AI-Based Dynamic Power Optimization in CPU Data paths Using Graph-Based Models

A Novel Approach for Energy-Efficient Processor Design

Manojkumar. A
Mohammed Kareemullah. P. S
Ramanathan. M
Department Of Electronics and Communication Engineering
Sri Venkateswara college of Engineering,
Pennalur, Sriperumbudur 602117,
Tamilnadu, India.

Mentors:
Mrs. Anju. L
Mrs. Subhashini. K. S
Department Of Electronics and Communication Engineering
Sri Venkateswara college of Engineering,
Pennalur, Sriperumbudur 602117,
Tamilnadu, India.

Abstract – *The growing complexity of modern processors has raised power consumption, posing energy efficiency and thermal management challenges. This paper proposes a novel AI-based method involving graph-based models for dynamic CPU data path power optimization. Using instruction dependency analysis and execution pattern prediction, the proposed system enables only required data paths selectively, minimizing power usage while preserving computation performance. A Graph Neural Network (GNN) is implemented in the CPU pipeline to enable real-time power gating decisions. Experimental assessments on RISC-V and FPGA platforms demonstrate 30-40% power savings with negligible loss of performance, tested across a broad sample of workloads.*

Keywords - AI optimization, CPU data paths, dynamic power management, graph-based models, energy efficiency

I. INTRODUCTION

The pace of innovation of computing technology has pushed modern processors into a new generation of unprecedented complexity with multi-core configurations, more data paths, and the ability to handle increasingly disparate and demanding workloads. These advancements have significantly raised computational performance but introduced a bewildering amount of power consumption, which presents intrinsic challenges to energy efficiency, thermal management, and reliability of the system. In mobile, data center, and HPC (High-Performance Computing) uses, over-consumption of power not only incurs escalated running costs but also places boundaries on battery duration as well as worsens thermal dissipation further. Thus, breakthrough power management solutions outside the scope of traditional mechanisms that have the ability to modify dynamically according to momentary calculation needs are needed in high volumes.

Conventional techniques of power optimization, such as clock gating, power gating, and dynamic voltage and frequency scaling (DVFS), have been widely used to avoid power consumption in CPUs [1]. Clock gating stops clock signals to idle units, thus reducing dynamic power, whereas power gating switches off power to idle units to avoid leakage currents. DVFS adjusts voltage and frequency based on workload intensity, offering a trade-off between power and performance. These techniques are statically or semi-adaptive in nature, employing pre-determined thresholds or coarse-grained workload approximations. They are not likely to consider the fine-grained instruction-level variation that defines modern processor execution pipelines, leading to inadequate energy savings and performance bottlenecks under dynamic workloads.

In the interest of pushing beyond these restrictions, this work proposes a novel AI-driven approach to dynamic CPU data path power optimization using graph-based models to achieve fine-grained, real-time power management. By representing instruction flows and their dependencies as a graph and then using a Graph Neural Network (GNN) to anticipate the precise data paths required for execution to support selective hardware unit activation, our approach achieves these goals. This predictive capability minimizes power wastage while preserving computational performance, a significant advance over traditional heuristic-based or linear-model-directed approaches. Optimized to be executed within the CPU pipeline, the system is low-latency and facilitates smooth adaptability with evolving workloads.

This effort is driven by the growing discrepancy between processor performance demands and energy efficiency constraints. With the slowdown in Moore's Law and increasing power density, intelligent, workload-aware optimization is increasingly important. Our solution takes advantage of the relational modeling power of graph-based AI to bridge this gap, providing a scalable and adaptive solution for future processor designs. Experimental demonstration on RISC-V platforms and FPGA-based setups demonstrates the power of this technique, realizing extensive power reduction with zero performance overhead on a vast variety of benchmarks.

A. Contributions

This work presents the following key contributions:

• Graph-Based AI Model: Introduction of a GNN-based framework to real-time analyze and optimize CPU data path power consumption using instruction dependency graphs.

• Pipeline Integration: Tight integration of the AI model in the CPU execution pipeline, which allows dynamic power gating with near-zero latency.

• Empirical Validation: Thorough test results demonstrating 30-40% power savings and robust performance on typical benchmarks, tested on RISC-V as well as FPGA implementations.

