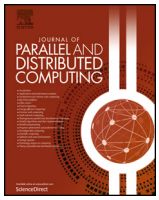
Journal of Parallel and Distributed Computing 186 (2024) 104778

Contents lists available at ScienceDirect

Journal of Parallel and Distributed Computing

journal homepage: www.elsevier.com/locate/jpdc



A survey of machine learning for Network-on-Chips

Xiaoyun Zhang, Dezun Dong *, Cunlu Li, Shaocong Wang, Liquan Xiao

College of Computer, National University of Defense Technology, Changsha, 410073, Hunan, China

ARTICLE INFO

Keywords:
Machine learning
Network-on-Chips
NoCs design optimization

ABSTRACT

The popularity of Machine Learning (ML) has extended to numerous disciplines, including the domain of Network-on-chips (NoCs), leading to a consequential impact. Recent works have explored ML models' applicability for NoCs design, optimization, and performance evaluation. ML-based NoCs design has demonstrated superior performance to heuristic methods employed by human experts in NoCs design. This has facilitated a tight collaboration between ML and NoCs research, offering novel perspectives and optimization strategies to advance NoCs design. In this paper, we present a comprehensive survey into implementing ML techniques for NoCs. Initially, we provide an overview of ML-based research for NoCs in two aspects: (i) the adoption of ML for performance modeling and prediction and (ii) ML-based for NoCs design, including individual components (such as routing algorithm, arbitration, traffic control, etc.). Subsequently, we summarize the challenges and difficulties in designing NoCs for applying ML techniques and discuss the preliminary solutions to these issues. Finally, we prospect the perspective on future research directions for expanding the application of ML techniques to diverse scenarios of NoCs, exploring the adoption of ML techniques for NoCs design automation. We expect this paper can be helpful for design experts to optimize NoCs using ML techniques, leading to high-performance, energy-efficient, and easy-to-implement NoCs.

1. Introduction

Machine learning (ML) endows computers with the capability to learn from a large amount of datasets [107], which has gained widespread popularity across various fields, typified by computer vision [18] and natural language processing [16], etc. In each domain, ML models are employed to uncover patterns and correlations inherent in the dataset, enabling the generation of predictions or informed decision-making. The surging demand for **Artificial Intelligence (AI)** has spurred the development of increasingly sophisticated ML models, leading to an exponential surge in the computational power necessary to train ML models [148]. In parallel, continuous advancements in chip manufacturing processes have made it possible to integrate billions of transistors onto a single chip, thus enabling the integration of numerous cores. This integration facilitates communication among the multiple cores through **Network-on-Chips (NoCs)** and boosts computing power, thereby meeting the rigorous demands required for training ML models [135]. Therefore, despite the potential benefits, NoCs design and innovation still face great challenges.

Previous NoCs design predominantly relied on designers' professional intuition and practical expertise in determining essential design aspects such as topology, routing decisions, etc. However, heuristic-based design methods cannot always achieve scalability and optimality, particularly when the constraints and demands of the NoCs are diverse. The complex interplay between components in a NoCs presents a significant challenge in designing nearly optimal, high-performance, and energy-efficient NoCs that can meet the demands of multiple applications. Recently, researchers have displayed a growing interest in utilizing ML techniques for NoCs, providing profound insights, discovering more efficient optimization methodologies, and enabling better decision-making processes. Existing research has shown that employing ML techniques in NoCs can explore the trade-off between performance and energy consumption. Furthermore, some previous works have explored the application of ML to improve NoCs design, including **dynamic voltage and frequency scaling (DVFS)** management [51,160], routing algorithm [48,138], and arbitration [152,154], etc. In short, ML techniques enable data-driven NoCs optimization and provide a new design framework for exploring the NoCs design space. This framework

* Corresponding author.
E-mail address: dong@nudt.edu.cn (D. Dong).

https://doi.org/10.1016/j.jpdc.2023.104778
Received 18 April 2023; Received in revised form 24 July 2023; Accepted 18 September 2023
Available online 20 November 2023

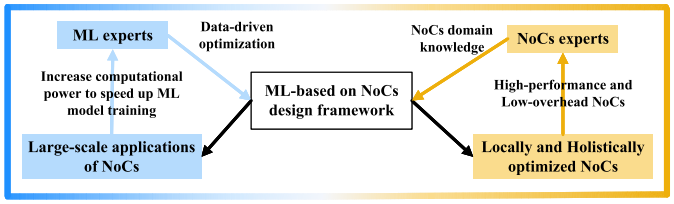


Fig. 1. The tight collaboration between the ML and NoCs domains has facilitated the use of ML techniques to optimize NoCs design. Simultaneously, this synergy has improved computational power instrumental in training ML models.

allows designers to optimize the design of some components of the NoCs. NoCs designers can gain insight by understanding the principles behind ML design decisions. By significantly reducing the workload and the cost of designing a NoCs, this framework can be commercialized for large-scale applications of NoCs. Simultaneously, in the ML field, this framework enables data-driven optimization for NoC systems to meet the computing power required to train ML models. Therefore, there is an interdependence between ML and NoCs design, as shown in Fig. 1.

In this paper, we review the related works of ML-based for NoCs from two aspects. **(1) Performance modeling and prediction.** The utilization of ML techniques in NoCs performance modeling and prediction plays a key role in ensuring the accurate evaluation of the NoCs system throughout the design process. NoCs performance predictions are traditionally evaluated using cycle-accurate simulators with theory-based methods. As the intricacy of the NoCs design space grows, the conventional modeling techniques and theories for performance evaluation become intricate and always entail high computational costs. ML-based approaches, in contrast, can balance simulation cost overhead and evaluation accuracy, holding immense promise for exploring the NoCs configuration space, among other applications [107]. **(2) Optimization of NoCs design.** ML techniques have been increasingly applied to optimize critical components in NoCs, including link management, routing algorithms, arbitration, traffic control, etc. ML can automatically extract essential features from datasets, improving the accuracy of training ML models based on the learning experience. In general, ML serves as an instrument for NoCs design to proactively and intelligently explore the design space and adopt different strategies for dynamic workloads, providing effective optimization solutions. Furthermore, we also discuss the common problems in applying ML for NoCs, analyze the potential directions of extending ML techniques to different scenarios of NoCs, and anticipate the prospect of employing ML techniques to automate the NoCs design process. We expect this paper can be helpful for architects to use ML to optimize NoCs, resulting in high-performance, energy-efficient, and easy-to-implement NoCs.

The rest of this paper is organized as follows. Section 2 provides relevant background knowledge on NoCs and ML techniques. Section 3 presents the existing work applying ML to NoCs performance modeling and prediction. Section 4 summarizes related works optimizing key components or addressing critical problems in NoCs using ML, while the ML models used in these efforts are also analyzed. Section 5 discusses issues for improvement in existing works while applying ML for NoCs and makes an outlook on future work, aiming to convey insights for consideration in designing NoCs. Finally, Section 6 concludes this paper.

2. Background

NoCs are a solution for inter-core communication within a chip, and it is also the essential component of the contact between various nodes. The design of a NoCs significantly impacts the overall performance of a multi-core system. In this section, we provide a brief introduction to NoCs and the ML techniques frequently employed in NoCs. Typical ML models include supervised learning, unsupervised learning, and **reinforcement learning (RL)**. Consequently, designers should determine a suited ML model to optimize the design of NoCs, based on the available dataset and the specific objective function.

2.1. NoCs

NoCs are fundamental to interconnecting the cores on a chip in high-performance computer systems, which act as the backbone of multi-core/crowd-core systems by linking multiple cores to form a robust communication subsystem. NoCs facilitate expedited and effective packet transfer between computing resources (computing cores and memory units), enabling optimal system performance. Notably, the evaluation of NoCs performance includes but is not limited to latency and throughput. For example, the design of NoCs necessitates considering area and power consumption factors to ensure it is viable and sustainable in real applications. Therefore, striking a balance between the hardware overhead and network performance in NoCs continues to pose a significant challenge.

A typical NoCs-based multi-core system has been presented in Fig. 2. As shown in Fig. 2, a 4×4 mesh interconnects 16 cores, where each node in the network houses a computing core and a router. Computing cores may comprise other hardware modules, such as accelerators or storage controllers, while routers primarily include caches, crossbar switches, allocators, and routing computation units. Routers can be connected by links to form a NoCs. From the perspective of NoCs design, the primary research includes optimization of critical components such as topology, routing algorithm, flow control mechanisms, and router microarchitecture. Topology forms the fundamental structure of NoCs, which determines the specific connection relationship between each network node and link, the number of paths between nodes, and the length of the links in each path. A range of classical network topologies, namely ring, mesh, cmesh, torus, and butterfly topology [77], etc., are commonly employed in NoCs. The primary task of a routing algorithm is to determine the efficient route for packets to travel from the source to the destination router while ensuring network load balancing. Routing algorithms can be classified into two categories: oblivious routing algorithms [20,124,131] and adaptive routing algorithms [53,55,94,98,112]. Flow control mechanisms mainly determine how to allocate network resources and the granularity of the allocated resources. Store-and-forward flow control [26], virtual cut-through flow control [74], and wormhole flow control mechanism [25] are the typical flow control mechanisms. The router microarchitecture is associated with the hardware implementation of NoCs components, such as routing algorithms, flow control mechanisms, etc., but also optimized to satisfy constraints such as the area and power overhead of the NoCs.

