HAS-RL: A Hierarchical Approximate Scheme Optimized With Reinforcement Learning for NoC-Based NN Accelerators

Li, Siyue ; Zhou, Shize ; Xue, Yongqi ; Fan, Wenjie; Cheng, Tong ; Ji, Jinlun ; Dai, Chenyang ; Song, Wenqing ; Gao, Chang; More Authors

DOI
10.1109/TCSI.2024.3359912

Publication date
2024

Document Version
Final published version

Published in
IEEE Transactions on Circuits and Systems I: Regular Papers

HAS-RL: A Hierarchical Approximate Scheme Optimized With Reinforcement Learning for NoC-Based NN Accelerators

Siyue Li, Shize Zhou, Yongqi Xue, Wenjie Fan, Tong Cheng, Jinlun Ji, Chenyang Dai, Wenqing Song, Qinyu Chen, *Member, IEEE*, Chang Gao, *Member, IEEE*, Li Li, *Member, IEEE*, and Yuxiang Fu, *Member, IEEE*

***Abstract*— Network-on-Chip (NoC) is a scalable on-chip communication architecture for the NN accelerator, but with the increase in the number of nodes, the communication delay becomes higher. Applications such as machine learning have a certain resilience to noisy/erroneous transmitted data. Therefore, approximate communication becomes a promising solution to improving performance by reducing traffic loads under the constraint of the acceptable maximum accuracy loss of neural networks. It is a key issue to balance the result quality and the communication delay for approximate NoC systems. The traditional approximate NoC only considers the node-to-node approximation-based dynamic traffic regulation. However, the dynamically changing traffic patterns across different nodes, different times, and different applications lead to a huge search space, which makes it hard to explore an optimal global approximation solution. In this paper, we propose a quality model for different neural networks, which presents the relationship between the quality loss and the data approximate rate. Then, a hierarchical approximate scheme optimized with reinforcement learning (HAS-RL) is proposed and we reduce the complexity of the HAS-RL by reducing the state space and action space, which will reduce the resource overhead as well. After that, we embed a global approximate controller in the NoC system, in which we deploy a policy network trained with the offline reinforcement learning algorithm to adjust the data approximate rates of each node at run time. Compared with the state-of-the-art method, the proposed scheme reduces the average network delay by 13.5% while their accuracies are similar. The proposed HAS-RL only causes an additional area overhead of 1.24% and power consumption of 0.77% compared with the traditional router design.**

***Index Terms*— Offline reinforcement learning, neural network, approximate communication, network-on-chip.**

I. Introduction

APPLICATIONS such as machine learning and image processing have a certain error-resilience of data. In essence, approximate computing is the computing paradigm that trades quality for performance [1]. Ye et al. [2] and Ha and Lee [3] designed an approximate adder and an approximate multiplier that can be used for approximate computing. Wang et al. [4] used software to identify the noncritical portion of computations that can be approximated. These works focus on approximate computing but ignore the approximation in communication. With the increase in the number of nodes, on-chip communication plays an increasingly important role in multi-core chips. The communication delay of NoC systems becomes higher and the real-time performance becomes worse, especially for data-intensive applications.

In previous works, some researchers have used global congestion awareness [5] or regional congestion awareness [6] adaptive routing algorithms, or different topologies like CMesh [7], 3D torus [8], and 3D butterfly fat-tree [9] to alleviate the congestion in the NoC. Ramakrishna et al. [5] proposed an adaptive routing algorithm based on global link states and congestion information for on-chip routers. Xu et al. [6] used both local and non-local/aggregated congestion information to estimate network congestion and proposed an adaptive routing based on efficient regional congestion. Ebrahimi et al. [10] designed a fully adaptive routing algorithm for 3D NoCs which uses the congestion information at the input buffer of the neighboring routers as the congestion metric to select among the output channels. However, these schemes can not reduce the number of flits transmitted, which are only applicable to the congestion situation caused by unbalanced loads rather than heavy loads.

Manuscript received 21 August 2023; revised 13 December 2023 and 5 January 2024; accepted 23 January 2024. Date of publication 12 February 2024; date of current version 29 March 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62104098, in part by the Joint Funds of the National Natural Science Foundation of China under Grant U21B2032, in part by the Natural Science Foundation of Jiangsu Province for Youth under Grant BK20210178, in part by the National Key Research and Development Program of China under Grant 2021YFB3600104, in part by the National Key Research and Development Program of China under Grant 2023YFB2806802, and in part by the "Xiaomi Youth Scholar—Innovation and Technology Award." This article was recommended by Associate Editor H. Stratigopoulos. *(Corresponding authors: Li Li; Yuxiang Fu.)*

Siyue Li, Shize Zhou, Yongqi Xue, Wenjie Fan, Tong Cheng, Jinlun Ji, Chenyang Dai, Wenqing Song, Li Li, and Yuxiang Fu are with the School of Integrated Circuits and the School of Electronic Science and Engineering, Nanjing University, Qixia, Nanjing, Jiangsu 210023, China, also with the Interdisciplinary Research Center for Future Intelligent Chips (Chip-X), Nanjing University, Suzhou 215163, China, and also with the State Key Laboratory of Spintronics Devices and Technologies, Nanjing University, Suzhou 215163, China (e-mail: lili@nju.edu.cn; yuxiangfu@nju.edu.cn).

Qinyu Chen is with the Institute of Neuroinformatics, University of Zurich and ETH Zurich, 8057 Zurich, Switzerland.

Chang Gao is with the Electronic Circuits and Architectures (ELCA) Group, Faculty of Electrical Engineering, Mathematics, and Computer Science (EEMCS), Delft University of Technology, 2628 CD Delft, The Netherlands.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2024.3359912.

Digital Object Identifier 10.1109/TCSI.2024.3359912

To solve these problems, some recent works have focused on approximate communications [11], [12], [13], [14], [15], [16], [17], [18], [19]. Approximate communication can reduce the average delay and the energy consumption of NoC systems, which will sacrifice some accuracy at an acceptable quality loss [11]. Boyapati et al. [12] compressed data in different compression patterns but did not consider network congestion. The data approximate method of the Approximation-Based Dynamic Traffic Regulation (ABDTR) [13] is a feedback process essentially, which senses congestion by monitoring the number of free buffer slots in different nodes and adjusts the approximate rate of the source node according to the congestion information. The Accuracy and Congestion-aware Dynamic traffic Control method (ACDC) [14], [15] contains a quality model and a lightweight heuristic algorithm to control each traffic route's approximate rate. The Slack-Aware Packet Approximation technique (SAPA) [17] includes a slack-aware control policy to identify low-slack packets and accelerates these packets using two approximation mechanisms.

However, for quality control, ABDTR simply considers that the data approximate rate has a linear relationship with the quality, which is not accurate. Meanwhile, a more complex nonlinear quality model has been proposed in ACDC and it is related to the statistical characteristics of input data, but it does not consider the data approximations in the middle layers. The two methods all use linear interpolation to recover data, but SAPA uses a 32-bit quantization method and approximates data through truncation. However, for neural networks, low-bit quantization is typically used, so the bit width that can be further truncated is limited. In this paper, we propose a quality model in the form of the quadratic function for neural networks, which drops part of the outputs of each layer rather than truncations and considers not only the data approximate in the input layer but also the middle layers.

Performance optimization at run time is also a complicated problem in these previous methods. In ABDTR, for example, in the NoC system with n routers, if each router can be adjusted with v available approximate rates, the total number of regulation combinations is as large as v^n in each control interval. The NoC system must respond quickly to the change of network congestion at run-time, which is a challenge for approximate system design. In ACDC, there are $n \times (n-1)$ different traffic routes. A lot of area resources and time are consumed in the process of finding the route of the maximum traffic, which results in limited performance improvements. SAPA, which allows high-slack packets to be dropped in halfway, also loses some performance because it makes the transmission of previous hops before the dropped node meaningless.

In this paper, we want to maximize the performance of the system with minimal quality loss, so there is a trade-off between the average delay and the data quality. Reinforcement learning is a better choice to solve the problem with a huge exploration space and performs well at sequential decision-making [20], [21], [22]. Based on the policy update algorithm, the agent can constantly learn from the environment to update the policy and find a better solution than traditional methods. Because online reinforcement learning requires repeatedly gathering experience through interaction with the environment, and in many cases, the cost of online learning, including money, energy, and so on, is high, training agents with pre-existing datasets is preferred [23]. The reinforcement learning algorithm can sequentially make a series of policies to adjust the data approximate rates to maximize the cumulative reward at run time and get a better trade-off between quality and performance. In this paper, we find the relationship between the number of free buffer slots and congestion and then explore the rationality of using local free buffer slots to represent congestion, which reduces the state space. We further reduce the action space by categorizing nodes according to different congestion situations, as shown in Fig. 1. Therefore, it reduces the resource overhead and improves the availability of reinforcement learning. To sum up, we propose a hierarchical approximate scheme optimized with reinforcement learning (HAS-RL) for accurately controlling the approximate rates of different nodes in different traffic patterns to achieve performance improvement.