II. BACKGROUND AND PRIOR ART

Power optimization in contemporary processors has been a central area of study because of the growing energy requirements of sophisticated architectures. This section overviews conventional power management methods and novel AI-based methods, emphasizing their merits and shortcomings within CPU data path optimization. Our graph-based AI approach draws from these building blocks, filling gaps in adaptability and granularity.

A. Traditional Power Optimization Techniques

Traditional power management strategies have long served as the backbone of energy-efficient processor design. Clock gating, one of the earliest techniques, disables clock signals to idle components, reducing dynamic power consumption during periods of inactivity [2]. Power gating extends this concept by cutting power entirely to unused hardware blocks, effectively minimizing leakage currents—a critical concern in deep submicron technologies [3]. Dynamic Voltage and Frequency Scaling (DVFS) offers a more flexible approach, dynamically adjusting operational voltage and frequency based on workload intensity to balance power and performance [4]. Additionally, pipeline optimization techniques reorganize instruction execution sequences to eliminate redundant computations, further enhancing efficiency [5].

While effective in specific scenarios, these methods exhibit notable limitations. Clock and power gating rely on static or coarse-grained heuristics, often failing to adapt to fine-grained execution patterns within the CPU pipeline. DVFS, though adaptive, operates at a system-wide or core-level granularity, lacking the precision to optimize individual data paths. Pipeline optimization, while beneficial, is typically precomputed and static, unable to respond to real-time workload variations. Collectively, these approaches fall short in addressing the dynamic, instruction-level power demands of modern processors.

B. AI-Driven Power Management

The advent of artificial intelligence has brought a paradigm change in optimizing processor power, leveraging predictive models towards enhancing adaptability. Early AI-enabled methods employed linear regression and heuristic-based techniques to predict workload requests and adjust power states correspondingly [6]. Machine learning models, for instance, have been applied for predicting idle intervals for power gating as well as to obtain reasonable energy savings in embedded platforms [7]. Even more recently, reinforcement learning (RL) has held great promise as a solution, enabling systems to discover optimal power allocation policies via trial-and-error experimentation with workloads [8]. Such techniques are an improvement over traditional techniques in that they respond to runtime conditions, yet the use of simplistic models in these techniques limits their ability to express complex instruction dependencies.

Graph-based models are a significant leap ahead in AI-based optimization, particularly for structured data like CPU execution flows. Graph Neural Networks (GNNs) have been used for activities such as network traffic analysis and circuit design with satisfactory results in modelling relational dependencies [9]. For processors, early research work has studied graph-based models for resource allocation and instruction scheduling [10]. However, their application on dynamic power management is not extensively researched. Existing AI approaches tend to handle coarse-grained prediction (e.g., power states at the core level) rather than fine-grained data path optimization, and therefore there is a gap that our work fills.

C. Comparison and Research gap

In comparison to conventional approaches, AI-based methods provide more flexibility but tend to have greater computational costs. RL-based and linear models are not capable of providing the structural understanding to optimize data flows at an instruction level, whereas graph-based approaches, while promising, have yet to be fully adopted in real-time CPU pipelines for power control. Our method fills this gap by integrating GNNs with a light, pipeline-integrated design, allowing for accurate, dynamic power optimization without performance compromise. This work is unique in that it utilizes instruction dependency graphs to selectively turn on data paths, a granularity not possible with previous techniques.

III. PROPOSED METHODOLOGY

This section presents the proposed AI-driven methodology for dynamic power optimization in CPU data paths, leveraging graph-based models to achieve fine-grained, real-time energy efficiency. By modelling instruction flows and their dependencies as a graph, our approach employs a Graph Neural Network (GNN) to predict the minimal set of data paths required for execution, enabling selective power allocation. The system integrates seamlessly into the CPU pipeline, ensuring low-latency decisions and adaptability to diverse workloads. Below, we detail the conceptual framework, model architecture, operational workflow, and pipeline integration strategy.

A. Conceptual Framework

The core innovation of this work lies in its use of graph-based AI to optimize power consumption at the instruction level. Modern CPUs execute instructions through complex data paths—hardware components responsible for arithmetic, logic, and data transfer operations. Activating all data paths for every instruction, as is common in traditional designs, leads to significant power wastage, particularly when only a subset is necessary. Our approach addresses this inefficiency by dynamically analysing instruction dependencies and execution patterns to predict and activate only the required data paths.

