Enabling Learning-Based Efficiency Optimizer With Shadow Cycles in Resource-Constrained Autonomous Embedded Systems

Xinkai Wang, *Graduate Student Member, IEEE*, Chao Li, *Senior Member, IEEE*, Yiyang Li, Lingyu Sun, *Graduate Student Member, IEEE*, Cheng Xu, *Graduate Student Member, IEEE*, Xiaofeng Hou, *Member, IEEE*, Jing Wang, *Member, IEEE*, Minyi Guo, *Fellow, IEEE*, and Yaqian Zhao, *Senior Member, IEEE*

***Abstract*—The emerging trend of autonomous embedded systems (AES) is promising to minimize human intervention in critical tasks. In the pursuit of maximal per-watt performance, the complex hardware and software of AES require intelligent energy efficiency optimizers (EO), and the stochastic runtime variances require continuous EO. However, deploying the desirable on-device EO causes severe performance slowdown due to contention on limited computing power with the AES pipeline. We find that there are ignored and underutilized heterogeneous resources within AES for costly EO, which results from unbalanced accelerator behaviors and misaligned parallel inference executions. We experimentally and theoretically analyze the *Shadow Cycles* within the realistic autonomous Bird's Eye View pipeline on commercial embedded platforms, categorizing them into vertical and horizontal types with distinct properties. In this paper, we introduce *SHEEO+*, a continuous and intelligent energy efficiency optimizer that utilizes ignored heterogeneous shadow cycles. It achieves continuous and lightweight AES monitoring with the observation module, as well as intelligent and efficient AES power management with the optimization module. On the one hand, *SHEEO+* observes both the internal runtime status and external environment variance with portable interfaces to capture shadow cycles and real-time states. On the other hand, *SHEEO+* optimizes power configurations per iteration based on deep reinforcement learning (DRL) methods. It tailors DRL for two types of shadow cycles and invocates optimization processes based on resource availability. To extensively evaluate *SHEEO+*, we implement a prototype and deploy it on realistic edge platforms. The evaluation results show that *SHEEO+* utilizes up to 74.2% shadow cycles and achieves up to 18.6% energy efficiency improvements compared to state-of-the-art energy efficiency optimizers with negligible deployment overheads.**

***Index Terms*—Energy efficiency optimization, shadow cycle, reinforcement learning, autonomous embedded system.**

Received 16 March 2025; revised 27 July 2025; accepted 7 December 2025. Date of publication 16 December 2025; date of current version 12 February 2026. This work was supported in part by STCSM International S&T Cooperation Project under Grant 23510713200, and in part by the National Natural Science Foundation of China under Grant U23A6007. An earlier version of this paper was presented at the ICCD'24 [DOI: 10.1109/ICCD63220.2024.00082]. Recommended for acceptance by X. Liu. *(Corresponding author: Chao Li.)*

Xinkai Wang, Chao Li, Yiyang Li, Lingyu Sun, Cheng Xu, Xiaofeng Hou, Jing Wang, and Minyi Guo are with the School of Computer Science, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: lichao@cs.sjtu.edu.cn).

Yaqian Zhao is with the IEIT Systems Company Ltd., Jinan 250014, China.

Digital Object Identifier 10.1109/TC.2025.3644184

I. Introduction

THE development of autonomous embedded systems (AES), characterized by their diminished reliance on human interaction, represents a significant trajectory within academic [1], [2] and industrial domains [3], [4]. AES, in contrast to conventional human-operated systems, leverages integrated embedded platforms to perform environmental sensing, object perception, and vehicle actuation [5]. The capability of AES to alleviate human labor has spurred substantial investments from prominent industrial entities, including Nvidia [3] and Tesla [4], into technology advancements.

The emerging AES exhibits hardware and software differences compared with conventional large-scale systems as shown in Fig. 1. Firstly, different from abundant server resources [6], the underlying embedded architecture incorporates diverse hardware accelerators, providing domain-specific processing capabilities [2]. However, the computing power and energy supply are strictly limited due to Size, Weight, and Power (SWaP) constraints [7], [8]. Secondly, different from request-driven loosely-coupled applications [9], the essential AES software operations are structured as an iterative three-stage pipeline, encompassing sensing, perception, and actuation, to automatically complete critical missions [1], [5]. The perception stage involves a fixed set of deep neural network-driven tasks that operate in parallel on heterogeneous accelerators [10], [11]. Moreover, the large-scale and on-device facilities are more complex to make the optimal decisions and induce higher costs [12], [13].

Given the energy and resource limits, optimizing energy efficiency is important but challenging for AES. It is crucial for AES to tackle the temporal and spatial variances towards better efficiency. The stochastic environment dynamics, such as vehicle speed and surrounding complexity, are changing with time, requiring continuous decision-making [14], [15]. Complex inferences on heterogeneous accelerators have distinct behaviors and constitute a large design space, requiring intelligent

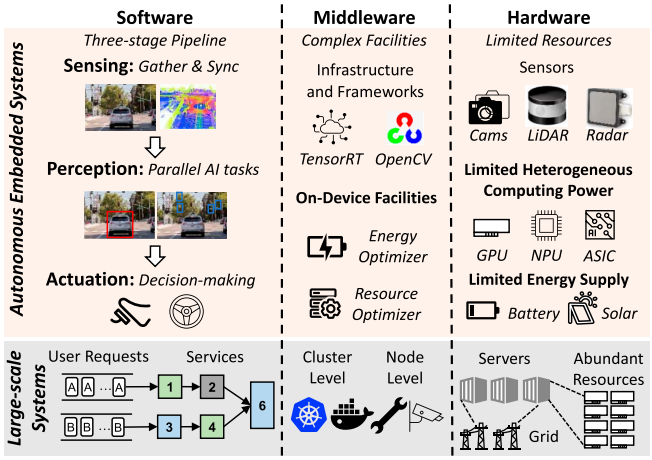

Fig. 1. Overview of autonomous embedded systems (AES) and comparison with large-scale systems.

optimization [16], [17]. Thus, an on-device machine learning-based optimizer operating at the iteration granularity is desirable for AES. However, deploying such a complex facility poses difficulties for resource-constrained AES. The computational demands of the optimizer are costly for embedded accelerators and interfere with mission-critical inference tasks, causing up to 13% AES performance slowdown.

To deploy the costly efficiency optimizer without slowdown, we envision the ignored opportunities of unexploited resources within the AES pipeline. The variable inferences executed on heterogeneous accelerators (i.e., GPU and DLA) in parallel lead to misaligned patterns, causing idle accelerator periods in AES execution, termed as *shadow cycles*. Based on distinct causes of occurrences, we categorize shadow cycles into two types. Firstly, *vertical shadow cycles (VSC)* result from misaligned execution of parallel perception tasks, denoting the waiting time until the slowest tasks within the perception stage. Secondly, *horizontal shadow cycles (HSC)* result from the inherent AES pipeline pattern, denoting the accelerator-idle sensing and actuation stages. We investigate shadow cycles within both simple parallel inference and modern Bird's Eye View (BEV) pipeline, whose proportions are 24-42% for VSC and 5% for HSC (detailed in Section III-B). They enlarge the resource visibility of resource-constrained AES and are valuable for complex on-device facilities. Different from regular resources, the preemptive VSC and HSC are volatile and fleeting in duration, presenting non-trivial challenges to capture and exploit them.

In this paper, we propose ***SHEEO+***, a **SH**adow cycles-based **E**nergy **E**fficiency **O**ptimizer that exploits the under-utilized resources to enable continuous and intelligent power management. To improve the efficiency manageability of AES, *SHEEO+* aims to make the best use of available shadow cycles to minimize AES pipeline interference and maximize energy efficiency. To improve manageability efficacy, it customizes deep reinforcement learning (DRL) methods to continuously observe AES variances and optimize power configurations of every iteration. Meanwhile, to improve manageability efficiency, it manages and identifies shadow cycles to deploy the learning-based optimizer without AES interferences.

More specifically, *SHEEO+* incorporates two cooperative modules to pursue optimal energy efficiency optimization. Firstly, the *observation module* detects shadow cycles based on light-weight signals and organizes them agilely. It also uses portable interfaces to collect real-time external environments and the internal status of AES for joint optimization. Secondly, the *optimization module* judges shadow cycle states to invocate the DRL agent, which uses VSC to fine-tune the actor-critic networks based on execution statistics and HSC to infer the optimal action based on the current AES state.

To evaluate *SHEEO+* thoroughly, we implement and deploy it on a commercial edge platform, Nvidia Jetson Orin [18]. Under changing scenarios and variances, we demonstrate that *SHEEO+* harvests up to 74.2% vertical shadow cycles and 39.9% horizontal shadow cycles within the AES pipeline to deploy continuous energy efficiency optimizers. Further, *SHEEO+* reduces up to 18.6% and 5.6% energy consumption over state-of-the-art workload-aware and learning-based power management baselines with little performance sacrifice. Moreover, the deployment costs and execution overheads of *SHEEO+* are negligible compared with conventional power management solutions.

In summary, this paper makes three main contributions:

1) **Analysis**: We identify the performance slowdown caused by intelligent energy efficiency optimizers and envision the latent potential of shadow cycles within heterogeneous AES to deploy the facility efficiently.
2) **Design**: We introduce an intelligent continuous efficiency optimizer, *SHEEO+*, utilizing shadow cycles and incorporating reinforcement learning. Two cooperative components are designed to facilitate the learning and optimization of AES efficiency.
3) **Evaluation**: *SHEEO+* is implemented and deployed on commercial edge platforms to validate its effectiveness in utilizing shadow cycles and enhancing AES energy efficiency across diverse operational scenarios.