In addition to electrical NoCs, some special NoCs have emerged in recent years, including 3D NoCs, Wireless NoCs, and Optical NoCs, as shown in Fig. 3. To overcome the challenges of shrinking the area and improving the network performance of 2D NoCs, 3D NoCs utilize vertical stacking technology to achieve higher integration and better performance [30]. In addition to vertical stacking technology, 3D NoCs also use various fabrication techniques, including silicon perforation technology, silicon inter-layer interconnect, and push-stack packaging technology, making it more flexible and adaptable to various application requirements [9]. While electrical NoCs are a commonly investigated communication technology, researchers are exploring innovative methods to develop novel NoCs. Wireless NoCs leverage wireless communication technologies to establish interconnections between diverse functional modules and processors within the chip, facilitating efficient packet transfer and communication [2]. Optical NoCs utilize optical signals instead of electrical signals for communication and packet transmission and have the advantages of high speed, low power consumption, and high integration [144]. Electrical NoCs and optical NoCs have similarities in topology and routing strategies, and both can use a similar topology for wired connections. However, wireless NoCs always use one-hop routing, while electrical NoCs and optical NoCs use one or

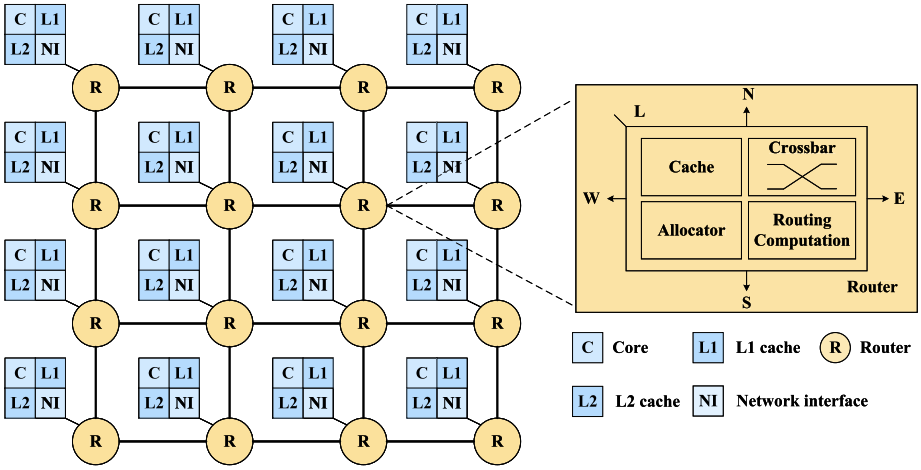


Fig. 2. A multi-core system with NoCs interconnection.

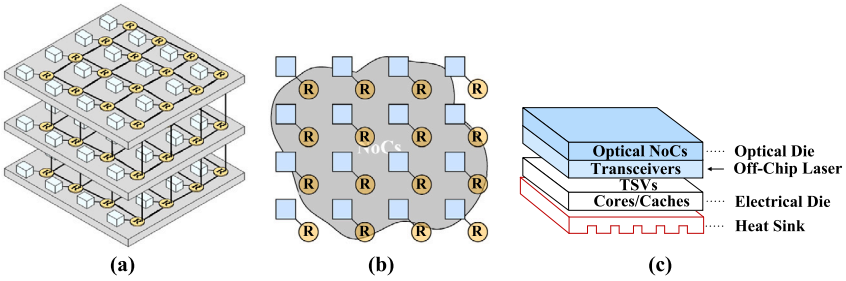


Fig. 3. Emerging NoCs have (a) 3D NoCs, (b) Wireless NoCs, and (c) 3D-integrated chips with optical components [103].

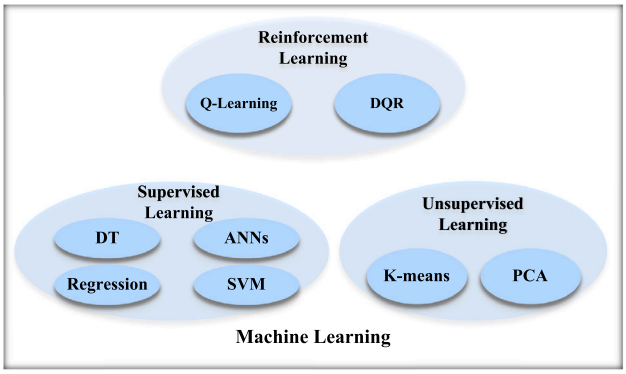


Fig. 4. Classification of ML techniques.

more hops for packet transmission, depending on the distance between source and destination routers. It should be noted that, if not specifically stated, the term NoCs refers to the 2D electrical NoCs in this paper.

2.2. ML techniques

ML is a fundamental branch of AI. The progress in ML research is manifested through its continued development across various domains, such as computer vision [18] and natural language processing [16], etc. ML can be segregated into three general frameworks: supervised learning, unsupervised learning, and RL, as depicted in Fig. 4. These ML models mainly distinguish what datasets are sampled and how these datasets are used to build ML models. In the following sections, we focus on ML models applied to NoCs. When selecting an appropriate ML model for a given practical problem, several factors should be considered, including hardware implementation overhead, available datasets, and the objective function.

2.2.1. Supervised learning

Supervised learning is an ML technique that aims to learn the mapping function between input and output data. The training of supervised learning models to predict unknown datasets is facilitated through the utilization of labeled datasets, which contain input-output pairs that have been marked beforehand. We briefly introduce several commonly used supervised learning techniques in NoCs, including Regression, **Support Vector Machines (SVM)**, **Decision Trees (DT)**, and **Artificial Neural Networks (ANNs)**, as shown in Fig. 5.

(1) Regression [90] aims to analyze the relationship between two or more variables. The widely used linear regression [123] is the basic form of regression with only one independent variable and one dependent variable. This model builds the connection between the independent and dependent variables, enabling predictions of the dependent variable's value and inferences about the causal relationship.

(2) SVM aims to separate datasets into different classes by finding a hyperplane in the dataset space [122]. However, several hyperplanes can be used to separate datasets, and the optimal hyperplane can be obtained by finding the hyperplane with the maximum margin. SVM can perform the input dataset's prediction, classification, and regression tasks. **Support Vector Regression (SVR)** is a variant of SVM that is useful for handling regression tasks [93], and it works by finding the relative position of the dataset to the hyperplane.

(3) DT [4,5,111,133] is a tree structure model to construct regression or classification models, similar to humans' decision-making mechanism. The outcome of this model training process is a decision tree. The decision tree usually comprises a root node, numerous internal nodes, and several leaf nodes. The decision nodes, consisting of the root and internal nodes, represent data features within a given dataset. Each decision node is associated with branches, which denotes a possible value for the respective feature. Every path from the root node to the leaf node represents a decision-determination sequence. The procedure of decision tree generation is a recursive process.

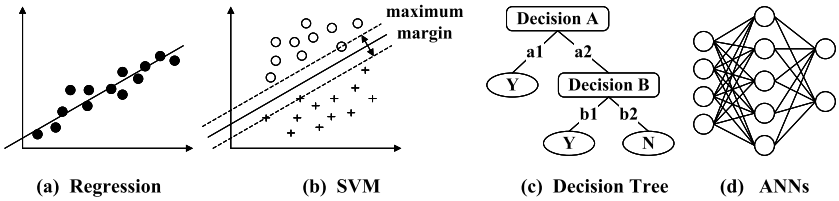


Fig. 5. Examples of supervised learning include (a) Regression, (b) SVM, (c) Decision Tree, and (d) ANNs.

(4) Neural networks, also called ANNs [118], are learning models that mimic biological neural networks' structure and function. They have been applied in solving diverse problems, such as speech and visual recognition. In neural networks, a fundamental structure is the single-layer neural network, commonly used to solve linear regression problems. It consists of a finite number of neurons, each taking a vector as input and producing a result value. The output of a single-layer neural network is also a vector, which can be used for various tasks such as classification and prediction. Multilayer neural networks can deal with nonlinear problems and are a network structure obtained by superimposing single-layer neural networks, hence the concept of layers. Different types of multilayer neural networks are designed for specific domains, such as **Convolutional Neural Networks (CNNs)** and **Recurrent Neural Networks (RNNs)**. CNNs utilize spatial features by sharing the same weights among neurons [56]. RNNs leverage temporal features to learn from sequences and histories [101].

Supervised learning models can exhibit different preferences for data features. For instance, Regression, SVM, and ANNs are proficient in handling data sets with multiple continuous features. On the other hand, DT models excel in dealing with datasets comprising discrete features. In NoCs, supervised learning is commonly used for performance modeling or predicting tasks. It is worth noting that supervised learning techniques need labeled training datasets before the model training phase, which typically requires substantial human expertise and engineering.

2.2.2. Unsupervised learning

Unsupervised learning is an ML technique that identifies patterns and relationships in unlabeled datasets. Unlike supervised learning, it does not require manual data labeling, making it suitable for analyzing large datasets. We mainly introduce two unsupervised learning models **Principal Components Analysis (PCA)** and Clustering, as shown in Fig. 6.

(1) PCA [39] is a statistical dimensionality reduction method that transforms multiple indicators into several principal components. PCA achieves reduced dimensionality of the high-dimensional variable space through the coordinate transformation of data statistics, thereby retaining the primary information of datasets. PCA transforms the original dataset into a set of linearly independent representations of each dimension by a linear transformation, which can be used to extract the main feature components of the data.

