BenQ: Benchmarking Automated Quantization on Deep Neural Network Accelerators

Zheng Wei, Xingjun Zhang, Jingbo Li, Zeyu Ji, Jia Wei
School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China
Email: frank.wei@stu.xjtu.edu.cn, xjzhang@xjtu.edu.cn

***Abstract*—Hardware-aware automated quantization promises to unlock an entirely new algorithm-hardware co-design paradigm for efficiently accelerating deep neural network (DNN) inference by incorporating the hardware cost into the reinforcement learning (RL) -based quantization strategy search process. Existing works usually design an automated quantization algorithm targeting one hardware accelerator with a device-specific performance model or pre-collected data. However, determining the hardware cost is non-trivial for algorithm experts due to their lack of cross-disciplinary knowledge in computer architecture, compiler, and physical chip design. Such a barrier limits reproducibility and fair comparison. Moreover, it is notoriously challenging to interpret the results due to the lack of quantitative metrics. To this end, we first propose *BenQ*, which includes various RL-based automated quantization algorithms with aligned settings and encapsulates two off-the-shelf performance predictors with standard OpenAI Gym API. Then, we leverage cosine similarity and manhattan distance to interpret the similarity between the searched policies. The experiments show that different automated quantization algorithms can achieve near equivalent optimal trade-offs because of the high similarity between the searched policies, which provides insights for revisiting the innovations in automated quantization algorithms.**

***Index Terms*—reinforcement learning, automated quantization, DNN accelerator, benchmark**

I. Introduction

Deep neural network (DNN) has become the most interesting domain with successful applications across a wide range of fields. However, the high computational complexity and huge hardware resource consumption have spurred a tremendously increased demand for improving hardware efficiency [1]. The solutions generally fall into two categories: 1) designing efficient DNN accelerators [2]; 2) designing efficient models or using model compression [3], [4].

There have been intensive studies on designing DNN accelerators. Early DNN accelerator designs [5], [6] are mainly based on the CMOS- or von-Neumann architecture, whose performances are bottlenecked by the *memory wall* problem. Recent progress in processing-in-memory (PIM) techniques, especially the resistive random-access memory (RRAM), has shown great potential for efficiently accelerating the matrix-vector multiplications (MVMs) compared to their digital counterparts. The commons of these specific accelerators are the highly parallel computing units and on-chip storage. However, they are not adept at full-precision computing and large model storage due to the limited resource budget, which compels researchers to use model quantization.

Model quantization refers to reducing the bitwidth of the data used to represent the parameters in DNN models. With the 32-bit floating-point (FP32) numbers quantized into low-bit integers, the computation in DNN is integer-only-arithmetic, friendly to hardware implementation, but may cause the accuracy loss. Thus, model quantization acts like a bridge between the algorithm and hardware, providing an opportunity to make a trade-off between accuracy and hardware efficiency. Manually selecting the bitwidth for each DNN layer is arduous, and it becomes more complicated when considering the hardware efficiency. Recently, hardware-aware automated quantization as a new algorithm-hardware co-design paradigm has sparked an explosion of researches, aiming to unlock further DNN applications and capabilities in resource-constrained environments. It can be formulated as a reinforcement learning (RL) problem by incorporating the hardware cost into the quantization strategy search process. Fig. 1 shows a general framework composed of RL agent, quantization method, and DNN accelerator performance predictor to accomplish this task. Previous works [7]–[9] can be viewed as the vertical integration and optimization of the above three components. A very missing part is how to compare these automated quantization algorithms to reveal the innovations in algorithm designs.

The **first challenge** is the absence of a benchmark framework with quantitative metrics to compare RL-based automated quantization. MQBench [10] proposes a model quantization benchmark to evaluate the handcrafted quantization algorithms but ignores the advancements in automated quantization. HW-NAS-Bench [11] focuses on benchmarking hardware-aware neural architecture search (HW-NAS). Strictly speaking, sandwiched between MQBench and HW-NAS-Bench, BenQ further explores and compares the RL-based automated quantization. Additionally, we empirically find that the overall performance is insufficient to interpret the results when different methods achieve similar overall performance. Therefore, we interpret the above observations from another angle (i.e., policy similarity).

