Multi-Objective Optimization Approach Using Deep Reinforcement Learning for Energy Efficiency in Heterogeneous Computing System

Journal:	*IEEE Internet of Things Journal*
Manuscript ID	IoT-28007-2023
Manuscript Type:	Regular Article
Date Submitted by the Author:	31-Jan-2023
Complete List of Authors:	Yu, Zheqi; University of Glasgow, School of Engineering Zhang, Chao; Shanghai Shiyue Computer Technology Machado, Pedro; Nottingham Trent University, Department of Computer Science Zahid, Adnan; Heriot Watt University, School of Engineering & Physical Sciences Fernandez-Hart, Tim; Sundance Multiprocessor Technology Imran, Muhammad Ali; University of Glasgow, School of Engineering Abbasi, Qammer; University of Glasgow, School of Engineering
Keywords:	Sensor Signal Processing < Sub-Area 1: Sensors and Devices for IoT, Energy Efficient Devices < Sub-Area 1: Sensors and Devices for IoT, Artificial Intelligence

Multi-Objective Optimization Approach Using Deep Reinforcement Learning for Energy Efficiency in Heterogeneous Computing System

Zheqi Yu, Chao Zhang, Pedro Machado, *Member, IEEE*, Adnan Zahid, *Member, IEEE*, Tim. Fernandez-Hart, Muhammad A. Imran, *Senior Member, IEEE*, and Qammer H. Abbasi, *Senior Member, IEEE*

***Abstract*—The growing demand for optimal and low-power energy consumption paradigms for Internet of Things (IoT) devices has garnered significant attention due to their cost-effectiveness, simplicity, and intelligibility. We propose an Artificial Intelligence (AI) hardware energy-efficient framework to achieve optimal energy savings in heterogeneous computing through appropriate power consumption management. A deep reinforcement learning framework is employed, utilizing the Actor-Critic architecture to provide a simple and precise method for power saving. The results of the study demonstrate the proposed approach's suitability for different hardware configurations, achieving notable energy consumption control while adhering to strict performance requirements. The evaluation of the proposed power-saving framework shows that it is more stable, and has achieved more than 23% efficiency improvement, outperforming other methods by more than 5%.**

***Index Terms*—Deep reinforcement learning, Actor-Critic architecture, Energy efficiency, Performance optimisation, Heterogeneous computing.**

I. Introduction

Traditional power management techniques rely on pre-trained AI algorithms and specialized energy-saving settings for different hardware. They typically follow predefined energy-saving rules, such as those used in Dynamic Voltage Frequency Scaling (DVFS) [1], where the algorithm is based on feedback control energy-saving actions to optimize voltage and frequency by modelling the power and performance of the system. This approach is limited in its application and effectiveness, as it strongly depends on the construction and preprocessing of the data model. An important area of research is focused on developing methods to maximise power usage while ensuring performance without the need for predefined energy-saving methods and data pre-processing.

In our previous research [2], we utilised a reinforcement learning algorithm [3] to observe the states of different hardware modules, and subsequently take actions to achieve energy savings. The algorithm's agent improves continuously through a specified learning method, ultimately learning the targeted optimal decisions for energy-saving actions. To enhance the performance of this method, we have now combined deep learning [4] with reinforcement learning, making use of the ability of neural networks to fit data [5] and the decision-making capability of reinforcement learning [6]. This results in a more robust deep reinforcement learning algorithm, which performs better in energy-saving tasks.

This paper presents a power-saving framework: a Power Measurement Utility Deep Reinforcement Learning (PMU-DRL) [7] model for controlling hardware power consumption. The proposed PMU-DRL power-saving framework is implemented on a LynSyn Lite Board (LSLB) [8], and is connected to the heterogeneous computing platform of NVIDIA Jetson TX2 (NJT2) [9] for implementation and evaluation. The results clearly demonstrate that excellent energy consumption control is achieved while meeting stringent performance requirements. The main contributions of this work are as follows:

- We propose a power-saving framework for Heterogeneous Computing platforms. Our proposed method does not require data pre-processing and uses self-adaption feedback through the Actor-Critic architecture, making it simpler and more precise for power saving strategies.
- We constructed a Deep Reinforcement Learning (DRL) framework, which is an updated version of a previously used reinforcement learning algorithm. Its self-balancing of the modules' working states provides the optimal energy-saving scheme for the entire system. It is different from traditional handcrafted states (such as Reinforcement learning using a Q-table to list all states), which achieved more significant energy-saving effects.

The structure of this article is as follows: In Section II, we explain how the power-saving framework operates on the heterogeneous computing platform and provide details on the signal processing and algorithm calculation workflow. Section III presents a quantitative evaluation of the DRL algorithm, demonstrating the power-saving effects achieved by the framework. In Section IV, we discuss the power-saving results and compare them with relevant research to demonstrate our contributions. Finally, in Section V, we provide a conclusion and suggest areas for future research.

The authors are with the James Watt School of Engineering, University of Glasgow, Glasgow G12 8QQ, U.K. (e-mail: z.yu.2@research.gla.ac.uk; muhammad.imran@glasgow.ac.uk; Qammer.Abbasi@glasgow.ac.uk) Chao Zhang with Shanghai Shiyue Computer Technology Co., Ltd, Shanghai, China. (e-mail: ChaoZhang.z@outlook.com;) Pedro Machado with the Computation Intelligence and Applications Group, Department of Computer Science, Nottingham Trent University, Nottingham, U.K. (e-mail: pedro.machado@ntu.ac.uk;) Adnan Zahid with the School of Engineering & Physical Sciences, Heriot Watt University, Edinburgh, EH14 4AS, U.K. (e-mail: a.zahid@hw.ac.uk;) Tim Fernandez-Hart with Sundance Multiprocessor Technology Ltd, London, UK. (e-mail:Tim.FH@sundance.com) Corresponding author: Qammer H. Abbasi.