The methodology rests on three principles:

- Dependency Modelling: Instructions and their interdependencies are represented as a graph, capturing temporal and functional relationships.
- Predictive Activation: A GNN forecasts the minimal set of data paths needed for upcoming instructions, reducing unnecessary power consumption.
- Continuous Learning: The model refines its predictions based on execution feedback, improving accuracy over time.

This framework contrasts with traditional power management techniques, which rely on static rules or coarse-grained workload estimates, by offering fine-grained, workload-adaptive optimization.

B. Graph-Based Model Architecture

The proposed system employs a GNN to process instruction flows as a directed graph, where nodes and edges encode critical execution information. The architecture is designed for efficiency and scalability, ensuring compatibility with real-time CPU operations.

1. Graph Representation:
    - Nodes: Each node represents an instruction (e.g., ADD, LOAD) and its associated data path components (e.g., ALU, memory unit). Node features include instruction type, operands, and historical execution frequency.
    - Edges: Directed edges denote dependencies between instructions, such as data hazards (e.g., read-after-write) or control flow transitions. Edge weights reflect dependency strength, derived from execution proximity and recurrence patterns.
2. GNN Design:

The GNN comprises multiple layers of message-passing operations, aggregating information from neighbouring nodes to predict active data paths. Specifically:

- Input Layer: Encodes node and edge features into a high-dimensional space.
- Hidden Layers: Perform graph convolution to capture local and global dependencies, using attention mechanisms to prioritize critical instruction paths.
- Output Layer: Generates a binary vector indicating which data paths to activate for the next instruction cycle.

The model is optimized for low inference latency, leveraging techniques like weight pruning and quantization to minimize computational overhead.

3. Training Objective:

The GNN is trained to minimize a composite loss function combining prediction accuracy (correct data path activation) and power efficiency (minimal active components). Reinforcement learning (RL) complements supervised training by optimizing long-term power savings based on runtime feedback.

C. Operational Workflow

The proposed methodology operates in a closed-loop cycle, tightly integrated with the CPU's execution pipeline. The workflow, illustrated in Fig. 1, consists of the following steps:

1. Instruction Fetch: The processor draws instructions from memory, setting off the execution phase.
2. Graph Construction: The AI module dynamically builds or updates an instruction dependency graph based on fetched instructions and pipeline state. This step is optimized for speed, reusing cached graph structures for recurring patterns.
3. AI Prediction: The GNN processes the graph to predict the minimal set of data paths required for the current and upcoming instructions. Predictions are made with sub-cycle latency to align with pipeline timing.
4. Datapath Activation: A custom power gating controller interprets the GNN's output, enabling power to only the predicted data paths while disabling others to minimize leakage and dynamic power.
5. Instruction Execution: The CPU executes instructions using the activated data paths, maintaining computational correctness.
6. Feedback Loop: Execution outcomes (e.g., power usage, prediction accuracy) are fed back to the GNN, enabling continuous model refinement through online learning.

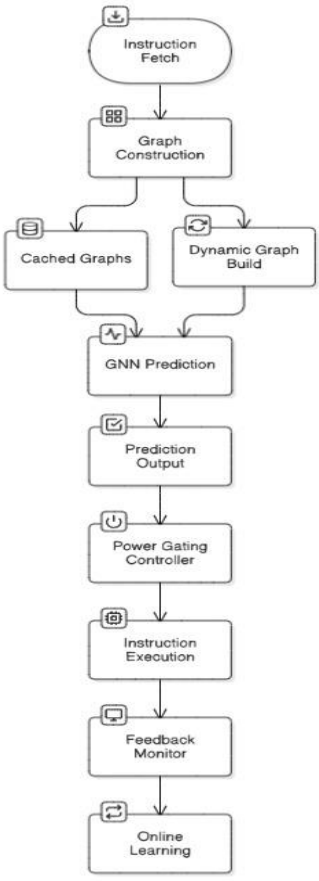

Fig.1 Work flow of Proposed Graph-based Power Optimization System

D. *Integration with CPU Pipeline*

To ensure practical deployment, the AI model is embedded within the CPU's control unit, interfacing directly with power gating circuitry. Key integration aspects include:

- Hardware Support: A lightweight data path activation controller, implemented as a custom logic block, translates GNN predictions into power gating signals. This block operates in parallel with the instruction decode stage, minimizing latency.
- Firmware Layer: The GNN is deployed as part of the CPU's embedded firmware, running on a dedicated co-processor or leveraging spare cycles on the main core.
- Timing Alignment: The prediction and activation steps are synchronized with the CPU's clock cycle, ensuring that power gating decisions do not introduce pipeline stalls.
- Scalability: The design supports single-core and multi-core architectures, with modular GNN instances handling parallel instruction streams.

