SHEEO: Continuous Energy Efficiency Optimization in Autonomous Embedded Systems

Xinkai Wang, Chao Li, Lingyu Sun, Qizheng Lyu, Xiaofeng Hou, Jingwen Leng, Minyi Guo

Department of Computer Science and Engineering, Shanghai Jiao Tong University

Emails: {unbreakablewxk, sunlingyu, machinelvqz}@sjtu.edu.cn, {lichao, hou-xf, leng-jw, guo-my}@cs.sjtu.edu.cn

***Abstract*—The emerging trend of autonomous embedded systems minimizing human intervention has raised new questions about continuously maximizing system energy efficiency faced with stochastic runtime variance, which is costly for resource-constrained autonomous embedded systems. Considering heterogeneous hardware and variable software, we envision opportunities for vertical and horizontal shadow cycles within the AES pipeline for management facilities. This paper introduces SHEEO, a continuous energy efficiency optimizer that exploits underutilized heterogeneous computing resources to pursue variability-aware power management. To achieve this, SHEEO constantly monitors inner and outer variances and customizes reinforcement learning into two phases for stochastic runtime variance. We implement and deploy SHEEO on a commercial edge platform. The evaluation results show that SHEEO harvests up to 88% shadow cycles and improves up to 39% energy efficiency compared to state-of-the-art power management techniques with negligible overheads.**

***Index Terms*—Energy Efficiency, Continuous Learning, Shadow Cycles, Autonomous Embedded System**

I. Introduction

With an emphasis on intelligent systems minimizing human intervention, autonomous embedded systems (AES) have recently become a promising trend in both academia [8] and industry [16]. AES surpasses traditional human-involved systems by sensing the environment, perceiving the objects, and actuating the vehicle on highly integrated embedded platforms [10]. Considering the ability of AES to reduce the burden on people, many industry leaders such as Nvidia [16] and Tesla [19] have invested large amounts of capital and engineering power in developing such systems.

AES has three major differences from traditional well-studied systems, as shown in Figure 1. Firstly, the underlying architecture comprises heterogeneous hardware accelerators, providing varied domain-specific computing capabilities [9]. Also, the hardware heterogeneity requires more comprehensive and adaptive management facilities [4]. Secondly, the mission-critical tasks of AES are organized into a three-stage pipeline executing iteratively, i.e., sensing, perception, and actuation [10]. The perception stage consists of a fixed number of AI-based tasks that execute concurrently on heterogeneous hardware [25]. Thirdly, AES faces more severe stochastic runtime variance than before due to unpredictable execution status and continuous interaction with the environment [6], requiring variability-aware and intelligent power management under severe energy budget [3].

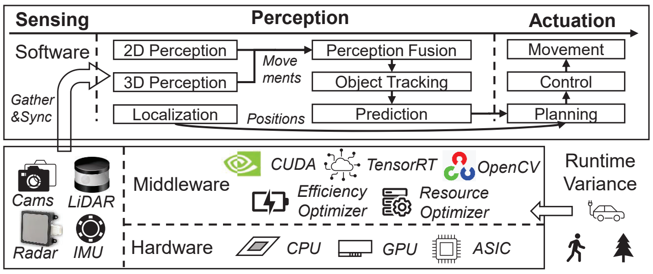


Fig. 1: Overview of autonomous embedded systems.

Faced with runtime variances, AES calls for a continuous and intelligent energy efficiency optimizer. Practically, the changing vehicle speed, outer environment surroundings, and inner task complexity constitute the stochastic nature of AES [8], which is ignored by current constraint-aware and workload-aware efficiency optimizer [2], [12]. Therefore, AES needs a variability-aware energy efficiency optimizer at the granularity of each three-stage iteration. What's worse, such continuous efficiency optimizer frequently occupies normal resources to make decisions, interfering with the mission-critical tasks in the pipeline.

The heterogeneous hardware and variable software result in underutilized heterogeneous computing resources in AES, which are desirable for deploying management facilities. We define the potential resource fragments in each three-stage iteration as *shadow cycles* and divide them into two categories. The *Vertical Shadow Cycles* (VSC) exist in the waiting time of heterogeneous hardware within the perception stage, and the *Horizontal Shadow Cycles* (HSC) exist in the sensing and actuation stages with idle accelerators. Inherently, the preemptive VSC and HSC are volatile and short-lived in duration, respectively. Shadow cycles are valuable for resource-constrained AES by enlarging the resource visibility, but it is challenging to transparently utilize them. Normal mission-critical tasks cannot be guaranteed on shadow cycles, but the auxiliary management tasks could seize the opportunities.

Considering the potential of underutilized resources and the need for runtime optimization, it is attractive to utilize shadow cycles for continuous energy efficiency optimizers. To this end, we propose **SHEEO**, a continuous **E**nergy **E**fficiency **O**ptimizer with **SH**adow cycles, which exploits the underutilized heterogeneous computing resources to enable variability-aware power management every iteration. SHEEO

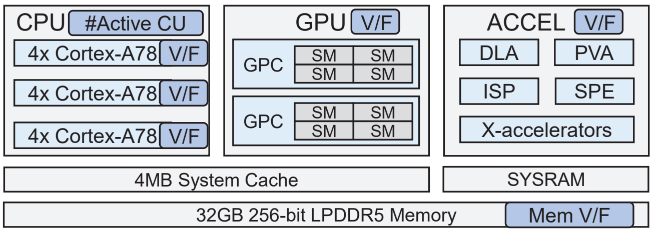


Fig. 2: Hardware architecture and power management tuning knobs of COTS embedded platforms.

is composed of two well-designed components to pursue optimal energy efficiency. It monitors the external environment dynamics and the internal execution status with *Real-time Monitor*. The *Continuous Learning Manager* customizes reinforcement learning based on AES properties into two phases to learn and optimize AES energy efficiency.

