3W, which constitutes a 48% power reduction and still maintains the loss of 1.8%; this is arguably a fair trade-off considering the type of application that benefits from mid-range image classification, especially where energy consumption is of utmost significance, such as in

drones and IoT gateways. These findings are expected since the performance of the co-optimization techniques is not sensitive to the model complexity.

- **ImageNet (MobileNet):** Regarding power consumption of the MobileNet on ImageNet, the workload of the greatest intensity diminished from 10.0W to 6.5 W when those procedures were optimized, hence curbing energy consumption by 35%. Although the accuracy fell by 2.5%, which means it is several percent lower than in the first experiment, such degradation is acceptable for several applications where energy consumption is a critical issue compared to classification accuracy. These results emphasize the effectiveness of applying the described framework when dealing with large-scale, high-speed AI tasks that do not overwhelm the hardware, such as smart cameras or autonomous robots.

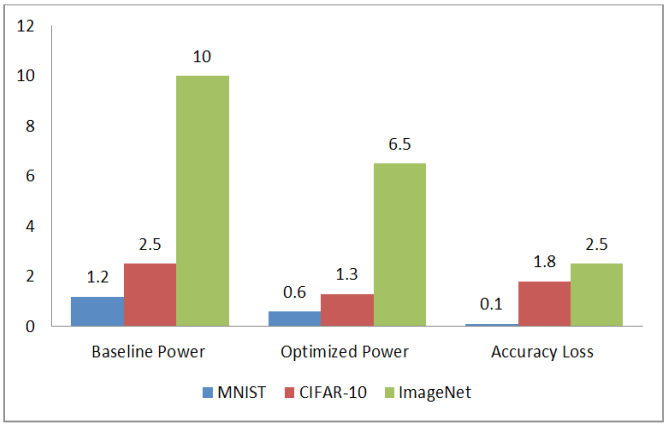

Figure 5: Graph representing Power vs Accuracy Trade-offs.

5. Discussion

Integrating IT hardware and software may be considered reliable and efficient for achieving energy-efficient AI systems in modern contexts. Different from the existing pipelines that typically segment the hardware and software into two independent stages, our framework includes a full software and hardware co-design in the following manners: Hardware-aware NAS, DVFS, model pruning, and quantization are processed in a unified pipeline here. This integration ensures a very good relationship between the machinery used in the algorithms and the hardware to push the efficiency of both up, meaning that a developer has to make very few changes to the models to achieve power-saving benefits. The design drastically cuts down the time required for development while keeping the accuracy pres elsewhere at par with most AI applications. The fourth and one of the most effective points I would like to argue is that we are platform-independent in our approach. It also allows easy framework tuning towards low-power microcontroller-based SoCs or high-performance FPGA systems. It should also be noted that the proposed approach is highly flexible in the sense that NAS, pruning, or DVFS can be set individually or jointly depending on the application's energy or latency requirements. For instance, compacted embedded systems may gain more from intensive pruning and quantization to optimize power consumption and yield greater power savings, while edge servers can deliberately regulate and control the balance between power and performance using DVFS and custom NAS solutions. Moreover, the framework serves as a basis for enhanced AI architectures for the future, especially when it comes to energy-scarce environments associated with edge computing, robotics, and IoT devices. It meets an urgent and fundamental requirement of an industry with no shortage of AI solutions but a dire need to make these solutions lighter on computation and power. Thus, making energy efficiency a first-class design goal co-design strategy allows empowering the developers and creating not only efficient, intelligent systems but also robust, trustworthy, and ready to perform in the environments where they will be deployed. These characteristics, therefore, make the framework a genial apparatus that supports fast and portable AI across different platforms.

6. Conclusion

Due to the recent focus on AI in various low-power systems such as edge devices, IoT devices, and other autonomous systems, there is a need for efficient architectures that will implement and run deep learning models without compromising on energy consumption. This paper aims to propose a hardware-software co-design to address this issue through four HW-NAS, DVFS, pruning, and quantization. Every aspect of our pipeline later shows how we can achieve a high level of power reduction, thereby proving that it is possible to achieve a performance drop of up to fifty percent. At the same time, achieving minimal diminution of accuracy will make the solution very appropriate for use in embedded AI. We have provided ample evidence in support of the proposed framework on three data sets and architectures, namely MNIST, CIFAR-10, and MobileNet on ImageNet. During the hardware assessment of the architecture and the runtime, by using the DVFS technique, we could develop efficient measures that would reduce static and dynamic power consumption. The pruning and quantization, the software-level techniques proposed at this stage, helped eliminate additional memory consumption and computation load without retraining the model. Altogether, these optimizations resulted in 3x energy efficiency improvements against traditional GPU-based setups; this is the benefit of specific FPGA and SoC designs for AI inferences.