II. Background

A. Heterogeneous Hardware and Software of AES

Autonomous embedded systems (AES) operate independently on various tough but important tasks, like autonomous driving [3] and space exploration [19], without requiring human interventions. Compared to datacenter-scale systems accommodating long-lasting workloads, AES mainly has two key differences. The first is heterogeneous and limited hardware resources. As shown in Fig. 2, typical edge platforms for AES such as Nvidia Jetson AGX Orin [18] are equipped with several heterogeneous computing units, including one 12-core Arm CPU, one 2048-core Ampere-architecture GPU, and two low-power Deep Learning Accelerator (DLA) [2]. GPU provides more TFLOPS performance but consumes more power than CPU and DLA. The computing units share a unified memory with limited capacity (32 GB) and bandwidth (204.8 GB/s) [20]. Each component includes a Voltage / Frequency (V/F) gate that can be adjusted via software, enabling flexible dynamic voltage and frequency scaling (DVFS) [21]. All components

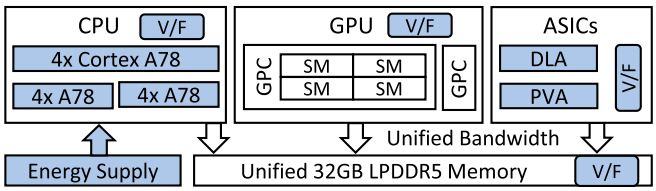

Fig. 2. Hardware architecture and power tuning knobs (in blue) of typical AES platforms.

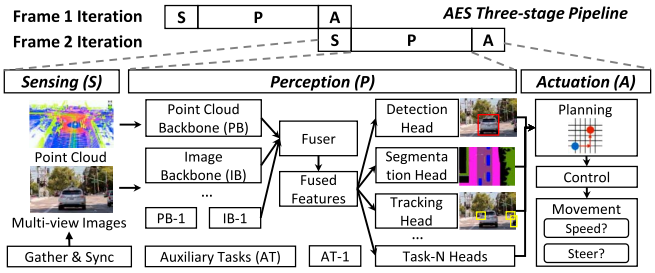

Fig. 3. Three-stage AES software pipeline with emerging bird's eye view (BEV) perception stage.

are integrated into a single System-on-Chips (SoC) with limited energy supply due to strict Size, Weight, and Power (SWaP) constraints [22], [23].

Apart from hardware limits, AES exhibits complex and iterative software execution. As shown in Fig. 3, a three-stage pipeline executes iteratively to process consecutive image and point cloud frames [24]. Firstly, the sensing stage gathers and synchronizes real-time data from multiple sensors [7]. Secondly, the perception stage understands the running conditions via parallel deep neural networks (DNN) [1]. Emerging perception fuses multi-modal sensors into BEV space with parallel backbone tasks, and performs various parallel task heads such as detection and segmentation [25]. Also, auxiliary tasks exist irregularly and interfere with regular BEV tasks [26]. Thirdly, the actuation stage makes control decisions based on the perception results [5]. Previous academic three-stage iteration frequency is around 10 FPS (100ms deadline) [1] but higher execution frequency enables faster reaction given new data from sensors (30-40 FPS) [2]. More than 90% execution time of each iteration performs perception, occupying heterogeneous accelerators concurrently [1].

B. Efficiency Optimization Requirements of AES

Despite limited energy supply and hardware resources, AES faces varied environments and complex hardware to manage, requiring better efficiency optimization. Firstly, the *temporal variance* due to external dynamics requires continuous optimization. The vehicle speed of AES changes with time and influences the latency constraints of every iteration [7]. Meanwhile, AES senses and interacts with the changing environments constantly, such as real-time traffic conditions, which force AES to adapt to the external variation at every iteration [8]. Due to changing outer situations, the task complexity to understand the surroundings varies in each iteration based on AES status, making it hard to decide the optimal efficiency settings statically [14]. The three changing factors require continuously choosing the optimal resource and power configurations for every AES iteration [21].

Secondly, the *spatial variance* due to different accelerator behaviors requires intelligent optimization. Heterogeneous processing units, including CPU, GPU, and DLA, have diverse performance and power consumption trade-offs, constituting a large optimization space [27], [28]. For a single iteration to optimize, AES must jointly consider the V/F settings and resource allocation of all components since optimization for a single component often conflicts when performed independently by separate controllers [29]. But simple methods relying on empirical models to optimize heterogeneous platforms fail to grasp the complex behaviors of AES [30]. Recent works have designed on-device machine learning (ML)-based efficiency optimizers for both cloud [31] and edge [32]. The models are required to rapidly learn and retrain in response to emergent data and shifting scenarios, ensuring sustained responsiveness [32]. In summary, an on-device continuous and intelligent energy efficiency optimizer is a must for AES.

III. Motivation

For resource-constrained AES, we first analyze the slowdown caused by the continuous and intelligent efficiency optimizer. We also present the ignored resource opportunities within the AES pipeline for harmless optimizer deployment.

A. AES Slowdown due to On-Device Efficiency Optimizer

Given the urgent need for continuous and intelligent efficiency optimizer in AES, the on-device facility deployment faces two-fold challenges. On the one hand, the intelligent optimizer is costly for resource-constrained AES, occupying significant hardware resources. As shown in Fig. 4, we deploy two popular intelligent efficiency optimizers based on deep reinforcement learning (DRL) on a typical edge platform Nvidia Jetson AGX Orin [18] to investigate their overheads. The first is Deep Q-Network (DQN) [33], a value-based method for discrete space, and the second is Deep Deterministic Policy Gradient (DDPG) [31], [34], an actor-critic method for continuous space. We vary the input sizes and running hardware to present overheads under various scenarios. We have two key findings. Firstly, the embedded CPU is improper for deploying intelligent optimizers since the inference latency is unacceptable for AES execution frequency. Secondly, the usage of GPU could accelerate DRL inference, but the optimizer occupying accelerators interferes with the AES pipeline. For a typical AES with 40ms constraints, even the DQN optimizer causes 5.7-13.3% performance slowdowns.

On the other hand, the costly optimizer continuously interferes with the AES pipeline. Current power/resource management facilities make decisions per iteration [28], [35] and even per layer [21], [36] in order to pursue the best performance. Further, on-device training for continuous adaptation in dynamic environment is essential for AES [32]. However,

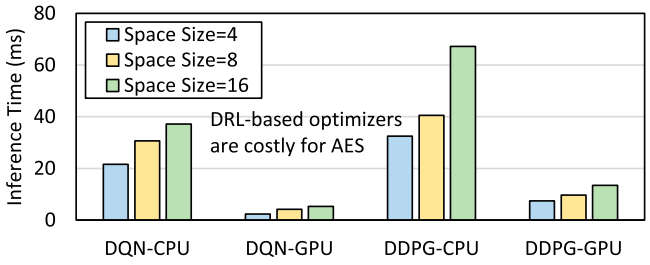

Fig. 4. DRL-based optimizers are costly for embedded accelerators, causing significant AES performance slowdown.

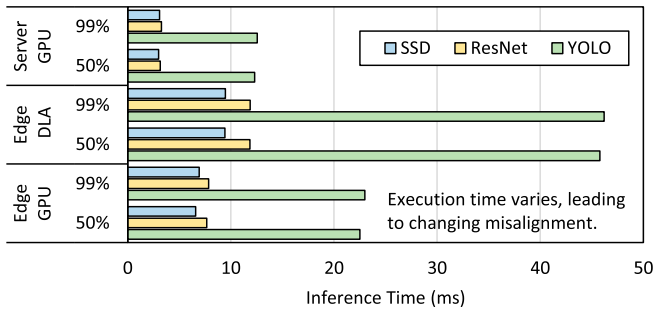

Fig. 5. Various inferences on heterogeneous accelerators present AES execution time variability and varying misalignment ratios. Misalignment varies with inference mapping [38].

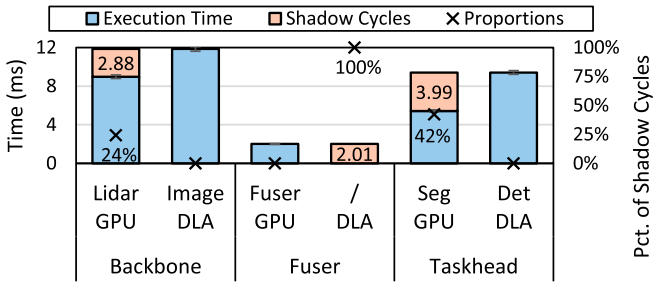

Fig. 6. Execution misalignment in BEV pattern leads to underutilized computing resources within AES perception stage.

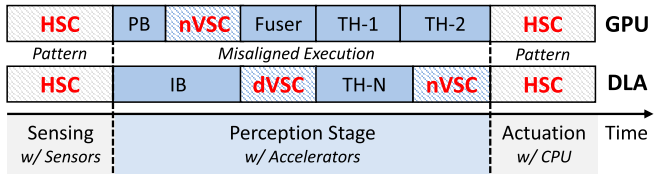

Fig. 7. Two types of underutilized shadow cycles within the execution pipeline of AES based on causes of occurrences.

high-frequency management worsens resource contention in accelerators, occupying AES resources in every iteration. Prior works consider the DVFS overheads to reduce management frequency [30], but most on-device efficiency optimizers fail to consider the realistic deployment overheads, causing cascading and significant AES performance slowdown [16], [37].

B. Tapping Into Shadow Cycles in AES

Considering the unacceptable slowdown of the on-device efficiency optimizer, AES presents unexploited resource opportunities for deploying the facility. The AES execution pattern that various DNNs execute on heterogeneous computing units [5] leads to misaligned DNN inferences and underutilized computing resources within the pipeline.