To evaluate SHEEO thoroughly, we implement and deploy it on a commercial edge platform, Nvidia Jetson Orin. Under various scenarios, we demonstrate that SHEEO harvests up to 88% shadow cycles within AES to deploy continuous energy efficiency optimizers. Further, SHEEO improves up to 39% and 27% energy consumption over state-of-the-art workload-aware and learning-based power management baselines via more aggressively tuning power with negligible overheads.

This paper makes the following key contributions:

1) **Analysis**: We envision the potential of shadow cycles within the heterogeneous autonomous embedded systems and the urgent need for continuously optimizing energy efficiency with runtime variances.
2) **Design**: We propose an intelligent continuous efficiency optimizer on shadow cycles SHEEO and design two components to learn and optimize AES efficiency.
3) **Evaluation**: We implement and deploy SHEEO on edge platforms to prove its effectiveness in harvesting shadow cycles and efficiency in improving AES energy efficiency under various scenarios.

The rest of this paper is organized as follows: Section II discusses the background and Section III introduces our motivations. Section IV proposes our detailed designs and Section V thoroughly evaluates SHEEO. Section VI presents discussion and Section VII concludes this paper.

II. Background and Related Work

A. Researches of Autonomous Embedded Systems

Autonomous systems operate independently without requiring human intervention, such as autonomous driving [16] and unmanned aerial vehicles [5]. They mainly have two differences with traditional systems. On the architectural aspect, unlike traditional systems deployed in large-scale datacenters with clear workload division, autonomous systems are usually deployed on edge embedded platforms such as commercial off-the-shelf (COTS) Nvidia Jetson AGX Orin [15] as shown in Figure 2. Such embedded platforms are equipped with abundant heterogeneous computing units, which are integrated into a single board and responsible for miscellaneous workloads.

Apart from architectural differences, AES exhibits distinct software execution patterns. It executes a horizontal three-stage pipeline iteratively, and diverse workloads run concurrently in a single stage [10]. Firstly, it gathers the environment information with diverse real-time sensors in the sensing stage [23]. Then, it performs concurrent processing and prediction of the objects with various deep neural networks to understand the external conditions in the perception stage. Finally, it makes decisions based on the processed information in the actuation stage. The perception stage mainly occupies heterogeneous computing power while the other two stages mainly occupy general computing power of CPU [9]. Every three-stage iteration has a dynamic latency constraint, and the AI-intensive perception stage takes up more than 90% execution time within each iteration [8].

Prior works from academia and industry focus on architecting and scheduling the AES [14], [22]. Shi-Chieh and Bo comprehensively present the architectural implications and design considerations of AES [8], [23]. Soroush proposes a predictable data-driven scheduler to manage the resources of AES [1]. However, these works ignore the hidden hetero-computing resource fragments due to the hardware and software heterogeneity within AES.

B. Intelligent and Heterogeneous Power Management

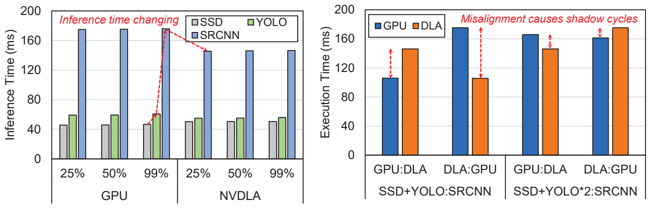
Faced with more complex hardware and software, researchers utilize machine learning methods to enhance traditional power management solutions. In cloud computing, workloads and runtime are more stable than AES, and researchers use predictive models to learn energy efficiency [11]–[13], [21], [24]. Such solutions target systems with little stochastic runtime variance and occupy additional hardware resources for learning and deciding. In the AES scenario, there are a few prior works on intelligent power management [2], [6], [17]. NeuOS is a latency-predictable multi-dimensional optimization framework for multi-DNN workloads in AES [2]. Such solutions pay more attention to the characteristics of workloads while overlooking the stochastic runtime variance and lacking real-time adaptation.

Further, power management for heterogeneous architectures is much more challenging. Dynamic voltage and frequency scaling (DVFS) for a single component often conflict when performed independently by separate controllers [4]. Prior works rely on empirical models to manage the power supply of CPU-GPU systems [7], but such static methods fail to grasp the changing behaviors of AES. Therefore, adaptive and intelligent power management is crucial to pursuing the efficiency optimals of heterogeneous AES.

III. Motivation

A. Harvesting shadow cycles in AES

In autonomous embedded systems, diverse neural networks execute on heterogeneous accelerators to understand the envi-



(a) Execution variability on heterogeneous hardware.

(b) Execution misalignment under various mappings.

Fig. 3: Different inferences exhibit much performance diversity on heterogeneous hardware.