(2) Clustering involves dividing a dataset into disjoint subsets or clusters, which can be used to classify the original dataset [104]. Determining appropriate metrics is critical to the effectiveness of clustering methods, often based on performance metrics or distance measures. This technique helps large datasets and identify patterns without manual labeling. K-means clustering is a popular and straightforward clustering technique.

In practice, evaluating the performance of unsupervised learning models and assessing control effectiveness is challenging due to the absence of labeled datasets. To overcome this challenge, semi-supervised learning has emerged as a promising solution that combines supervised and unsupervised learning techniques [134]. This model is first trained on a limited labeled dataset in semi-supervised learning. This model result is then used to infer labels for the unlabeled dataset, which is subsequently incorporated into the original dataset to retrain the model

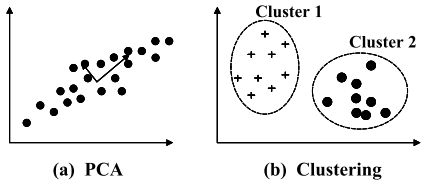


Fig. 6. Examples of unsupervised learning include (a) PCA and (b) Clustering.

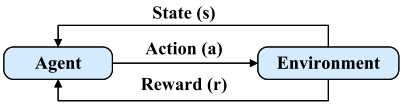


Fig. 7. A framework for RL.

in a cyclic process. This approach significantly reduces the need for manual data labeling, while this model's performance can be further improved with each cycle.

2.2.3. Reinforcement learning

RL is a type of ML that involves training an agent to maximize rewards or achieve specific goals by learning strategies through interaction with an environment. The RL framework comprises several components: agent, environment, state, action, and reward. In RL, the agent takes action and learns from the feedback received from the environment, which allows it to improve its decision-making capabilities [71]. RL is a dynamic approach to continuous training learning where an agent interacts with the environment through actions, states, and rewards, as depicted in Fig. 7. At each time step, the agent selects an action a based on the current state s, and the environment provides feedback as a reward r. The agent then updates its decision-making process based on the reward signal received and selects the next action a' based on the probability of receiving greater positive feedback. It should be noted that the agent's actions not only affect the current state but also influence the next state s' and the final decision. Therefore, the RL is a "trial and error" approach, in which the agent learns from their experiences with the environment to achieve a specific goal or maximize reward [71]. RL is distinguished from supervised and unsupervised learning methods in that it operates by continuously making decisions and receiving feedback from its environment rather than relying on direct feedback about the appropriateness or efficiency of its actions. Common RL techniques include Q-learning [143] and **Deep Q-Network (DQN)** [45], among others.

In Q-Learning, an agent tries to learn a function $Q(s,a)$, where s denotes the current state, a represents the action taken, and $Q(s,a)$ means the cumulative reward obtained by taking action a in state s. The update of the Q-value relies on the iterative process of the Bellman equation, which represents the relationship between the current state's Q-value and the next state's Q-value, as shown in the following equation.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma maxQ(s',a') - Q(s,a)) \quad (1)$$

where α is the learning rate, γ is the discount rate, and $maxQ(s',a')$ is the maximum Q-value for the next state. Q-table is used in Q-Learning

Table 1
Summary of applying ML models for performance modeling and prediction.

Problems	Specific Tasks	ML models
Performance Modeling and prediction	Construction of latency evaluation model	MCTS and RNN [61]
	Construction of an accurate cost model	Nonparametric Regression [66]
	Prediction of average channel wait time and traffic flow latency	SVR [109,110]
	An NoCs design space exploration methodology	Linear Regression [120]
	An efficient framework for accurate measurements of NoCs latency parameters	SVR and ANNs [83]
	Prediction of latency parameters for different traffic patterns	SVR [84]
	An evaluation framework to quickly obtain different performance metrics	Markov reward models [60]
	Prediction of NoCs architecture components and applications	Random Forest [126]
	Prediction of link utilization and link direction	DT [36]
	Prediction of buffer utilization	Ridge Linear Regression [22]
	Prediction of the number of packets injected into each router	Ridge Linear Regression [145]

to store Q-value. A neural network is used in DQN to approximate the Q function. Given a state s, the neural network can output an approximate Q-value action for each possible state. The RL process is a Markov decision-based process well-suited for solving control or decision problems. As such, RL can be naturally applied to the design problem of routing algorithms in NoCs. Specifically, the routing direction (either X-dimension or Y-dimension) is selected based on the network state dataset input. The reward function is then defined based on the latency or throughput to optimize network performance. RL can learn how to achieve optimal network performance in routing algorithm design by considering the interaction between the routing direction and network state.

2.2.4. Discussion

The underlying principles and algorithms of ML models determine different implementation complexity and computational overhead of different ML models. Supervised learning aims to learn a function from a labeled training dataset that maps input and output datasets. In contrast, unsupervised learning does not require labeled datasets and seeks to discover patterns and structures in a dataset, performing tasks such as clustering, dimensionality reduction, and anomaly detection. RL learns optimal behavior by trial and error to maximize rewards. However, the input data features' size and learning task goals must be considered when choosing a suitable ML model. Supervised learning models are usually used with labeled datasets, which helps perform known or specified datasets. Compared to unsupervised learning, supervised learning is less complex. Unsupervised learning is more appropriate when there is no explicit mapping between input and output datasets. The complexity of RL is intermediate between supervised and unsupervised learning. RL adaptations learn policies from experience or are deployed in unknown environments. Simultaneously, RL may apply to multi-objective optimization problems, where multiple optimization objectives need to be understood [132].

3. ML for NoCs performance modeling and prediction

In this section, we review research on accurate NoCs performance modeling and prediction using ML techniques. In NoCs, commonly used performance metrics are latency, energy consumption, etc. The NoCs performance is usually evaluated using a cycle-accurate simulator, and classical performance models are employed for analysis and statistics [76]. These models have certain constraints, e.g., packet length and traffic obey Poisson distribution [87]. In contrast, ML-based performance modeling and prediction approaches show great potential in NoCs. Existing work mainly uses supervised learning to explore the modeling and prediction of NoCs performance. Table 1 summarizes the application of ML techniques for performance modeling and prediction.

(1) Latency prediction. Qian et al., [109], argue that the key is to assume strict restrictions, try to relax these assumptions, and use an SVR model to predict channel average wait time and traffic flow latency. This paper demonstrates the accuracy of the performance modeling by using an independent test dataset [110]. Sangaiah et al. [120] proposed a regression model-based approach for design space exploration and performance prediction, which simulates the impact of memory hierarchies and NoCs on performance. This method of constructing a NoCs-based **Chip Multi-Processor (CMP)** model first determines the relevant memory and NoCs design configurations; then determines the interdependencies between the configurations; and finally, uses sampling techniques to generate the sample data set needed for the model. Kumar et al. [83] propose an effective ML framework to predict latency using SVR and ANNs models accurately. This approach considers topology size, virtual channels, traffic patterns, and injection rates as data features fed into the ML model. Kumar et al. [84] also proposed a latency prediction framework based on SVR. This framework mainly predicts the performance parameters of different traffic patterns in 2D and 3D network topologies: average network latency, average packet latency, average flit latency, and the average number of hops. Hu et al. [61] model NoCs design space exploration as a **Markov Decision Process (MDP)**, which uses **Monte Carlo Tree Search (MCTS)** as a search inspiration in terms of designing application-specific NoCs design. They propose an RNN-based latency evaluation model that can evaluate arbitrary NoCs topologies. Hou et al. [60] propose an ML executable evaluation framework that models the NoCs as a Markov reward model and uses this model to quickly obtain different performance metrics considering faulted routers and various simulation parameters: the calculation of fault resilience and communication time. Silva et al. [126] study the latency of NoCs performance based on the characteristics of NoCs microarchitecture and applications. This work proposes to use a random forest model to predict the latency during the design process, which accelerates the design of space exploration for the microarchitecture of NoCs.

(2) Energy consumption evaluation or ML-based predictors are often used for saving energy in NoCs. Jeong et al. [66] use nonparametric regression techniques to develop an accurate architecture-level NoCs router cost model. Compared with existing models, our model reduces the evaluation error by up to 89% on average. DiTomaso et al. [36] proposed using predictions from the DT model to decide link utilization and direction, when to power gate the channel and router, and increase the channel bandwidth to maximize power savings while minimizing performance loss. Also, this method combines dormant link storage SLS units to save static and dynamic energy consumption. Clark et al. [22] provide a proactive power management strategy, LEAD, which predicts buffer utilization, variation in buffer utilization, or a combined power and throughput metric. It uses a ridge regression model to select the appropriate voltage/frequency per-router basis, minimizing this model's error and making the most accurate predictions possible. In optical NoCs, Winkle et al. [145] use a ridge regression model to scale the power the photonic link consumes and predict the number of packets injected into each router in the next window. This prediction technique adjusts the window appropriately to reduce the static energy consumption of the photonic link.

Table 2
Summary of applying ML models for link management and DVFS.