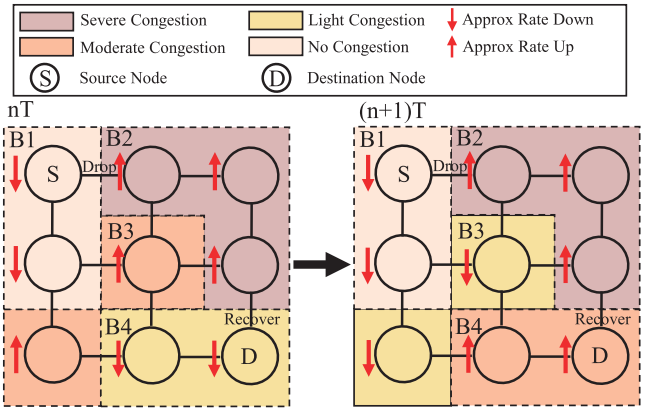


Fig. 1. Concept of the hierarchical approximate scheme optimized with reinforcement learning (HAS-RL), which classifies the network according to the congestion cases and adjusts the approximation rate at run time. The flit is dropped at the source node and recovered at the destination node.

We make the following contributions to this paper:

(1) A quality model for the neural networks is proposed. We obtain the result accuracy of different neural networks under different approximate rates. Then the optimization objective function is determined according to the quality model.

(2) To optimize the objective function, The hierarchical approximate scheme is proposed, which is composed of one global approximate controller optimized with offline reinforcement learning and n local data controllers. We train the policy network used for decision-making in offline reinforcement learning with software and then deploy it on the global approximate controller with hardware. The local data controller consists of a data dropper and a data recoverer, which is used to conduct data approximate and recovery operations.

(3) We use local free buffer slots to represent the state to reduce the state space and categorize nodes with different congestion conditions to reduce the action space, which reduces resource overhead and improves the availability of reinforcement learning.

(4) Compared with two recent works [15], [17], experimental results show that the HAS-RL can achieve a better balance between performance and data quality. This method can adapt to different mappings of different applications. Compared with [15] and [17], the average network delay is reduced by 9.3% and 13.5%, respectively. And the data quality improved by 0.5% compared with [15] and is similar compared with [17].

The rest of this paper is organized as follows. We first review the related works in Section II. The quality model and the objective function are proposed in Section III. We elaborate on the HAS-RL structure and its working mechanism in Section IV. In Section V, we evaluate the average delay, quality loss, energy consumption, throughput, and hardware overhead. Finally, we summarize our work in Section VI.

II. Related Works

A. Approximate Communication in NoC

Approximate communication provides a feasible high-performance communication solution by relaxing the restrictions on the accuracy of data transmitted, especially in data-intensive applications. We divide the previous approximate communication schemes into two classes: approximate the entire data in flits and approximate LSBs of data in flits.

1) Approximate the Entire Data in Flits: Wang et al. [13] proposed Approximation-Based Dynamic Traffic Regulation (ABDTR), which drops packets at the source node and recovers packets at the destination node. It uses a simple negative feedback adjustment mechanism to control the data approximate rate and monitors congestion by using the number of free slots of buffers per node but does not distinguish the direction, resulting in inaccurate control. Xiao et al. [14], [15] proposed the Accuracy and Congestion-aware Dynamic traffic Control method (ACDC), which puts forward the model of on-chip network congestion under the constraint of the result quality. At the same time, the lightweight heuristic algorithm is proposed which can relieve congestion in the NoC. However, the lightweight heuristic algorithm can not solve congestion very well because it may not detect congested links correctly. In this case, the effect of this control method is not obvious. Wang et al. [24] proposed the Approximate Bufferless method (ABNoC), which reduces network congestion and packet retransmission through an approximate allocation mechanism and a packet approximation method. Ahmed et al. [25] proposed a two-type voltage management method (AxNoC), but transmissions with low voltages can cause unexpected bit reverses and lead to some errors.

2) Approximate LSBs of Data in Flits: Boyapati et al. [12] proposed a hardware data approximation framework (APPROX-NoC), which reduces the amount of data transmitted in the NoC by providing a variety of different approximate patterns for accurate data. Stevens et al. [16] proposed an approximate bus architecture framework (AxBA), which is a compression technique based on approximate deduplication. Chen et al. [18] proposed an approximate framework based on Dynamic Error Control (DEC-NoC), which can dynamically adjust the data accuracy based on the error tolerance to reduce retransmissions significantly and achieve half latency reduction. Reza and Ampadu [19] summarized the approximate techniques based on the truncations in NoCs and proposed a general approximate communication scheme based on the error tolerance to explain the importance of approximate communication techniques in NoCs. Chen et al. [11] proposed an approximate communication framework (ACF), which analyzes the source code and marks error-resilient variables to reduce transmission. Chen et al. [17] proposed the Slack-Aware Packet Approximation technique (SAPA), which includes a slack-aware control policy to identify low-slack packets and accelerates these packets using two approximation mechanisms. However, SAPA approximates data through truncation, whose performance is limited for neural networks using low-bit quantization.

All the above studies are based on the optimization of network architecture, but their control methods are insufficient to solve the key problem of how to balance the result quality and the communication delay for approximate NoC systems. They have room for performance improvements using approximate communication under the limited quality loss.

B. ML for NoC Designs

Machine learning methods have exploded in many fields in recent years. Some researchers have applied machine learning algorithms to NoC optimization. Chen and Liao [26] proposed a Predictive Dynamic Thermal Management (PDTM) method, which uses machine learning to control the temperature of multicore systems. Reza et al. [27] proposed a Neuro-NoC model, which utilizes neural networks to dynamically monitor, predict, and configure NoC resources. Rao et al. [28] proposed a method to realize the design of high-dimensional NoC using machine learning, which can quickly produce near-optimal NoC design. Rapp et al. [29] proposed the run-time algorithm PCMig which is based on a lightweight neural network to predict the performance impact of task migrations.

Reinforcement learning can make better decisions than traditional design and is widely used in architecture optimization. Chen et al. [30] proposed an Online Distributed Reinforcement Learning (OD-RL) based DVFS control algorithm for the multicore system under power constraints. Reza [31] used a reinforcement learning algorithm to proactively configure NoC link bandwidth in heterogeneous architecture to improve energy efficiency. Chen et al. [32] proposed learning-based quality management, which uses a reinforcement learning algorithm to assign error thresholds to different variables in programs to achieve approximate effects.

Compared with the previous works, this paper uses the offline reinforcement learning algorithm to establish a smart global approximate controller that can adjust the data approximate rate of each node dynamically. It achieves a better balance between performance and quality requirements.

III. Problem Modelling

In this section, we establish a quality model for neural networks. Based on this quality model, we propose the optimization goal.

Algorithm 1 The Approximate Function Integrated Between Two Adjacent Layers to Establish the Quality Model of Approximate Communications

Input: $D_o{}^y(C, L, W)$: the layer outputs.
Output: $D_i{}^{y+1}(C, L, W)$: the next layer approximate inputs.
1 Initialize Apr for each layer and f_n;
2 **for** *each* $i \in [0, C-1]$ **do**
3 **for** *each* $j \in [0, L-1]$ **do**
4 **for** *each* $k \in [0, W-1]$ **do**
5 $R_n \Leftarrow \frac{rand()}{RAND\_MAX}$;
6 **if** $R_n < Apr$ **then**
 // The middle body flit
7 **if** $k > 0$ *and* $k < W-1$ **then**
8 $D_i^{y+1}(i, j, k) \Leftarrow \frac{D_o^y(i,j,k-1)+D_o^y(i,j,k+1)}{2}$;
9 **end**
 // The first body flit
10 **if** $k = 0$ *or* $k\%f_n = 0$ **then**
11 $D_i^{y+1}(i, j, k) \Leftarrow D_o^y(i, j, k+1)$;
12 **end**
 // The tail flit
13 **if** $k = W-1$ *or* $k\%f_n = f_n - 1$ **then**
14 $D_i^{y+1}(i, j, k) \Leftarrow D_o^y(i, j, k-1)$;
15 **end**
16 **else**
17 $D_i^{y+1}(i, j, k) \Leftarrow D_o^y(i, j, k)$;
18 **end**
19 **end**
20 **end**
21 **end**

A. The Quality Model

We deploy several widely-used neural networks such as AlexNet [33] and VGG16 [34] for CIFAR-10 datasets to the NoC-based NN accelerator. Each layer of the neural network is mapped to one or several processing elements (PEs). We store a single output value into a flit, and approximate communication is often achieved by reducing a number of flits between nodes, which means we approximate data by dropping part of the outputs of each layer rather than truncating some output values.