Misaligned DNN Inferences: The changing network inference behaviors and the parallel inference pattern result in misalignment. We characterize the performance of three representative DNN inferences in BEV pipeline [26] on Orin platform and server GPU RTX 4090: (1) Resnet50-v1 [39] with input size of 224×224 (`Resnet`), performing image processing backbone tasks, (2) SSD-Mobilenet-v1 [40] with input size of 300×300 (`SSD`), performing object detection task heads, (3) Yolov3-tiny [41] with input size of 416×416 (`YOLO`), performing auxiliary tasks. Each network executes on GPU and DLA at the highest frequency level 100 times to reduce biases. As shown in Fig. 5, we have two key observations. Firstly, inference time changes among networks with different sizes and input sizes, while the time distributions for specific networks are relatively stable. Secondly, inferences using different accelerators vary greatly and GPU generally finishes earlier than DLA. Further, if we map the parallel inferences to heterogeneous accelerators in different manners, the misalignment ratio could be 5-38% on edge GPU and DLA based on prior analysis [38].

In addition, we present the parallel inference execution of the BEV pipeline in Fig. 6. The backbones execute in parallel to process multi-modal input, where the image backbone `Resnet` [39] uses DLA and the lidar backbone `Pointpillar` [42] uses GPU. The `Fuser` [25] starts when all backbones are finished and only uses GPU. Different task heads use fused features to work in parallel, where the detection head `SSD` [40] uses DLA and the segmentation head `Second` [43] uses GPU. Parallel workloads allocated on different accelerators execute for varied durations and the accelerator that finishes ahead remains idle until all parallel tasks are finished. Generally, GPU executes faster than DLA so GPU presents more idle periods. For backbone and taskhead periods, around 24% and 42% GPU time is wasted for waiting for DLA. For fuser periods occupying GPU only, the whole DLA is idle. This misalignment phenomenon is widespread and more complex for realistic AES perception stages with more variable and concurrent workloads.

Underutilized Computing Resources: Based on misalignment characterizations, we envision the opportunities of unexploited heterogeneous computing resources within the AES pipeline, termed as *shadow cycles*. These underutilized resources refer to idle accelerator periods in AES execution and are ignored in current end-to-end workload mapping solutions [1]. In this paper, we mainly consider shadow GPU and DLA cycles on the Nvidia Jetson platform for two reasons: Firstly, the two accelerators are mostly used for DNN in AES [1]. Secondly, the embedded CPU cores are too weak to perform heavyweight inferences [7]. Based on distinct causes of occurrences, we categorize shadow cycles within AES into two types as shown in Fig. 7.

1) *Vertical Shadow Cycles (VSC)*: They result from misaligned execution of parallel AES perception tasks, denoting the idle accelerator periods within the waiting time until the slowest workload in the perception stage. VSC in fuser periods is deterministic in position and length, called *dVSC*. VSC in backbone and taskhead periods are nondeterministic due to latency-variable inferences, called *nVSC*.
2) *Horizontal Shadow Cycles (HSC)*: They result from the horizontal AES pipeline pattern, denoting the idle accelerator periods within the sensing and actuation stages. Tasks in the two stages mostly occupy sensors and CPU, leaving idle accelerators to be utilized.

In summary, there are three main factors that contribute to shadow cycle variance. First is AES workload patterns. AES pipeline organization and task complexity affect the proportions of shadow cycles, such as the unique dVSC in the BEV pattern. Second is AES accelerator specialty. Currently, we experiment with a GPU with 200 TOPS and two DLAs with 90 TOPS on the Nvidia Orin platform, and shadow cycles increase with more diverse accelerators performing tasks differently. Third is AES workload mapping. Distinct workload mapping choices causes varying misalignment, leading to variable shadow cycles [38], while it is hard to design an optimal balanced mapping without nVSC [26].

C. Analysis and Challenges of Shadow Cycles

Given the existence of precious shadow cycles within the AES pipeline, there are three main differences compared with regular accelerator resources. Firstly, shadow cycles are *preemptive*. Ensuring AES safety is more important than harvesting underutilized resources, so tasks using shadow cycles may be interrupted for additional auxiliary tasks [26]. Secondly, VSC is *volatile* in its duration. The mapping of parallel inferences on accelerators varies, leading to variable degrees of misalignment within in the perception stage. Despite the VSC ratio in Fig. 6, we have observed VSC that occupies 10%-50% perception time under various BEV settings. Thirdly, HSC is *fleeting* in its duration. The accelerator-idle sensing and actuation stages occupy a small proportion of the AES pipeline. For a typical AES iteration, HSC occupies around 5% execution time [1], namely a 1-2ms scale for a 40ms latency constraint. These shadow cycles provide a beneficial expansion of resource visibility for resource-constrained AES.

Moreover, the ignored shadow cycles are valuable for resource-constrained AES. Current rule-based AES middleware, such as energy efficiency optimizer, becomes more intelligent and costly for AES. The millisecond-scale management latency is unacceptable when using accelerators for AES pipelines or low-power CPUs on edge platforms. Shadow cycles, taking up around 20% accelerator time, are suitable for deploying on-device intelligent facilities without intrusion into AES pipeline. Utilizing these underutilized resources could fully exploit the heterogeneous accelerators in AES. Moreover, due to the high DVFS latency of more than one millisecond [30], it is inefficient to shut down the accelerators during the idle periods and we do not tune the frequency during shadow cycle periods. Beyond the individual edge platform like Orin, more heterogeneous and complex platforms, such as dual-SoC architecture in autonomous driving [4], present more significant shadow cycles within the various accelerators.

There will also be situations in which accelerators are always busy and shadow cycles are scarce, which may result from tighter constraints or well-designed workload balancing among accelerators. Under such scenarios, *nVSC* becomes rare, while *dVSC* and *HSC* still exist due to AES pipeline patterns. The existing fragments could be utilized to determine the optimal power configurations but the model optimization lacks sufficient shadow cycles to avoid AES intrusions, which presents new challenges in selecting a cloud-AES path to optimize the efficiency model on the cloud with on-device data [13]. However, current AES with diverse workloads fails to align various accelerators and mostly presents abundant shadow cycles that can be utilized.

Overall, utilizing the fragmented shadow cycles mainly faces two key challenges. 1) *How to efficiently capture and manage shadow cycles?* Given the volatile VSC and fleeting HSC, it is crucial to capture and manage them for utilization quickly. Time-consuming strategies fail to seize such opportunities. 2) *How to transparently exploit shadow cycles for on-device facilities?* Even if we could capture shadow cycles as wished, it is more complex to exploit them perfectly. Due to its preemptive nature, we should carefully select auxiliary rather than mission-critical tasks to fulfill shadow cycles. More importantly, exploiting shadow cycles should be transparent to AES pipeline execution, causing no performance degradation.

IV. Design of *SHEEO+*

A. Design Considerations

Based on the aforementioned characterizations, we outline two key design considerations to enhance the AES manageability from data observation and decision optimization.

1) **Observe AES Externally and Internally.** Optimizing energy consumption requires continuous monitoring of AES surrounding variances. The *external* environment variances, such as vehicle speed and traffic conditions, must be jointly considered to determine the most energy-efficient configurations. Effectively leveraging idle resources across heterogeneous accelerators requires a precise and timely understanding of *internal* hardware and software runtime status. Variations in software inferences on hardware accelerators need to be grasped for fragmented shadow cycles.
2) **Optimize AES Intelligently and Efficiently.** Optimizing energy consumption requires agile and *intelligent* reactions to both temporal and spatial variances. Deep reinforcement learning (DRL) offers an effective solution for managing power in response to runtime variances. It learns optimal behaviors by interacting with the environment in a tight feedback-loop, and can be easily adapted to manage heterogeneous hardware. Harvesting shadow

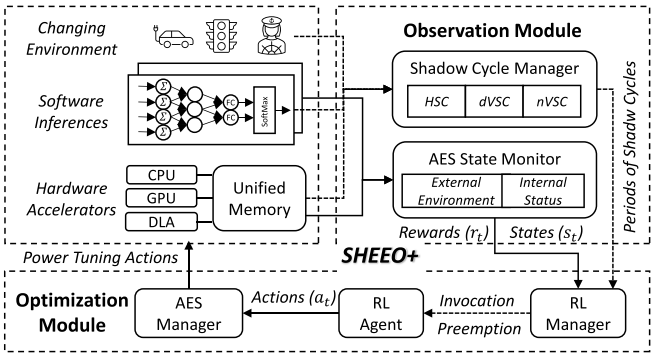

Fig. 8. Design overview and execution workflow of the two components of *SHEEO+*.

cycles for executing the learning-based optimizer is suitable and *efficient* for resource-constrained AES. VSC is well-suited for the model fine-tuning process, which is preemptive and can be executed across heterogeneous computing units with varying sizes. In contrast, HSC is appropriate for the inference process, which is relatively lightweight and carried out in each iteration.

B. Design Overview

In this work, we propose a continuous and intelligent **SH**adow cycle-enabled **E**nergy **E**fficiency **O**ptimizer (***SHEEO+***), which leverages the underutilized heterogeneous resources to enable learning-based power management at the granularity of each iteration. We aim to make the best use of the available shadow cycles for lower pipeline interference and higher energy efficiency. Fig. 8 depicts the workflow of two main components of *SHEEO+*, where the *observation module* is responsible for collecting internal and external information at runtime, and the *optimization module* is responsible for using shadow cycles for customized deep reinforcement learning (DRL) algorithms in the pursuit of optimal energy efficiency.

Moreover, compared with the original SHEEO [38], we further extend it in the following three aspects.

1) *We Devise a More Fine-Grained AES Observation Module.* We present respective strategies for three types of shadow cycles within the AES pipeline and newly designed structures to record AES status.
2) *We Extend SHEEO from Simple Q-Learning to the DRL Method.* It relies on actor-critic architecture to make decisions in large and continuous optimization space, improving the performance and stability of SHEEO.
3) *We Provide More Experiments to Validate Our SHEEO+ Design.* Extensive evaluation results show that *SHEEO+* is able to significantly harvest shadow cycles to avoid slowdown and improve energy efficiency.