Problems	Specific Tasks	ML models
Link Management	Dynamically calculation of link utilization thresholds	ANNs [121]
	Prediction of link utilization and link direction	DT [36]
	Dynamically configuration of link bandwidth	Q-learning [114,115]
DVFS	LLC DVFS power management	ANNs [146]
	Proactive power management strategy	Ridge Linear Regression [21]
	Learning critical parameters in routers and caches to obtain control strategies	Q-learning and ANNs [160]
	Prediction of buffer utilization	Ridge Linear Regression [22]
	Prediction of performance state from some available input features	Bayesian Classifier [70]
	Prediction of the number of packets injected into each router	Ridge Linear Regression [145]
	An Efficient power prediction framework	Semi-supervised RL [68]
	The power-constrained performance optimization issue	Q-learning [164]

4. ML for NoCs design

This section presents studies that utilize ML techniques to design NoCs. NoCs primarily handle core-to-core communication and packet transfer between cores and the storage layer. As the number of integrated cores on each chip increases, so does the complexity of NoCs. As such, the NoCs's role in ensuring efficient and reliable packet transfer has become increasingly vital. The design of NoCs usually requires attention to several critical components and issues. First, energy consumption is a challenging problem as the size of transistors gets smaller, and the circuits in the NoCs become more complex, which leads to increased energy consumption. Second, the individual components in the NoCs should be designed to accommodate different workloads and applications, including routing algorithms, arbitration, traffic control strategies, etc. Third, as technology shrinks into the nanometer domain, transistors and links in NoCs are prone to various faults, making the reliability and fault tolerance of NoCs design a critical issue. Fourth, creating the topology and general design from a higher level, usually with a larger design space and complex search process, is nearly impossible using manually optimized methods. ML techniques have powerful learning and generalization capabilities and can adjust strategies according to dynamic workloads and network status in NoCs design. ML-based methods are more suitable for solving these problems and designing crucial components. As many issues in NoCs design can be framed as combinatorial optimization or decision-making problems, ML techniques are extensively explored and exploited. At present, classical ML techniques are well applied to decision-making problems, and RL techniques are widely used in NoCs space exploration, routing algorithms, etc.

4.1. Link management and DVFS

With the development of process technology and the increase in the number of cores integrated into the chip, the energy consumption of NoCs has been increasing. Therefore, it is necessary to design energy-efficient NoCs. Generally, energy consumption can be divided into static and dynamic [13]. In the current research, **power gating (PG)** can effectively reduce static energy consumption by lowering unused or lightly used network components without impacting performance [100]. DVFS can effectively mitigate dynamic energy consumption [28]. DVFS can scale the voltage and frequency of NoCs components (routers, links) in proportion to the workload without degrading network throughput. To achieve energy savings in NoCs, sophisticated energy control schemes are required, which provide an excellent opportunity to design NoCs using ML-based active control schemes. At present, related work on the energy-efficient method of the NoCs using ML mainly designs the control scheme from the following two aspects: the design of the link management scheme and the creation of an active approach for selecting diverse DVFS modes. Table 2 summarizes the research works of ML for link management and DVFS.

(1) Link management. Savva et al. [121] use ANNs to design dynamic link power management mechanisms. This mechanism is used to dynamically calculate link utilization thresholds, which are used to determine which links can be turned off or on at every time interval, thereby achieving power savings for NoCs. However, this method cannot well-balance the trade-off between hardware overhead and network latency. DiTomaso et al. [36] propose a LESSON to reduce static and dynamic energy consumption framework. This work utilizes predictions from the DT model to decide link utilization and direction, when to power gate the channel and router, and increase the channel bandwidth to maximize power savings while minimizing performance loss. Reza et al. [114,115] utilize the RL technique to dynamically configure the bandwidth of NoCs links according to the various application communication tasks. This paper uses table-based Q-learning. Q-learning technique actively configures link bandwidth based on historical information (link utilization, buffer utilization, etc.), where increasing or decreasing the link bandwidth can change the waiting time of packets or the number of flits transmitted on the link. This work uses the distributed RL agent to configure the link bandwidth for each router and verifies that this approach is feasible in the dynamic configuration of large-scale systems. Although this work aims to reduce power consumption while meeting application requirements, it does not emerge from this work how to describe application requirements. Meanwhile, the distributed RL of each router and the use of table-based Q-Learning technology undoubtedly increase the hardware resources of NoCs.

(2) Optimization from the perspective of proactively selecting different DVFS modes. Won et al. [146] was the first to propose utilizing an ANN for NoCs and Last-Level Caches (LLC) DVFS power management. They found predictable patterns in uncore utility that point towards the potential of a proactive approach to uncore power management to address responding poorly to abrupt workload and uncore utility. This paper uses the ANNs model to exploit noncore utility pattern predictions actively. Compared with the existing state-of-the-art method, the method improves the energy latency of noncore systems by 27%. Clark et al. [21] provide a proactive energy management strategy, LEAD, which uses an offline-trained ridge linear regression model to reduce hardware overhead and achieve the trade-off between energy and performance. LEAD predicts buffer utilization, variation in buffer utilization, or a combined energy and throughput metric. LEAD uses a ridge linear regression model to select the appropriate voltage/frequency on a per-router, reducing energy consumption. Zheng et al. [160] propose an RL-based method for designing energy-efficient NoCs, which combines PG and DVFS techniques to reduce static and dynamic energy consumption. RL can automatically observe the interplay between the combined technology and various network parameters to change the control strategy of the V/F level dynamically. Specifically, a per-router-based RL agent learns multiple parameters related to the NoCs and cache and obtains the optimal per-router control policy. Also, ANNs are introduced to reduce the hardware overhead of efficiently implementing RL on each router. Clark et al. [22] propose an adaptive

Table 3
Summary of applying ML models as the design methodology for routing algorithm.

Problems	Specific Tasks	ML models
Routing Algorithm	Each router uses local information to ensure the minimum transmission time	Q-Learning [3]
	Optimization of A-value in Q-Routing	Q-Learning [17,57,86]
	Optimization of the size of the Q-table	Q-Learning [46–49,108]
	Prediction of the hotspots or congestion	ANNs [72,128,129]
	Network congestion alleviation	Q-Learning [41]
	Isolation of endpoint congestion and implementation of load balance	DQN [137,138]
	Prediction of global network state and congestion information	ANNs [81]
	Prediction of buffer and link utilization information to select the optimal routing algorithm	Q-Learning [116]
	Adaption of different applications	Q-Learning [73]

power management technology, DozzNoCs, which effectively combines PG and DVFS technologies to reduce static and dynamic energy consumption. At the same time, this method also combines the ridge linear regression model to enhance further the proposed power management technology, which can actively realize workload prediction and the selection of the DVFS model. Jung et al. [70] propose a Bayesian classifier framework for power management. This framework uses a probability-based Bayesian classifier to predict the processor's performance state for each input task by examining some available input features. Then it uses that predicted state to find the optimal power management action from an existing policy table. Otoom et al. [105] propose a novel global power management technique that uses Q-Learning to optimize the power allocation of all cores. At the same time, power allocation is dynamically adjusted at runtime according to the application's needs to maximize NoCs performance and reduce energy consumption. Juan et al. [68] propose a DVFS approach based on semi-supervised RL to maximize network performance while efficiently utilizing the NoCs power budget. By exploiting spatial and temporal hierarchies, Zhuo et al. [164] propose an online distributed reinforcement learning OD-RL-based DVFS control method to improve network performance under power constraints. This method splits the power-constrained performance optimization problem into two sub-problems with different spatial and temporal granularity. A per-router RL method is adopted at the fine-grained level to learn the optimal control policy for DVFS. An efficient global power budget is employed at coarse-grained to maximize the overall NoCs performance. Sheng et al. [125] propose a new adaptive energy minimization method for heterogeneous systems that uses regression model measurements to generate an energy/performance model at runtime. This model accurately maps applications onto computing resources to ensure the minimum energy consumption for a given application performance requirement. Winkle et al. [145] apply ML to the optical NoCs and use a linear regression model to scale the power consumed by the photonic link and predict the number of packets injected into each router in the next window. This method is an active approach to determining the amount of laser power required by each router within the reserved window. This dynamic prediction technique shows power savings of $40\% - 65\%$ by dynamically adjusting the window appropriately to reduce the static energy consumption consumed by the photonic link.

In applying ML to proactively select different DVFS modes, previous works mainly optimize from two perspectives. (1) Predict network status parameters, such as buffer utilization or performance and energy consumption indicators, enabling the selection of an appropriate DVFS model that achieves a trade-off between energy consumption and performance. (2) Dynamically adjusting power consumption allocation based on workload requirements aims to reduce energy consumption. However, the above previous works lack optimization from the perspective of specific hardware deployment. It remains unclear whether it is possible to form related datasets in a unified format for offline training to predict which DVFS mode to choose to reduce energy consumption while maximizing network performance.

4.2. Routing algorithm

The routing algorithm aims to select a transmission path for a packet that can reach the destination router from the source router and fully utilize the network resources. Therefore, routing algorithms have a significant impact on NoCs performance. With the increasing diversity and irregularity of workloads and traffic patterns, routing algorithms designed from traditional perspectives cannot adapt well to complex traffic patterns and multi-objective optimization problems. To address these problems, researchers have started to try to design routing algorithms using ML techniques. ML techniques can help experts take valuable insights from NoCs' status information, resulting in better design of routing algorithms. Most existing related works employ Q-Learning to design routing algorithms and show better performance through experimental evaluation. In the ML-based routing algorithm design research, the routing algorithm is currently designed from two main perspectives. First, the routing algorithm is designed from the perspective of optimizing the Q-Learning algorithm, how to quickly update the Q-value, etc., to compensate for the shortcomings of this learning algorithm in low workload and how to reduce the size of the Q-table to design a low hardware overhead routing algorithm. Second, the routing algorithm is designed for specific goals or issues, such as congestion and dynamic load balancing problems. Table 3 summarizes the research work of ML on the routing algorithm.