As shown in Algorithm 1, we add the approximate function between two adjacent layers to establish the quality model of approximate communications, whose inputs D_o^y are the outputs of the current layer and outputs D_i^{y+1} are the inputs of the next layer. For the CNN, the output shape is composed of $C \times L \times W$, where C, L, and W represent the dimensions of the channel, length, and width, respectively. The data approximate can be simulated by modifying some transmitted data between the layers. We label some data that can be approximated based on their magnitude, which takes advantage of the fact that some data with small magnitude (close to the value zero) have little impact on the overall accuracy even if they are approximated [35]. As shown in line 6 in Algorithm 1, we randomly select some data to change with the probability of Apr, where Apr represents the data approximate rate. In lines $7 \sim 9$, we replace the original value as the average value of the neighboring data in the case of the middle body flit under the assumption that the output data of each layer is packaged in rows, and in lines $10 \sim 15$, we replace the original value as the value of their neighboring data in the case of only one neighbor. The f_n is used to describe the number of flits in a packet.

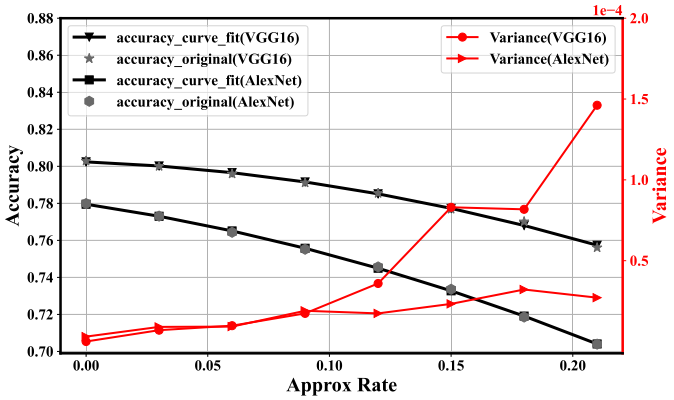


Fig. 2. Accuracies of neural networks and variance of accuracies for 50 simulations under different data approximate rates.

Fig. 2 shows the relationship between the data approximate rate and the accuracy of the neural network. The scatter points are the original data obtained from software simulations, and the line is their fitting curve. The relationship between data approximate rate and quality loss is obtained by data fitting, which can be shown as follows:

$$Aq = \eta_1 \times Apr^2 + \eta_2 \times Apr + \eta_3, \tag{1}$$

where Aq is used to describe the accuracy of the neural network, namely the data quality, and Apr is the data approximate rate. η_1, η_2, and η_3 are different coefficients for different neural networks.

We approximate neuron data randomly in the input layer and middle layers 50 times under the same data approximate rate, whose accuracy variance is also shown in Fig. 2. It can be obtained that the variance is very small when the data approximate rate is not large. It means that it has almost nothing to do with the positions of approximated data in the labeled data. Therefore, we conclude that in neural networks, data quality loss is mainly related to the data approximate rate. We need to determine the coefficients in (1) for different neural networks.

B. Optimization Goals

According to the quality model, the accuracy of the neural network decreases when the data approximate rate increases. A threshold is defined by the user to avoid unacceptable quality loss of the results after approximate transmissions, which we assume 4% accuracy loss in this paper. On the one hand, if the NoC is not congested, the increase in the data approximate rate may bring only a small reduction in delay. In this case, quality is the most critical objective. On the other hand, when the NoC is congested, a little bit of quality loss may bring a significant delay reduction. Therefore, both quality and average delay need to be considered, and a penalty is given if the quality loss exceeds the threshold. The goal of optimization is a trade-off between the data quality and the

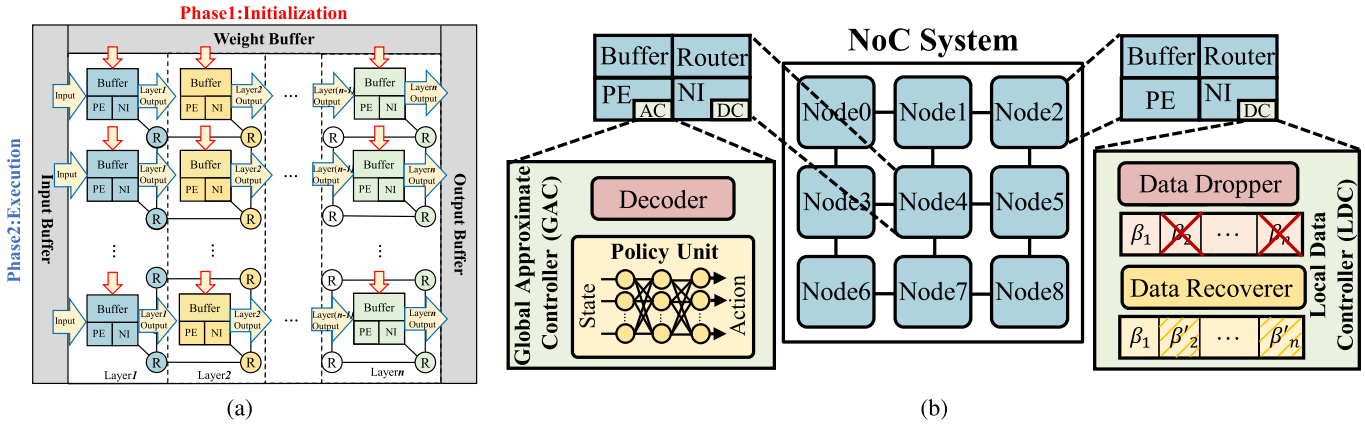


Fig. 3. Overview of HAS-RL. (a) is the NN execution flow in the NoC-based NN accelerator (Take the column-based mapping as an example). (b) is the architecture overview of HAS-RL, which is composed of one global approximate controller and n local data controllers. The global approximate controller is located at the center (Node4) and the local data controller is located at each node of the NoC.

average delay. It can be formulated as (2).

$$Goal = \begin{cases} \xi_1 \times Aq + \xi_2 \times Ad, & \text{for } Aq \geq Aq_{th}, \\ Pn, & \text{otherwise,} \end{cases} \tag{2}$$

where

$$Ad = \frac{1}{\sum_{i=0}^{n} m_i} \times \sum_{i=0}^{n} \sum_{j=0}^{m_i} (1 - \frac{Pd_{ij}}{Avd\_na}). \tag{3}$$

In (2), Aq is the data quality in (1), and Ad is the reward of network performance. ξ_1, ξ_2 are weights of each component. Aq_{th} is the threshold that is determined by the user. Pn is a penalty term that prevents the data approximate rate from exceeding the quality loss threshold and becoming unacceptable. In (3), we use Pd to reflect the delay of every packet, m_i to represent the number of packets sent by node i and Avd_na to be the average delay without approximation. The optimization process is to maximize the goal function.

IV. HAS-RL: Hierarchical Approximate Scheme Optimized With Reinforcement Learning

In this section, firstly, we introduce the execution flow of the applications and the architecture overview of the proposed HAS-RL. Secondly, we show the global approximate controller optimized with offline reinforcement learning and reduce the complexity of HAS-RL by reducing the state space and the action space. Finally, we provide a case study and the hardware implementation is presented.

A. The Architecture Overview of HAS-RL

We design an NoC-based neural network accelerator. As shown in Fig. 3a, in this architecture, each node contains a router and a PE. The router is used for packet transmission, and the PE is used for computing neural networks. In the initialization phase, different layers of the neural network are mapped to different nodes, and the built-in buffer of each node stores the weight and bias needed for the computation. In the execution phase, the nodes which are assigned to the first layer get the required input data from the input buffer. After receiving the input data required by the node, the computation starts. Then when the computation of the first layer is completed, the output data will be transmitted to the nodes of the next layer as their inputs. The nodes of the last layer finally write the output data to the output buffer.

As shown in Fig. 3b, the HAS-RL is composed of one global approximate controller and n local data controllers. Instead of using a distributed RL approximate controller, we employ a global RL approximate controller as the policy unit for four main reasons:

Challenges with distributed RL approximate controller

1. It is difficult to control the quality in the distributed RL approximate controller because the accuracy of the NN applications is related to the global approximation rate instead of the approximation rate of a single node.

2. It is difficult to achieve global optimization, because the actions between nodes may affect each other.

Scalability of the global RL approximate controller

1. Since the control signal has the highest priority, and the interval between each control is relatively long, the control delay is acceptable.

2. The number of control packets is only a small portion of data packets.

The global approximate controller includes a policy unit running the policy network and a decoder for converting the output of the policy network into control packets.

The local data controller consists of a data dropper and a data recoverer. The data dropper compresses data by selectively dropping flits based on the data approximate rate information sent by the global approximate controller. The data dropper includes a random number generator which generates a random number between 0 and 1 and a comparator which is used to compare the data approximate rate with the random number to determine whether the labeled flit is allowed to be sent. If the flit is not authorized to send, the location of the flit is marked in the head flit and the subsequent flits are moved forward in order to shorten the packet length. The data recoverer recovers data at the destination node by identifying the approximated packet and using the average value of its

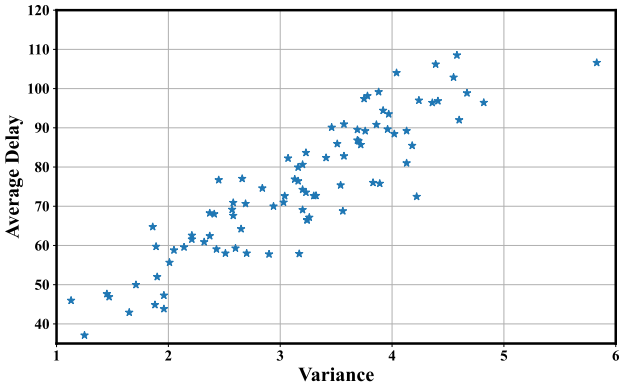


Fig. 4. Relationship between the delay and the variance of free buffer slots under different mappings of AlexNet.

neighboring flits. For a special case where a flit has only one neighbor such as the first or the last body flit, the value of its neighbor is copied.