Manuscript received 2023

II. Materials and Methods

In this paper, we propose the PMU-DRL framework for reducing power consumption on heterogeneous architectures that include both a Central Processor Units (CPU) [10] and a Graphics Processor Units (GPU) [11]. The PMU-DRL algorithm balances the trade-off between energy and computational performance by using power consumption information to make decisions and control the working state of the GPU. The algorithm assesses and predicts the optimal GPU operational states by analysing the overall topological structure and estimating the optimal working state using power consumption data. We evaluate the performance of the PMU-DRL by running YOLO v4 [12] on the NJT2, measuring the power consumption of the on-chip CPU and GPU. Figure 1 has shown the demonstrated YOLO V4 algorithm workflow on the hardware.

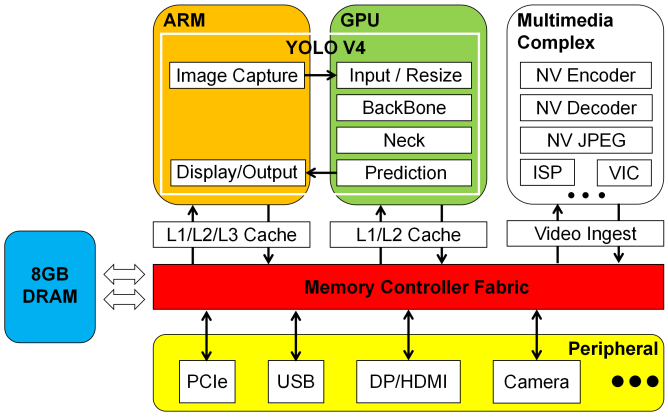

Fig. 1. YOLOv4 neural network computation stages on the Nvidia Jetson TX2 of CPU and GPU.

A. PMU-DRL Framework

In this section, we present the architecture of the proposed PMU-DRL framework. The setup used to evaluate the power consumption and computational performance of the NJTX2 is illustrated in Figure 2.

Figure 2 illustrates the system architecture used for evaluating the performance of the proposed PMU-DRL framework. The setup comprises the NJTX2 as the host hardware, and a Sundance Lynsyn Lite (SLL). for monitoring the hardware. The SLL is equipped with a Microcontroller Unit (MCU) [13] and power consumption sensors that are used to operate the PMU-DRL framework. The SLL [14] captures voltage and current information of the NJTX2 at a rate of 2 kilo samples per second (kS/s) and sends it to the PMU-DRL framework for analysis. The framework, based on a deep reinforcement learning algorithm, processes the power information and provides feedback for optimization actions. Finally, the PMU-DRL framework makes decisions to control the scheduling of the GPU based on the energy-saving analysis of the entire system.

B. Power Measurement Unit

The SLL[1] (see Figure 3) Power Measurement Utility (PMU), as described in [2], is specifically designed for analysing power consumption in embedded systems. Despite being optimized for use with Field Programmable Gate Arrays (FPGAs), it can also be adapted for use with the NVIDIA Jetson TX2 (NJTX2). The SLL is equipped with three power sensors for monitoring the power consumption of the unit under test. Additionally, it features a series design with two USARTs (universal synchronous/asynchronous receiver/transmitter) that are synchronized through the EFM32 PRS trigger system [15]. This setup allows for accurate and efficient monitoring of power consumption in the NJTX2.

In addition, when the board connects to the host workstation via Joint Test Action Group (JTAG). The power measurements can be read directly from the SLL Program Counter (PC) [14]. It can be configured to sample the target signals with frequencies up to 10kHz, and can be used as a generic remotely as an amperimeter or voltmeter. The PMU visualization tool can be

[1]Available online, https://store.sundance.com/product/lynsyn-lite/, last accessed 27/01/2023

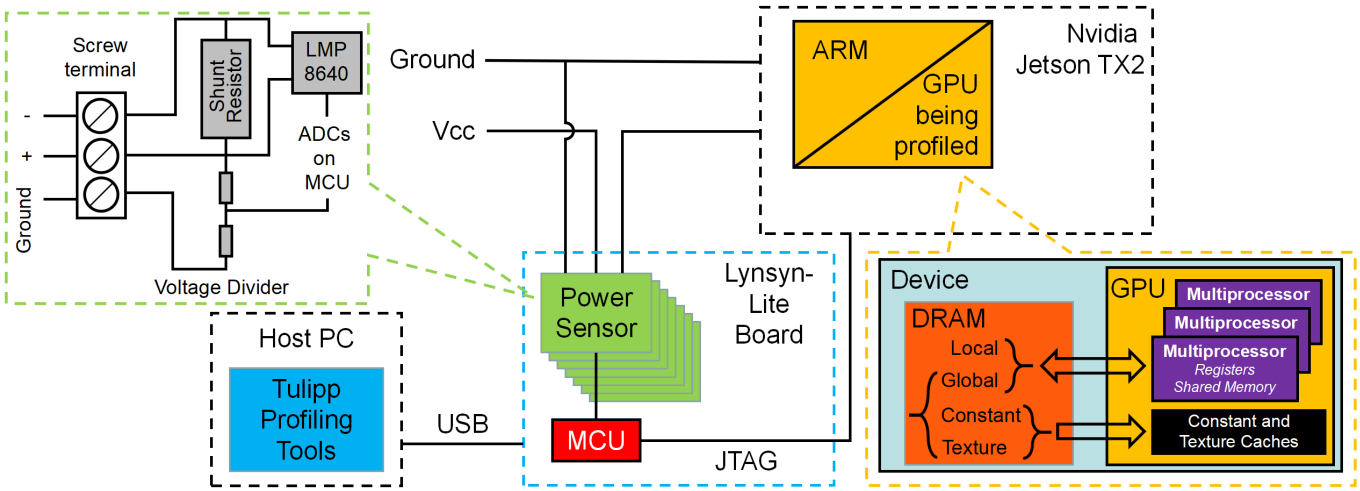

Fig. 2. Setup used to assess the power consumption and computational performance of PMU-DRL. The NJTX2 monitored by SLL.