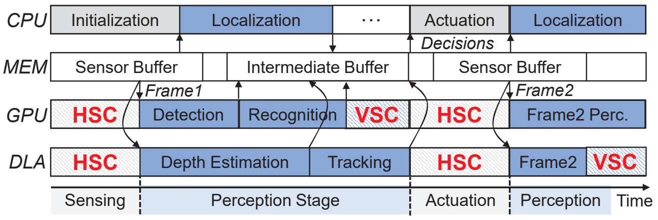


Fig. 4: Two types of underutilized shadow cycles within the execution pipeline of AES.

ronment [10], causing underutilized heterogeneous computing resources within the pipeline. We characterize representative workloads of AES [8], including object detection and object tracking, on COTS edge platform Nvidia Jetson AGX Orin [15]. We select three typical neural network inferences: 1) ssd-mobilenet-v1 (SSD), 2) yolov3-tiny-416 (YOLO), and 3) super-resolution-bsd500 (SRCNN) to constitute the perception stage. We average the results of 100 repeated experiments for each setting to reduce random biases.

Figure 3 reports the average performance, and we have two key observations. On the one hand, Figure 3a presents the inference time distribution of three tasks on GPU and deep learning accelerator (DLA). We find that the inference time varies among neural networks on different accelerators due to varied model sizes and hardware capabilities. On the other hand, Figure 3b shows the execution misalignment of various tasks within the perception stage. We select four typical mapping schemes that allocate different tasks to underlying accelerators. Tasks allocated on GPU and DLA are executed for varied duration and the unit that finishes ahead remains idle until the latter finishes. The misalignment phenomenon is widespread in AES with execution variability and is more complex with more concurrent tasks in realistic scenarios.

Given the AES characterizations, we envision the potential of underutilized heterogeneous computing resources within the execution pipeline, termed as *shadow cycles*. In this paper, we focus on the shadow GPU and DLA resources since they are mostly utilized in AES. The shadow cycles are familiar in traditional systems, such as multi-core CPUs that utilize idle cores for work stealing. We discuss more in Section VI-B. However, AES executes in a fixed and repetitive manner, making shadow cycles more fragmented and variable. As shown in Figure 4, we categorize two types of shadow cycles:

1) **Vertical Shadow Cycles (VSC)**: The idle hetero-computing resources within the waiting time until the slowest task in the perception stage.
2) **Horizontal Shadow Cycles (HSC)**: The idle hetero-computing resources in the sensing and actuation stages of the horizontal pipeline.

More specifically, shadow cycles are different from normal heterogeneous computing resources. Firstly, the underutilized resource fragments are *preemptive* since the normal functionalities are most important. Secondly, VSC results from the misaligned execution of concurrent tasks within the perception stage, so it is *volatile* in its duration due to varied task and hardware mappings. For example, the VSC in Figure 3b ranges from 14-70 milliseconds, depending on the perception stage settings. Thirdly, HSC results from the horizontal pipeline with CPU-intensive actuation stages, so it is *short-lived* in its duration since the actuation stage occupies a small proportion in each iteration. For a typical 100ms end-to-end latency constraint, the HSC exists around 5-10 milliseconds [8] in every 100ms AES pipeline.

In resource-constrained AES, utilizing shadow cycles is both valuable and challenging. Exploiting these underutilized resources could fully release the heterogeneous computing power in AES but faces two key challenges. (1) *How to capture shadow cycles efficiently?* Given the volatile VSC and short-lived HSC, it is crucial to capture and organize them for utilization quickly. (2) *How to exploit shadow cycles transparently?* Even if we could manage shadow cycles as wished, the preemptive property makes it suitable to deploy auxiliary tasks on them instead of normal mission-critical tasks. We analyze that VSC fits tasks executing adaptively in length while HSC fits tasks executing rapidly and demanding little hardware resources.

B. *Continuous Learning for Energy-Efficient AES*

Running AES is a dynamic system continuously changing and interacting with the environment, requiring an adaptive energy efficiency optimizer. The runtime dynamics mainly contain three parts: 1) *vehicle speed*, which changes with time and influences the latency constraints of AES [23]; 2) *surrounding changes*, which denotes the external environment variation and determines the understanding complexity for AES [5]; 3) *task complexity*, which varies in each iteration based on AES status. Such dynamic factors exist continuously such that the current constraint-aware efficiency optimizer [2] fails to make optimal power management decisions.

As shown in Figure 5, for a running AES, *case 1* is straightforward, and the optimal decision is to decrease the power supply (i.e., decrease V/F level). However, AES in *case 2 and 3* experiences runtime dynamics, and the traditional method fails to make continuous adaptations. Latency constraints in *case 4* are tight, but reducing power supply is safe and energy-efficient in such scenarios with an understanding of current

AES	Urban Road	Traffic Jam	Overtake	Highway
Factors	Case 1	Case 2	Case 3	Case 4
Vehicle Speed	Low	Low	Fluctuated	High
Surrounding Change	Low	Fluctuated	Low	Low
Task Complexity	Low	High	High	Low
Power Management Decisions	↓	↓	↓	↓
Current Method	Decrease V/F		Increase V/F	
Expected Method	V/F ↓	Adapting V/F with time		V/F ↓

Fig. 5: Current energy efficiency optimizers cannot manage the stochastic runtime variance faced by AES.

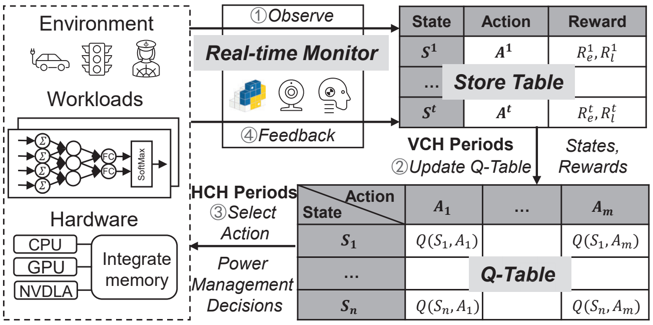


Fig. 6: Workflow of the two components of SHEEO.

surroundings. In summary, AES calls for a continuous and intelligent energy efficiency optimizer.