B. Optimizing Policy Unit With the Offine RL Algorithm in the Global Approximate Controller

An RL agent is integrated into the global approximate controller which adjusts the data approximate rate of each node by monitoring the congestion of the network. We use the Deep Q Network (DQN) algorithm, which predicts the probability of each action by calculating the Q value and selects the action with the maximum Q value as the set of data approximate rates of each node in the next period.

The reinforcement learning algorithm includes four elements: *Environment*, *State*, *Action*, and *Reward*.

1) Environment: The environment is the NoC-based NN accelerator, which consists of routers, PEs, and links. When the nodes take different actions, the environment can generate different rewards depending on the current state.

2) State: Some works analyze the congestion of the NoC by using hardware utilization. For example, ABDTR [13] and HREN [36] monitor the congestion by analyzing the free slots of buffers, and ACDC [15] obtains the congestion information by monitoring the link utilization in the NoC. For the link utilization in a 3D NoC with the size of $l \times w \times h$ (assuming $l = w = h$), the total number of links is $3 \times l^2 \times (l-1)$. With the increase in the number of nodes, the state space will be too large, and the power consumption and area overhead will rise rapidly by using the link utilization as the $state$.

For the NoC with router buffers, we conduct a pre-experiment to reflect the relationship between the free buffer slots and congestion. Fig. 4 shows the relationship between the delays of different random mappings and the variances of free buffer slots of each node under the AlexNet. It can be seen that the two variables are strongly positively correlated, and the Pearson coefficient is 0.9. Therefore, for different mappings with the same traffic load volume, the less uniform the traffic distribution is, the worse the NoC performance will be, and the higher the average delay will be. We can use the number of free buffer slots of every node to directly reflect the network congestion. However, each node has 7 buffers for 3D NoCs, the state space is still large.

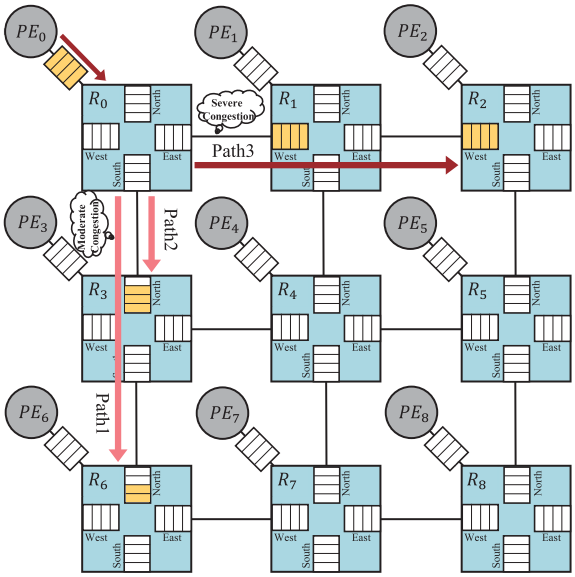


Fig. 5. An example of illustration that the number of local free buffer slots represents the congestion levels.

In Fig. 5, if the west direction of node 1 is congested, the data transmitted from node 0 to node 2 will be blocked. However, if congestion is detected in the west direction of node 1, node 1 cannot be adjusted directly because the congestion is caused by excessive data sent from node 0. ABDTR adopts an emergency channel to send the control message. It finds the source node from the head flit and then controls the source node by sending the control message through the emergency channel. This brings the overhead of the emergency channel and increased power consumption due to the additional control information. Due to congestion in the west direction of node 1, the packets generated by node 0 that should be transmitted to node 2 are blocked in the local buffer, which leads to a very high local buffer occupancy rate. We obtained the numbers of all free buffer slots in the transmission paths from node 0, including the west direction of R1, the west direction of R2, the north direction of R3, and the north direction of R6. We get their minimum value and marked it as PB_0. The number of local free buffer slots in node 0 is marked as LB_0. For the j_{th} source node, we define the minimum number of free slots in all buffers that the data will pass through to be PB_j, and the number of local free buffer slots is marked as LB_j. Fig. 6 shows the relationship between PB_j and LB_j. Their Pearson coefficient is 0.93, indicating that they are very correlated.

If we want to monitor the minimum number of free buffer slots on all routes, the overhead is very large. Therefore, we use an indirect but accurate enough scheme to monitor network congestion, that is, using the number of local free slots. The size of state space is only n, where n is the number of nodes in the NoC system. The state space is 7 and $2 \sim 3$ times smaller than the buffer-based method and link-based method, respectively. We set the $State$ to be a one-dimensional vector with n values. Each value in the vector is the average number of the local free buffer slots during the period t.

3) Action: First, we assume that the maximum data approximate rate cannot exceed $f\%$. Then we initialize the data

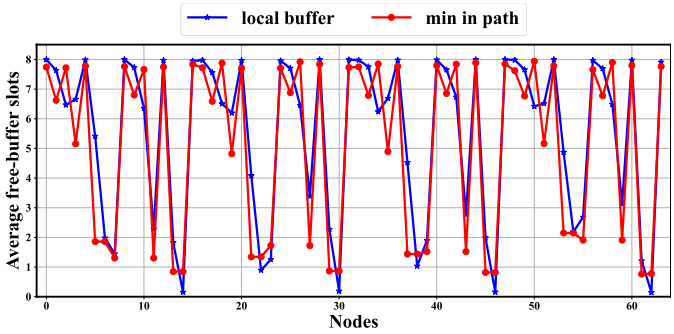


Fig. 6. Relationship of local buffer free slots and the minimum buffer free slots of transmission paths from a node.

approximate rate to $f/2\%$ in each node at the beginning. The adjustment method is as follows: each node can decrease or increase the data approximate rate by $i\%$ at a time. For an NoC system with n nodes, the number of possible configurations of approximate control grows at an exponential rate 2^n, thereby creating an astronomically large decision space when n becomes large. Consequently, we confront two pivotal challenges: firstly, determining the appropriate number of categories for partitioning nodes across varying NoC scales, and secondly, devising a method for effectively distributing these nodes among those categories.

We classify nodes into C_n categories according to the local free buffer slots and the number of nodes and propose a hardware-constrained classification optimization scheme. Firstly, we define the maximum duration of decision-making in the policy unit as t_max (assuming $t\_max = 400$ cycles in this paper) ensuring the global approximate controller completes the selection of actions within t_max. Then the number of multiply-accumulator (MAC) units required to complete the decision in t_max determines the additional hardware overhead in the global controller. To maintain practicality, we impose a restriction that the extra hardware overhead does not exceed more than $h\%$ of a traditional router (assuming $h = 2$ in this paper). Under this hardware constraint, we maximize the output number of the policy unit to minimize control granularity, which will obtain better performance. The classification optimization scheme can be formulated as (4):

$$C\_n = \left\lfloor \log_2 \max(output\_n) \right\rfloor \quad \text{under} \begin{cases} D\_t < t\_max \\ H\_o < h\% \times T\_ho \end{cases} \tag{4}$$

where $output\_n$ is the output number of the policy unit. D_t, H_o, and T_ho represent decision time, additional hardware overhead, and the hardware overhead of the traditional router, respectively.

For the challenge of how to divide nodes into C_n categories, we first pre-run different applications and their different mappings on an NoC-based system. Then, we sort the occupancy of the local free buffer slots of all nodes, and all nodes are placed evenly into each category from the lowest to the highest occupancy. Therefore, the principle of classification is to divide nodes into each category as evenly as possible instead of choosing fixed slot occupancy thresholds to divide different nodes, which overcomes the challenge of determining different thresholds for different applications. For example, we could divide the 64 cores into 4 different categories according to the classification optimization scheme and the congestion: severe congestion (SC), moderate congestion (MC), light congestion (LC), and no congestion (NC). Then the nodes belonging to the same congestion category are grouped into the same block, and we treat each block as a whole when adjusting the data approximate rate.

The data approximate rate can only be adjusted between 0% and $f\%$. If the data approximate rate is 0% and the action is to decrease the data approximate rate, we will keep the data approximate rate to 0%. If the data approximate rate is $f\%$ and the action is to increase the data approximate rate, we will keep the data approximate rate to $f\%$.

4) Reward: The RL agent outputs the action according to the congestion in the current state. The environment generates rewards according to the optimization function. The optimization function can be formulated as (2). The reward determines how well the action is taken and the Deep Q Network algorithm maximizes the cumulative reward, which is a better trade-off between average delay and quality.

5) Training RL Model Using DQN Algorithm: As shown in Fig. 7, the offline reinforcement learning framework has the following steps. First, the dataset is obtained from the NoC simulator. And then the agent is trained on the software platform. Finally, the trained policy network is deployed to the hardware platform.