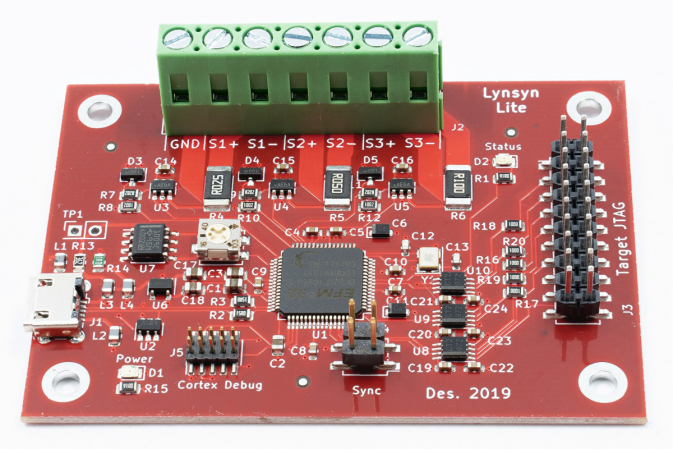

Fig. 3. SLL Hardware Platform.

configured to run on a host workstation to display and control the NJTX2 via JTAG.

The SLL delivers efficient power measurement tools ideal to perform power analysis on embedded systems and can be used as a general power analysis tool. Therefore, the SLL is fully compatible with major development platforms including Xilinx ZynqMP [16], NJTX2 [17], Raspberry Pi 4 [18], and Sundance VCS-1 [19].

1) Power Saving Concept for GPU: Context switching is the process of switching between different tasks or threads by the processor. This is done to ensure that the maximum number of tasks are executed in the shortest possible time. When a context switch occurs, the processor saves the current task's kernel context, including the processor registers and program counter, in a Process Control Block [20] (Switch Frame). This information is then used to load the context of the new task and jump to its location in memory, allowing it to run.

The cost of context switching in a CPU is high, as it requires saving and restoring the data from the registers to Random Access Memory (RAM) [21], which takes time. In contrast, the context switch of a GPU uses the CPU resources in a time-sharing manner, and is designed to process parallelisable data, which is less efficient for a CPU to compute sequentially. A GPU utilises an efficient work pool to ensure continuous task execution without idle hardware resources, and when a task encounters a pipeline stall, other tasks can run in parallel and utilize the idle pipelines. This is called a latency hiding work model, where the latency of one task is hidden by the progress of other tasks.

Context switching in a GPU is very fast, as it does not need to save the value of the register to memory as in a CPU. The GPU uses the SIMD (Single Instruction, Multiple Data) mode [22], which means that one instruction can process multiple different data, and has many cores built-in that can simultaneously process multiple threads. Additionally, the GPU can schedule threads at the hardware level, making context switching particularly fast with low overhead.

Overall, the GPU enables significantly reducing the context-switching in parallelisable tasks, which releases the CPU to process sequential (non-parallelisable) tasks. This improves the overall performance of the system by utilizing the resources of both the CPU and GPU effectively.

C. Deep Reinforcement Learning

DRL is a type of machine learning that involves an agent learning to interact with an environment in order to achieve a specific goal. There are two main types of algorithms in DRL, namely, 1) Behaviour policy: This is the strategy that the agent uses to interact with the environment in order to produce data; 2) Target policy: This is the strategy that the agent learns to accurately evaluate the Q value, which is the strategy that needs to be optimised.

When the two policies are the same strategy, the method is referred to as an on-policy method. When the policies are different, the method is referred to as an off-policy method. In this project, the Proximal Policy Optimization (PPO) algorithm [23] was used to make smart scheduling of hardware for power optimisation. PPO is an on-policy DRL, and the data used in the buffer is obtained by the target policy. This means that the Replay Buffer stores the data collected by the same policy for updating the network.

The PPO algorithm includes two network structures: the Actor Network [24] and the Critic Network [25]. The Actor Network takes the state as input and outputs either the action probability for discrete action spaces or the action probability distribution parameters for continuous action spaces [24]. The Critic Network takes the state as input and outputs the state values [25]. The output of the algorithm is considered better if the action output by the actor network can make the advantage larger, and if the value of the state output by the critic network is more accurate.

The PPO agent consists of two parts: interaction with the environment to collect samples, and decision-making of the actions quality. One advantage of the Actor-Critic (AC) network architecture used in PPO is that it can solve DRL problems related to continuous action spaces. Unlike discrete action spaces, it does not require dictionaries or storing value functions as tables or matrices. Instead, the AC network architecture directly uses the Policy-Based method, which uses various policy gradient methods to directly optimize the deep neural network parameterization's policy. This allows the deep neural network to directly output the action. Additionally, the noise variance of the action can be represented as a trainable vector, rather than being output by the network. This helps prevent overlarge noise variance from generating numerous boundary actions, which can negatively affect the performance of the agent and make the algorithm difficult to explore certain states. PPO is also well-suited for modelling continuous data of GPU energy consumption, which can be used as input for the deep reinforcement learning algorithm. The algorithm's output is the action with energy saving effect.

Figure 4 shows the deep reinforcement learning algorithm architecture. The pseudo-code of the algorithm is demonstrated in Algorithm 1 and depicted in Figure 4.