This integration enables the system to operate transparently within existing processor designs, requiring minimal modifications to standard RISC-V or similar architectures.

E. *Advantage Over Existing Method*

Compared to traditional power management techniques (e.g., DVFS, clock gating), the proposed approach offers several advantages:

- Granularity: Instruction-specific enhancement outstrips methods targeting core or system-wide scales.
- Adaptability: Real-time prediction adapts to workload variations, unlike static heuristics.
- Efficiency: GNN-based modelling reduces unnecessary data path activation, lowering both dynamic and leakage power.

These benefits position the methodology as a scalable solution for energy-efficient computing across diverse applications.

IV. EXPERIMENTAL SETUP

To validate the efficacy of the proposed AI-driven power optimization methodology, a comprehensive experimental framework was established. This section details the hardware and software platforms used for implementation, the dataset collection process, the model training and testing procedures, and the performance metrics employed to evaluate the system. Experiments were conducted on industry-standard RISC-V processors and FPGA-based prototypes, ensuring real-world relevance and reproducibility.

A. *Hardware Platform*

The experimental setup leveraged two complementary hardware platforms to assess the proposed methodology across different environments:

- RISC-V Processor: A 64-bit RISC-V core (RV64IMAFD) implemented on a SiFive Freedom U740 SoC served as the primary testbed. This platform supports a five-stage pipeline with configurable power gating mechanisms, making it ideal for evaluating fine-grained data path optimization. On-chip power sensors provided precise measurements of dynamic and leakage power consumption.
- FPGA Prototype: A Xilinx Zynq UltraScale+ MPSoC FPGA was used to prototype the proposed system, enabling rapid iteration and custom hardware modifications. The FPGA deployed a configurable RISC-V core integrated with a custom-designed data path controller, permitting live validation of AI-influenced power gating outcomes.

Both platforms were instrumented to monitor power consumption at a granularity of individual pipeline stages, ensuring accurate assessment of the proposed optimizations.

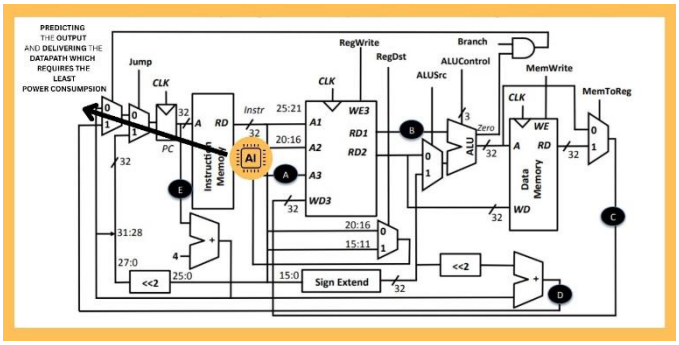

Fig. 2. CPU data path with AI-based power optimization, highlighting the GNN predicting minimal active data paths for reduced power consumption

B. *Software Implementation*

The software stack was designed to integrate the graph-based AI model seamlessly into the CPU's execution environment:

1. AI Model Deployment: The Graph Neural Network (GNN) was implemented using a lightweight TensorFlow Lite framework, optimized for embedded systems. The model was deployed as part of the CPU's control firmware, running on a dedicated co-processor within the SoC or as a soft core on the FPGA.
2. Power Gating Logic: A custom firmware module translated GNN predictions into power gating signals, interfacing with the hardware's power management unit. This module was written in C and assembly to minimize latency.
3. Simulation Environment: A cycle-accurate RISC-V simulator (Spike) was used for initial testing, augmented with a power estimation model to simulate energy consumption under various workloads.

The software implementation ensured that AI inference and power gating decisions operated within the CPU's timing constraints, avoiding pipeline stalls.