Offline reinforcement learning is trained on offline datasets. There is no interaction with the environment during the training. In previous works such as [37] and [38], it has been demonstrated through experiments that a better model can be obtained by increasing the diversity and the size of datasets. Therefore, we added the diversity of data and increased the size of the dataset to ensure the effectiveness of offline RL. To increase the data diversity, some traffic variations are considered, including adding some burst transmission, throttling, and so on. At the same time, different datasets corresponding to different mappings are added. On the other hand, we collected enough datasets to approximately approach the performance of online learning.

The datasets consist of a series of strategies. A single policy is composed of a series of actions taken during one episode. After executing one action, the agent gets a reward from the environment and observes the next state. When an episode is completed, the states, actions, and rewards of each step are stored in an offline dataset. The amount of buffer D that stores offline datasets depends on the size of the action space and the state space.

The DQN algorithm is adopted here. It uses a neural network instead of the Q table in the Q-learning algorithm. Q value can be updated by using (5) and (6). DQN algorithm is suitable for the data approximate rate control due to two reasons: the first is the empirical playback mechanism, which can solve the problem of high state correlation between successive steps. The second is that there are two neural networks in this algorithm, named Q online network and Q target network respectively, which make fitting parameters easier. At the same

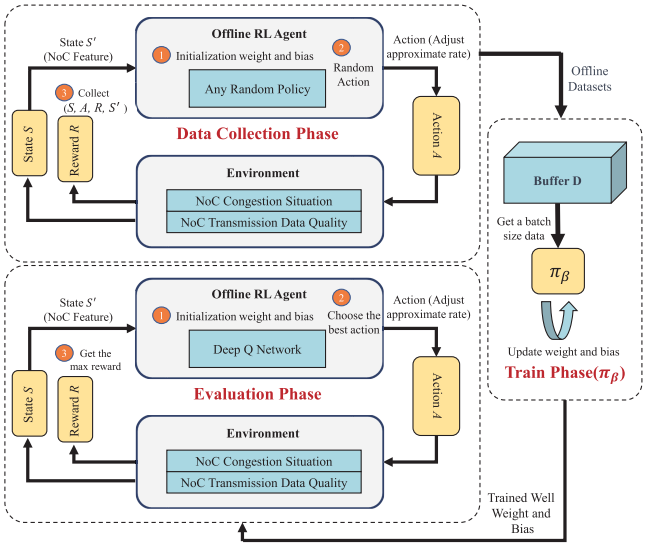


Fig. 7. Approximate rate control optimized with offline RL.

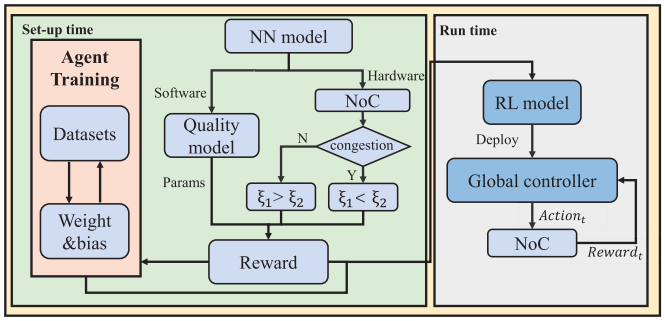


Fig. 8. Process of HAS-RL, which includes two stages. At set-up time, the quality model and the policy network are obtained. At run time, the policy network is deployed and the RL model outputs the control command to adjust the approximate rate of each node.

time, other hyperparameters are adjusted to make the network fit better.

$$y_j(s_{j+1}) = \begin{cases} r_j, & \text{if } s_{j+1} \text{is terminal,} \\ r_j + \gamma max_a \hat{Q}(s_{j+1}, a; \theta'), & \text{otherwise,} \end{cases} \tag{5}$$

$$Loss = (y_j(s_{j+1}) - \hat{Q}(s_j, a_j; \theta'))^2, \tag{6}$$

where r_j is the reward at the current state. The discount factor γ determines the importance of future rewards. s_{j+1} is the next state. The $max_a \hat{Q}(s_{j+1}, a; \theta')$ represents the largest Q in the s_{j+1} state of θ' policy. $y_j(s_{j+1})$ is the target Q value. (6) is the loss function, where $\hat{Q}(s_j, a_j; \theta'))$ is the prediction Q value and $Loss$ is the loss value.

C. Case Study

We use VGG16 as an example to provide a case study. As shown in Fig. 8, the process of HAS-RL includes two stages as follows:

Set−up time:

1. We adopt a random mapping method to map different layers of the neural network to the NoC-based NN accelerator. At the same time, we simulate the quality model of VGG16 with software to determine the coefficients $\eta_1 = -0.78$, $\eta_2 = -0.05$, and $\eta_3 = 0.802$. Based on this, we can obtain the quality model of VGG16.

2. The rewards of reinforcement learning are determined by the optimization goal in (2). In the NoC-based NN accelerator, batch processing and pipeline calculation methods are employed. Besides, due to the large size of the VGG16, the network load is heavy, so we set $\xi_1 = 2$ and $\xi_1 = 3$ to ensure $\xi_1 < \xi_2$. In this case, the user-defined maximum quality loss is defined as 4% accuracy loss. The penalty term Pn whose absolute value is much larger than the normal case, which means $|Pn| \geq \xi_1 + \xi_2$, so in this case, we set $Pn = -5$.

3. We generate different offline datasets using random policies for VGG16. The size of the offline datasets is approximately 10000 episodes, which is large enough for offline learning [37], with 30 steps in each episode executing different actions.

4. We use the DQN algorithm for training. After each iteration, we will use the real NoC platform for evaluation and get the cumulative reward under this policy. When the reward converges, we will save the weights and biases under this policy. We use the policy network with fully connected layers, with a size of $64 \times 128 \times 32 \times 16$ in this case.

Run time:

5. Deploy the trained policy network to the global approximate controller.

6. At run time, the global approximate controller obtains the congestion information of each node and outputs the action as the control command through the policy network. Then the control command is broadcast to each node. Finally, the data dropper adjusts the approximate rate of each node according to the control command.

D. Hardware Implementation

We implement the well-trained policy network in hardware. The input of the neural network is the state value. The output is the Q values for different actions. Then the agent selects the action with the maximum Q value.

The basic hardware architecture is shown in Fig. 3b. The global approximate controller is located at the center of the NoC, which includes a decoder and a policy unit. Because the essence of neural networks is the multiplication and accumulation of matrices, as shown in Fig. 9a, the policy unit includes a set of multiply-accumulator (MAC) units, which use low-bit fixed-point multipliers and adders to reduce area overhead. At the same time, parallel MAC units are used to accelerate computing. The number of output nodes depends on the size of the action space. The agent takes the action with the maximal value as the final output. The decoder decodes the action into a control packet and transmits it to all nodes.

For the local data controller, as shown in Fig. 9b, we use a 7-stage linear feedback shift register (LFSR) to generate a sequence of random numbers and determine the feedback by choosing whether $g_0 \sim g_7$ is true or false. The random number is compared with the current data approximate rate to determine whether the flit is dropped. We recover the data by simply using addition and shift following Algorithm 1.

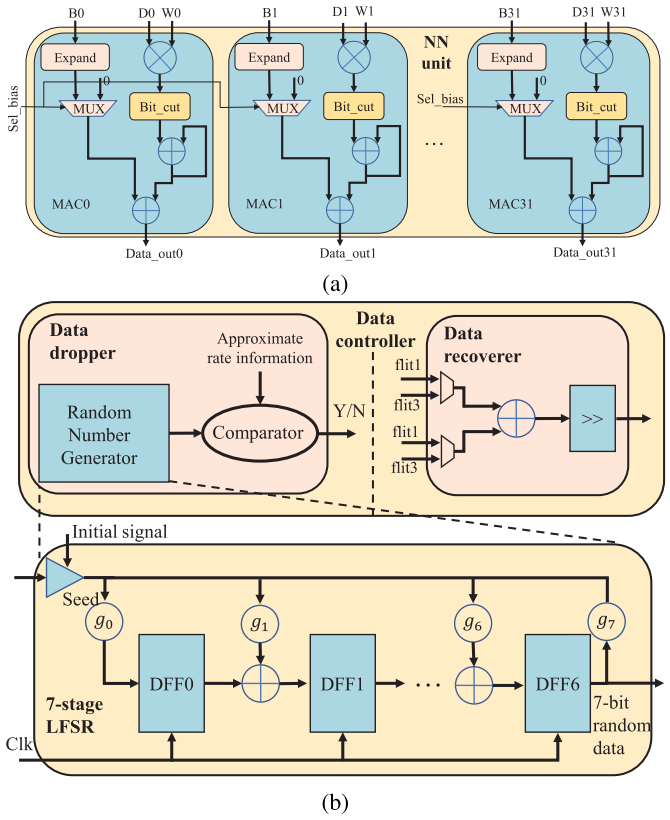


Fig. 9. Hardware implementation. (a) is the policy unit in the global approximate controller. (b) is the local data controller.