The **second challenge** is how to generate the hardware cost. Collecting hardware cost data for each DNN model on real devices is time-consuming. Moreover, many accelerators still stay at academic research before real implementation. Building an accurate performance predictor is extremely difficult as it requires cross-disciplinary knowledge in computer architecture, compiler, and physical chip design. Thus, motivated by the unified hardware cost collection pipeline proposed in [11], BenQ considers two off-the-shelf performance predictors, both

978-3-9819263-6-1/DATE22/©2022 EDAA

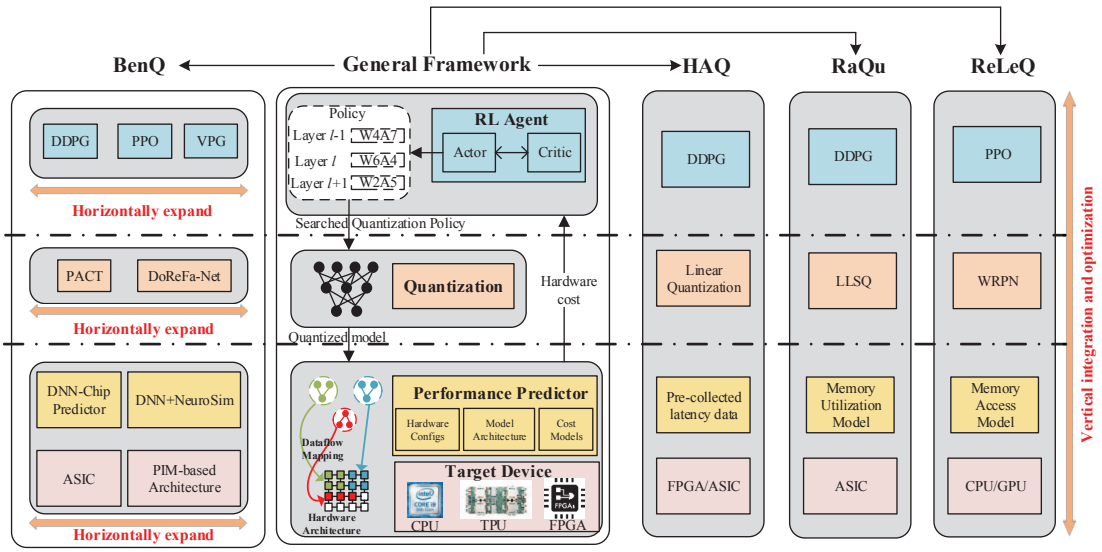


Fig. 1. The general framework to accomplish the hardware-aware automated quantization task. The RL agent searches the quantization policy, with which the DNN model is quantized into low-bit. The performance predictor takes the quantized DNN model and hardware specifications as inputs and outputs the estimated hardware cost. Previous works [7]–[9] can be viewed as the vertical integration and optimization of the above three components.

of which take the model architecture and hardware specifications as inputs and output the estimated hardware cost. For Eyeriss-like architecture, we modify the DNN-Chip Predictor [12] to estimate the energy consumption of the mixed-precision quantized DNN model. For PIM-based architecture, we refer to the weight mapping method in DNN+NeuroSim [13] and use the mapped tile number as the hardware cost.

To summarize, this paper makes the following contributions:

- We propose a benchmark framework, BenQ, to enhance reproducibility and fair comparisons of RL-bases automated quantization.
- It covers three representative RL-based automated quantization algorithms and two off-the-shelf performance predictors.
- We use cosine similarity and manhattan distance to measure the policy similarity and interpret the results, which is less explored in previous works.
- The results show that: 1) different automated quantization algorithms can achieve near equivalent trade-offs between accuracy and hardware cost because of the high similarity between the searched policies; 2) it provides insights for revisiting the innovations in RL-based automated quantization algorithms.

The remainder of this paper is organized as follows. Section II describes the related works, Section III describes the detailed implementation, Section IV presents the experimental results, and Section V concludes the paper.