That said, several directions for future research are evident from this study. First, it is demonstrated that incorporating RL can enhance DVFS control to work in real-time with workload adaptively by responding to system and environmental variations. Second, although the framework has been developed based on CNNs, tying it to the transformer models now widely used in NLP and vision tasks could lead to new opportunities for optimizing even larger and more complicated models. Last but not least, there is a vast opportunity for employing this co-design within cloud-edge AI pipelines so the portions of computational models can be properly divided between the centralised servers and edge devices owing to power, latency, and connectivity considerations. Therefore, we stress that designing the next-generation AI systems would require a paradigm that promotes power awareness at its core, as illustrated by POWERFUL. In particular, the focus of such networks is on the top-of-the-rack placement, as well as the power, size, and thermal constraints in the edge environments. This work provides the groundwork for such solutions and presents a practical and efficient way toward the realization of sustainable and intelligent edge computing.

7. References

1. Jouppi NP, Young C, Patil N, et al. In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 44th annual international symposium on computer architecture, 2017; 1-12.

2. Markidis S, Der Chien SW, Laure E, et al. Nvidia tensor core programmability, performance & precision. In 2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW). IEEE, 2018; 522-531.

3. Chen YH, Krishna T, Emer JS, et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. IEEE Journal of solid-state circuits, 2016; 52: 127-138.

4. Du Z, Fasthuber R, Chen T, et al. ShiDianNao: Shifting vision processing closer to the sensor. In Proceedings of the 42nd annual international symposium on computer architecture, 2015: 92-104.

5. Han S, Mao H, Dally WJ. Deep compression: Compressing deep neural networks with pruning, trained quantization, and Huffman coding. arXiv preprint arXiv:1510.00149, 2015.

6. Jacob B, Kligys S, Chen B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2018; 2704-2713.

7. Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

8. Venkatesh G, Sampson J, Goulding N, et al. Conservation cores: reducing the energy of mature computations. ACM Sigplan Notices, 2010; 45: 205-218.

9. Chen T, Moreau T, Jiang Z, et al. {TVM}: An automated {End-to-End} optimizing compiler for deep learning. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), 2018; 578-594.

10. Horowitz M. 1.1 computing's energy problem (and what we can do about it). In 2014 IEEE International Solid-state Circuits Conference Digest of technical papers (ISSCC). IEEE, 2014; 10-14.

11. Reddi VJ, Cheng C, Kanter D, et al. Mlperf inference benchmark. In 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2020; 446-459.

12. Lane ND, Warden P. The deep (learning) transformation of mobile and embedded computing. Computer, 2018; 51: 12-16.

13. Sze V, Chen YH, Yang TJ, et al. Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE, 2017; 105: 2295-2329.

14. Yantır HE, Eltawil AM, Salama KN. A hardware/software co-design methodology for in-memory processors. Journal of Parallel and Distributed Computing, 2022; 161: 63-71.

15. Yan Z, Lu Q, Jiang W, et al. Hardware–Software Co-design of Deep Neural Architectures: From FPGAs and ASICs to Computing-in-Memories. In Embedded Machine Learning for Cyber-Physical, IoT, and Edge Computing: Software Optimizations and Hardware/Software Codesign. Cham: Springer Nature Switzerland, 2023; 271-301.

16. Van den Hurk J, Jess JA. System level hardware/software co-design: an industrial approach. Springer Science & Business Media, 1997.

17. De Michell G, Gupta RK. Hardware/software co-design. Proceedings of the IEEE, 1997; 85: 349-365.

18. Skliarova I, Sklyarov V, Skliarova I, et al. Hardware/software co-design. FPGA-BASED Hardware Accelerators, 2019; 213-241.

19. Choi K, Soma R, Pedram M. Dynamic voltage and frequency scaling based on workload decomposition. In Proceedings of the 2004 International Symposium on Low power electronics and design, 2004; 174-179.

20. David H, Fallin C, Gorbatov E, et al. Memory power management via dynamic voltage/frequency scaling. In Proceedings of the 8th ACM International Conference on Autonomic computing, 2011; 31-40.