V. Experimental Results

In this section, we first introduce our experiment settings. Then we analyze the experiment results.

A. Experiment Settings

To evaluate the proposed HAS-RL, the AccessNoxim [39] simulation platform is used to evaluate the $4 \times 4 \times 4$ NoC system. The activation (16 bits) and the neuron index (16 bits) determine the first body flit size (32 bits) and two activations (32 bits) determine the rest of the body flits size. To simplify the problem without loss of generality, we set the virtual channel to 1. In the case of multiple virtual channels, we can average the utilization of different virtual channels in the local direction and then classify them into different levels. Table I shows the simulation parameter configurations and NoC parameter configurations. The RL agent takes the approximate rate adjustment decision every 10,000 cycles. We generate offline datasets using AccessNoxim to train the policy network, and offline RL configuration parameters are shown in Table II. The discount factor is set to 0.99. The batch size is set to 16. The Q target network is updated every 240 steps.

We map three different neural networks (AlexNet [33], VGG16 [34], ResNet [40]) to the NoC-based NN accelerator to verify its practicability. To reflect the influence of mappings, which will bring different traffic patterns to the HAS-RL, we generate several different mappings randomly. The proposed scheme is compared with the state-of-the-art approximate schemes including ABDTR [13], ACDC [15] and SAPA [17]. In different neural networks, we all set the maximum quality loss as 4% accuracy loss and set the f and i to be 20 and 1 respectively ($f\%$ is the maximum allowable data approximate rate per packet, $i\%$ is the stride by one time).

TABLE I
Configurations Used in the AccessNoxim

Simulation Parameters			
Simulation cycles	310 000	Warmup cycles	10 000
NoC Parameters			
Core number	$4 \times 4 \times 4$	NoC flit size	32-bit
Packet size	12 flits	NoC buffer depth	7×8 flits
Routing algorithm	XYZ routing	Virtual channel	1
Router delay	2 cycles	Link delay	1 cycles

TABLE II
Configurations Used in the Offline RL

Parameters	Settings	Parameters	Settings
Algorithm	DQN	Discount factor	0.99
Learning rate	0.0005	Training steps	40 000 steps
Replay capacity	10 000 steps	Iteration number	1000 times
Batch size	16	Target update period	240 steps

B. Experiment Results

1) Train Agent: We can use the ξ_1 and ξ_2 to adjust the optimization goal. It depends on which one you consider more. In this experiment, we set $\xi_1 = 2$ and $\xi_2 = 3$ to train the policy network. We try different policy network sizes to evaluate their influence on optimization results. A large network will occupy too many cycles to generate the required actions in the next state, which cannot meet the real-time requirement, whereas a small network may result in non-convergence. Therefore, we finally adopted a fully connected neural network with two hidden layers, and the number of neurons in each hidden layer is 128 and 32 respectively. Due to the small scale of the network, the training is very fast, and we terminate the training after the convergence has lasted a period. It is worth noting that we use the same size of the policy network for different neural network applications, and we evaluate different applications without changing the hardware implementations of HAS-RL, just updating the parameters of the policy network.

2) Performance and Quality Analysis: As shown in Fig. 4, the average delay of NoC is strongly positively correlated with the free buffer slot variance under the same traffic load. We can get the free buffer slot variance at runtime under a certain mapping of VGG16 and AlexNet, which is shown in Fig. 10a and Fig. 10b respectively. The free buffer slot variance of the scheme without approximate communication is much larger than the other four schemes with approximate communication. As simulation time elapses, the variance of HAS-RL decreases significantly and tends to be stable, while the variance of the other three schemes is larger than HAS-RL. Therefore, HAS-RL can alleviate congestion more effectively by adjusting the data approximate rate.

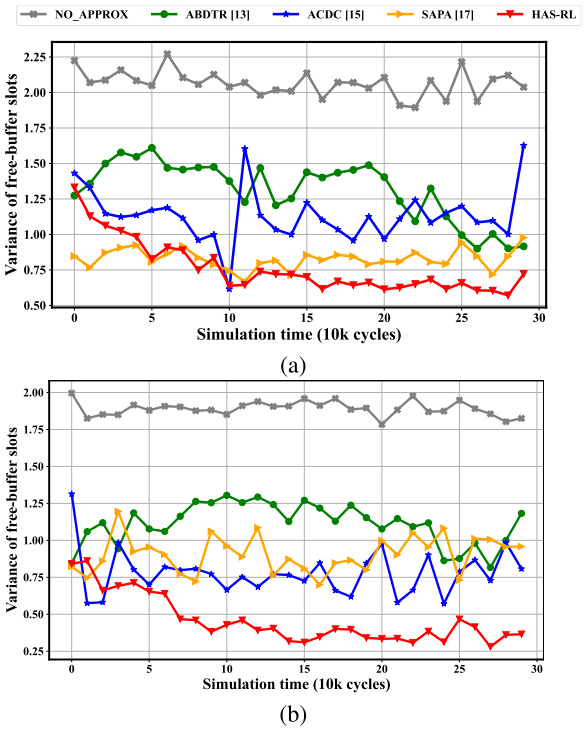


Fig. 10. Variances of free slots of buffers change with simulation time under different approximate schemes. (a) is VGG16 and (b) is AlexNet.

Then we compare HAS-RL with the scheme without approximate communication (NO_APPROX), ABDTR, ACDC and SAPA in terms of the average delay, quality loss, total energy consumption, and throughput. We normalized based on the average delay, total energy consumption, and throughput of NO_APPROX, and we also normalized the error threshold when comparing quality loss. To ensure that the quality loss is in an acceptable range, the sensitivity of congestion sensing in ABDTR is adjusted so that it does not exceed the error threshold at runtime. As shown in Fig. 11a, HAS-RL reduces the average delay by 23.95% to 48.1% (37.72% on average), 6.9% to 36.8% (19.59% on average), 1.1% to 15.4% (9.3% on average) and 2.2% to 21.5% (13.5% on average) compared with NO_APPROX, ABDTR, ACDC, and SAPA, respectively. As shown in Fig. 11b, the quality loss of HAS-RL and SAPA is less than 85% of the error threshold on average, while that of ACDC is very close to the error threshold, and that of ABDTR is about 65% of the error threshold on average. The output accuracy is decreased by 0.64% to 2.81% (1.68% on average), 0.04% to 0.82% (0.4% on average) compared with NO_APPROX and ABDTR, and improved by 0.1% to 2% (0.5% on average) compared with ACDC. And the output accuracy of SAPA is similar to HAS-RL on average. As shown in Fig. 11c, the total energy consumption of ACDC is the lowest due to the minimum number of flits transmitted. Compared with NO_APPROX, ABDTR, and SAPA, HAS-RL reduces energy consumption by about 12.05%, 2.9%, and 1.3% on average. As shown in Fig. 11d, HAS-RL improves the throughput by 12.5% to 27.8% (20.23% on average), 1.8% to 13% (7.18% on average), 0.8% to 6% (3.5% on average), and 0.8% to 9.2% (4.8% on average) compared with NO_APPROX, ABDTR, ACDC, and SAPA, respectively.

TABLE III
HARDWARE OVERHEAD (1GHZ@28NM)

Original router	Area(um^2)	28925.8	Power(mW)	9.84

Submodules of approximate controller		HAS-RL	ABDTR [13]	ACDC [15]	SAPA [17]
Local data controller	Area(um^2)	242.42	344.9	378.16	579.24
	Power(mW)	0.06	1.0	0.36	0.18
Global approximate controller	Area(um^2)	7439.29	—†	—*	—†
	Power(mW)	1.01	—†	—*	—†
Equivalent shared global$ controller of each node	Area(um^2)	116.24	—†	—*	—†
	Power(mW)	0.016	—†	—*	—†
Equivalent additional # overhead of each node	Area(um^2)	358.66	344.9	378.16	579.24
	Power(mW)	0.076	1.0	0.36	0.18

† There is no a global approximate controller in ABDTR and SAPA.
* ACDC uses software to implement a global approximate controller.
$ Dividing the area and power of the global approximate controller by the number of nodes.
# The sum of the equivalent shared global approximate controller of each node and the local data controller.

ABDTR can easily lead to underutilization or overutilization. The experiment shows that its quality loss is far below the error threshold and the average delay is much higher than the other two methods. ACDC makes full use of the error threshold to drop the flits, so its quality loss is very close to the error threshold. However, it will ignore other congested links in the NoC once the traffic load in the route is large but does not exceed the link capacity. Therefore, it does not achieve the purpose of minimizing congestion. SAPA approximates data through truncation. For low-bit quantization neural networks, the benefits of truncation are limited. Meanwhile, because of the different truncation bits for different data, it is necessary to insert marks to distinguish the different data, which will limit the shortening of the packet length.

Although it is acceptable that the quality loss is less than the error threshold, the increase in the data approximate rate still leads to a decrease in the output quality. Therefore, we need to maximize performance with as little quality loss as possible. In this paper, the proposed HAS-RL can monitor global congestion well and achieve a better balance between average delay and quality by using reinforcement learning algorithms.