(1) Design of routing algorithm from the perspective of optimizing the Q-Learning algorithm. Boyan et al. [3] propose a Q-routing algorithm with an RL module embedded in each router of NoCs, where each router uses local information to ensure minimum transmission time. Likewise, it is demonstrated that the design of adaptive routing algorithms is a natural application of RL. Q-routing can discover efficient routing decisions in dynamically changing NoCs status. Kumar et al. [86] propose an online adaptive routing DRQ-Routing algorithm, which is improved and designed based on Q-Routing. DRQ-Routing combines the principle of RL to dynamically change routing decisions according to the current network's state. This routing decision learns based on the information from the neighboring router. DRQ-Routing is a dual reinforcement mechanism that adds backward exploration to forward exploration used to update the Q-value, which ensures this algorithm adapts to the current state information of the network and the recent historical information. Choi et al. [17] propose a memory-based routing algorithm, PQ-Routing, which is also improved and optimized Q-Routing. PQ-Routing mainly makes up for the two problems of Q-Routing: the inability to fine-tune the routing strategy and the failure to learn an optimal routing strategy in the case of low workload. RQ-Routing improves the learning rate by storing past network information while selecting a routing strategy by predicting traffic trends. PQ-Routing uses the recovery rate to evaluate the Q-value under low network workloads rather than making selection actions based on the current Q-value alone. Since the Q-value in Q-Routing does not accurately reflect the current network state, this routing policy may be inaccurate. To address this problem, Gupta et al. [57] propose two novel Q-learning framework-based routing strategies. One is a routing strategy

Fig. 8. Q-tables of different routing algorithms in 4×4 mesh (a) Q-Routing, (b) C-Routing, (c) LCQ routing [49].

credit-based Q-Routing, CrQ-Routing, which uses a variable learning rate to increase the learning speed and dynamically captures the current congestion state of the network by adding a parameter. The other is an extension of probabilistic credit-based Q-Routing, PCrQ-Routing, which uses credit values (C-values) to measure the uncertainty in the corresponding Q-values and calculate the learning rate. Puthal et al. [108] propose a new hierarchical cluster-based adaptive routing, C-Routing. The core idea of the C-Routing algorithm is to reduce the size of the Q-table, reducing the storage overhead and the actual deployment area cost. Each router maintains a Q-table containing overhead information for reaching a neighboring destination in its cluster and overhead information for other clusters. Farahnakian et al. [46] propose an adaptive routing algorithm CQ-Routing, which optimizes Q-Routing to obtain the performance of the learning method while keeping the area overhead as small as possible. The idea of this algorithm is to divide the network into several clusters, and each cluster maintains a routing table instead of each router. The key to distinguishing C-Routing is to cluster-level traffic monitoring as not router-level. This approach reduces the size of the Q-table by a factor of four while also improving network performance. Farahnakian et al. [49] also propose a congestion-aware routing algorithm based on Q-Routing to avoid congested areas in NoCs. This routing algorithm is also a low-weight clustering LCQ routing algorithm that divides the network into several clusters, and each cluster maintains a CQ table. To update the CQ table more frequently, both learning and data packets are involved in the propagation of congestion information, so a bidirectional ground-weight clustering routing algorithm Bi-LCQ is proposed. Also, Farahnakian et al. [47,48] develop a congestion-aware routing algorithm based on dual reinforcement Q-Routing, DuQAR. Meanwhile, this paper proposes a congestion detection method that measures the average of free buffer slots in a specific time interval. DuQAR adaptively learns the optimal routing policy and can select a path with less congestion among the available paths from the source node to the destination node. Finally, the Q-tables of three RL-based routing algorithms are summarized, in which Q-routing, C-Routing, and LCQ routing algorithms decrease in size of Q-table sequentially, as shown in Fig. 8.

(2) Design of routing algorithm in terms of specific goals and problems. Soteriou et al. [128,129] and Kakoulli et al. [72] propose a novel proactive hotspot prevention routing algorithm, HPRA, which uses prior knowledge obtained from hotspot predictors of ANNs to assist routing selection in reducing the occurrence of unforeseen hotspots. ANNs model collects buffer utilization to predict hotspots and informs HPRA in time to take appropriate actions to prevent the hotspots from forming. Nonetheless, the algorithm still uses local information to predict hotspots, leading to doubts about its ensured prediction accuracy. Ebrahimi et al. [41] propose a new highly adaptive routing algorithm based on Q-Learning, HARAQ. HARAQ can be minimum routing or non-minimum routing, providing multiple alternative paths between source and destination routers. HARAQ adopts the Q-Learning model as the output selection function for an effective output port selection strategy, evaluates the latency from each output channel to the destination node, and selects a less congested output channel. However, in the ML model, a distributed table must be maintained to store global congestion information from different network regions, and a selection function selects a less congested output channel. The algorithm poses a significant challenge for resource-constrained NoCs due to its substantial storage space requirements. Wang et al. [137,138] propose a comprehensive RL framework for adaptive routing, RELAR. RELAR applies to multiple traffic patterns and solves multi-objective optimization problems. RELAR uses the DQN model to perform routing decisions and is distributed on each router. RELAR first collects the network state, then selects the output port by the decision policy, and finally gets feedback from the NoCs and continuously adjusts the routing policy. RELAR can effectively isolate endpoint congestion when dealing with hotspots and burst traffic patterns and achieve dynamic load balancing to alleviate congestion when encountering uniform traffic patterns to adapt to diverse application scenarios. However, this framework lacks an explanation of how the DQN model determines the relationship between traffic patterns and optimization goals and also lacks an analysis of actual hardware deployments. Kinsy et al. [81] uses a multi-layer neural network model and uses the predicted global network state and congestion information to route network traffic efficiently to make routing decisions for better network performance and power consumption efficiency. However, this work lacks the analysis and interpretation of how the neural network model predicts the congestion state and makes routing decisions. Reza et al. [116] utilizes RL to learn from multiple available routing algorithms (XY routing, random oblivious routing, adaptive West-First routing algorithms) and uses the buffer and link utilization information in the NoCs, to select the optimal routing algorithm to improve the performance of the NoCs. Kao et al. [73] analyze the role of RL in routing algorithms and design a routing framework based on RL for NoCs. This framework mainly uses a learning agent to configure routing strategies in NoCs. Experimental results verify that this framework can learn near-optimal routing strategies under different applications.

4.3. Arbitration and traffic control

The arbiter is a critical component in the router, and the arbitrator's design significantly impacts the router's performance. An arbiter ensures the effective transmission of packets from input ports to output ports without resource components. Arbitration policy aims to grant the output port to one of the multiple competing input ports to transmit packets. An efficient arbitration policy is essential for global fairness in allocating network resources and improving network performance [8,91]. As NoCs microarchitecture becomes more complicated, it increases the difficulty for experts to design effective arbitration strategies. Therefore, experts try to use ML techniques to extract useful heuristic information from sufficient datasets to assist experts in designing arbitration strategies. Related research is currently focused on using ML to develop arbitration strategies by Yin et al. They initially attempt to create an arbitration strategy using DQL, then interpretability for neural networks to design simple arbitration strategies. Finally, they implement an arbitration strategy that operates automatically and can be deployed in actual hardware. In addition to the arbitration policy, ML techniques also can design traffic control, which is a promising approach. Researchers have mainly focused on controlling the rate at which packets are injected. Table 4 summarizes the research works of ML on arbitration and traffic control.

(1) Arbitration policy. Yin et al. [152] initially demonstrated an idea of using DQL to design an effective arbitration strategy. This model is trained through online learning to make long-term maximization decisions. This work verifies that DQL can help researchers explore potentially useful feature information to design effective arbitration strategies. To investigate DQL deeply to design an arbitration strategy,

Table 4
Summary of applying ML models for arbitration and traffic control.

Problems	Specific Tasks	ML models
Arbitration and Traffic Control	Design of an arbitration strategy	DQN [152] and ANNs [154]
	Design of an implementable arbitration logic	linear tree-based [162]
	Design of self-learning throttling mechanism	Q-Learning [33]
	Prediction of the appropriate injection rate for each router	ANNs [136]

Yin et al. [154] propose another novel arbitration strategy that effectively deals with competing network resources. The design process of this arbitration strategy relies on the interpretability of neural networks and domain knowledge to extract critical feature information by analyzing neural network weights. This work needs domain expert knowledge to select and reason about this information and design an arbitration strategy that can be implemented. The above two related works show that arbitration policies designed with ML techniques perform well in experimental simulations. However, the high hardware overhead is a thorny issue when deploying the DQL model directly into routers. To solve this problem, Zhou et al. [162] exploit a linear tree-based model to bridge the DQL model and circuit implementation. First, the result of DQL training is used as the input of the linear tree-based model; then, the linear tree-based model is trained, and the result is converted into an implementable circuit; finally, the arbitration logic can be obtained directly from the simulation process. The design process shows that this model fits well with the design of the arbitration strategy and can also improve the interpretability of the learned policy. Although this work utilized a lightweight ML model (linear tree model) to transform the arbitration logic into a realizable circuit logic, it did not consider whether it could improve network performance under different application loads.