C. Observation Module

The foundational component of *SHEEO+* is a continuous observation module. As Fig. 8 shows, it has two main components: 1) an agile shadow cycle manager and 2) a lightweight AES state monitor.

1) Shadow Cycle Manager: The *Shadow Cycle Manager (SCM)* is responsible for detecting the generation of shadow cycles, organizing their state, and managing them for executing optimizers. Firstly, the execution pipeline of AES is closely monitored. We modify the tasks within AES to report their start and end times to *SCM* to produce the shadow cycle states *[type, period, device]*. For HSC, the end of the slowest task head signals the start, and the *period* is the fixed time of actuation stage on *devices* GPU and DLA. For dVSC, the start of the fuser signals the start, and the *period* is the estimated time of the fuser on *device* DLA since the current fuser can only execute on GPU. For nVSC, when a task in the perception stage is completed without subsequent tasks, *SCM* signals the start of VSC periods, and the *period* is estimated by the remaining time of the running task. In addition, when a new shadow cycle is detected on the idle device, the periods of shadow cycles are accumulated without adding new indexes for continuity. To avoid the intrusion into the AES pipeline due to shadow cycle utilization, SCM is responsible for preempting shadow cycles for mission-critical tasks. When the perception tasks for the next frame are ready, SCM terminates the ongoing RL agent and removes the corresponding shadow cycles. Overall, the shadow cycle states are efficiently captured and managed, and then passed to the *RL manager* for deployment of learning-based optimizers.

2) AES State Monitor: The *AES State Monitor* is responsible for collecting and organizing the real-time external environments and internal status to optimize energy efficiency. The AES state contains two parts for every iteration. On the one hand, the *external environments* include vehicle speed and surrounding variance. The former is obtained from the vehicle domain controller, and the latter is obtained by comparing adjacent perception results. Higher speed and variance represent more complex scenarios that require tighter latency constraints for the next iteration, while AES can adaptively reduce energy consumption with looser constraints. On the other hand, the *internal status* contains offline software metadata and online hardware utilization. The former is calculated via network architectures, and the latter is obtained via *jetson_stat* tool. More heavy-weight inference and higher hardware usage lead to more aggressive power configurations in the next iteration. Overall, the collected AES state is organized into an ordered array indexed by AES iterations and is provided for the *RL agent* to make decisions.

D. Optimization Module

After observing AES variance, the next question is how to optimize AES power configurations to pursue the best energy efficiency. The optimization module relies on the information collected by the observation module to make informed decisions. As Fig. 8 shows, it has three main components: 1) a flexible RL manager, 2) an intelligent RL agent, and 3) a timely AES manager.

1) RL Manager: The *RL Manager* is responsible for allocating shadow cycles to the *RL agent* considering system status. As shown in Fig. 9, it organizes the unexploited AES states in the *Store Table* and embeds RL training and inference into VSC

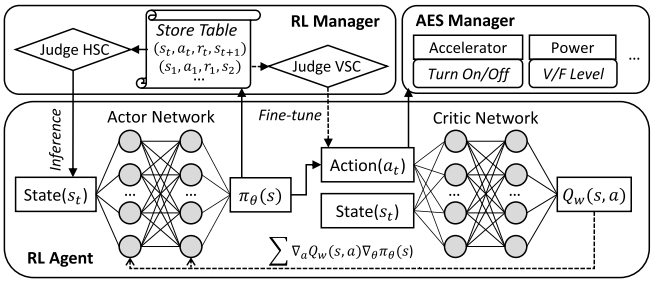

Fig. 9. Design details and DRL architectures of the optimization module under different scenarios.

and HSC, respectively. The most recent AES states are placed at the top of the table and are discarded after being used for training. For HSC, it provides AES states of the latest iteration to the *RL agent* for power adjustments in the next iteration. If there is insufficient HSC to complete one RL inference, the RL manager refrains from triggering the *RL agent* and instead informs the AES manager directly to maintain the current configurations. For VSC, it outputs AES states to the *RL agent* iteratively to fine-tune the optimization model. To avoid the intrusion of the RL agent into the AES pipeline, it first compares the shadow cycle periods with the average inference/fine-tuning execution time. Since the latency of each training update is relatively stable, if the remaining VSC is shorter than one training step, the manager avoids sending new states to the agent to prevent wasting shadow cycles. If the execution time of *RL agent* exceeds shadow cycle periods, the manager terminates the facility proactively to ensure the mission-critical workloads. The RL agent remains silent for current iteration and uses previous configurations. Overall, shadow cycles are exploited without intrusion into the AES pipeline and remain transparent to AES users.

2) RL Agent: The *RL Agent* is responsible for optimizing power management policies for long-term reward in dynamic AES environments based on DRL.

RL Selection. *SHEEO+* utilizes the deep deterministic policy gradient (DDPG) algorithm [34], which is a model-free, actor-critic RL framework. RL is particularly effective for learning power adjustment policies since it offers a strong feedback loop for exploring the action space and generating optimal policies without relying on inaccurate assumptions. It also enables direct learning from actual AES inner workloads and outer environments, allowing an understanding of how adjustments of low-level resources impact AES performance. We select DDPG mainly for two reasons: 1) model-free RL does not require the ergodic distribution of states or the precise modeling of environment variances (state transitions), which are often challenging to define accurately. When AES workloads are updated, the state-transition simulation used in model-based RL changes. 2) Actor-critic methods integrate both policy-based and value-based approaches, making them well-suited for continuous and stochastic environments. They offer faster convergence and lower variance compared to other methods like Q-learning and DQN.

RL Design. To decide power configurations for an AES iteration, we formulate it as a sequential decision-making problem that can be solved by RL framework as shown in Fig. 9. For every discrete iteration t, the agent observes AES state $s_t \in S$, and performs an action $a_t \in A$ based on its policy $\pi_\theta(s)$ (parameterized by θ), which maps state space S to action space A. At the following iteration $t+1$, the agent observes an reward $r_t \in R$ given by a reward function $r(s_t, a_t)$, denoting the changes from s_t to s_{t+1} via action a_t. The tuple (s_t, a_t, r_t, s_{t+1}) is called one transition. *SHEEO+* optimizes the policy π_θ to maximize the expected cumulative discounted reward $\sum_{k=0}^{T} \gamma^k r_{t+k}$. The discount factor $\gamma \in (0, 1]$ penalizes the predicted future rewards. A smaller discount factor means that consecutive states have a weak relationship, and it gives less weight to rewards in the near future, thus it is suitable for AES with a stochastic nature. *SHEEO+*'s policy learning is an actor-critic approach, where the critic estimates value function, i.e., the cumulative discounted reward under a given policy, and the actor updates the policy in the direction suggested by the critic. The value function of critic $Q_w(s_t, a_t)$ with parameter w is defined as Equation 1.

$$Q_w(s_t, a_t) = E[r(s_t, a_t) + \gamma Q_w(s_{t+1}, \pi(s_{t+1}))] \quad (1)$$

The corresponding loss function is defined as Equation 2.

$$L(w) = \sum_i (r_i + \gamma Q'_{w'}(s_{i+1}, \pi'_{\theta'}(s_{i+1})) - Q_w(s_i, a_i))^2 \quad (2)$$

The target networks $Q'_{w'}(s, a)$ and $\pi'_{\theta'}(s)$ are introduced in DDPG to mitigate the problem of instability and divergence when directly implementing the agent. The actor maintains $\pi_\theta(s)$ to specify the current policy by deterministically mapping states to a specific action. We select the Adam optimizer with different learning rates for the actor and critic networks. The critic network requires a larger learning rate than the actor network for stable policy updates, and a larger critic learning rate leads to faster convergence. We use the replay buffer to reduce the dependency between samples. The actor is updated as via the critic network in Equation 3.

$$\nabla_\theta J = \sum_i \nabla_a Q_w(s = s_i, a = \pi(s_i)) \nabla_\theta \pi_\theta(s = s_i) \quad (3)$$

State: Every AES iteration contains several neural network inferences on heterogeneous accelerators. The multi-dimensional state s_t is composed of three parts: 1) software features, especially network architecture. For AES workloads, we consider the number of convolutional layers s_t^{conv}, fully-connected layers s_t^{fc}, and recurrent layers s_t^{rc}, which are closely correlated with the efficiency and performance of AES execution. Moreover, the state includes software latency $s_t^{latency}$. 2) hardware features, i.e., resource usage. For the heterogeneous platform, the state includes the utilization of computing units $s_t^{compute}$ and memory s_t^{mem}. Also, the state includes the energy consumption s_t^{energy}, including *VDD_GPU_SOC* and *VDD_CPU_CV* within $s_t^{latency}$ on Orin platform. 3) external environments, including the vehicle speed s_t^{speed} and the surrounding variances s_t^{var}.

Action: The actions available to the *RL agent* are to adjust hardware resources and their corresponding power configurations. $a_t^{disable}$ denotes the core disabling mechanism to shut

down the idle cores. a_t^{freq} denotes the DVFS mechanism on different components, including CPU, GPU, two DLAs, and memory. For instance, if the latency constraints are met in the previous iteration and the environment becomes less complex, it may be possible to deactivate specific computing units and lower the frequency of active units to conserve energy. The set of actions is currently defined at the hardware layer and can be expanded by incorporating additional software tuning knobs such as workload mapping [28] and approximation [21].