The DRL parameters are listed in Table I. The algorithm utilizes the power consumption information of the environment to divide the scheduling strategies for the CPU and GPU. It

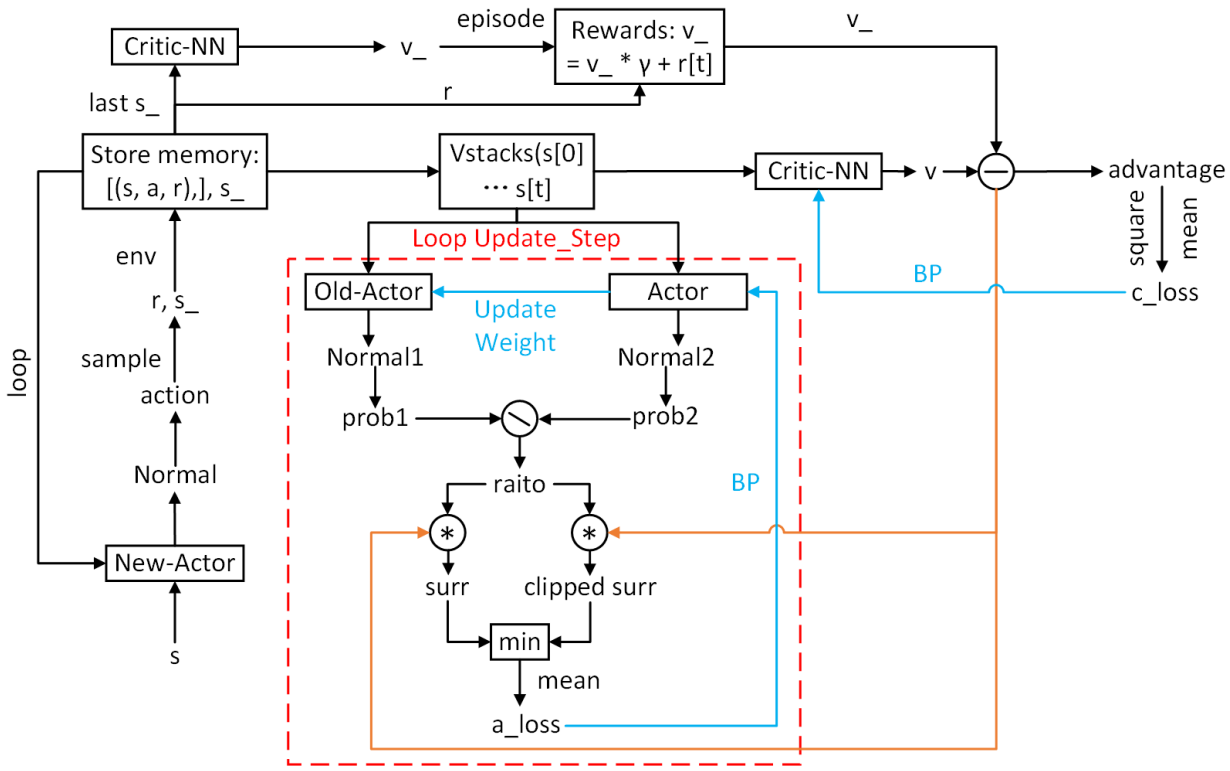

Fig. 4. DRL block diagram.

TABLE I
PPO ALGORITHM HYPER-PARAMETERS.

Hyper-parameter	Value
Sample_Step	3000
Reuse_Times (epochs)	200
Gamma	0.99
Lambda_Entropy	0.97
Clip Epsilon	0.2
Policy_Learning_Rate	3e-4
Value_Function_Learning_Rate	1e-3
Train_Policy_Iterations	80
Train_Value_Iterations	80
Target_kl	0.01
Hidden_Sizes	(64, 64)

TABLE II
DEEP REINFORCEMENT LEARNING OF STATES AND ACTION.

State \ Action	1	2	3	4	5	6	7	8	9
CPU	Up	Up	Up	Keep	Keep	Keep	Down	Down	Down
GPU	Up	Keep	Down	Up	Keep	Down	Up	Keep	Down

categorizes the load levels into high, low, and stable, based on the smoothness of the power consumption, taking into account the direction of power change trend. The different load levels correspond to different strategies, such as power up, power down, and maintaining the current state. In the decision-making stage, the algorithm takes into account the combined effect of the CPU and GPU working states, resulting in a total of 9 classes of action outputs. These outputs are illustrated in an action/state table, which can be referred to in Table II. This table helps to understand the relationship between the state of the system and the action taken by the algorithm, which can be used to optimize the performance of the algorithm.

D. Simulation Environment: Yolo-v4 and NJTX2

To evaluate the performance of our deep reinforcement learning-based PMU-DRL framework, we set up a simulated environment for an energy-saving algorithm using the NJTX2 hardware platform. This platform is known for its low power consumption of 15 watts [26] and is designed to run efficient state-of-the-art AI applications. The NJTX2 is a heterogeneous platform that includes the NVIDIA Jetpack Software Developer Kit (SDK) which provides libraries for GPU-accelerated computing, Linux drivers, and the Ubuntu operating system. The on-chip GPU can be programmed using NVIDIA's Compute Unified Device Architecture (CUDA) [27] for accelerating parallelisable algorithms and the CUDA Deep Neural Network (cuDNN) library, which is optimized for Deep Neural Network (DNN), is also installed to ensure maximum computational performance. Additionally, popular AI frameworks such as PyTorch and TensorFlow are integrated with the NVIDIA libraries to simplify the use of the GPU for AI developers. The NJTX2 is widely used in fields such as computer vision, data classification [28], deep learning [4] and other areas that require intensive computing. It has a wide range of applications, including Intelligent Video Analysis (IVA), drones, robots, gaming devices, virtual reality (vr), augmented reality (ar) and portable medical devices.