Considering the analysis of shadow cycles and runtime dynamics of AES, it is necessary and suitable to utilize shadow cycles for continuous efficiency optimizer. Firstly, the optimal power management adapting to runtime dynamics is continuous learning, which learns behaviors through trial-and-error interactions with the environment and easily scales to heterogeneous hardware management. However, it is costly to occupy normal heterogeneous computing resources in every iteration [12]. Thus, deploying the continuous efficiency optimizer with shadow cycles is necessary for resource-constrained AES. Secondly, continuous learning fits well with shadow cycles within the AES pipeline. VSC is suitable for the training (learning) process that is preemptive on heterogeneous computing units and can be performed in variable sizes. HSC is suitable for the inference (control) process that is relatively lightweight and performed in every iteration.

IV. Design

Based on the above analysis, it is of great help for autonomous embedded systems to exploit shadow cycles for continuously optimizing energy efficiency. In this work, we propose **SHEEO**, a continuous **E**nergy **E**fficiency **O**ptimizer with **SH**adow cycles, which exploits the underutilized heterogeneous computing resources to perform variability-aware power management at iteration granularity. SHEEO automates the above process through a real-time monitor and a continuous learning manager, as shown in Figure 6.

A. Real-time Monitor

SHEEO abstracts away the complexity of the external environment and internal software and hardware through the real-time monitor. During runtime, it observes both the underutilized periods and the execution conditions with several ready-to-use systems interfaces and methods. Firstly, it closely monitors the execution pipeline of AES. We modify the tasks of AES to report their start time and end time to the monitor in real-time. When an individual task in the perception stage finishes without subsequent candidates, the monitor marks the start of VSC periods for the continuous learning manager. When the slowest task of the perception stage finishes, the monitor marks the end of VSC periods. In other stages, the monitor marks HSC periods until the start of perception tasks.

In addition, the real-time monitor is responsible for interacting with the environment and system. It monitors the vehicle speed and environment changes as runtime dynamics and records the metadata of executing workloads for every iteration by collecting the computing and memory utilization through the *jetson_stat* tool. The collected information is organized into an ordered array indexed by iteration and fed into the continuous learning manager to make decisions. Equipped with the monitor, SHEEO can harvest shadow cycles accurately and continuously optimize AES energy efficiency.

B. Continuous Learning Manager

The continuous learning manager of SHEEO takes into account real-time knowledge to manage energy efficiency. Among the various reinforcement learning methods, such as Q-learning, TD-learning, and deep-RL, we choose Q-learning for the continuous efficiency optimizer since it employs a lookup table to find the best action with low overhead. We implement the continuous learning manager by Q-learning with value function $Q(S, A)$, which takes state S and action A as parameters and is called a Q-Table. To fully utilize the historical information with shadow cycles, we design a Store Table to record the unexploited data.

As shown in Figure 6, the monitor ① observes the environment status and system states and stores them in the Store Table. During VSC periods, the manager ② updates the Q-Table according to historical information in the store table. During HSC periods, the manager ③ selects the optimal power management actions with the best energy efficiency based on the Q-table. After the execution is completed, the manager ④ gets feedback from the observing objects and calculates the reward to the store table. The continuous learning manager is composed of three key components (state, action, reward) and two well-designed algorithms.

State: As shown in Table Ia, the multi-dimensional state contains workload features and runtime features. For neural network inference tasks, the number of convolutional layers (CONV) S_{CONV}, fully-connected layers (FC) S_{FC}, and recurrent layers (RC) S_{RC} is deeply correlated with the energy efficiency and performance of inference execution. Based on

TABLE I: Design details of continuous learning manager.

(a) Multi-dimensional state-related features

State		Descriptions	Discrete Values
Workload Features	S_{CONV}	Number of CONV layers	L (<30), M (<50), H (≥50)
	S_{FC}	Number of FC layers	L (<10), H (≥10)
	S_{RC}	Number of RC layers	L (<10), H (≥10)
Runtime Dynamics	S_{Speed}	Current vehicle speed	L (<10mph), M (<40mph), H (≥40mph)
	S_{Var}	Variation from last execution	L (<10%), M (<30%), H (≥30%)
	S_{Com}	Computing units' utilization	L (<25%), M (<75%), H (≥75%)
	S_{Mem}	Memory utilization	L (<25%), M (<75%), H (≥75%)

(b) Power tuning actions

Actions	Descriptions
S_X	Turn On/Off Component X
N_{CPU}	Number of Active Core
F_{CPU}	V/F Level of CPU Core
F_{GPU}	V/F Level of GPU
F_{DLA1}	V/F Level of DLA1
F_{DLA2}	V/F Level of DLA2
F_{MEM}	V/F Level of Memory

Algorithm 1: Algorithm for continuous learning

```
Input: Pre-trained Q-table $Q(S, A)$, learning rate $\alpha$,
       discount factor $\gamma$, exploration probability $\epsilon$
Output: Fine-tuned Q-table and Action A for each
        iteration
1  while $\exists$ VSC periods and $\exists$ Store Table do
2      Calculate recorded reward $R_{energy}$ and $R_{latency}$;
3      For consecutive $S$ and $S'$ and chosen action $A$;
4      Choose action $A'$ with the largest $Q(S', A')$;
5      $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$;
6      $S \leftarrow S'$
7  end
8  foreach HSC periods do
9      Gather $S$ and locate in $Q(S, A)$;
10     if rand() $< \epsilon$ then
11         Choose action $A$ randomly;
12     else
13         Choose action $A$ with the largest $Q(S, A)$;
14     end
15     Store Table $\leftarrow$ (A, S') ;
16 end
```

the analysis above, the fluctuated vehicle speed S_{Speed} and the surrounding changes S_{Var} (represented by the variation between adjacent perceptions) represent stochastic runtime variance. In addition, the current utilization of computing units S_{Com} and memoryS_{Mem} is vital for future energy efficiency decisions. We summarize the above into a seven-tuple state and set the discrete values for each feature based on prior characterization and analysis.