Reward: The goal of the *RL agent* is, given a iteration t, to determine an optimal policy π_t that results in as low energy consumption as possible ($min_{\pi_t} s_t^{energy}$) while keeping the AES execution performance close to quality-of-service (QoS) constraints (min_{π_t}QoS $- s_t^{latency}$). The reward defines the optimization goals in each iteration, so we design a two-fold reward r_t, which is the weighted sum of normalized performance and energy reward. The performance is normalized to QoS, and the energy is normalized to the power supply (ELimit). If the execution latency satisfies QoS requirements, $r_t = \alpha \cdot s_t^{latency}/\text{QoS} - \beta \cdot s_t^{energy}/\text{ELimit}$. Otherwise, $r_t = -\beta \cdot s_t^{energy}/\text{ELimit}$ to denote the importance of performance guarantee. The energy reward is multiplied by -1 to increase the reward for lower energy consumption. *SHEEO+* prioritizes energy consumption so it sets α as 0.1 and β as 0.2 for the following experiments. But we can use a higher α if AES workloads require higher QoS guarantees or a higher β if AES requires lower energy consumption.

Workflow: The workflow of the *RL Agent* is closely related to shadow cycles. The actor-critic networks are pre-trained offline and fine-tuned on AES with runtime state-action transitions. The actor and critic networks contain two fully connected hidden layers with the ReLU activation function. The actor network has nine state inputs and six action outputs. To optimize energy efficiency, the *RL agent* utilizes VSC for fine-tuning and HSC for inference as shown in Algorithm 1. For the pre-trained networks, if the VSC periods are enough and the store table is not empty, it iteratively fine-tunes the networks with historical records. With consecutive state transition (line 2), it updates current critic and actor networks by Equations 2 and 3 (lines 3-4). Then, it updates the target networks considering consecutive penalties and gradient descent (lines 5-6). Meanwhile, if the HSC periods are enough, the *RL agent* selects actions maximizing energy efficiency with just the actor network (line 10). It also calculate the reward of the last iteration to complete the store table (lines 11-13).

3) AES Manager: The *AES Manager* is responsible for performing the actions generated by the RL Agent and executing them accordingly. To tune AES power configurations, it modifies the *set_freq* file at the beginning of every iteration. If the *RL agent* decides to decrease the frequency of a component running at the lowest frequency, the *AES manager* turns off specific units to reduce static power. For every component, there are predefined V/F levels (e.g., GPU has 12 fixed levels). The tuning range of each knob is defined by the power mode configurations and is limited by the overall power supply available to the AES (e.g., a maximum of 40W for Orin). Moreover, to prevent additional complexity and costs due to DVFS latency [30], *SHEEO+* performs the optimal action at the granularity of AES iterations to minimize DVFS influences rather than inference tasks or network layers.

Algorithm 1: Pseudocode of RL Agent Execution.

Input: Pre-trained $Q_w(s,a)$ and $\pi_\theta(a \mid s)$ with weights w and θ. Observation state of iteration t.
Output: Fine-tuned networks Q_w and π_θ. Action a_t for iteration t.

1 **foreach** *VSC periods and Store Table* $\neq \emptyset$ **do**
2 For a batch of s_t and s_{t+1} and chosen action a_t;
3 Update critic by minimizing the loss $L(w)$;
4 Update actor by sampled policy gradient $\nabla_\theta J$;
5 $w' \leftarrow \gamma w + (1-\gamma)w'$;
6 $\theta' \leftarrow \gamma\theta + (1-\gamma)\theta'$;
7 **end**
8 **foreach** *HSC periods* **do**
9 Receive observation state s_t of iteration t;
10 $a_t \leftarrow \pi_\theta(s_t)$;
11 Calculate reward r_{t-1};
12 Add (r_t, s_t) to (s_{t-1}, a_{t-1});
13 Store Table $\leftarrow (s_t, a_t)$;
14 **end**

V. Evaluation

Our evaluation wants to answer three questions:
1) How can *SHEEO+* utilize AES shadow cycles? (§V-B)
2) How can *SHEEO+* optimize energy efficiency? (§V-C)
3) How about the deployment costs of *SHEEO+*? (§V-D)

A. Evaluation Methodology

Implementations: To evaluate the performance of *SHEEO+* in realistic AES, we implement a prototype of the two main components in Python, and the actor-critic DRL agent is implemented with Pytorch. As for the hyperparameters in *SHEEO+*, we select the learning rates of 3×10^{-4} for the actor network and 3×10^{-3} for the critic network. The actual values are determined following prior works [16], [44] and validated on our platforms. Secondly, we set the discount factor γ as 0.4. We compare 0.1, 0.4, and 0.9 based on prior works [15], [16] and we choose the optimal 0.4. Thirdly, we train the RL agent offline for 50 iterations to initialize the pre-trained networks. The number of pre-training iterations is determined by the empirical learning curve in our optimization space. We use the *INA3221 power monitor* at I2C address 0x40 [45] at 5ms intervals to collect energy consumption and the *Python time module* to measure the execution latency. To reduce experiment result biases, the evaluated metrics are averaged from 10 repeated experiments.

System Setup: We perform all experiments on a typical embedded platform, Nvidia Jetson AGX Orin [18], and the specifications are shown in Table II. The power mode of Orin is set at maximum power mode (*MAXN*) consistently. We mainly consider the shadow cycles on GPU and NVDLA since they are mostly used for AES workloads. To mimic AES software

TABLE I
STATE AND ACTION SPACE OF *SHEEO+*

State Space S	
Software Features	Workload architecture. Execution latency.
Hardware Features	Component utilization. Energy consumption.
External Env.	Vehicle speed. Surrounding variation.
Action Space A	
Switch Knobs	Disable idle components.
Power Knobs	Adjust V/F levels of different components

TABLE II
EVALUATION PLATFORM SPECIFICATIONS

Platform	Nvidia Jetson AGX Orin Module
CPU	8-core ARM Cortex-A78AE v8.2
GPU	1792-core Ampere GPU with 200 TOPS
Memory	32GB 256-bit LPDDR5 with 204.8 GB/s
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

pipeline, we construct with a lidar backbone `Pointpillar` [42], an image backbone `Resnet` [39], a BEVFusion `Fuser` [25], an object detection task head `SSD` [40], and a scene segmentation task head `Second` [43]. All models are implemented in *Pytorch 1.12.1*, converted to *ONNX 1.15.0*, and optimized by *TensorRT 8.5.2* in FP16.

Baselines: To comprehensively compare the performance of *SHEEO+*, we compare it with three types of state-of-the-art (SOTA) baselines: 1) Workload-Aware Control (WAC) methods [21]. 2) Learning-Based Control (LBC) methods [13]. 3) Q-learning RL (SHEEO) methods [38]. WAC employs a static model for workload profiling and formulates power management decisions based on system limitations, such as latency analysis. LBC utilizes a machine learning model and historical data to derive optimal power management decisions. We present a comparative analysis of *SHEEO+* against SOTA implementations from all of these categories.

B. Shadow Cycles Utilization

To answer the first question, we evaluate *SHEEO+*'s usage of underutilized shadow cycles under various scenarios.

Pipeline Utilization: Fig. 10 illustrates a representative iteration within the three-stage AES pipeline, encompassing sensing, perception, and actuation phases. The efficacy of *SHEEO+* is demonstrated through a comparative analysis of heterogeneous hardware utilization, specifically focusing on GPU, which exhibits a later completion time than DLA. With *SHEEO+* implemented, as indicated by the blue line, the vertical and horizontal shadow cycles within the AES pipeline are effectively utilized to execute power management facilities. On average, compared with execution without shadow cycles utilization in the gray line, VSC periods achieve over 65.4% resource utilization for DRL networks fine-tuning, while HSC periods attain approximately 41.2% resource utilization, attributed to the relatively lightweight DRL inference. Compared with SHEEO with Q-learning in the green line, *SHEEO+* could harvest more shadow cycles with advanced DRL methods.

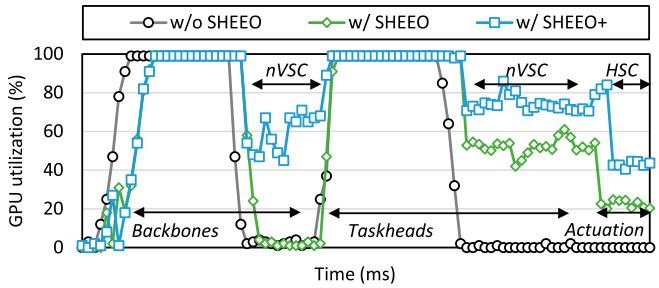

Fig. 10. *SHEEO+* utilizes two types of shadow cycles within the AES pipeline. It utilizes shadow cycles better than SHEEO with the more powerful DRL agent.

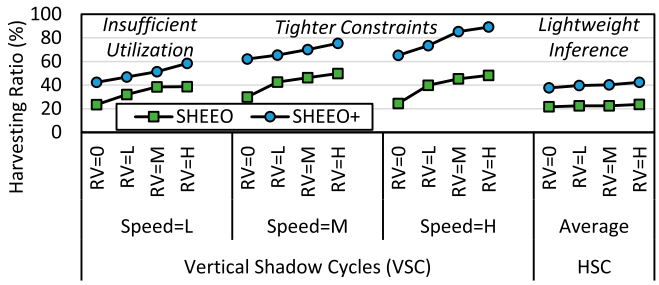

Fig. 11. The ratio of harvesting shadow cycles via *SHEEO+* under various execution conditions. Higher ratios mean effectively utilizing more idle resources for facilities.

Overall, *SHEEO+* enables resource-constrained AES to fully utilize two types of idle accelerator periods to avoid regular resources for energy efficiency optimizers.