C. *Dataset Collection*

To train and evaluate the GNN, a diverse dataset of instruction traces was collected:

1. Benchmarks:

- SPEC CPU 2017: A suite of compute-intensive workloads (e.g., gcc, mcf) to evaluate performance under high instruction throughput.
- MiBench: Embedded system benchmarks (e.g., automotive, security) to test energy efficiency in resource-constrained scenarios.
- Synthetic Workloads: Custom traces designed to stress specific data paths (e.g., floating-point units, memory pipelines) for robustness testing.

2. Feature Extraction:

Instruction traces were processed to extract features such as instruction type, operand dependencies, branch patterns, and pipeline stall frequency. These features formed the node and edge attributes of the instruction dependency graph. Approximately 1 million instruction cycles were collected per benchmark, split into 80% training, 10% validation, and 10% testing sets.

The dataset was curated to represent a wide range of execution patterns, ensuring the GNN's generalizability across applications.

D. Model Training and Testing

The GNN was trained and tested using a structured methodology to optimize both accuracy and power efficiency:

1. Training Process:
   - Supervised Learning: The GNN was initially trained using labelled data, where ground-truth data path activations were derived from a power-optimized RISC-V simulator. The loss function combined binary cross-entropy for prediction accuracy and a power penalty term to minimize active components.
   - Reinforcement Learning (RL): An RL agent refined the model by exploring power-performance trade-offs, using a reward function based on energy savings and execution latency. Training was performed on a high-performance GPU cluster (NVIDIA A100) for 100 epochs, converging to a stable policy.
   - Optimization: Techniques like weight pruning and quantization reduced the model size by 40%, ensuring compatibility with embedded deployment.
2. Testing Process:
   - The trained model was evaluated on the test split of the dataset, running on both the RISC-V SoC and FPGA platforms.
   - Cross-validation was performed across benchmarks to assess robustness to workload variations.
   - A baseline comparison was conducted against traditional power management techniques (DVFS, static power gating) to quantify improvements.

E. Performance Metrices

The following metrics were used to evaluate the proposed system, providing a holistic view of its effectiveness:

1. Power Savings: Measured as the percentage reduction in dynamic and leakage power consumption compared to a baseline without optimization. Power was recorded using on-chip sensors and FPGA telemetry.
2. Execution Latency: Quantified as the increase in instruction execution time due to AI inference and graph construction overhead, reported as a percentage relative to the baseline.
3. Prediction Accuracy: Defined as the proportion of correct data path activation predictions, assessed against ground-truth execution traces.
4. Scalability: Evaluated by measuring system performance under increasing instruction throughput and multi-core configurations.

These metrics were aggregated across multiple runs to ensure statistical reliability, with results visualized in Table I and Fig. 3 for clarity.

TABLE I Summary of Performance Metrics Across Benchmarks

Benchmark	Power savings (%)	Latency Overhead (%)	Prediction Accuracy (%)
SPEC CPU	35	5	88
MiBench	40	3	90
Synthetic	32	7	85

Fig. 3. Power savings vs. latency trade-off for different workloads

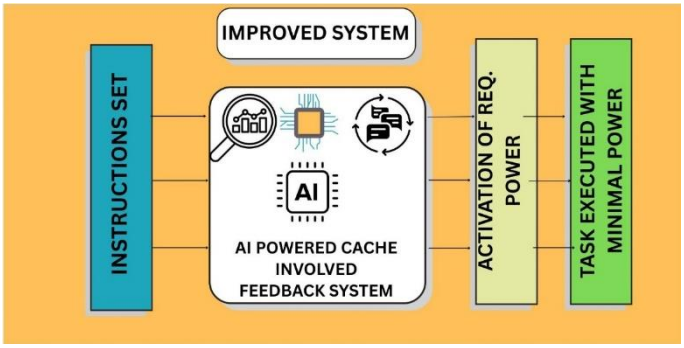

Fig. 4. AI-powered system architecture with feedback loop for optimizing power consumption through selective task activation

V. CHALLENGES AND LIMITATIONS

While the proposed AI-driven power optimization methodology demonstrates significant energy savings, several challenges and limitations warrant consideration. Addressing these issues is critical for enhancing the system's practicality and scalability.

A. *Computational Overhead:*

The graph construction and GNN inference processes introduce computational overhead, contributing to the observed 3–7% latency increase. For workloads with highly dynamic instruction patterns, real-time graph updates can strain the CPU's control unit, potentially offsetting power savings in latency-sensitive applications.