(2) Traffic control. Daya et al. [33] use prioritization, bypassing, and throttling mechanisms, which push bandwidth by maximizing opportunities to use non-minimal paths. A distributed RL throttling mechanism controls injection throttling using Q-Learning to maximize multi-hop performance and improve bandwidth allocation fairness and network performance. Wang et al. [136] propose an admission control method, ANN-AC, a centralized ANNs traffic admission controller. ANN-AC determines the appropriate injection rate and control strategy for each node in the NoCs. Each router's **network interface (NI)** is responsible for periodically collecting network state data within a certain sampling window and sending it to the ANN-AC. ANN-AC implements dynamic control of packets entering the network to improve the performance and efficiency of the NoCs. Two previous works on ML-based flow control the injection rate by analyzing network state information. However, these works will increase the packet information feedback, affecting the packet transmission time.

4.4. Fault tolerance

As technology shrinks to the nanometer realm, transistors and links are more prone to various failures, making ensuring the reliability of NoCs a critical issue. To enhance the reliability of NoCs, researchers have proposed different fault-tolerance techniques, including reactive and proactive strategies. Reactive technology is a typical handling fault technique that responds to faults after the fault occurs, making the recovery process inefficient in energy consumption and latency [159]. Proactive technology proactively predicts the occurrence of faults and mitigates these faults by implementing balanced load techniques [147] or deflective routing algorithms [52]. From these, reactive techniques are not the optimal method to mitigate failures, and this technique may increase power consumption and latency. Proactive techniques require manual design strategies to proactively predict the occurrence of faults and require a large workforce and engineering with relevant domain knowledge. Therefore, proactive techniques give a suitable environment for applying ML techniques to address fault tolerance issues and attract some researchers to adopt ML techniques to design fault tolerance methods. Table 5 summarizes the research work of ML on fault tolerance.

Table 5
Summary of applying ML models for fault-tolerance.

Problems	Specific Tasks	ML models
Fault Tolerance	A reconfigurable fault-tolerance deflection routing algorithm	Q-Learning [50]
	Handling of link and router failures	Q-Learning [119]
	Efficient prediction of network faults	DT [35]
	Dynamically selection of fault-tolerance solutions	Q-Learning [139–141]
	Detection of channel congestion and alternative routes selection	Spiking Neural Network [7]
	Explore application error tolerance	Q-Learning [15]

Feng et al. [50] utilize Q-Learning to design a reconfigurable fault-tolerance deflection routing algorithm, FTDR. This algorithm uses 2-hop fault information to construct routing tables for efficient routing decisions to avoid faults. At the same as, a hierarchical Q-Learning-based deflection routing algorithm, FTDR-H, is also proposed to reduce the routing table size. The area overhead of FTDR-H is reduced by 27% compared to FTDR. Samala et al. [119] uses Q-Learning to design a fault-tolerance routing algorithm, which provides optimal routes between the source and destination under faulty and fault-free grid topology conditions, effectively resolving router and link failures. Both of the above works use the Q-Learning model to achieve efficient fault-tolerant routing decisions. However, the storage overhead brought by the Q-table cannot be ignored, and how to further optimize the size of the Q-table is also an important research direction. DiTomaso et al. [35] exploit a DT model to predict network faults, which adjusts the fault tolerance scheme before the effect of faults. Based on this prediction model, the faults are dynamically mitigated by error correction codes (ECC) and relaxed timing transmission to improve the network performance. Although this solution uses a lightweight ML model (DT model), the training model requires a manual collection of datasets and labeling to ensure the dataset's quality. This work requires the assistance of experts with specialized domain knowledge to provide a high-quality dataset, which is also a time-consuming and labor-intensive task. Wang et al. [140] propose a new active fault tolerance scheme that allows routers to switch between four fault-tolerance operations. This scheme uses Q-Learning to design control policy, and each router agent can dynamically select four fault-tolerance schemes. Each agent updates the control policy by observing the network state information. This control policy detects and corrects soft errors while achieving minimal network latency and lower power consumption. Compared with the above works, Wang et al. [139,141] take a holistic approach to optimize performance, power consumption, and reliability by proposing an intelligent NoCs framework, IntelliNoC. This framework introduces NoCs microarchitecture innovations and uses RL to control for implementation dynamically. The novel NoCs microarchitecture includes multi-functional adaptive channels (MFACs), adaptive error detection/correction and retransmission control, and stress-relaxed bypass design. Active control policies use Q-Learning to handle the dynamic interactions between these technologies. Each RL agent learns from the NoCs state and updates the control policy to select the best mode of operation while providing better fault tolerance and reducing power and area overhead. Carrillo et al. [7] uses a spiking neural network to detect channel congestion and select alternative routes to adapt to NoCs router congestion and connection interruption (failure). Chen et al. [15] explore application error tolerance and propose an RL-based

Table 6
Summary of applying ML models for topology and general design.

Problems	Specific Tasks	ML models
Topology and General Design	NoCs resources for monitoring, prediction, and configuration	ANNs [117]
	Rapidly search and evaluation of NoCs design	Random Forest, Neural Network, Linear Regression [113]
	Routeless NoCs design as an example of designing an optimal loop layout	DQN [95,96]
	Control policies to optimize topology selection for subNoCs	DQN [161]
	Multi-objective optimization problem	STAGE [67,78,79]
	Optimization of the layout of planar and vertical communication links	STAGE [29,31,32]
	Prediction of FPGA resource utilization	Linear Regression, DT regression, and random forest regression [85]

accuracy management technique to automatically analyze application error tolerance and adjust the accuracy in packets to reduce the power consumption and latency of NoCs.

4.5. Topology and general design

With the growing number of cores per chip, the design of the NoCs involves a trade-off between network latency, throughput, various hardware overheads, cabling resources, and various performance targets, which complicates optimizing these goals in NoCs simultaneously. This will lead to an increase in the design space of NoCs, where applying exhaustive strategies to explore the design space is infeasible. Researchers started experimenting with ML methods such as simple regression and neural networks that have been proposed as alternative search strategies. Table 6 summarizes the research work of ML on topology and general design.

(1) Topology. Zheng et al. [161] design a flexible NoCs microarchitecture, Adapt-NoCs, with an RL-based control strategy. Adapt-NoCs structure mainly comprises three parts: adaptive routers, adaptive links, and centralized links. Based on the running applications, the fabric dynamically allocates the network into multiple disjoint regions called subNoCs. These subNoCs can be reconfigured into different topologies, such as mesh, cmesh, torus, or tree, to meet the communication needs of other applications. RL is employed to design an effective control strategy to optimize the topology selection of the subNoCs and improve the performance and energy efficiency of the subNoCs. However, the sizes of different topologies configured in this work are fixed, and if the topology size is changed, it needs to be reconfigured again. This work can be applied to the Chiplet structure, and different topological structures are set for each layer to improve the performance and energy efficiency of the NoCs.

(2) General design. Reza et al. [117] use distributed clustering and global neural network models to dynamically monitor, predict, and configure NoCs resources and propose a Neuro-NoCs model. Divide the NoCs into a cluster of uniform size; each cluster has a neural network for resource monitoring and configuration; the Neuro-NoCs model can achieve the local optimum of each cluster and the global optimum of the entire NoCs. This work uses offline training to implement local and global classifiers, which lack the consideration of application load characteristics. Although online learning is considered in this work, maintaining both local and global neural network models also increases hardware overhead. Rao et al. [113] uses ML to rapidly search and evaluate many solutions and determine the best NoCs design, ML-NoCs. MLNoCs uses many real-world and extensive training datasets of integrated designs to learn which design solution suits different requirements and constraints. Using trained ML models then enables rapid reduction of the design space, significantly speeding up convergence on the needs and conditions of the NoCs. Lin et al. [95,96] utilize reinforcement learning and Monte Carlo tree search (MCTS) implemented, efficiently exploring the design space under conditions and designing the optimal loop layout as an example of routing-free NoCs design. The works proposed by Rao et al. [113] and Lin et al. [95,96] use ML techniques to achieve efficient search design spaces. However, when the application is unknown, these studies do not sufficiently illustrate the significant impact of application-driven methods on energy consumption. Moreover, they do not explore applying unsupervised learning methods to address this issue. Kumar et al. [85] use an ML model for FPGA synthesis under different network topology designs, which utilize ML techniques (multiple linear regression, decision tree, and random forest regression) to predict FPGA resource utilization for NoCs. With available application and network configurations, Le et al. [88] propose a mixed integer linear programming (MILP) model to quickly and efficiently evaluate feasible reconfigurable designs to minimize the number of hybrid links used and reduce the estimation time. However, this work evaluates the predictive model on training datasets collected from the results of the CPLEX optimizer. It lacks instructions on collecting the training datasets and guaranteeing the dataset quality.