The hardware information for the NJTX2 system is shown in

Algorithm 1 Proposed PMU-DRL Algorithm

1: Input s (environmental information) into the *Actor-New* network, and get two values representing the distribution of action. One is *mu* (mean of Normal distribution), the other is *sigma* (variance of Normal distribution).
2: Following the Normal distribution to sample an action a.
3: Input $s\_$ into the *Actor-New* network to Loop steps 1 and 2 until stored a certain amount of [(s, a, r), ...]. The *Actor-New* network is not updated during this process.
4: Input the $s\_$ obtained in the last step of the loops into the *Critic-NN* to get the $v\_$ value of the state.
5: Calculate discount rewards: R[t] = r[t] + γ * r[t+1] + γ^2 * r[t+1] + ... + γ^{T-t+1} * r[T-1] + γ^{T-t} * $v\_$, and then achieve R = [R[0], R[1], ... ,R[t],...,R[T]]. (T is the last time step)
6: Input all stored s combinations into *Critic-NN*, get $V\_$ values for all states, and then calculate A_t = R – V_
7: c_loss = mean(square(A_t)), and then backpropagation updates the *Critic-NN*
8: Input all stored s combinations into the *Actor-Old* and *Actor-New* networks (there are two network structure are the same), and get the Normal distribution of *Normal 1* and *Normal 2*.
9: Input all stored action combinations as actions into the Normal distribution of *Normal 1* and *Normal 2* that achieves the corresponding probability of *prob 1* and *prob 2* for each action.
10: Divide *prob 2* by *prob 1* to get the important weight, which is *ratio*
11: Following equation a_loss = mean(min((ration * A_t, clip(ratio, 1-ξ, 1+ξ) * A_t))) to update *Critic-NN* by backpropagation
12: Following some Loops by steps 8 11 to update *Actor-Old* network from *Actor-New* network's weight.
13: Looping all above steps to achieve the finally output.

Table III. This information is used to set up the simulation environment for the study. The You Only Look Once (YOLO)v4 algorithm, as described in [12], is utilized to demonstrate the typical computational loads of large-scale deep neural networks applications.

YOLO is a state-of-the-art Convolutional Neural Network (CNN) that is able to accurately classify objects in real-time [12]. The algorithm processes the entire image using a single neural network, then divides the image into parts and forecasts bounding boxes and probabilities for each object [29]. The predicted probability weighs these bounding boxes. The technique "only looks once" at the image, since it only does one forward propagation loop through the neural network before making predictions [30].

One of the main advantages of the YOLOv4 algorithm is that it is written in C++, which results in lower latency compared to other CNNs written in Python. This means that it does not need to rely on third-party libraries, such as PyTorch or TensorFlow, to support AI computing. The C++ language project can directly call the CUDA instruction set, and GPU's Video RAM utilization efficiency is higher than that of the Python language. As a result, it is widely used in computing resource-constrained embedded environments.

In contrast, Python is a scripting language that requires an interpreter to interpret the code line by line and then turn it into machine instructions every time the project is called, allowing the processor to run the program. On the other hand, the C/C++ language is a static language that only needs to use a compiler, such as gcc, to disposable compile the program into a system execution of machine code file. This way, the program can be executed multiple times without affecting efficiency. This difference in the underlying logic of the hardware system shows that the execution efficiency of Python is not as good as that of the C/C++ language. The CUDA library can also be called by C/C++ language projects for accelerated computation on the project, not just for Python project development.

The proposed hardware/software setup is able to demonstrate typical loads introduced by state-of-the-art CNNs running in edge devices. It can truly reproduce the real operations, such as detect, observe, recognise, and identify objects, which are normally done by humans monitoring Closed-Circuit Television (CCTV) systems. The application is realized through the hardware platform, and then the real running power consumption information of the hardware can be obtained through the SLL of sensors. Finally, a simulation environment is built based on the collected information. The PMU-DRL framework works on this simulation environment to learn and understand the application's hardware requirements for power consumption and performance. Thereby, the PMU-DRL framework can optimise GPU scheduling to save power on the hardware platform.

TABLE III
NJTX2 SPECIFICATIONS

Hardware Information	NJTX2
CPU	Dual-Core for NVIDIA Denver 2 and Quad-Core for Application Response Measurement (ARM) Cortex A57
GPU	256-core NVIDIA Pascal with CUDA
Pipeline	18-Stages, Out-of-Order, 3-way Issue
Cache	48kB L1-I, 32kB L1-D, 2 MB L2
Memory	8 GB LPDDR4
Operating System	Ubuntu 20.04 LTS

III. Experimental Evaluation

The performance of the DRL algorithm of the PMU-DRL framework is illustrated in Figure 5. This figure displays the three key learning curves of the algorithm, providing insight into the training process of the agent. Figure 5.A shows the rewards per epoch of the PMU-DRL framework, generated during the training phase. The results indicate that the algorithm gradually makes more reasonable hardware control actions that lead to increasing rewards, improving the system's energy efficiency. The PPO algorithm is observed to converge quickly, reaching a stable solution in only 25 iterations.