Harvesting Ratios: Fig. 11 depicts the shadow cycle harvesting ratio of *SHEEO+* across varied execution conditions. The *harvesting ratio* is quantified as the increase in resource utilization within shadow cycle durations compared to that without *SHEEO+*. We vary the vehicle speed and runtime variance (RV) to mimic different latency constraints and AES complexity. On average, *SHEEO+* achieves VSC harvesting ratios of 56.5%, 61.8%, 68.8%, and 74.2% under different runtime variances, and an average HSC harvesting ratio of 39.9%. Theoretically, *SHEEO+* identifies the underutilized resources with the shadow cycle manager and utilizes two types of shadow cycles with the RL agent tasks, resulting in high harvesting ratios under various scenarios. Further, we have four key observations to deepen the understanding of *SHEEO+*'s inherent advantages. Firstly, the harvesting ratio increases under high vehicle speed conditions, where strict latency constraints result in shorter VSC durations for processing equivalent perception tasks. Under such scenarios with tighter constraints and scarce nVSC, the VSC harvesting ratio is higher due to sufficient utilization of dVSC. Secondly, the harvesting ratio increases with greater runtime variances, which demands more frequent DRL model fine-tuning. Thirdly, scenarios with low vehicle speed and runtime variance exhibit an abundance of shadow cycles, offering potentials for deployment of more compute-intensive management facilities. Lastly, a stable decision-making inference is too lightweight to occupy the HSC, which could be utilized more. Overall, compared with light-weight SHEEO, *SHEEO+*

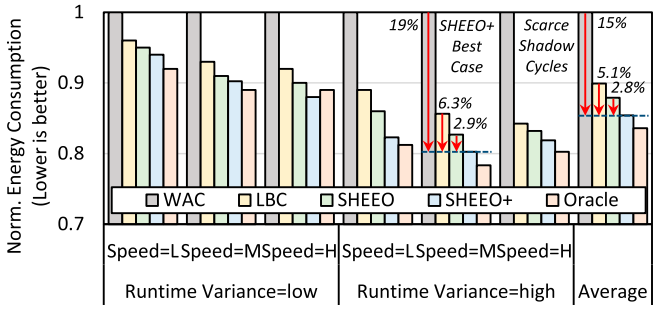

Fig. 12. *SHEEO+* achieves better energy consumption than SOTA baselines under various scenarios. Values are normalized to the WAC baseline. *SHEEO+* outperforms the most under high variance and medium speed scenario.

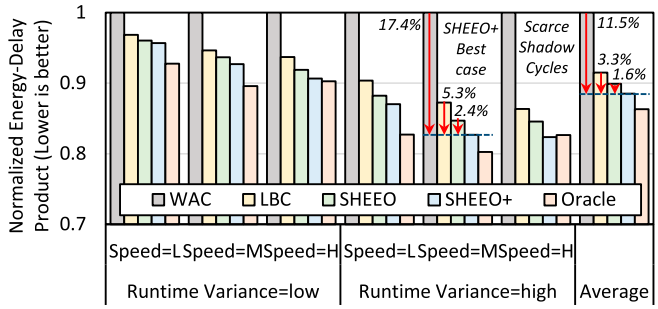

Fig. 13. *SHEEO+* achieves better energy efficiency with little cost of execution performance. *SHEEO+* outperforms the most under high variance and medium speed scenario.

with DRL methods achieves higher utilization of two types of shadow cycles. However, shadow resource utilization only shows the effectiveness of *SHEEO+* in harvesting underutilized resources, we further evaluate its efficiency improvements via AES overall power consumption.

C. Energy Efficiency Improvements

To answer the second question, we evaluate AES energy efficiency with *SHEEO+* and SOTA baselines.

Energy Consumption Reduction: Fig. 12 illustrates a comparative analysis of energy consumption across various energy optimization schemes. The *Oracle* represents an idealized offline power management decision. For a more direct comparison, all results are normalized against the WAC baseline. On average, *SHEEO+* shows 15% and 5.1% improvements over heuristic-based and learning-based methods, respectively. For low runtime variance scenarios, *SHEEO+* exhibits 9.3% and 3.2% energy efficiency improvements over WAC and LBC, since the minimal environmental changes reduce the necessity of adaptive management. For high runtime variance scenarios, *SHEEO+* surpasses the aforementioned baselines by 18.6% and 5.6%, respectively, due to the baselines' inability to adapt to dynamic conditions. Theoretically, *SHEEO+* reduces AES energy consumption by selecting power configurations that offer both performance guarantees and runtime adaptation. It optimizes heterogeneous accelerator efficiency in the variable execution environments. The efficiency improvements result from the joint work of learning changing environments in VSC and deciding the optimal configurations in HSC. In addition, we have two observations to deepen the understanding of *SHEEO+*'s inherent advantages. Firstly, workload-aware power management without complex learning is sufficient for achieving near-optimal energy efficiency under low runtime variances. Secondly, *SHEEO+* achieves the maximal 19.8% energy reduction under high runtime variance and medium vehicle speed, since the stringent latency constraints imposed by high speed limit the scope of energy optimization.

Energy-Performance Trade-offs: Fig. 13 illustrates the energy-delay product (EDP) of *SHEEO+*, providing a compressive assessment of AES efficiency by considering both energy consumption and execution latency. On average, *SHEEO+* exhibits a 11.5% improvement over WAC and a 3.3% improvement over LBC. The results show that *SHEEO+* prioritizes energy efficiency, potentially incurring a marginal increase in execution latency. With specific designs of performance guarantee in DRL reward, *SHEEO+* keeps the AES execution latency within the QoS requirements and minimizes the performance slack. Notably, under high runtime variances, *SHEEO+* demonstrates adaptability to varying latency constraints and environmental fluctuations.

Frequency Tuning Decisions: To further examine the energy efficiency gains achieved by *SHEEO+*, the frequency tuning decisions per iteration are depicted in Fig. 14. The default power mode of the Orin platform is modified to allow for 12 selectable frequency levels for the GPU and DLA. The GPU frequency can be adjusted within the range of 306 to 930 MHz, while the DLA frequency can be varied from 320 to 1408 MHz. We mimic three distinct phases to represent realistic AES execution conditions in previous experiments. During the *Low RV* phase, *SHEEO+* exhibits performance comparable to LBC and SHEEO, as all systems effectively learn the AES behaviors. During the *Speed Changing* phase, characterized by fluctuating latency constraints, *SHEEO+* makes decisions similar to WAC, but with more aggressive frequency adjustments to accelerate or decelerate execution. During the *High RV* phase, where the runtime conditions are fluctuating, *SHEEO+* implements more varying frequency tuning decisions to optimize the efficiency of the current state, resulting in decreased energy consumption. *SHEEO+* tends to reduce the frequency of energy-hungry GPU and increase the frequency of energy-efficient DLA. However, the reduction in frequency may lead to a slight increase in execution latency.

D. Deployment Cost Analysis of SHEEO+

To answer the third question, we present the execution details of *SHEEO+* from three key aspects.

Runtime Overhead: The overheads of *SHEEO+* come from two sources: the cost of AES observation and the cost of efficiency optimization. Fig. 15 presents the AES execution overheads with *SHEEO+* and other baselines. For observation module overheads, the slight slowdown results from the state observation and shadow cycle control overheads. Compared

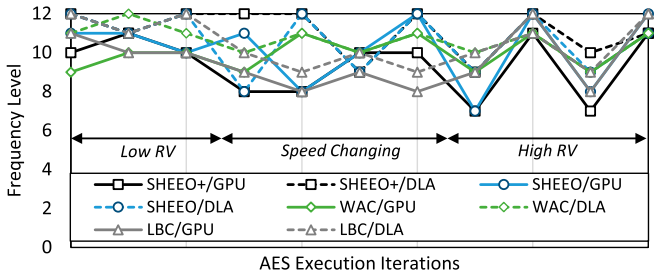

Fig. 14. *SHEEO+* tunes hardware frequency with awareness of variability under changing scenarios, leading to optimized energy consumption and AES performance.

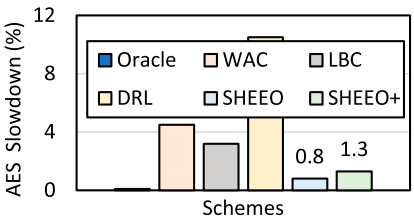

Fig. 15. AES slowdown caused by observation and optimization overheads.

with SHEEO using the same state monitor, the AES state monitor incurs 0.8% overhead for every iteration, including the vehicle domain controller for speed, identifier for variance, and jetson-stat tool for utilization. Moreover, the shadow cycle manager incurs an additional 0.5% overhead to organize and manage three types of shadow cycles, including instrumentation into AES pipeline tasks to obtain shadow cycles and following management. For optimization module overheads, *SHEEO+* theoretically reduces AES slowdown due to optimization since the training utilizes VSC and the inference utilizes HSC without interfering with the AES pipeline. *SHEEO+* and SHEEO incur negligible AES performance slowdown compared with similar DRL methods. For additional energy consumption to execute *SHEEO+*, we find that it consumes 7.6 mJ for every training step and 12.3 mJ for every inference process, corresponding to only 1.5% of the total energy consumption. As for the memory overhead, *SHEEO+* uses 45.2 MB memory to store the states and models, occupying only 0.1% of the 32 GB capacity of the evaluated AES platform.

Learning Curve: Fig. 16 shows that the reward of *SHEEO+* converges with the training process, denoting that the DRL agent is getting better at optimizing energy efficiency and guaranteeing performance. When training the model from scratch, the reward converges after around 100 iterations and experiences fluctuations with runtime condition changes. With a pre-trained actor-critic model, *SHEEO+* experiences little reward fluctuations and can further use real-time state-action transitions to fine-tune the offline DRL model.

QoS Violation: Fig. 17 demonstrates the AES QoS violation caused by the energy decisions via different optimizers. With aggressive power tuning, *SHEEO+* violates at most 3.2% latency constraints of AES iterations, resulting from the lower V/F level settings for better energy efficiency. Compared with

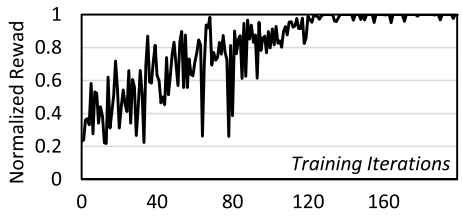

Fig. 16. Learning curve of *SHEEO+* showing the reward convergence.