II. Related Works

A. Automated Quantization

There have been intensive studies on automated quantization algorithms to balance both accuracy and hardware efficiency. When incorporating the hardware cost into the reward function, it evolves into hardware-aware automated quantization. HAQ [7] leverages the deep deterministic policy gradient (DDPG) [14] to generate the quantization policy that meets the resource constraints by querying the pre-collected hardware-cost look-up tables. ReLeQ [8] employs the proximal policy optimization (PPO) [15] with a reshaped reward to enable joint optimization of accuracy and reduced operations. RaQu [9] proposes an array-aware grouping method with DDPG to improve resource utilization for the RRAM-based accelerator. Besides the above RL-based methods, neural architecture search (NAS) -based methods [16] simultaneously adjust the model architecture, quantization policy, and hardware configurations to achieve the optimal performance. To our best knowledge, there lacks a benchmark framework to enhance fair comparisons of the RL-based automated quantization.

B. DNN Accelerator Performance Predictors

As most DNN accelerators still stay at academic research, there raises an intensive demand for performance predictors to evaluate the innovations in hardware designs before real implementation. Table I summarizes the state-of-the-art performance predictors. Designing accurate performance predictors for specific accelerators usually requires cross-disciplinary knowledge in computer architecture, circuits, and physical device design, which is not the algorithm experts' specialty. Thus, we make small modifications on two off-the-shelf performance predictors (i.e., DNN-Chip Predictor and DNN+NeuroSim) to support generating the hardware cost for the mixed-precision quantized DNN models.

C. Benchmark Frameworks

Benchmarking enables systematically comparing and revisiting the innovations in automated quantization and is, therefore, the fundament to drive the algorithm-hardware co-design to reach maturity. MQBench [10] is a model quantization benchmark to reproduce and compare various handcrafted quantization algorithms, but ignore the advantages of automated quantization. HW-NAS-Bench [11] is a unified benchmark to enhance the reproducibility and accessibility of hardware-aware neural architecture search (HW-NAS). Sandwiched between the MQBench and HW-NAS-Bench, BenQ focuses on benchmarking the RL-based automated quantization.

III. The Proposed BenQ

As Fig. 1 illustrates, BenQ horizontally expands the general framework to enhance fair comparisons. This section describes the implementation details of each part.

A. RL Algorithms

The state, action, reward, and agent are the four fundamentals of any RL algorithms. Hence, we first align the settings of these four parts.

1) State Space: As Table II shows, the state is characterized by a ten-dimensional feature vector, covering both the DNN architecture properties (i.e., the first 9 dimensions) and the action (i.e., the quantization bitwidth). If the l-th layer is a fully-connected (FC) layer, we set $k_{\mathrm{w}}^{l} = k_{\mathrm{h}}^{l} = 1$, $stride^{l} = 0$. Each dimension in the state is scaled within $[0, 1]$ before being passed to the agent. The symbols in Table II have the same meaning throughout this paper unless specifically stated.

TABLE I
State-of-the-art DNN Performance Predictors

Reference	Target Device	Programming Language	Open Source
Eyeriss Estimation Tool [6]	Eyeriss-like architecture	MATLAB	×
DNN-Chip Predictor [12]	Eyeriss-like architecture	Python	✓
Timeloop+Accelergy [17]	ASIC/FPGA	Python	✓
DNN+NeuroSim [13]	PIM-based architecture	C++/Python	✓
XPESim [18]	PIM-based architecture	Python	✓
MNSIM 2.0 [19]	PIM-based architecture	Python	✓

TABLE II
Embeddings of the State

I.D.	Description
l	Layer index
c_{in}^{l}	Input channels
c_{out}^{l}	Output channels
if_{w}^{l}	Input feature map width
if_{h}^{l}	Input feature map height
k_{w}^{l}	Kernel width
k_{h}^{l}	Kernel height
$stride^{l}$	Stride size
$\mathbb{I}_{\frac{\mathrm{weight}}{\mathrm{act}}}^{l}$	Indicator for weight or activation
a^{l-1}	The action