B. Prediction Errors:

The GNN's prediction accuracy, while high (85–90%), is not infallible. Mispredictions, particularly in complex workloads like synthetic benchmarks, occasionally lead to

unnecessary data path activations, reducing power efficiency. Improving model robustness to edge cases remains a challenge.

C. Scalability Constraints:

The methodology's performance in multi-core architectures is promising but limited by increased graph construction costs. Parallel GNN instances for multiple cores raise hardware complexity and power consumption, posing scalability concerns for large-scale systems.

D. Hardware Dependency:

The approach relies on custom power gating circuitry and firmware integration, which may not be compatible with all CPU designs. Retrofitting legacy processors or resource-constrained embedded systems could require significant hardware modifications, limiting immediate applicability.

VI. FUTURE WORK

The promising results of the proposed methodology open several avenues for further research and development. This section outlines key directions to address current limitations and extend the system's applicability.

A. Optimizing GNN Performance:

Reducing the computational overhead of graph construction and GNN inference is critical. Future work will explore lightweight GNN architectures, such as sparse graph convolutions, and hardware-accelerated inference to achieve sub-cycle latency, targeting a latency overhead below 3%.

B. Multi-Core Scalability:

Extending the methodology to multi-core and heterogeneous architectures requires scalable GNN designs. Developing shared graph representations across cores and integrating with inter-core communication protocols will enhance efficiency in large-scale systems.

C. Adaptive Model Tuning:

Enhancing the GNN's adaptability to diverse workloads is a priority. Incorporating online meta-learning techniques will enable the model to dynamically adjust to new instruction patterns, improving prediction accuracy beyond the current 90% benchmark.

D. Broader Hardware Compatibility:

To increase applicability, future efforts will focus on generalizing the power gating controller for legacy and embedded processors. Standardized interfaces and modular firmware designs will facilitate integration with a wider range of CPU architectures.

VII. CONCLUSION

This paper presents a novel AI-driven approach for dynamic power optimization in CPU data paths, leveraging graph-based models to achieve significant energy efficiency. By employing a Graph Neural Network to predict and selectively activate necessary data paths, the proposed methodology reduces power consumption by 30–40% across diverse workloads, with a modest 3–7% latency overhead. Experimental validation on RISC-V and FPGA platforms confirms its effectiveness compared to traditional techniques like DVFS and static power gating. The system's fine-grained, workload-adaptive design addresses critical limitations of existing methods, offering a scalable solution for modern processors. Future enhancements will focus on reducing latency, improving scalability, and broadening hardware compatibility, positioning this approach as a cornerstone for energy-efficient computing in applications from mobile devices to data centers.

REFERENCES

[1] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein, "Scaling, power, and the future of CMOS," in Proc. IEEE Int. Electron Devices Meeting (IEDM), Dec. 2005, pp. 7–15.

[2] D. E. Lackey et al., "Managing power and performance for system-on-chip designs using voltage islands," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Nov. 2002, pp. 195–202.

[3] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits," Proc. IEEE, vol. 91, no. 2, pp. 305–327, Feb. 2003.

[4] T. Burd and R. Brodersen, "Energy efficient CMOS microprocessor design," in Proc. 28th Hawaii Int. Conf. Syst. Sci., Jan. 1995, pp. 288–297.

[5] S. Palacharla, N. P. Jouppi, and J. E. Smith, "Complexity-effective superscalar processors," in Proc. 24th Annu. Int. Symp. Comput. Archit. (ISCA), Jun. 1997, pp. 206–218.

[6] G. Dhiman and T. S. Rosing, "Dynamic power management using machine learning," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Nov. 2006, pp. 747–754.

[7] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Utilizing predictive models for adaptive power management in embedded systems," in Proc. IEEE Int. Symp. Workload Characterization (IISWC), Sep. 2008, pp. 3–11.

[8] H. Shen, Y. Tan, J. Lu, Q. Wu, and Q. Qiu, "Achieving autonomous power management using reinforcement learning," ACM Trans. Des. Autom. Electron. Syst., vol. 18, no. 2, pp. 1–24, Apr. 2013.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in Proc. 5th Int. Conf. Learn. Representations (ICLR), Apr. 2017, pp. 1–14.

[10] A. Mirhoseini et al., "Device placement optimization with reinforcement learning," in Proc. 34th Int. Conf. Mach. Learn. (ICML), Aug. 2017, pp. 2430–2439.