Action: Table Ib lists the power tuning actions of the manager, which represent the adjustable knobs of heterogeneous hardware in AES. N_{CPU} denotes the core throttling mechanism to shut down the idle cores. F_X denotes the DVFS mechanism on different components. The actions constitute a six-tuple for each iteration to decide the configuration for each component. The tuning range for each knob is determined by the power mode configurations and is restricted by the overall power supply of AES (e.g., maximum 50W for Orin). For example, if the latency constraints are satisfied in the last iteration and the environment becomes easier to understand, it is possible to shut down some computing units and reduce the frequency of active units to save energy. The set of actions is determined at the hardware layer for now and can be augmented by considering more software tuning knobs.

Reward: The reward determines the optimization objectives in each iteration, and we design a two-fold reward R to pursue optimal energy efficiency under latency constraints. We define the reward of SHEEO as the weighted sum of performance reward $R_{latency}$ and efficiency reward R_{energy}. R_l is the measured latency of the perception stage for selected actions from the real-time monitor. R_e is the measured energy usage of the targeted hardware defined as below. Here we demonstrate Nvidia Jetson AGX Orin, where VDD_GPU_SOC and VDD_CPU_CV represent the power of all targeted hardware units and T_b (T_e) represents the begin (end) time.

$$R = aR_{latency} + bR_{energy} \text{ s.t. } R_l \leq \text{Constraint}$$

$$R_{latency} = \max_{t \in \text{Perception}} T^t_{end} - T_{start}$$

$$R_{energy} = \sum_i E^i_{Unit} = \sum_i \int_{T_b}^{T_e} P_f \delta t + P_{idle} \times t_{idle}$$

Aside from three elements, the continuous learning manager pursues agile decision-making and uses Q-learning to exploit the low decision latency. We carefully design a continuous learning algorithm to utilize the VSC and HSC. Algorithm 1 shows the process of continuously training the Q-table within VSC periods. It inputs a pre-trained Q-table $Q(S, A)$, learning rate α denoting the importance of rewards, and discount factor γ denoting the relationship between consecutive states. It will iteratively fine-tune the Q-table until the VSC period ends or historical information is all used. For each iteration within VSC periods (line 1), it calculates the reward (line 2), gets consecutive information from the store table (line 3), chooses the corresponding action (line 4), updates the $Q(S, A)$ based on the equation (line 5). With algorithm 1, SHEEO can utilize vertical shadow cycles adaptively and fine-tune the Q-table for more accurate decisions constantly.

Concerning HSC, Algorithm 1 shows the details for selecting actions maximizing energy efficiency. To deal with the exploitation versus exploration dilemma, SHEEO employs the epsilon-greedy method with parameter ϵ, which chooses the action with the highest reward or a uniformly random action based on an exploration probability. For the observed S in the Q-table (line 9), it evaluates a random value compared with ϵ (line 10). If the random value is smaller than ϵ, the algorithm chooses A randomly for exploration (line 11). Otherwise, it

TABLE II: Evaluation platform specifications.

Device	Nvidia Jetson AGX Orin Module
CPU	8-core ARM Cortex-A78 v8.2
GPU	1792-core Ampere GPU with 56 tensor cores
Memory	32GB 256-bit LPDDR5
Accelerator	2x NVDLA v2, 1x PVA v2
System	Linux 5.10.104-tegra with Jetpack 5.1.1
Software	CUDA 11.4 and TensorRT 8.5.2

chooses A with the largest $Q(S, A)$ (line 13). After that, the information is recorded in the store table for future usage (line 15). SHEEO can select actions agilely with maximized energy efficiency, and the decision process fits HSC perfectly.

V. Evaluation

In this section, we thoroughly evaluate SHEEO. Specifically, we want to answer three questions:

1) How can SHEEO harvest shadow cycles in AES?
2) How can SHEEO continuously optimize efficiency?
3) How about the deployment overhead of SHEEO?

A. Evaluation Methodology

Implementations To evaluate SHEEO in realistic AES, we implement a prototype of the two components in Python. As for the hyperparameters in SHEEO, we set the learning rate α as 0.9 to reflect the reward more to the Q values and the discount factor γ as 0.1 to decrease the relationship between consecutive states due to the stochastic nature of AES. Also, we empirically set the exploration probability ϵ as 0.1 to prefer exploitation instead of exploration.

Specifications: We perform the evaluations on a typical COTS embedded platform, Nvidia Jetson AGX Orin [15], and Table II summarizes the platform specifications. We focus on the shadow cycles on GPU and NVDLA since they are mostly used for AES inference tasks. During the execution, we record the hardware utilization with *jetson_stat* tool and measure the power with the onboard power meter updated at 5ms intervals to calculate energy consumption. We overwrite the default power mode setting to set 12 frequency levels for both GPU and DLA. As for the software, we take three typical neural networks: SSD, YOLO, and SRCNN with FP16 precision, as detailed in Section III-A, and we combine them together to mimic the perception stage of the AES pipeline.