The performance of the AC, which is responsible for the output value of state, is displayed in Figure 5.B. The loss

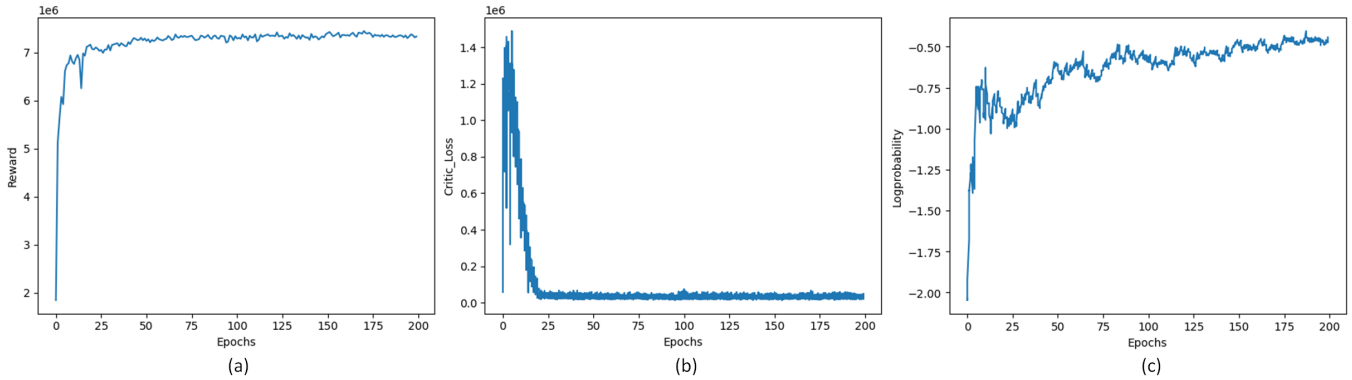

Fig. 5. Deep Reinforcement Learning Training Results.

curve of this network decreases dramatically as the training iteration deepens, indicating that value losses follow policy losses through the interaction of the AC architectures. The use of the PPO algorithm is observed to prevent over-learning.

Lastly, Figure 5.C illustrates the trend of Log_Probability. The algorithm calculates the entropy of random processes based on the average log probability, representing the measure of uncertainty of random variables. As the training proceeds, the Log_Probability approaches zero, indicating that the model is moving in the right direction, and the learned skills are being fully utilized to achieve high-performance output.

DRL requires a delay in checking the current decision's effect, which needs more time to determine if the correct outcome is achieved. This means that the negative effects of decisions are not solely caused by recent actions, but may also be influenced by errors that occurred in the previous period. The fundamental principle of DRL is to quickly escape the current state when making a mistake and carefully adjust to achieve convergence when making the right actions.

Figure 6 illustrates the learning output of the PMU-DRL framework for the effect of energy-saving power consumption, and it is compared to the normal power consumption of the system without the PMU-DRL framework. According to the simulation environment settings, the PMU-DRL algorithm can improve the energy efficiency of the system by understanding the appropriate time for GPU context switching scheduling rules for different processing conditions required by hardware computing.

The DRL agent is based on on-policy learning, which avoids the difference between the effect of the optimization result and the theoretical value affecting the output of the decision. The simulation environment test reflects the different simulations of the running time of the popular program to achieve the overall reaction energy-saving effect. The resulting data is presented in Table IV, which shows the power consumption comparison and energy efficiency improvement. The influence of different time periods on energy consumption is examined through the continuous multi-object detection of the YOLOv4 deep neural network algorithm in the test. The PMU-DRL framework increases the energy efficiency of the hardware by more than 23%. This means that the algorithm can save power without compromising performance requirements, which depends on the algorithm to output more efficient scheduling rules for GPU context switching.

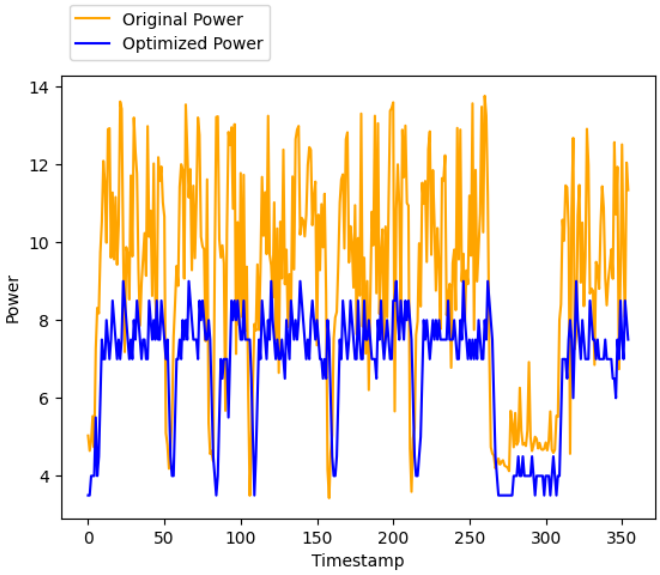

Fig. 6. Comparison of power consumption effects with and without PMU-DRL framework.

IV. Discussion

In this article, we propose an energy-efficient framework for edge AI hardware platforms that achieves a balance of performance and power consumption. The work simulates the running environment of the real program, and then demonstrates how to use the sensors of the LSLB to capture the power consumption information, and control the hardware through the DRL algorithm. Finally, to achieve energy saving effect.

Meanwhile, our proposed method is different from traditional techniques such as DVFS [34] and Adaptive Voltage Scaling (AVS) [35], which are required data pre-processing of frequency and voltage information. Moreover, compared with the DRL algorithm we used in previous work [2], the PMU-DRL framework of giving energy-saving actions through self-adaption feedback by AC architecture is more accurate

TABLE IV
ENERGY EFFICIENCY EVALUATION

Energy \ Times / Work	1s	10s	30s	60s	90s
Normal Work Mode	2.325 mWh	23.309 mWh	70.682 mWh	141.903 mWh	214.417 mWh
DRL for Hardware Control	1.771 mWh	17.803 mWh	53.916 mWh	108.153 mWh	163.049 mWh
Energy Efficiency Improvement	23.828%	23.622%	23.720%	23.784%	23.957%