2) Action Space: The value of the bitwidth intrinsically is an integer, indicating that the action space should be discrete. However, some RL algorithms (e.g., DDPG) can only be used for continuous action space, leading to an unavoidable discretization step before quantization. If so, we use the same discretization method as HAQ. Otherwise, no action transformation is needed. We limit the value of the bitwidth in the range of $[2, 8]$ due to that: 1) the accuracy loss of the 8-bit quantized model is negligible compared with the full-precision model; 2) many specific accelerators are only adept for low-precision computing (i.e., below 16-bit); 3) many related works [7], [9] use such action space.

3) Reward Function: We use the following resource-constrained reward function:

$$\mathcal{R} = \begin{cases} acc_{\mathrm{quant}}, & if\ \frac{cost_{\mathrm{quant}}}{cost_{\mathrm{W8A8}}} \leq th \\ -1. & else \end{cases} \tag{1}$$

where the subscript W8A8 and quant represent the 8-bit model-wise and mixed-precision quantized model, respectively. The th is a predefined threshold to impose the resource constraint, the $acc\_$ is the top-1 classification accuracy, and the $cost\_$ is the estimated hardware cost to be illustrated in Section III-C. The proposed reward function penalizes the situation that the hardware cost exceeds the resource constraint and encourages the agent to find an optimal quantization policy that simultaneously achieves maximum acc_{quant} and satisfies the resource constraint.

4) Agents: Table III lists the supported RL agents implemented with the Spinning Up package[1]. All agents consist of an actor and a critic network, which are two solely multi-layer perceptron (MLP) models taking the state vector from the last step as inputs. The size of the actor network is {512, 512, act_dim}, while that for the critic network is {512, 512, 1}. The act_dim is the category of the action. The learning rates of the actor and the critic network are 3×10^{-4} and 10^{-3}, respectively. We use ADAM [20] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to optimize the agent. The rest hyperparameters of each agent are directly taken from the Spinning Up package.

B. Quantization Method

We mainly consider the linear quantization since non-linear quantization will complicate the hardware design [3], [10]. At this time, BenQ adopts two off-the-shelf quantization methods, PACT [21] and DoReFa-Net [22], both of which are per-tensor and symmetric quantization. As for quantization granularity, only the weights (W) and activations (A) for each convolution (CONV) and fully-connected (FC) layer are quantized into low-bit except for the first and last layer (8-bit). The model is trained with the simulated quantization where the backward pass is modeled as a straight-through-estimator (STE) [3].

C. DNN Accelerator Performance Predictors

Table IV lists the performance predictors considered in BenQ. We make minor modifications to enable them to generate the corresponding hardware cost for mixed-precision quantized

[1]The codes are available at *https://github.com/openai/spinningup*

TABLE III
THE SUPPORTED RL AGENTS IN BENQ

Agents	On/Off-Policy	Action Space	Actor-Critic Network
VPG[a]	On-Policy	Discrete/Continuous	MLP
DDPG	Off-Policy	Continuous	MLP
PPO	On-Policy	Discrete/Continuous	MLP

[a] The implementation of Vanilla Policy Gradient (VPG) is the same as Spinning Up.

TABLE IV
TWO OFF-THE-SHELF PERFORMANCE PREDICTORS CONSIDERED IN BENQ

Devices	Eyeriss-like accelerator	RRAM-based accelerator
Collected Metrics	Energy	Mapped tile number
Collecting Method	Estimated	Estimated
Performance Predictor	DNN-Chip Predictor	DNN+NeuroSim
Architecture	von-Neumann	PIM-based

DNN models. It is noteworthy that the estimated hardware cost only covers the CONV and FC layers as BenQ only quantized these two operations right now. The hardware cost computation is straightforward as they directly relate to the quantization bitwidth. As our goal is not designing the hardware accelerators, we refer readers to the original papers for more details on architecture-, circuit- and device-level design.