Baselines: To fully evaluate SHEEO, we compare it with two types of state-of-the-art (SOTA) baselines: Workload-Aware Control (WAC) methods [2] and Learning-Based Control (LBC) methods [12]. The former relies on the static model to profile workloads and makes power management decisions based on the system constraints (e.g., LAG analysis). The latter uses a machine learning model and historical statistics to make optimal power management decisions. We compare the SOTA works of the two categories with SHEEO.

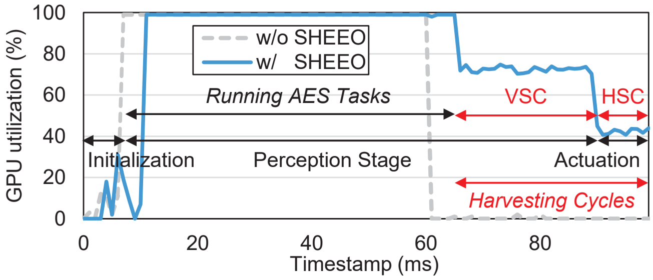


Fig. 7: RQ1: SHEEO harvests two types of shadow cycles.

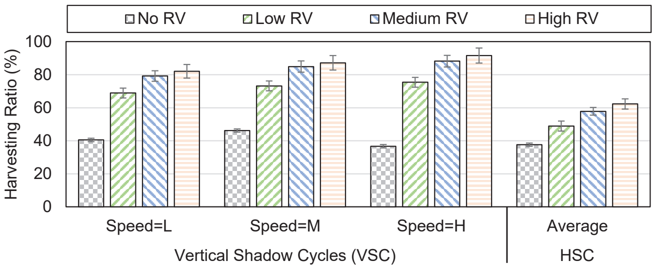


Fig. 8: RQ1: The harvesting ratio under different scenarios.

B. Evaluation Results

1) Effectiveness (RQ1): The primary goal of SHEEO is harvesting the underutilized resources within autonomous embedded systems. Figure 7 shows an example iteration in the three-stage AES pipeline, composed of sensing, perception, and actuation stages. We present the effectiveness by comparing the heterogeneous hardware utilization with and without SHEEO, especially GPU since it finishes later than DLA here. With SHEEO in the blue line, AES utilizes the virtual and horizontal shadow cycles in the pipeline to finish the power management training and inference. On average, the VSC periods harvest more than 70% resources, and the HSC periods harvest around 40% resources since Q-learning inference is lightweight and agile. SHEEO rescues resource-constrained AES from ignoring hetero-computing power within the pipeline and wasting normal resources for power management functionalities.

Moreover, Figure 8 presents the harvesting ratio of SHEEO under different scenarios. The harvesting ratio is defined as the average improved utilization within shadow cycle periods compared to that without SHEEO. SHEEO harvests 41%, 72%, 84.1%, and 88% VSC for different runtime variances and 51% HSC on average. We have three key findings. Firstly, the harvesting ratio is higher under high vehicle speed scenarios. The tighter latency constraints leave shorter VSC periods with processing the same perception tasks. Secondly, the harvesting ratio is higher with more significant runtime variance, which requires more frequent Q-table training. The standard deviation under such scenarios is higher than that of others since the duration of VSC periods is unstable. Thirdly, the low vehicle speed and low runtime variance scenarios have more shadow cycles, which presents opportunities for heavier management functionalities. Overall, SHEEO makes use of underutilized hetero-computing resources within normal execution instead of occupying additional hardware resources for continuous power management.

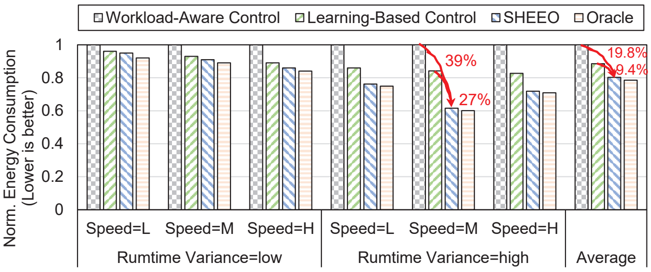


Fig. 9: RQ2: SHEEO reduces the energy consumption of AES execution under various scenarios.

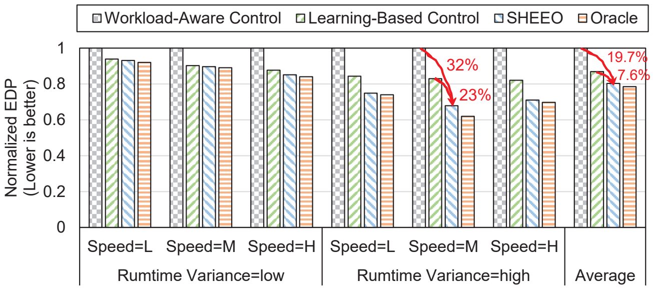


Fig. 10: RQ2: SHEEO optimizes energy consumption with little cost of execution latency.