In addition to conventional 2D NoCs, some related works focus on 3D NoC designs. Kim et al. [67,78] explore 3D NoCs systems and describe each research problem as a different multi-objective optimization problem, proposing an ML-based design space exploration technique, MOO-STAGE. The STAGE algorithm iteratively alternates between two phases, where the base search attempts to find local optima based on learned evaluation functions. The meta-search uses SVM to learn evaluation functions that can significantly speed up the multi-objective search and determine the best design solution for heterogeneous architectures [79]. Das et al. [29,31,32] focus on the design space of 3D NoCs architecture based on Small-world (SW) networks, using ML to intelligently explore the design space to optimize the layout of planar and vertical communication links. They use the STAGE learning algorithm to analyze the design space to optimize the structure for low latency and less power consumption.

4.6. Other

4.6.1. Thermal management

With the various workloads distribution and high power densities in NoCs, thermal issues have become challenging for NoCs design. Traditional thermal management techniques typically rely on some a priori knowledge of the thermal model and information about the workload application being executed. However, this prior knowledge does not reflect the spatial and temporal uncertainties and variations from the environment, hardware, and workload [106]. For example, traditional **dynamic thermal management (DTM)** controls the system's temperature [151], usually built based on a few specific physical parameters, which may lead to inaccurate prediction results and reduce the benefits of DTM. Researchers have attempted to address this problem using ML techniques. In other words, ML techniques can potentially adapt to changing system conditions and workloads to improve decisions about thermal management. Table 7 summarizes the research work of ML on thermal management.

Chen et al. [11,12] combines ANNs and minimum mean square LMS adaptive filtering theory to propose an adaptive ML-based temperature prediction model. A temperature prediction model based on ANNs is first used to predict the NoCs temperature. Then the LMS adaptive filtering theory adapts dynamically to the temperature change behavior.

Table 7
Summary of applying ML models for thermal management and security issues.

Problems	Specific Tasks	ML models
Thermal Management and Security Issue	Prediction of temperature	ANNs [11,12]
	A thermal-aware adaptive routing algorithm	Q-Learning [158]
	Adaptation of inter- and intra-application thermal variations	Q-Learning [27]
	Dynamically change the thermal management strategy	Q-Learning [40]
	Accurate temperature prediction	Locally Weighted Projection Regression [80]
	Satisfy with the power and temperature constraints	Q-Learning [63]
	Fast and accurate prediction of instantaneous temperatures	Autoregressive [69]
	Identification and isolation of HT injection failure behavior	ANNs [142]

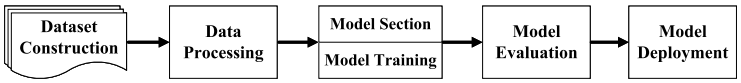


Fig. 9. General steps of ML-based NoCs.

This model can adapt to the temperature behavior during the operation of the NoCs, resulting in an average error reduction of 37.2% to 62.3%. Zhang et al. [158] propose a thermal-aware adaptive routing algorithm based on tableless approximate Q-Learning to optimize thermally induced optical power loss in optical routing systems and find the optimal low-loss path in NoCs temperature variations. Compared with the traditional table-based Q-Routing algorithm, the advantages of this routing algorithm are fast convergence and linear function approximation of the Q-table, which reduces the area overhead while obtaining close optimization results. Das et al. [27] propose an adaptive thermal management method that adapts to inter- and intra-application thermal variations. This method uses a reinforcement learning model to learn the relationship between the mapping of threads to cores, the frequency of the cores, and the temperature to control the average temperature and thermal cycling. Ebi et al. [40] propose an RL-based thermal management scheme to dynamically change the thermal management strategy before reaching the thermal threshold based on past recorded observations. This scheme can reduce the peak temperature ratio and keep it below the thermal threshold through RL. Kim et al. [80] propose a learning-based solution for dynamic thermal management (LDTM) to control the temperature. This method needs to predict future thermal before the temperature becomes high. Then, depending on the temperature variation pattern, it must apply adapted DTM techniques. Finally, it uses the LWPR (locally weighted projection regression) algorithm for accurate temperature prediction to reduce the temperature by decreasing the operating frequency. The previous studies aim to leverage past recorded network state information to predict the future temperature of the NoCs. However, this research does not analyze whether the windows for collecting network state datasets under different workloads must be dynamically adjusted. Coskun et al. [23] uses online learning to adapt to dynamic changes in workload and select a policy to provide a trade-off between performance and thermal management. This online learning evaluates the thermal management policy at runtime by considering thermal behavior and performance and ensures convergence to the policy with the most favorable desired performance-temperature trade-off. Iranfar et al. [63] propose an RL-based power and thermal management algorithm that uses heuristics to speed up the convergence of RL models. Using heuristics to help the RL agent select the most appropriate action from a set of actions improves the network performance. It reduces the learning space to satisfy the power and temperature constraints. Juan et al. [69] use a learning-based autoregressive AR to learn the different thermal distributions of the NoCs to achieve fast and accurate prediction of instantaneous temperatures. This work mainly analyzes the predicted temperature situation in 2D NoCs. However, this work can further extend this framework to predict the transient temperature of 3D CMP.

4.6.2. Security issue

In addition to the classical problems in NoCs design, the security issue of NoCs is also worth studying. Since the primary responsibility of the NoCs is to interconnect various components, NoCs can be effectively used to implement security countermeasures [10]. When maliciously implanted hardware trojans (HTs) are planted in the NoCs, they leave the network in a saturated state, resulting in packet leakage from the side channels while causing network performance degradation. While traditional techniques can predict and isolate HTs to protect NoCs, they inevitably lead to inaccurate detection appearances and severe impacts on network performance and power consumption. Table 7 summarizes the research work of ML on security issues. Wang et al. [142] design a secure and efficient NoCs design framework based on ML, TSA-NoCs, which focuses on real-time detection of HTs using ANNs to identify abnormal networks to HT injection fault behavior and improve the accuracy of HT detection. Then, the enhanced HT isolation problem using SmartRoute can optimize network performance by dynamically classifying routers into HT-free and HT-infected routers after HT detection. However, this study lacks an introduction to the dataset used as input into the ANN model. Besides utilizing buffer utilization as a data feature in this paper, other network state information can also be employed as data features for learning and training purposes. Sudusinghe et al. [130] propose an ML-based DoS attack detection mechanism based on NoCs, which employs the trained model to classify network traffic behavior as normal or attack during the running phase to detect DoS attacks.

5. Discussion and future work

The general steps in applying ML techniques for NoCs are illustrated in Fig. 9, comprising the construction of datasets, data processing, selection and training of the ML model, model evaluation, and model deployment. This section highlights the limitations and prospects of using ML techniques in NoCs. The enhancement of ML techniques in NoCs hinges on addressing data processing issues, selecting appropriate ML models, implementing ML models effectively, and specific deployment. Additionally, we explore the potential benefits of NoCs design automation and practices in future endeavors.

5.1. Bridging the data gap

Datasets are the basis of ML technology to mine valuable knowledge and ML technology will be meaningless without datasets. Therefore, the dataset's quality plays a crucial role in the performance of ML techniques. At present, there is no unified format and standard for collecting datasets when ML technology is applied in the NoCs. At the same time, most previous works did not give clear steps to construct

the dataset. Determining the features of the NoCs heavily relies on expert knowledge. For example, the network features used in previous works typically involve buffer utilization, buffer queue size, etc., which require careful analysis and selection by personnel familiar with the NoC to determine. Therefore, applying ML technology to constructing high-quality datasets in the NoCs also requires domain experts to provide meaningful and high-quality datasets.

5.1.1. Dataset construction

The dataset construction is an essential part of ML techniques, which is the basis for training and testing ML models and determines the accuracy and generalization ability of the ML techniques. As ML models usually require enough datasets to learn statistics and make decisions, dataset quality is a critical factor that limits ML techniques' capabilities. However, constructing higher-quality datasets in NoCs is a challenging task. **(1) The methodology for acquiring datasets in NoCs is arduous and costly.** Obtaining data through simulation or evaluation is expensive and scarce in hardware deployment problems. Synthetic data generation using simulation tools is a popular alternative. The dynamic nature of NoCs applications and the variability in network state and performance necessitate the collection of datasets in real time while the network is operational. However, monitoring network status and collecting data without compromising network performance is difficult. **(2) Variety of data types.** Noise and errors may mar the dataset, leading to imprecise training and suboptimal application results. The non-uniform distribution of the dataset, determined by NoCs topology, traffic workload, and other factors, may also impact dataset quality. Meanwhile, the variety of data involved in NoCs requires pre-processing methods to improve the dataset's quality. For example, normalize data by scaling the properties of the sample to some specified range [34]. **(3) Properly labeled datasets or transform dataset forms to suit different ML models.** Supervised learning models require labeled datasets, and NoCs designers must have domain knowledge and determine evaluation metrics. Without labeled training datasets, we can utilize unsupervised learning models or a combination of supervised and unsupervised models [1,54] to address the critical problem of designing a NoCs. At the same time, RL is also a potential method to generate training datasets.

5.1.2. Feature selection

Feature selection identifies a proper subset of features from the original dataset for modeling and prediction purposes. ML strongly depends on feature selection, where input data features are used to build training models and generate predictions. The feature selection can greatly affect the performance and generalization ability of the model. Hence, feature selection is vital to curtail the number of features and simplify the learning model, mitigating issues such as overfitting or computational overhead. Classical feature selection methods include filter methods, wrapper approach, embedded methods, and **Deep learning (DL)** methods. **(1) Filter methods**, which use statistical or information-theoretic methods to analyze and score features and select a subset of features with high ratings. However, this method ignores the interconnection between features and may not guarantee the quality of the final selected feature set [89]. **(2) Wrapper approach**, which uses a learning algorithm to train the model and compare the performance of the models obtained by evaluating different feature subsets and selecting the best-performing feature subset [58]. **(3) Embedded methods** that embed feature selection into the learning model, i.e., the best subset of features is selected directly during the training of the learning model to provide a trade-off between filter and wrapper methods [82]. **(4) DL methods**, which learn the representation of features by building deep neural networks and backpropagation algorithms, choose different loss functions depending on the target task and thus select the optimal subset of features [92].