1) DNN-Chip Predictor for Eyeriss-like Architecture: DNN-Chip Predictor [12] supports three kinds of dataflows (i.e., row/weight/output stationary [6]). It uses a nested *for-loop* to describe spatial and temporal dataflow mapping. The technology-dependent unit costs in [12] are measured for 16-bit data. Therefore, we first divide the original unit cost by 16 and recalculate the new data volume $V^{'}_{ref_{i,j}}$ with the quantization bitwidth. Then we can get the projected hardware cost. For instance, the projected energy consumption of data movement ($E^{'}_{DM}$) is reformulated as:

$$E^{'}_{DM} = \sum_{i}\sum_{j} N_{ref_{i,j}} \times V^{'}_{ref_{i,j}} \times \frac{e_{DM_{i,j}}}{16} \tag{2}$$

where i, j, $N_{ref_{i,j}}$, and $e_{DM_{i,j}}$ have the same meaning as the original paper. We perform such projection on all energy models and sum them up to get the projected energy consumption of the mixed-precision quantized model.

2) DNN+NeuroSim for RRAM-based Accelerator: Compared to the complicated circuit- and device-level cost models in DNN+NeuroSim [13], the weight mapping method at the architecture level is more straightforward and deterministic given the corresponding dataflow mapping. DNN+NeuroSim provides two mapping methods: a traditional im2col-like and a novel mapping method. For simplicity, we use the former, and take the total number of the mapped tiles ($\#tiles$) as a proxy signal for hardware cost:

$$\#tile = \sum_{l=1}^{L} \lceil \frac{\lceil \frac{k_{\mathrm{w}}^{l} \cdot k_{\mathrm{h}}^{l} \cdot c_{\mathrm{in}}^{l}}{R_{\mathrm{row}}} \rceil \cdot \lceil \frac{c_{\mathrm{out}}^{l} \cdot a^{l} / P_{\mathrm{cell}}}{R_{\mathrm{col}}} \rceil}{N_{\mathrm{Array}} \cdot N_{\mathrm{PE}}} \rceil \tag{3}$$

where a^l is the bitwidth allocated for weights, P_{cell} is the precision of the RRAM cell, R_{row} and R_{col} are the row and column number of each RRAM crossbar array, N_{Array} is the number of RRAM crossbar array per processing engine (PE), N_{PE} is the number of PE per Tile. Throughout this paper, we set $P_{\mathrm{cell}} = 1$, $R_{\mathrm{row}} = R_{\mathrm{col}} = 128$, $N_{\mathrm{Array}} = 4$, $N_{\mathrm{PE}} = 4$.

IV. EXPERIMENTAL RESULTS

In this section, we focus on the classical image classification tasks and evaluate a modified VGG[2] on CIFAR-10 dataset, and ResNet-18 on ImageNet (ILSVRC'12) dataset. We first train the W8A8 quantized model, the parameters of which are initialized with the pre-trained full-precision model. Then, during exploration, for large datasets like ImageNet, we first search the bitwidth on the randomly selected dataset and then finetune the model on the whole dataset to recover the accuracy. During training and finetuning, we use SGD with a learning rate of 0.01, weight decay of 4×10^{-5}, and momentum of 0.9.

A. Overall Performance

Table V and VI show the comparisons of accuracy and hardware cost between different automated quantization with different performance predictors. It proves that:

- the automated quantization can **reduce the hardware cost with negligible accuracy loss**, which is consistent with previous works. For instance, compared with the W8A8 model-wise quantization, the automated quantization can reduce the mapped $\#tile$ by **1.26-1.95**$\times$ for ResNet-18 on ImageNet. For VGG on CIFAR-10, it reduces mapped $\#tile$ by **1.26-1.91**$\times$.
- **different automated quantization algorithms can achieve a similar maximum top-1 accuracy under the same hardware constraint** (i.e., th). For instance, when $th = 0.8$, the difference of the top-1 accuracy is less than 2%. This is not fully explored in previous works.