TABLE V
COMPARISON ENERGY EFFICIENCY WITH OTHERS WORK

Project	Hardware	Method	Energy Efficiency Improved
Our Current work	NJTX2 and SLL	Deep Reinforcement Learning algorithm to hardware control with GPU context switch rule	More than 23% average energy efficiency improved with the normal model
Our Previous work [2]	VCS-1 board based on Xilinx ZYNQ UltraScale+ Multi-Processor system-on-Chip (MPSoC) Chip and Sundance Lynsyn board	Reinforcement Learning on the MCU to control and stop the clocks when no data are being exchanged via the I/Os	Up to 18% power reduction compared with the original model
Saroj *et al* [31]	NVIDIA Jetson Nano	An application-deadline-aware data offloading scheme using deep reinforcement learning and Dynamic Voltage and Frequency Scaling (DVFS)	The most energy reduction in the range of 9.68 - 10.35%
Jose Nunez-Yanez [32]	ZYNQ Z7020 and Zynq Ultrascale+ ZU9	The extension and application of an adaptive voltage scaling framework called Elongate to a high-performance and reconfigurable binarised neural network	Energy efficiency between 5%-23% via a +/-1% accuracy variation.
Sadrosadati *et al* [33]	NVIDIA Tesla P100	A idle-time-aware power management technique, which use finite state machine to effectively reduce the static energy consumption of GPU execution units by exploiting their idleness	Improves the static energy savings by an average of 16.9%

and stable. The PMU-DRL framework only needs the top-level system's power consumption information to plan the performance requirements of different hardware, and finally gives the optimal energy-saving scheme for the whole system.

Table V shows a comparison of the results obtained in this work with several other similar projects. Our designed method is based on the previously researched reinforcement learning architecture upgraded to a deep reinforcement learning algorithm. In contrast, not only the efficiency is improved by more than 5%, but also the effect output is more stable. On the other hand, traditional methods are based on voltage and frequency control technology, such as functions of DVFS and AVS. There are rely on data pre-processing methods to obtain massive data sources for analysis. Then, they perform the voltage and frequency scaling activities on the system to save energy. Furthermore, in comparison to other similar projects, our method stands out in its ability to achieve energy efficiency without compromising performance requirements and its adaptability to different hardware platforms.

Finally, our research demonstrates that our proposed method is adaptable to various platforms through its theoretical and algorithmic feasibility as seen in our test results. The deep reinforcement learning algorithm at the core of our adaptive architecture allows for easy matching of different hardware configurations without the need for modifying the decision-making environment. Furthermore, the algorithm's ability to adaptively match unknown hardware parameters via self-feedback adjustment of the AC architecture enables it to effectively handle changes in hardware.

V. CONCLUSION

This research proposed an energy-efficient framework for edge AI hardware platforms that balances performance and power consumption through the use of a deep reinforcement learning algorithm. The results demonstrate that this method is highly effective, achieving a 23% improvement in energy efficiency for sustainable system operation. Additionally, the proposed approach is based on self-adaption feedback by the AC architecture and does not require data pre-processing. This makes it a significant improvement over traditional methods such as DVFS and AVS, which rely on data pre-processing of frequency and voltage information to achieve optimal performance.

The results of this study provide strong theoretical support and algorithmic feasibility for extending this method to different platforms and hardware configurations in the future. One possible direction would be to investigate the scalability of the proposed method to larger and more complex edge AI systems. Additionally, further experimentation and testing could be conducted on different types of hardware platforms to validate the universality of the method. Another area for future research could be to explore the integration of other AI algorithms, such as neural networks or evolutionary algorithms, to enhance the performance and energy efficiency of the proposed method. Finally, it would be interesting to explore the potential of this method to be applied to other types of systems, such as data

centres or cloud computing environments, to further reduce energy consumption and improve performance.

ACKNOWLEDGMENTS

This research project has been made possible with the support of Sundance Multiprocessor Technology Ltd, through the European Union's Horizon 2020 research and innovation program projects: Optimization for Energy Consumption and Performance Trade-off, under grant agreement No. 779656 and VineScout, under grant agreement No. 737669. The hardware support power monitor system used in this research was provided by Sundance Multiprocessor Technology Ltd. We would like to express our gratitude for their invaluable assistance and support. (Figure 3 illustrates the hardware support provided by Sundance Multiprocessor Technology Ltd.).

REFERENCES

[1] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, S. U. Khan, and P. Li, "A double deep q-learning model for energy-efficient edge scheduling," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 739–749, 2018.

[2] Z. Yu, P. Machado, A. Zahid, A. M. Abdulghani, K. Dashtipour, H. Heidari, M. A. Imran, and Q. H. Abbasi, "Energy and performance trade-off optimization in heterogeneous computing via reinforcement learning," *Electronics*, vol. 9, no. 11, p. 1812, 2020.

[3] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1179–1191, 2020.

[4] Z. Yu, A. Taha, W. Taylor, A. Zahid, K. Rajab, H. Heidari, M. A. Imran, and Q. H. Abbasi, "A radar-based human activity recognition using a novel 3d point cloud classifier," *IEEE Sensors Journal*, 2022.

[5] Z. Yu, A. Zahid, A. Taha, W. Taylor, J. L. Kernec, H. Heidari, M. A. Imran, and Q. H. Abbasi, "An intelligent implementation of multi-sensing data fusion with neuromorphic computing for human activity recognition," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[6] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: a survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[7] C. Wulf, M. Willig, and D. Göhringer, "Low power scheduling of periodic hardware tasks in flash-based fpgas," in *2020 IEEE Nordic Circuits and Systems Conference (NorCAS)*. IEEE, 2020, pp. 1–7.

[8] C. Wulf, M. Willig, and D. Goehringer, "Rtos-supported low power scheduling of periodic hardware tasks in flash-based fpgas," *Microprocessors and Microsystems*, p. 104566, 2022.