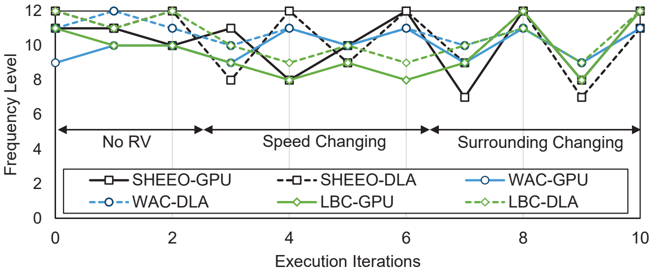


Fig. 11: RQ2: SHEEO tunes hardware frequency with variability-awareness to optimize energy consumption.

2) Efficiency (RQ2): With harvested shadow cycles, SHEEO aims to provide a more agile and accurate power management solution. Figure 9 presents the comparison of energy consumption via different power management schemes. The *Oracle* denotes ideally offline power management, making optimal energy under all scenarios. All results are normalized to the WAC baseline. On average, SHEEO outperforms WAC and LBC by 9.4% and 3% with low runtime variance, where the surrounding changes are less for adaptive management. With high runtime variance, SHEEO outperforms the two baselines by 21% and 16%, respectively, since the baselines are ignorant of the changing conditions. In addition, we have two findings. Firstly, with low runtime variance, power management with workload awareness is adequate for near-optimal AES energy efficiency. Secondly, the best 27% improvement of SHEEO happens with medium vehicle speed and high runtime variance since the tight latency constraints under high speed leave little power tuning space for SHEEO.

Moreover, Figure 10 presents the energy-delay product (EDP) of SHEEO to further evaluate the efficiency. EDP is mostly used as a more comprehensive metric to evaluate both energy efficiency and performance. On average, SHEEO outperforms WAC by 19.7% and LBC by 7.6%, respectively. Combined with energy consumption results, we find that SHEEO tends to sacrifice a little execution latency for better energy efficiency. With high runtime variance, SHEEO can adapt to the changes on the latency constraints, surrounding changes, and workload complexity.

Further, we present the frequency tuning process as shown in Figure 11 to investigate the energy efficiency improvements of SHEEO. We modify the default power mode of Orin to set 12 optional frequency levels for GPU and DLA. The GPU frequency can be adjusted from 306MHz to 1.3GHz. Figure 11 contains three different phases to represent realistic AES execution environments. In the *No RV* phase, SHEEO performs similarly to LBC since they both learn the AES behaviors. In the *Speed Changing* phase with changing latency constraints, SHEEO makes similar decisions with WAC but tunes the frequency more to speed up or slow down the execution more aggressively. In the *Surrounding Changing* phase, there are more runtime variances, and SHEEO makes more varied frequency tuning decisions to make the optimal decision in the current status, resulting in reduced energy consumption. However, the reduced frequency could sacrifice a little execution latency for the pursuit of energy efficiency, like iteration 7.

3) Overhead (RQ3): Another critical aspect of SHEEO is the deployment overhead in resource-constrained AES. The overhead contains two parts: training overhead and control overhead. The training utilizes VSC, and the inference utilizes HSC, which has no latency effect on the AES pipeline. As for energy overhead, SHEEO consumes 4.5 mJ for each training epoch and 1.7 mJ for each control process, corresponding to less than 1% reduced energy. Baseline power management schemes require normal resources and incur both latency and energy overheads. Moreover, the memory requirement of SHEEO is 1.3MB, translating to only 0.01% of the 32 GB DRAM capacity of the evaluated COTS AES platform.

VI. Discussion

A. Broader Utilization of Shadow Cycles

In this paper, we envision the underutilized shadow cycle opportunity in AES to expand the resource visibility of sophisticated designs on normal resources [1], [18], [20]. However, continuous learning energy efficiency is just a suitable and necessary management functionality that can be offloaded to shadow cycles, and this opportunity could be seized by more costly middleware in the resource-constrained AES. Except for normal mission-critical tasks with the highest priorities, the management tasks (e.g., power management, task mapping, cloud synchronization) can be offloaded to the volatile VSC and short-lived HSC. Therefore, SHEEO acts as an enlightening case for the broader utilization of shadow cycles.

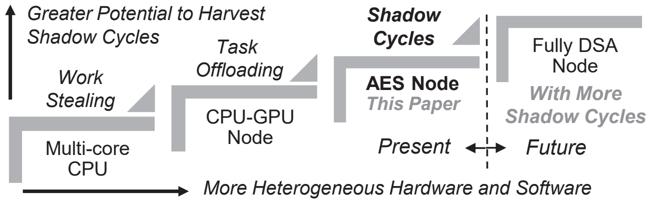


Fig. 12: Foresight of shadow cycles in future architectures.

B. Foresight of Shadow Cycles

Shadow cycles of AES result from misalignment in nature, which is a long-lasting topic in different system architectures. As shown in Figure 12, the work-stealing mechanism aims for unbalanced multi-core processors, and time-consuming tasks could be offloaded to GPU in CPU-GPU architecture. In resource-constrained AES, shadow cycles exhibit unique properties and are properly utilized by SHEEO. Future platforms will be equipped with more diverse domain-specific accelerators (DSA), offering varied computing capabilities for various tasks. With more heterogeneous hardware and variable software, the potential of underutilized and misaligned resources is even more worth exploring.

VII. Conclusion

This paper analyzes the potential of underutilized shadow cycles in AES and the need for variability-aware energy efficiency optimizers faced with runtime variance. We propose SHEEO to exploit shadow cycles for continuously optimizing energy efficiency. The evaluations show that SHEEO harvests up to 88% shadow cycles and improves up to 39% energy efficiency compared with SOTA works with negligible overheads. We expect SHEEO could motivate broader resource visibility for the management facilities of AES.