B. Quantization Policy Comparison

For simplicity, this section takes the VGG model as an example to illustrate the difference in the searched quantization policy. Fig. 2 and Fig. 3 plot the searched quantization policy corresponding to the maximum top-1 accuracy in Table V and VI, demonstrating that the searched policies of different automated quantization algorithms are different even though their maximum top-1 accuracies are close. It may be that the model redundancy provides a huge explorable quantization space for algorithms to make a trade-off between accuracy and hardware cost.

Fig. 2 shows that the agents perform more aggressive quantization (i.e., 4 to 5-bit for weights) on the last few layers (i.e., Conv2d-18 and Conv2d-22) to satisfy the resource constraints. The reason is that the parameters of the last few layers accounted for approximately 76% of the entire model and reducing the bitwidth of these huge layers can significantly decrease the $\#tile$. Such observation is not found in Fig. 3 as the hardware cost estimation is different and device-specific, leading to a different reward function. Therefore, it is necessary to use the same performance predictor with a unified hardware

[2]The model file is available at *https://github.com/aaron-xichen/pytorch-playground*

TABLE V
COMPARISON OF DIFFERENT RL-BASED METHODS USING THE MODIFIED DNN+NEUROSIM

Method	Bitwidth	th	VGG on CIFAR-10 Top-1	VGG on CIFAR-10 #Tile	ResNet-18 on ImageNet Top-1	ResNet-18 on ImageNet #Tile
DDPG+PACT	*flexible*	0.8	**92.56**	66	**69.23**	96
		0.7	91.96	58	68.35	83
		0.6	91.20	48	67.40	72
PPO+PACT	*flexible*	0.8	92.44	68	68.96	89
		0.7	91.34	53	67.80	78
		0.6	90.50	45	65.93	62
VPG+PACT	*flexible*	0.8	92.28	68	69.02	92
		0.7	92.07	58	67.90	83
		0.6	90.67	51	67.42	71
PACT [21]	W8A8	1	92.69	86	69.20	121
DoReFa-Net [22]	W8A8	1	92.59	86	68.13	121

TABLE VI
COMPARISON OF DIFFERENT RL-BASED METHODS USING THE MODIFIED DNN-CHIP PREDICTOR

Method	Bitwidth	th	VGG on CIFAR-10 Top-1	VGG on CIFAR-10 Energy
DDPG+PACT	*flexible*	0.8	**90.89**	2.78 mJ
PPO+PACT	*flexible*	0.8	90.50	2.77 mJ
VPG+PACT	*flexible*	0.8	89.30	2.76 mJ
PACT [21]	W8A8	1	92.69	3.50 mJ
DoReFa-Net [22]	W8A8	1	92.59	3.50 mJ

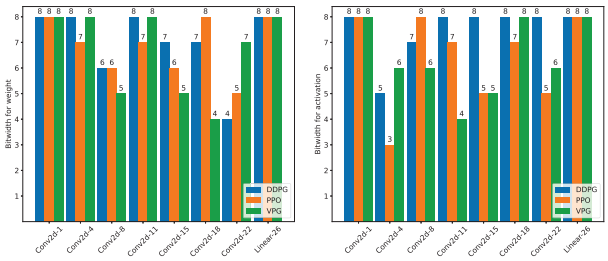


Fig. 2. The searched quantization policy corresponding to the maximum top-1 accuracy in Table V when $th = 0.8$.

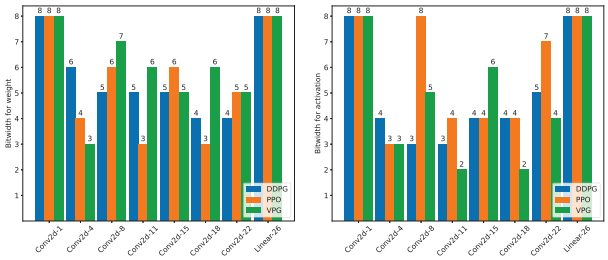


Fig. 3. The searched quantization policy corresponding to the maximum top-1 accuracy in Table VI when $th = 0.8$.

cost collecting method to ablate the role of various hardware-level optimizations (e.g., data reuse, pipeline, and resource allocation).