3) Hardware Overhead: We implemented the proposed approximate controller using Verilog and then evaluated the area and power consumption using Design Compiler under CMOS 28nm technology. As shown in Table III, in the global controller, 32 sets of 4-bit fixed-point MAC units are used to accelerate the computation of the policy network, with an area of $7439.29um^2$. Because only the first 400 cycles are used for computation in every 10,000 cycles, the average power consumption is only $0.016mW$. For each local data controller, it consists of the data dropper and the data recoverer, whose area and power consumption are 242.42 um^2 and 0.06 mW respectively.

Because the global controller in the proposed HAS-RL is shared by all nodes, the equivalent additional overhead of each node is the sum of the overhead of the equivalent shared global controller of each node and the overhead of the local data controller. And the overhead of each node's

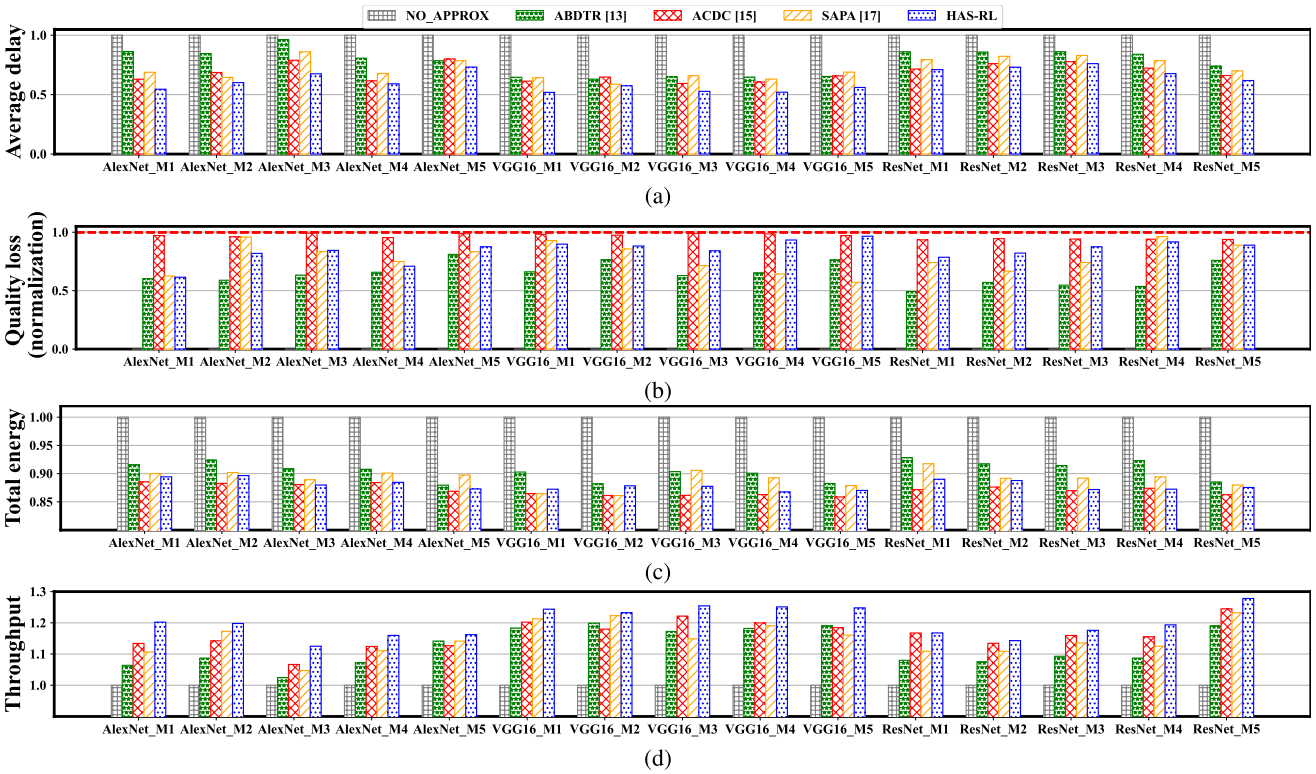


Fig. 11. Comparisons of different mappings in different neural networks under different approximate schemes. (a) is the average delay. (b) is quality loss. (c) is total energy. (d) is throughput.

equivalent shared global controller is equal to the overhead of the global controller divided by the number of nodes. The HAS-RL only causes an additional area overhead of 1.24% and power consumption of 0.77% compared with the traditional router design. Compared with ABDTR, the area overhead of HAS-RL is increased by 4%, but the power consumption is reduced by 92.4%. Because the global controller of HAS-RL only works for a short time, the power consumption is much lower than ABDTR. Compared with ACDC and SAPA, the area overhead of HAS-RL is reduced by 5.2% and 38.1%, and the power consumption is reduced by 78.9% and 57.7%, respectively.

VI. Conclusion

In this paper, to balance performance and output accuracy well, we propose a hierarchical approximate scheme optimized with reinforcement learning (HAS-RL), consisting of one global controller and n local data controllers. We first get the quality model under different neural networks. Then, we reduce the state space by using the number of free local slots to monitor the congestion in the network and reduce the action space by categorizing nodes with different congestion conditions. Finally, the policy network is obtained by the offline RL algorithm and deployed to the global controller. Compared with the state-of-the-art method, the proposed scheme reduces the average network delay by 13.5% while their accuracies are similar. The proposed HAS-RL results in only 1.24% additional area overhead and 0.77% power consumption over the traditional router design.

References

[1] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Des. Test*, vol. 33, no. 1, pp. 8–22, Feb. 2016.

[2] R. Ye, T. Wang, F. Yuan, R. Kumar, and Q. Xu, "On reconfiguration-oriented approximate adder design and its application," in *Proc. ICCAD*, 2013, pp. 48–54.

[3] M. Ha and S. Lee, "Multipliers with approximate 4–2 compressors and error recovery modules," *IEEE Embedded Syst. Lett.*, vol. 10, no. 1, pp. 6–9, Mar. 2018.

[4] Y. Wang, J. Dong, Y. Liu, C. Wang, and G. Qu, "A machine learning based approximate computing approach on data flow graphs: Work-in-progress," in *Proc. EMSOFT*, 2020, pp. 37–39.

[5] M. Ramakrishna, P. V. Gratz, and A. Sprintson, "GCA: Global congestion awareness for load balance in networks-on-chip," in *Proc. 7th IEEE/ACM Int. Symp. Networks-Chip (NoCS)*, Apr. 2013, pp. 1–8.

[6] S. Xu, J. Wu, B. Fu, M. Chen, and L. Zhang, "Efficient regional congestion awareness (ERCA) for load balance with aggregated congestion information," in *Proc. 25th Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. (PDP)*, Mar. 2017, pp. 93–99.

[7] J. Balfour and W. J. Dally, "Design tradeoffs for tiled CMP on-chip networks," in *Proc. ICS*. New York, NY, USA: Association for Computing Machinery, 2006, pp. 390–401.

[8] M. A. Khan and A. Q. Ansari, "Quadrant-based XYZ dimension order routing algorithm for 3-D asymmetric torus routing chip (ATRC)," in *Proc. Int. Conf. Emerg. Trends Netw. Comput. Commun. (ETNCC)*, Apr. 2011, pp. 121–124.

[9] A. Bose, P. Ghosal, and S. P. Mohanty, "A low latency scalable 3D NoC using BFT topology with table based uniform routing," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, Jul. 2014, pp. 136–141.

[10] M. Ebrahimi, X. Chang, M. Daneshtalab, J. Plosila, P. Liljeberg, and H. Tenhunen, "DyXYZ: Fully adaptive routing algorithm for 3D NoCs," in *Proc. 21st Euromicro Int. Conf. Parallel, Distrib., Netw.-Based Process.*, Feb. 2013, pp. 499–503.

[11] Y. Chen and A. Louri, "An approximate communication framework for network-on-chips," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1434–1446, Jun. 2020.