Acknowledgment

We sincerely thank all the anonymous reviewers for their valuable feedback. This work is supported by the STCSM International S&T Cooperation Project (23510713200) and the National Natural Science Foundation of China (No. U23A6007). The corresponding author is Chao Li.

References

[1] S. Bateni and C. Liu, "Predictable data-driven resource management: an implementation using autoware on autonomous platforms," in *2019 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2019, pp. 339–352.

[2] ——, "Neuos: A latency-predictable multi-dimensional optimization framework for dnn-driven autonomous systems," in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, 2020, pp. 371–385.

[3] P. H. Becker, J. M. Arnau, and A. González, "Demystifying power and performance bottlenecks in autonomous driving systems," in *2020 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 2020, pp. 205–215.

[4] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini, "Coscale: Coordinating cpu and memory system dvfs in server systems," in *2012 45th annual IEEE/ACM international symposium on microarchitecture*. IEEE, 2012, pp. 143–154.

[5] R. Hadidi, B. Asgari, S. Jijina, A. Amyette, N. Shoghi, and H. Kim, "Quantifying the design-space tradeoffs in autonomous drones," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021.

[6] Y. G. Kim and C.-J. Wu, "Autoscale: Energy efficiency optimization for stochastic edge inference using reinforcement learning," in *2020 53rd Annual IEEE/ACM international symposium on microarchitecture (MICRO)*. IEEE, 2020, pp. 1082–1096.

[7] T. Komoda, S. Hayashi, T. Nakada, S. Miwa, and H. Nakamura, "Power capping of cpu-gpu heterogeneous systems through coordinating dvfs and task mapping," in *2013 IEEE 31st International Conference on computer design (ICCD)*. IEEE, 2013, pp. 349–356.

[8] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving: Constraints and acceleration," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, 2018, pp. 751–766.

[9] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi, "Computing systems for autonomous driving: State of the art and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6469–6486, 2020.

[10] S. Liu, J. Tang, Z. Zhang, and J.-L. Gaudiot, "Computer architectures for autonomous driving," *Computer*, vol. 50, no. 8, pp. 18–25, 2017.

[11] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM special interest group on data communication*, 2019, pp. 270–288.

[12] N. Mishra, C. Imes, J. D. Lafferty, and H. Hoffmann, "Caloree: Learning control for predictable latency and low energy," *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 184–198, 2018.

[13] N. Mishra, H. Zhang, J. D. Lafferty, and H. Hoffmann, "A probabilistic graphical model-based approach for minimizing energy under performance constraints," *ACM SIGARCH Computer Architecture News*, vol. 43, no. 1, pp. 267–281, 2015.

[14] M. A. Neggaz, I. Alouani, P. R. Lorenzo, and S. Niar, "A reliability study on cnns for critical embedded systems," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*. IEEE, 2018, pp. 476–479.

[15] Nvidia, "Nvidia drive agx orin," Available: https://www.nvidia.cn/self-driving-cars/drive-platform/hardware/, aug, 2022.

[16] ——, "Solutions for self-driving cars & autonomous vehicles," Available: https://www.nvidia.com/en-us/self-driving-cars/, aug, 2022.

[17] L. Sun, X. Hou, C. Li, J. Liu, X. Wang, Q. Chen, and M. Guo, "A2: Towards accelerator level parallelism for autonomous micromobility systems," *ACM Transactions on Architecture and Code Optimization*, 2024.

[18] L. Sun, C. Li, X. Hou, T. Huang, C. Xu, X. Wang, G. Bao, B. Sun, S. Rui, and M. Guo, "Jigsaw: Taming bev-centric perception on dual-soc for autonomous driving," in *45th IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2024.

[19] Tesla, "Autopilot," Available: https://www.tesla.com/autopilot.

[20] X. Wang, H. He, Y. Li, C. Li, X. Hou, J. Wang, Q. Chen, J. Leng, M. Guo, and L. Wang, "Not all resources are visible: Exploiting fragmented shadow resources in shared-state scheduler architecture," in *Proceedings of the 2023 ACM Symposium on Cloud Computing*, 2023, pp. 109–124.

[21] X. Wang, C. Li, L. Zhang, X. Hou, Q. Chen, and M. Guo, "Exploring efficient microservice level parallelism," in *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2022, pp. 223–233.

[22] Y. Xue, J. Ji, X. Li, S. Li, S. Zhou, T. Cheng, L. Li, and Y. Fu, "Aome: Autonomous optimal mapping exploration using reinforcement learning for noc-based accelerators running neural networks," in *2022 IEEE 40th International Conference on Computer Design (ICCD)*. IEEE, 2022, pp. 364–367.

[23] B. Yu, W. Hu, L. Xu, J. Tang, S. Liu, and Y. Zhu, "Building the computing system for autonomous micromobility vehicles: Design constraints and architectural optimizations," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 1067–1081.

[24] L. Zhang, C. Li, X. Wang, W. Feng, Z. Yu, Q. Chen, J. Leng, M. Guo, P. Yang, and S. Yue, "First: Exploiting the multi-dimensional attributes of functions for power-aware serverless computing," in *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2023, pp. 864–874.

[25] H. Zhao, Y. Zhang, P. Meng, H. Shi, L. E. Li, T. Lou, and J. Zhao, "Driving scenario perception-aware computing system design in autonomous vehicles," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*. IEEE, 2020, pp. 88–95.