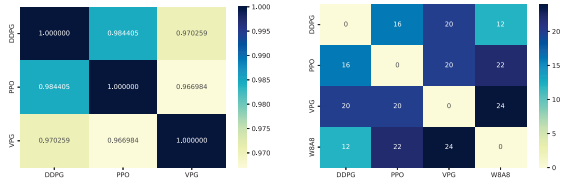
C. Policy Similarity Analysis

In this section, we try to answer the question: **why different policies lead to similar accuracy?** We propose two metrics to mesuare the policy similarity: 1) **cosine similarity**; 2) **manhattan distance**. As Fig. 4(a) shows, the cosine similarity between different searched quantization policies is close to 1, providing a clue that they can achieve a similar top-1 accuracy. Fig. 4(b) demonstrates the manhattan distance between different quantization policies and W8A8. The manhattan distance between the quantization policy searched by the DDPG and W8A8 is 12, which is smaller than the other two methods (i.e., 22 for PPO and 24 for VPG), indicating that its corresponding accuracy is closest to that of W8A8. This observation is consistent with the results in Table V.

Manhattan distance can also be used to measure the cost of the bitwidth adjustment step in HAQ. We replace the latency lookup table in HAQ with (3) to get the hardware cost (i.e., the normailized $\#tile$) for the VGG model. The top of the Fig. 5 plots the normalized $\#tile$ during the first 300 episodes when $th = 0.8$. The bottom of the Fig. 5 demonstrates the corresponding policies before and after the adjustment step at the 43rd episode. The manhattan distance between the blue and orange bars in Fig. 5 is 16, indicating that it needs 16 adjustment steps, each of which reduces 1-bit, to meet the resource constraint.

D. Impact of Reward Function

The reward function in HAQ only relates to the accuracy, directing the agent to select higher bitwidth to achieve higher accuracy but easily break the resource constraint after 200 episodes. Therefore, the bitwidth adjustment step is necessary in HAQ. In contrast, Fig. 6 demonstrates that the proposed reward function in Section III-A3 enables the agent to learn a quantization policy satisfying the resource constraint without any manual adjustment. Therefore, the reward functions with different optimization goals indeed influence the learning behavior of the agent. We argue for using the same reward function to ablate the role of the reward function reshaping when comparing different automated quantization methods.



(a) Cosine similarity (b) Manhattan distance

Fig. 4. The similarity measurement between different quantization policies.

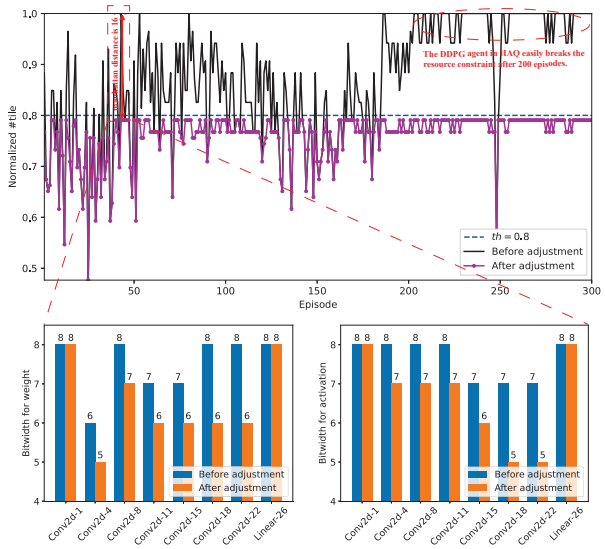


Fig. 5. The normailized $\#tile$ during the first 300 episodes when $th = 0.8$ (top) and the corresponding policies at the 43rd episode (bottom).

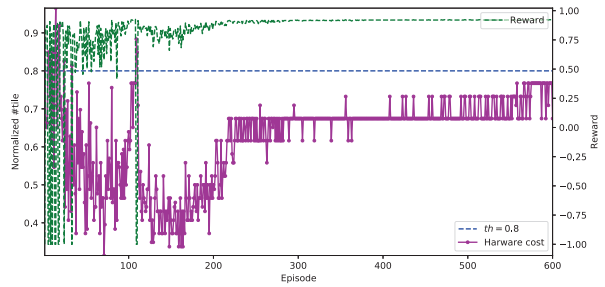


Fig. 6. The reward and hardware cost during the first 600 episodes when $th = 0.8$.