[9] A. Wilson, A. Kumar, A. Jha, and L. R. Cenkeramaddi, "Embedded sensors, communication technologies, computing platforms and machine learning for uavs: A review," *IEEE Sensors Journal*, vol. 22, no. 3, pp. 1807–1826, 2021.

[10] A. C. Elster and T. A. Haugdahl, "Nvidia hopper gpu and grace cpu highlights," *Computing in Science & Engineering*, vol. 24, no. 2, pp. 95–100, 2022.

[11] M. Shafiabadi, H. Pedram, M. Reshadi, and A. Reza, "An accurate model to predict the performance of graphical processors using data mining and regression theory," *Computers & Electrical Engineering*, vol. 90, p. 106965, 2021.

[12] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[13] B. Sudharsan, J. G. Breslin, and M. I. Ali, "Ml-mcu: A framework to train ml classifiers on mcu-based iot edge devices," *IEEE Internet of Things Journal*, 2021.

[14] A. Djupdal, B. Gottschall, F. Ghasemi, and M. Jahre, "Lynsyn and lynsynlite: The sthem power measurement units," in *Towards Ubiquitous Low-power Image Processing Platforms*. Springer, 2021, pp. 93–114.

[15] G. Callebaut, G. Leenders, J. Van Mulders, G. Ottoy, L. De Strycker, and L. Van der Perre, "The art of designing remote iot devices—technologies and strategies for a long battery life," *Sensors*, vol. 21, no. 3, p. 913, 2021.

[16] S. Sabogal, A. George, and G. Crum, "Recon: A reconfigurable cnn acceleration framework for hybrid semantic segmentation on hybrid socs for space applications," in *2019 IEEE Space Computing Conference (SCC)*. IEEE, 2019, pp. 41–52.

[17] T. Amert, N. Otterness, M. Yang, J. H. Anderson, and F. D. Smith, "Gpu scheduling on the nvidia tx2: Hidden details revealed," in *2017 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2017, pp. 104–115.

[18] A. A. Süzen, B. Duman, and B. Şen, "Benchmark analysis of jetson tx2, jetson nano and raspberry pi using deep-cnn," in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*. IEEE, 2020, pp. 1–5.

[19] B. G. Oliveira and J. Lobo, "Interactive demonstration of an energy efficient yolov3 implementation in reconfigurable logic," in *2019 5th Experiment International Conference (exp. at'19)*. IEEE, 2019, pp. 235–236.

[20] Z. M. Khaing, Y. Naung, and P. H. Htut, "Development of control system for fruit classification based on convolutional neural network," in *2018 IEEE conference of russian young researchers in electrical and electronic engineering (EIConRus)*. IEEE, 2018, pp. 1805–1807.

[21] M. Lobur, Y. Salo, I. Farmaha, O. Senkovych, and K. Pytel, "Automated arm cpu-based cloud system for the industrial internet of things," in *2021 IEEE XVIIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*. IEEE, 2021, pp. 25–28.

[22] A. Tino, C. Collange, and A. Seznec, "Simt-x: Extending single-instruction multi-threading to out-of-order cores," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 17, no. 2, pp. 1–23, 2020.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[24] B. Huang and J. Wang, "Deep-reinforcement-learning-based capacity scheduling for pv-battery storage system," *IEEE Transactions on Smart Grid*, vol. 12, no. 3, pp. 2272–2283, 2020.

[25] J. Su, S. Adams, and P. Beling, "Value-decomposition multi-agent actor-critics," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 13, 2021, pp. 11 352–11 360.

[26] H. Cui and N. Dahnoun, "Real-time stereo vision implementation on nvidia jetson tx2," in *2019 8th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, 2019, pp. 1–5.

[27] R. Shams, R. Kennedy *et al.*, "Efficient histogram algorithms for nvidia cuda compatible devices," in *Proc. Int. Conf. on Signal Processing and Communications Systems (ICSPCS)*. Citeseer, 2007, pp. 418–422.

[28] Y. Zhao, X. Yang, Y. Yu, B. Qin, X. Du, and M. Guizani, "Blockchain-based auditable privacy-preserving data classification for internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2468–2484, 2021.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[30] K. R. B. Legaspi, N. W. S. Sison, and J. F. Villaverde, "Detection and classification of whiteflies and fruit flies using yolo," in *2021 13th International Conference on Computer and Automation Engineering (ICCAE)*. IEEE, 2021, pp. 1–4.

[31] S. K. Panda, M. Lin, and T. Zhou, "Energy efficient computation offloading with dvfs using deep reinforcement learning for time-critical iot applications in edge computing," *IEEE Internet of Things Journal*, 2022.

[32] J. Nunez-Yanez, "Energy proportional neural network inference with adaptive voltage and frequency scaling," *IEEE Transactions on Computers*, vol. 68, no. 5, pp. 676–687, 2018.

[33] M. Sadrosadati, S. B. Ehsani, H. Falahati, R. Ausavarungnirun, A. Tavakkol, M. Abaee, L. Orosa, Y. Wang, H. Sarbazi-Azad, and O. Mutlu, "Itap: Idle-time-aware power management for gpu execution units," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 16, no. 1, pp. 1–26, 2019.

[34] Q. Fettes, M. Clark, R. Bunescu, A. Karanth, and A. Louri, "Dynamic voltage and frequency scaling in nocs with supervised and reinforcement learning techniques," *IEEE Transactions on Computers*, vol. 68, no. 3, pp. 375–389, 2018.

[35] M.-A. LaCroix, H. Wong, Y. H. Liu, H. Ho, S. Lebedev, P. Krotnev, D. A. Nicolescu, D. Petrov, C. Carvalho, S. Alie *et al.*, "6.2 a 60gb/s pam-4 adc-dsp transceiver in 7nm cmos with snr-based adaptive power scaling achieving 6.9 pj/b at 32db loss," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2019, pp. 114–116.