[12] R. Boyapati, J. Huang, P. Majumder, K. H. Yum, and E. J. Kim, "APPROX-NoC: A data approximation framework for network-on-chip architectures," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, Jun. 2017, pp. 666–677.
[13] L. Wang, X. Wang, and Y. Wang, "ABDTR: Approximation-based dynamic traffic regulation for networks-on-chip systems," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Nov. 2017, pp. 153–160.
[14] S. Xiao, X. Wang, M. Palesi, A. K. Singh, and T. Mak, "ACDC: An accuracy- and congestion-aware dynamic traffic control method for networks-on-chip," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 630–633.
[15] S. Xiao, X. Wang, M. Palesi, A. K. Singh, L. Wang, and T. Mak, "On performance optimization and quality control for approximate-communication-enabled networks-on-chip," *IEEE Trans. Comput.*, vol. 70, no. 11, pp. 1817–1830, Nov. 2021.
[16] J. R. Stevens, A. Ranjan, and A. Raghunathan, "AxBA: An approximate bus architecture framework," in *Proc. ICCAD*, 2018, pp. 1–8.
[17] Y. Chen, A. Louri, S. Liu, and F. Lombardi, "Slack-aware packet approximation for energy-efficient network-on-chips," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 1, pp. 120–132, Jan. 2023.
[18] Y. Chen, M. F. Reza, and A. Louri, "DEC-NoC: An approximate framework based on dynamic error control with applications to energy-efficient NoCs," in *Proc. ICCD*, 2018, pp. 480–487.
[19] M. F. Reza and P. Ampadu, "Approximate communication strategies for energy-efficient and high performance NoC: Opportunities and challenges," in *Proc. Great Lakes Symp. VLSI*, New York, NY, USA, May 2019, pp. 399–404.
[20] J. A. Bagnell and J. G. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," in *Proc. ICRA*, vol. 2, 2001, pp. 1615–1620.
[21] R. Agarwal, D. Schuurmans, and M. Norouzi, "An optimistic perspective on offline reinforcement learning," 2019, *arXiv:1907.04543*.
[22] H. Yu, S. Park, A. Bayen, S. Moura, and M. Krstic, "Reinforcement learning versus PDE backstepping and PI control for congested freeway traffic," *IEEE Trans. Control Syst. Technol.*, vol. 30, no. 4, pp. 1595–1611, Jul. 2022.
[23] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," 2020, *arXiv:2005.01643*.
[24] L. Wang, X. Wang, and Y. Wang, "An approximate bufferless network-on-chip," *IEEE Access*, vol. 7, pp. 141516–141532, 2019.
[25] A. B. Ahmed, D. Fujiki, H. Matsutani, M. Koibuchi, and H. Amano, "AxNoC: Low-power approximate network-on-chips using critical-path isolation," in *Proc. 12th IEEE/ACM Int. Symp. Netw.-Chip (NOCS)*, Turin, Italy, Oct. 2018, pp. 1–8.
[26] K.-C.-J. Chen and Y.-H. Liao, "Adaptive machine learning-based temperature prediction scheme for thermal-aware NoC system," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, p. 14.
[27] M. F. Reza, T. T. Le, B. De, M. Bayoumi, and D. Zhao, "Neuro-NoC: Energy optimization in heterogeneous many-core NoC using neural networks in dark silicon era," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5.
[28] N. Rao, A. Ramachandran, and A. Shah, "MLNoC: A machine learning based approach to NoC design," in *Proc. 30th Int. Symp. Comput. Archit. High Perform. Comput. (SBAC-PAD)*, Sep. 2018, pp. 1–8.
[29] M. Rapp, A. Pathania, T. Mitra, and J. Henkel, "Neural network-based performance prediction for task migration on S-NUCA many-cores," *IEEE Trans. Comput.*, vol. 70, no. 10, pp. 1691–1704, Oct. 2021.
[30] Z. Chen and D. Marculescu, "Distributed reinforcement learning for power limited many-core system performance optimization," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2015, pp. 1521–1526.
[31] M. F. Reza, "Reinforcement learning based dynamic link configuration for energy-efficient NoC," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2020, pp. 468–473.
[32] Y. Chen and A. Louri, "Learning-based quality management for approximate communication in network-on-chips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 11, pp. 3724–3735, Nov. 2020.
[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 2, 2012, pp. 1097–1105.
[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[35] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 1–13.
[36] P. Bhamidipati and A. Karanth, "HREN: A hybrid reliable and energy-efficient network-on-chip architecture," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 537–548, Apr. 2022.
[37] R. Agarwal, D. Schuurmans, and M. Norouzi, "Striving for simplicity in off-policy deep reinforcement learning," 2019, *arXiv:1907.04543*.
[38] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2052–2062.
[39] K.-Y. Jheng, C.-H. Chao, H.-Y. Wang, and A.-Y. Wu, "Traffic-thermal mutual-coupling co-simulation platform for threedimensional network-on-chip," in *Proc. VLSI-DAT*, 2010, pp. 135–138.
[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

Siyue Li received the B.E. degree in microelectronics science and engineering from Yangzhou University, Yangzhou, China, in 2021. He is currently pursuing the M.E. degree with the School of Electronic Science and Engineering, Nanjing University.

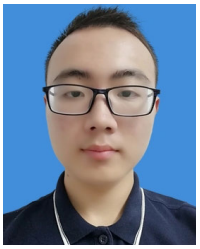
His current research interests include network-on-chip mapping optimization, neural networks, and AI for chip architecture design.

Shize Zhou received the B.E. degree from the School of Electronic Science and Engineering, Nanjing University, Nanjing, China, in 2020, where he is currently pursuing the M.E. degree with the School of Electronic Science and Engineering.

His current research interests include network-on-chip architecture design, neural network accelerators, and approximate communication.

Yongqi Xue received the B.E. degree in electronic and information engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2021. She is currently pursuing the M.E. degree with the School of Electronic Science and Engineering, Nanjing University, China.

Her current research interests include network-on-chip mapping optimization, neural networks, and AI for chip architecture design.



Wenjie Fan received the B.E. degree from the School of Electronic Science and Engineering, Nanjing University, Nanjing, China, in 2022, where he is currently pursuing the M.E. degree with the School of Electronic Science and Engineering.

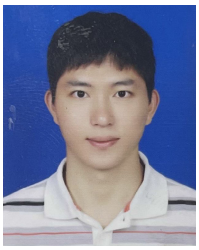
His current research interests include network-on-chip (NoC) and neural network accelerators.

Tong Cheng received the B.E. degree from the School of Electronic Science and Engineering, Nanjing University, Nanjing, China, in 2022, where he is currently pursuing the M.E. degree with the School of Integrated Circuit Engineering.

His research interests include network-on-chip design and thermal management for NoC-based multicore systems.

Jinlun Ji received the B.S. degree in physics from Nanjing University, Nanjing, China, in 2022, where he is currently pursuing the M.E. degree with the School of Electronic Science and Engineering.

His current research interests include AI for chip architecture design and network-on-chip mapping algorithms/architectures.



Chenyang Dai received the B.E. degree from the School of Electronic Information, Soochow University, Suzhou, China, in 2022. He is currently pursuing the M.E. degree with the School of Electronic Science and Engineering, Nanjing University.

His current research interests include network-on-chip (NoC) and neural network accelerators.

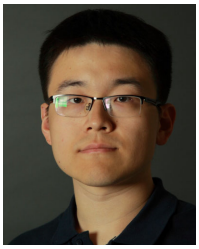
Wenqing Song received the B.S. degree from the School of Electronic Science and Engineering, Southeast University (SEU), Nanjing, China, in 2017. She is currently pursuing the Ph.D. degree in electronic science and engineering with Nanjing University (NJU), Nanjing.

Her current research interests include polar coding algorithms, homomorphic encryption, and efficient reconfigurable architecture.

Qinyu Chen (Member, IEEE) received the Ph.D. degree in electronic science and technology from Nanjing University, Nanjing, China, in June 2021.

In February 2024, She joined the Leiden Institute of Advanced Computer Science, Leiden University, Leiden, the Netherlands, as an Assistant Professor, before that, she was a Postdoc at the Institute of Neuroinformatics, University of Zürich and ETH Zürich, Zurich, Switzerland. In 2022, she received a Bridge Fellowship Grant from the Swiss National Science Foundation (SNSF) and Innosuisse. Her current research interests include the seamless neuromorphic artificial intelligence system at the edge, and its application in healthcare, AR/VR with a focus on event-based processing.

Dr. Chen serves as a member of the Neural Systems and Applications (NSA) Technical Committee in the IEEE Circuit and System Society (CASS).



Chang Gao (Member, IEEE) received the joint Ph.D. degree (Hons.) in neuroscience from the Institute of Neuroinformatics, University of Zurich, and ETH Zurich, Zürich, Switzerland, in March 2022.

In August 2022, he joined the Department of Microelectronics, Delft University of Technology, The Netherlands, as an Assistant Professor. His current research interests include computer architectures for deep learning, with an emphasis on recurrent neural networks. He received the 2022 Misha Mahowald Early Career Award in Neuromorphic Engineering, the 2022 Marie-Curie Post-Doctoral Fellowship, and the Title of 2023 MIT Technology Review Innovators Under 35 in Europe.

Li Li (Member, IEEE) received the B.S. and Ph.D. degrees from the Hefei University of Technology, Hefei, China, in 1996 and 2002, respectively.

She is currently a Professor with the School of Electronic Science and Engineering, VLSI Design Institute, Nanjing University, Nanjing, China. Her current research interests include VLSI design for digital signal processing systems reconfigurable computing and multiprocessor system-on-a-chip (MPSoC) architecture design methodology. She is a member of Circuits & Systems for Communications (CASCOM) TC of the IEEE CAS Society.

Yuxiang Fu (Member, IEEE) received the B.S. degree in microelectronics and solid-state electronics and the Ph.D. degree in electronic science and technology from Nanjing University, Nanjing, China, in 2013 and 2018, respectively.

In 2018, he joined the School of Electronic Science and Engineering, Nanjing University, where he is currently an Assistant Professor with the School of Integrated Circuits. His current research interests include AI for chip architecture design, network-on-chip algorithms/architectures, low-power digital systems, and 3D IC design.