V. Conclusion

This paper proposes a benchmark framework, BenQ, which covers three representative RL-based algorithms and two off-the-shelf performance predictors to enhance the fair comparisons of RL-based automated quantization algorithms. Empirical observations show that different RL-based methods can achieve near equivalent optimal trade-offs. We propose two policy similarity measurements to interpret the above results, which is less explored in previous works. We expect it will provide insights for revisiting the *de facto* innovations in RL-based automated quantization.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant 62172327.

References

[1] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[2] Y. Chen, Y. Xie, L. Song, F. Chen, and T. Tang, "A survey of accelerator architectures for deep neural networks," *Engineering*, vol. 6, no. 3, pp. 264–274, 2020.

[3] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *CVPR*, 2018, pp. 2704–2713.

[4] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *ICLR*, 2016.

[5] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 1, pp. 269–284, 2014.

[6] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *ISCA*, 2016, pp. 367–379.

[7] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: Hardware-aware automated quantization with mixed precision," in *CVPR*, 2019, pp. 8612–8620.

[8] A. T. Elthakeb, P. Pilligundla, F. Mireshghallah, A. Yazdanbakhsh, and H. Esmaeilzadeh, "Releq : A reinforcement learning approach for automatic deep quantization of neural networks," *IEEE Micro*, vol. 40, no. 05, pp. 37–45, 2020.

[9] S. Qu, B. Li, Y. Wang, D. Xu, X. Zhao, and L. Zhang, "Raqu: An automatic high-utilization cnn quantization and mapping framework for general-purpose rram accelerator," in *DAC*, 2020.

[10] Y. Li, M. Shen, J. Ma, Y. Ren, M. Zhao, Q. Zhang, R. Gong, F. Yu, and J. Yan, "Mqbench: Towards reproducible and deployable model quantization benchmark," in *NeurIPS*, 2021.

[11] C. Li, Z. Yu, Y. Fu, Y. Zhang, Y. Zhao, H. You, Q. Yu, Y. Wang, C. Hao, and Y. Lin, "Hw-nas-bench: Hardware-aware neural architecture search benchmark," in *ICLR*, 2021.

[12] Y. Zhao, C. Li, Y. Wang, P. Xu, Y. Zhang, and Y. Lin, "Dnn-chip predictor: An analytical performance predictor for dnn accelerators with various dataflows and hardware architectures," in *ICASSP*, 2020, pp. 1593–1597.

[13] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "Dnn+neurosim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in *IEDM*, 2019, pp. 32.5.1–32.5.4.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning." in *ICLR*, 2016.

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

[16] W. Jiang, Q. Lou, Z. Yan, L. Yang, J. Hu, X. S. Hu, and Y. Shi, "Device-circuit-architecture co-exploration for computing-in-memory neural accelerators," *IEEE TC*, vol. 70, no. 4, pp. 595–605, 2020.

[17] F. Wang, Y. N. Wu, M. Woicik, J. S. Emer, and V. Sze, "Architecture-level energy estimation for heterogeneous computing systems," in *ISPASS*, 2021, pp. 229–231.

[18] W. Zhang, X. Peng, H. Wu, B. Gao, H. He, Y. Zhang, S. Yu, and H. Qian, "Design guidelines of rram based neural-processing-unit: A joint device-circuit-algorithm analysis," in *DAC*, 2019.

[19] Z. Zhu, H. Sun, K. Qiu, L. Xia, G. Krishnan, G. Dai, D. Niu, X. Chen, X. S. Hu, Y. Cao, Y. Xie, Y. Wang, and H. Yang, "Mnsim 2.0: A behavior-level modeling tool for memristor-based neuromorphic computing systems," in *GLSVLSI*, 2020, pp. 83–88.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[21] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[22] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.