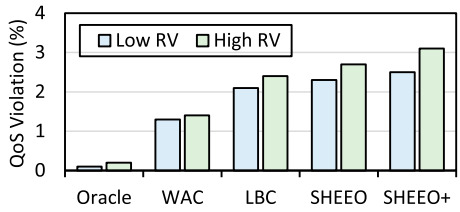

Fig. 17. AES QoS violation caused by different management schemes.

conservative LBC and SHEEO, *SHEEO+* experiences more frequent long-latency executions, especially under higher runtime variance scenarios. However, the DRL model of *SHEEO+* corrects itself with specific reward designs to mitigate further QoS violations and guarantee overall performance.

VI. Discussion and Future Work

In this paper, we exploit AES shadow cycles to deploy energy efficiency optimizers. In the future, broader existence and wider utilization of shadow cycles could be more promising.

A. Shadow Cycles in More Heterogeneous Systems

AES shadow cycles stem from inherent misalignment, which is an enduring consideration across past and future system architectures. Traditionally, the work-stealing techniques are designed to address misalignment in multi-core processors [46], while the task offloading mechanisms are used to accelerate heavy-weight tasks in CPU-GPU architectures [47]. For distributed shared-state datacenters, there are more underutilized resource fragments due to update misalignment [48]. Shadow cycles within AES go beyond traditional resource harvesting since repetitive AES makes shadow cycles more fragmented and variable. Future embedded platforms incorporate more domain-specific accelerators [2], [49] and variable AI workloads [11], leading to more severe misalignment. More specifically, BEV pipelines on dual-GPU platform induce 10-20% dVSC and more than 30% nVSC [26], and application performance on future GPU-NPU-ASIC platforms is varying more significantly [50], [51]. Therefore, the potential for underutilized and ignored resources warrants further investigation in more heterogeneous system architectures.

B. Shadow Cycles for More Complex Workloads

SHEEO+ is used to harvest AES shadow cycles for power management facilities, while the extended visibility on underutilized resources could be exploited for more intelligent tasks.

Firstly, diverse best-effort workloads, such as DNN training [52], data processing [48], and perspective-view tasks [26], could utilize shadow cycles to improve overall performance. We can adaptively divide the workloads to fit the volatile VSC and fleeting HSC. For example, the millisecond-scale training kernel for AES tasks [52] and periodic traffic light detection [26] could benefit from shadow cycle utilization. Secondly, more complex facilities could utilize shadow cycles for non-intrusive deployment. Besides energy efficiency optimizers just deciding power tuning knobs, more costly middleware, such as resource management [31], cloud-edge synchronization [15], and task mapping [28], could also exploit shadow cycles to hide the decision overheads. For example, the sub-accelerator selection [28] and coherence orchestration [37] with 3-6% overheads could utilize more horizontal shadow cycles within heterogeneous accelerators. More suitable auxiliary tasks and heavy-weight management inferences could further improve the current low HSC utilization. Therefore, *SHEEO+* serves as a demonstrative example of the broader application of ignored shadow cycles.

VII. Related Works

A. Researches on Autonomous Embedded Systems

Existing research from academic and industrial sectors has primarily concentrated on the architectural design and scheduling of AES [5], [14], [17], [23]. To build more heterogeneous AES systems, prior works provide in-depth analyses of the architectural implications and design considerations like autonomous driving [1], micromobility vehicles [7], and drones [8]. At the hardware layer, Srivatsan proposes a systematic approach to develop domain-specific embedded system-on-chip (SoC) architectures to improve hardware heterogeneity [2]. At the software layer, researchers propose more complex autonomous software paradigm [11], [25] and many data-driven mechanisms for resource management [20], [35], task mapping [26], [28], and cache coherence [37]. *However, these works concentrate on increasing specific capabilities and utilizing regular resources, failing to harvest the ignored heterogeneous resource fragments from the inherent hardware and software diversity within AES.*

B. Researches on Energy Efficiency Optimization

In response to the increasing complexity of AES hardware and software, researchers have adopted machine learning methodologies to enhance conventional energy efficiency optimizers. Within cloud environments, characterized by relatively stable workloads and runtime, predictive models are employed to optimize energy efficiency [13], [16], [27], [53]. Frameworks such as LEO utilize probabilistic graphical models to discern Pareto-optimal trade-offs [53]. These solutions, however, are tailored for systems with minimal stochastic runtime variances and necessitate dedicated hardware resources for learning and decision-making processes. Conversely, the domain of AES has seen limited prior works on intelligent power management [15], [21], [35]. NeuOS, for instance, is a latency-predictable, multi-dimensional optimization framework designed for multi-DNN workloads in AES [21]. These approaches tend to prioritize workload characteristics, while often neglecting stochastic runtime variance and lacking real-time adaptation capabilities [32]. Moreover, there is research on optimizing independent DVFS conflicts [29] and frequent DVFS overheads [30] for better energy efficiency. *However, prior works fail to construct continuous and intelligent power management solutions to agilely grasp runtime variance and precisely make optimal decisions, which is SHEEO+'s focus.*

VIII. Conclusion

This paper examines the potential of underutilized shadow cycles within autonomous embedded systems and addresses the necessity for energy efficiency optimizers capable of adapting to runtime variations. We introduce *SHEEO+*, a methodology designed to leverage shadow cycles for continuous energy efficiency optimization with deep reinforcement learning. Experimental results demonstrate that *SHEEO+* achieves up to 74.2% shadow cycle utilization and enhance energy efficiency by up to 18.6% when compared with state-of-the-art solutions, with minimal deployment overheads. It is anticipated that *SHEEO+* will inspire increased resource visibility for future intelligent management facilities of AES.

Acknowledgment

We sincerely thank all the anonymous reviewers for their valuable comments that helped us to improve the paper.

References

[1] S.-C. Lin et al., "The architectural implications of autonomous driving: Constraints and acceleration," in *Proc. 23rd Int. Conf. Archit. Support Program. Lang. Operating Syst. (ASPLOS)*, New York, NY, USA: ACM, 2018, pp. 751–766.

[2] S. Krishnan et al., "Automatic domain-specific SoC design for autonomous unmanned aerial vehicles," in *Proc. 55th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 300–317.

[3] Nvidia, "Solutions for self-driving cars & autonomous vehicles," Feb. 2025. [Online]. Available: https://www.nvidia.com/en-us/self-driving-cars/

[4] E. Talpes et al., "Compute solution for tesla's full self-driving computer," *IEEE Micro*, vol. 40, no. 2, pp. 25–35, Mar./Apr. 2020.

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

[6] C. Li, R. Zhou, and T. Li, "Enabling distributed generation powered sustainable high-performance data center," in *Proc. IEEE 19th Int. Symp. High Perform. Comput. Archit. (HPCA),* 2013, pp. 35–46.

[7] 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 *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 1067–1081.

[8] R. Hadidi, B. Asgari, S. Jijina, A. Amyette, N. Shoghi, and H. Kim, "Quantifying the design-space tradeoffs in autonomous drones," in *Proc. 26th ACM Int. Conf. Archit. Support Program. Lang. Operating Syst. (ASPLOS)*, New York, NY, USA: ACM, 2021, pp. 661–673.

[9] X. Wang, C. Li, L. Zhang, X. Hou, Q. Chen, and M. Guo, "Exploring efficient microservice level parallelism," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, Piscataway, NJ, USA: IEEE Press, 2022, pp. 223–233.

[10] L. Liu et al., "Computing systems for autonomous driving: State of the art and challenges," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6469–6486, Apr. 2021.

[11] X. Zhou et al., "Vision language models in autonomous driving: A survey and outlook," *IEEE Trans. Intell. Veh.*, early access, May 16, 2024, doi: 10.1109/TIV.2024.3402136.

[12] X. Wang et al., "EXIST: Enabling extremely efficient intra-service tracing observability in datacenters," in *Proc. 30th ACM Int. Conf. Archit. Support Program. Lang. Operating Syst., (ASPLOS)*, vol. 2, 2025, pp. 355–372, doi: 10.1145/3676641.3716283.

[13] N. Mishra, C. Imes, J. D. Lafferty, and H. Hoffmann, "CALOREE: Learning control for predictable latency and low energy," in *Proc. 23rd Int. Conf. Archit. Support Program. Lang. Operating Syst. (ASPLOS)*, New York, NY, USA: ACM, 2018, pp. 184–198.

[14] Y. Qi et al., "Tackling variabilities in autonomous driving," 2021, *arXiv:2104.10415*.

[15] Y. G. Kim and C.-J. Wu, "AutoScale: Energy efficiency optimization for stochastic edge inference using reinforcement learning," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, 2020, pp. 1082–1096.

[16] Y. Wang, W. Zhang, M. Hao, W. Kong, and Y. Wen, "Dynamic power management through multi-agent deep reinforcement learning for heterogeneous systems," *ACM Trans. Archit. Code Optim.*, vol. 22, no. 2, p. 23, Jun. 2025, Art. no. 46.

[17] P. H. Becker, J. M. Arnau, and A. González, "Demystifying power and performance bottlenecks in autonomous driving systems," in *Proc. IEEE Int. Symp. Workload Charact. (IISWC)*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 205–215.

[18] NVIDIA, "NVIDIA drive AGX orin," Feb. 2025. [Online]. Available: https://www.nvidia.cn/self-driving-cars/drive-platform/hardware/

[19] W. F. Truszkowski, M. G. Hinchey, J. L. Rash, and C. A. Rouff, "Autonomous and autonomic systems: A paradigm for future space exploration missions," *IEEE Trans. Syst., Man, Cybern., Part C (Appl. Reviews)*, vol. 36, no. 3, pp. 279–291, May 2006.

[20] S. Bateni, Z. Wang, Y. Zhu, Y. Hu, and C. Liu, "Co-optimizing performance and memory footprint via integrated CPU/GPU memory management, an implementation on autonomous driving platform," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Piscataway, NJ, USA: IEEE Press, 2020, pp. 310–323.

[21] S. Bateni and C. Liu, "NeuOS: A latency-predictable multi-dimensional optimization framework for DNN-driven autonomous systems," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf. (ATC)*. USENIX Assoc., 2020.

[22] NVIDIA, "NVIDIA drive end-to-end solutions for autonomous vehicles," Feb. 2025. [Online]. Available: https://developer.nvidia.com/drive

[23] S. Krishnan, Z. Wan, K. Bhardwaj, N. Jadhav, A. Faust, and V. J. Reddi, "Roofline model for UAVs: A bottleneck analysis tool for onboard compute characterization of autonomous unmanned aerial vehicles," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Piscataway, NJ, USA: IEEE Press, 2022, pp. 162–174.

[24] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," *Sensors*, vol. 19, no. 3, 2019, Art. no. 648.

[25] Z. Liu et al., "BEVFusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *Proc. IEEE Int. Conf. Rob. Autom. (ICRA)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 2774–2781.

[26] L. Sun et al., "Jigsaw: Taming bev-centric perception on dual-SoC for autonomous driving," in *Proc. 45th IEEE Real-Time Syst. Symp. (RTSS)*, Piscataway, NJ, USA: IEEE Press, 2024.

[27] C. Wan, M. Santriaji, E. Rogers, H. Hoffmann, M. Maire, and S. Lu, "ALERT: Accurate learning for energy and timeliness," in *Proc. USENIX Annu. Tech. Conf. (ATC),* 2020, pp. 353–369.

[28] S.-C. Kao and T. Krishna, "MAGMA: An optimization framework for mapping multiple DNNs on multiple accelerator cores," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*," Piscataway, NJ, USA: IEEE Press, 2022, pp. 814–830.

[29] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini, "CoScale: Coordinating CPU and memory system DVFS in server systems," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, IEEE Computer Soc., 2012, pp. 143–154.

[30] X. Hou, P. Tang, T. Xu, C. Xu, C. Li, and M. Guo, "CPM: A cross-layer power management facility to enable QoS-aware AIoT systems," in *Proc. 2024 IEEE/ACM 32nd Int. Symp. Qual. Serv. (IWQoS)*, 2024, pp. 1–10.

[31] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proc. ACM Special Interest Group Data Commun.*, 2019, pp. 270–288.

[32] Z. Li, A. Samanta, Y. Li, A. Soltoggio, H. Kim, and C. Liu, "R^3: On-device real-time deep reinforcement learning for autonomous robotics," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, IEEE Computer Soc., Dec. 2023, pp. 131–144.

[33] V. Mnih et al., "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[34] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[35] L. Sun et al., "A^2: Towards accelerator level parallelism for autonomous micromobility systems," *ACM Trans. Archit. Code Optim.*, vol. 21, no. 4, Nov. 2024.

[36] S. Bateni, H. Zhou, Y. Zhu, and C. Liu, "PredJoule: A timing-predictable energy optimization framework for deep neural networks," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 107–118.

[37] J. Zuckerman, D. Giri, J. Kwon, P. Mantovani, and L. P. Carloni, "Cohmeleon: Learning-based orchestration of accelerator coherence in heterogeneous SoCs," in *Proc. 54th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, 2021, pp. 350–365.

[38] X. Wang et al., "SHEEO: Continuous energy efficiency optimization in autonomous embedded systems," in *Proc. IEEE 42nd Int. Conf. Comput. Des. (ICCD)*, 2024, pp. 496–503.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[40] W. Liu et al., "SSD: Single shot multibox detector," in *Proc. Comput. Vis.–ECCV 2016: 14th Eur. Conf., Amsterdam, The Netherlands, October 11–14, 2016, Proc., Part I 14*, Berlin/Heidelberg, Germany: Springer, 2016, pp. 21–37.

[41] A. Farhadi and J. Redmon, "YOLOv3: An incremental improvement," in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1804, Berlin/Heidelberg, Germany: Springer, 2018, pp. 1–6.

[42] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recogni. (CVPR),* 2019, pp. 12697–12705.

[43] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, 2018, Art. no. 3337.

[44] H. Qiu, S. S. Banerjee, S. Jha, Z. T. Kalbarczyk, and R. K. Iyer, "FIRM: An intelligent fine-grained resource management framework for SLO-oriented microservices," in *Proc. 14th USENIX Symp. Operating Syst. Des. Implementation (OSDI),* 2020, pp. 805–825.

[45] NVIDIA, "NVIDIA Jetson Linux developer guide," Feb. 2025. [Online]. Available: https://docs.nvidia.com/jetson/archives/r34.1/DeveloperGuide/index.html#

[46] R. D. Blumofe and C. E. Leiserson, "Scheduling multithreaded computations by work stealing," *J. ACM (JACM)*, vol. 46, no. 5, pp. 720–748, 1999.

[47] S. Mittal and J. S. Vetter, "A survey of CPU-GPU heterogeneous computing techniques," *ACM Comput. Surv. (CSUR)*, vol. 47, no. 4, Jul. 2015.

[48] X. Wang et al., "Not all resources are visible: Exploiting fragmented shadow resources in shared-state scheduler architecture," in *Proc. ACM Symp. Cloud Comput.*, 2023, pp. 109–124.

[49] S. Moon, J. Cha, H. Park, and J.-Y. Kim, "Hybe: GPU-NPU hybrid system for efficient LLM inference with million-token context window," in *Proc. 52nd Annu. Int. Symp. Comput. Archit. (ISCA),* 2025, pp. 808–820, doi: 10.1145/3695053.3731051.

[50] J. Rogers, T. Soliman, and M. Jahre, "AIO: An abstraction for performance analysis across diverse accelerator architectures," in *Proc. ACM/IEEE 51st Annu. Int. Symp. Comput. Archit. (ISCA)*, Piscataway, NJ, USA: IEEE Press, 2024, pp. 487–500.

[51] D. Xu et al., "Fast on-device LLM inference with NPUs," in *Proc. 30th ACM Int. Conf. Archit. Support Program. Lang. Operating Syst., (ASPLOS)*, vol. 1, 2025, pp. 445–462, doi: 10.1145/3669940.3707239.

[52] W. Zhang et al., "PilotFish: Harvesting free cycles of cloud gaming with deep learning training," in *Proc. USENIX Annu. Tech. Conf. (ATC).* USENIX Assoc., 2022, pp. 217–232.

[53] N. Mishra, H. Zhang, J. D. Lafferty, and H. Hoffmann, "A probabilistic graphical model-based approach for minimizing energy under performance constraints," in *Proc. 20th Int. Conf. Archit. Support Program. Lang. Operating Syst. (ASPLOS).* New York, NY, USA: ACM, 2015, pp. 267–281.

Xinkai Wang (Graduate Student Member, IEEE) received the B.S. degree in computer science from Shanghai Jiao Tong University. He is currently working toward the Ph.D. degree with the School of Computer Science, Shanghai Jiao Tong University. His research interests include datacenter observability, energy efficiency optimization, and cloud computing.

Chao Li (Senior Member, IEEE) received the Ph.D. degree from University of Florida, in 2014. He is a Full Professor with the School of Computer Science, Shanghai Jiao Tong University. His research interests include computer architecture, datacenter scale computing, and emerging AI computing systems. He received several Best Paper Awards and Finalist recognitions at top-tier international conferences including ISCA and HPCA.

Yiyang Li is currently working toward the bachelor's degree (Hons.) with the School of Computer Science, Shanghai Jiao Tong University. His research interests include edge computing and energy efficiency optimization.

Lingyu Sun (Graduate Student Member, IEEE) received the B.S. degree in computer science from Shanghai Jiao Tong University. He is currently working toward the Ph.D. degree with the School of Computer Science, Shanghai Jiao Tong University. His research interests include real-time system, autonomous system, and AI acceleration architecture.

Cheng Xu (Graduate Student Member, IEEE) received the bachelor's and master's degrees in computer science from Shanghai Jiao Tong University. He is currently working toward the Ph.D. degree with the School of Computer Science, Shanghai Jiao Tong University. His research interests include architecture and system acceleration for multi-modal computing.

Xiaofeng Hou (Member, IEEE) received the Ph.D. degree from Shanghai Jiao Tong University, in 2020. She is currently an Assistant Professor with the School of Computer Science, Shanghai Jiao Tong University. She worked as a Postdoctoral Fellow with Hong Kong University of Science and Technology. Her research interests include high-performance and energy-efficient architectures and systems for intelligent computing centers. She received seven Best Paper Awards/Nominees.

Jing Wang (Member, IEEE) received the Ph.D. degree from SJTU, in 2024. She is a Postdoctoral Researcher with the School of Computer Science, Shanghai Jiao Tong University. Her research interests include disaggregated memory, graph processing, resource scheduling, heterogeneous architecture, etc.

Minyi Guo (Fellow, IEEE) received the Ph.D. degree in computer science from University of Tsukuba, Japan. He is currently a Zhiyuan Chair Professor with the School of Computer Science, Shanghai Jiao Tong University. His research interests include parallel/distributed computing, compiler optimizations, embedded systems, pervasive computing, big data, and cloud computing. He is currently on the editorial board for IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON CLOUD COMPUTING, and JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING. He is a fellow of CCF.

Yaqian Zhao (Senior Member, IEEE) is a Senior Scientist with IEIT Systems Company Ltd., Jinan, China. Since her Ph.D., she has been working on AI computational acceleration technologies. Her research interests include scenario-oriented AI acceleration and algorithm optimization techniques.