Currently, feature selection in NoCs relies heavily on the expertise of domain experts, particularly regarding the interpretability of features. However, DL-based methods can replace and supplement domain expert knowledge after analyzing classical feature selection techniques. By leveraging deep neural networks' nonlinear feature extraction capabilities, such techniques can identify the most distinctive features to enhance a model's generalization ability and predictive accuracy. For instance, Yin et al. [154] utilize insights from neural networks to discover more effective designs for arbiters. Through heat maps, they visualize the average weights of hidden layers to assess the importance of features. However, there remains a necessity for designers to explore features and translate their observations into actionable arbitration strategies.

5.2. Model selection and training

When ML technology is applied to the NoCs, one of the primary considerations is selecting an appropriate ML model. Model selection and building datasets are closely related. Supervised learning or unsupervised learning models can be selected depending on the availability of labeled training datasets. The specific problem of the NoCs determines the selection of the ML model. Prior research indicates that for supervised learning models, the predominant utilizes encompass regression, decision tree, and ANNs models. On the other hand, for RL models, Q-Learning or DQL models are primarily used. Supervised learning models are well-suited for tasks involving buffer utilization prediction and monitoring, such as performance prediction, DVFS model selection, and addressing security issues. The RL model is adept at handling multi-objective tasks, particularly those involving trade-offs between energy consumption and network performance, addressing diverse application requirements. This includes tasks related to routing algorithms, arbitration, fault-tolerance routing design, temperature management, etc. When applying ML technology to NoCs, the choice of training method for the ML model must also be considered. Offline and online training are commonly utilized techniques in ML. Online training involves deploying and running the ML model directly on the NoCs. However, online training methods are constrained by actual hardware limitations, such as power consumption and area. Offline training, on the other hand, takes place in an environment separate from the actual deployment of NoCs, allowing for the use of large datasets to train ML models with higher complexity and overhead. Nonetheless, offline training requires high-quality datasets and comes with the cost of training/prediction time. Additionally, offline training necessitates expert interpretation of the trained ML model to extract its essential logic for effective problem-solving and deployment within NoCs to accomplish desired tasks.

5.2.1. Model selection

In general, various factors must be considered when faced with ML model selection. First, the nature of the problem should be carefully typed. Different models should be selected depending on whether the application domain requires classification or regression. Second, the size and type of the dataset should be considered. More complex models may be necessary in cases where the dataset is large, while smaller datasets may require simpler models. Different data types, such as numeric, image, and text, may also require other models. Third, feature engineering should be considered as a basis for model selection. Factors such as the ease of feature engineering and the possibility of automatic feature extraction should be considered.

In previous studies concerning ML applications on NoCs, supervised learning models such as decision trees, neural networks, and linear regression models were commonly employed. On the other hand, Q-learning and DQL models were frequently utilized in NoCs. Supervised learning models were extensively used in function approximation [137] and prediction tasks [36], including buffer and link utilization prediction. DiTomaso et al. [36] use a DT model to predict link utilization changes and dynamically provide additional bandwidth to maximize

power savings with minimal performance loss. Given that the traditional table-based Q-Routing's table overhead increases with the network size, Zhang et al. [158] design a heat-aware adaptive routing with table-free Q-Learning using linear function approximation, which can obtain close optimization results. RL models are widely used in designing policy problems [3,152] (routing policies, arbitration policies, etc.). RL models may be configured with target reward functions as required to address the power and performance trade-off.

5.2.2. Training model: offline vs. online

We have taken into account different application scenarios when designing and deploying ML-based technology in the NoCs. In ML, offline and online training techniques are widely used. The choice of training technique depends on the specific application scenario and design constraints involved. Offline training utilizes large datasets to train ML models, enabling them to predict new datasets accurately. Offline training technology has the advantage of high training efficiency and does not require the storage space of real-time datasets in the system [42]. Online training involves the continuous updating of ML model parameters using real-time datasets, gradually improving the model's accuracy as it adapts to the evolving characteristics of the datasets [24]. Online training demands increased computational resources and storage space to facilitate better adaptation to dynamic datasets.

The main differentiation between offline and online training techniques can be summarized as follows:

(1) NoCs dataset requirements. Offline training technology relies on large datasets of higher-quality NoCs. The construction of these NoCs datasets requires manual collection and organization. Online training technology requires higher real-time performance of NoCs datasets, enabling ML models to train models and update parameters through real-time datasets incrementally.

(2) Training process. Offline training technology can fully utilize computing resources to perform batch processing and optimize ML models during training. Offline training technology can perform more complex calculations and model tuning. For example, in exploring the design space of NoCs and evaluating performance models. Online training technology requires more efficient and lightweight model updates. For example, more lightweight ML models (ANNs and DTs) are employed in link management and DVFS.

(3) Model implementation and hardware overhead. Offline training technology can generate a complete training model after the ML model training and perform prediction and inference in practical applications. The implementation of offline training technology largely depends on the overhead of traditional design methods. Therefore, optimizing offline training technology should focus on improving dataset quality and overall ML model accuracy. Online training technology continuously updates the ML model during training. In real-world applications, the constant reception of new datasets and the continuous updating of ML models are essential. It requires more computational resources and storage space to better adapt to dynamic datasets. Due to power consumption and area constraints, online training ML models may prove challenging. The hardware implementation of online training technology is mainly limited to the storage space of datasets and models. This leads us to prioritize efficient and less complex models (e.g., ANNs and DTs) when choosing ML models. Consequently, researchers have investigated dedicated hardware implementations to deploy ML models on hardware and minimize execution time overhead. For example, Esmaeilzadeh et al. [43] utilize specialized hardware to implement a neural network processor for function approximation. Nevertheless, the semi-online training mode presents a promising compromise, combining the advantages of online and offline training methods [148]. Specifically, ML models learn from previous training datasets in semi-online training mode and are incrementally updated. However, offline training remains a prevalent approach for training ML models in NoCs.

5.3. Mining ML model

Although ML models have made notable progress in the field of NoCs, we anticipate the development of new algorithms or models that can further enhance NoCs system modeling and optimization. Moreover, we expect to achieve more breakthroughs by exploring the potential of ML models in areas like knowledge and interoperability. Despite the current advancements in ML-based NoCs design, there is still ample room for improvement, and we look forward to witnessing more remarkable advances in this field. This paper provides two recommendations for optimizing ML models. First, we suggest combining the ML models. While individual ML techniques can achieve great results, adopting a combined approach that integrates different ML techniques or combines ML techniques with heuristics can offer more opportunities to leverage ML techniques. Second, we suggest enhancing the interpretability and implementation of ML models. In the previous works on ML-based NoCs, there is a lack of interpretation of ML models and optimization design schemes from the perspective of practical deployment. Simultaneously, the existing works demonstrate that employing ML techniques for optimizing problems in NoCs has yielded performance advantages compared to heuristic methods. However, the absence of an explanation of the principles behind the ML technology optimization problem has led to challenges in deploying these solutions to hardware units.

5.3.1. Combination of ML model

The majority of works in the realm of ML-based NoCs rely on a singular ML approach for learning and control. However, in the design of routing algorithms in NoCs, researchers utilize a distributed approach to optimization that involves multiple agents, which is also applicable to multi-agent DQN [157]. Through collaboration, these agents can achieve multi-objective optimization in NoCs. Fettes et al. [51] proposed three supervised learning models that accurately predicted network parameters for DVFS mode selection by considering buffer utilization, variation in buffer utilization, and variation in power consumption/throughput. RL directly uses the DVFS schema as the action space. At the same time, the combination of these two models can perform the task well. Zhou et al. [162] employ a hybrid approach, combining DQN and the DT model to devise an arbitration policy. The method relies on a DT model trained using the output of a pre-trained DQN model. More precisely, DQN is used to annotate the dataset, which is then trained as the input dataset to the DT model. Finally, the complex model is transformed into implementable hardware logic. Although a single ML model may demonstrate commendable outcomes, amalgamating diverse models or combining multiple ML models with heuristics can present even more possibilities.

In addition to the above combination, we recommend combining unsupervised learning and supervised learning models to improve the dataset's quality. Unsupervised learning finds applications in predictive models for constructing valuable feature representations by capturing intrinsic structures and patterns within NoCs datasets. These features can be used as input to supervised learning models, providing representations of more prosperous and valuable information, thereby improving the performance of the models. At the same time, the predicted model can also use the labeled NoCs dataset to fine-tune the model to adapt to specific supervised learning tasks. In addition, combining self-supervised and supervised learning models is also a viable approach. Self-supervised learning model means that the annotations used for ML come from the data rather than from manual annotations to learn more with fewer data samples or experiments. Self-supervised learning models can complement the labeled datasets required by supervised learning models for training. Therefore, the combination of self-supervised learning and supervised learning models can improve datasets' efficiency and generalization ability. Self-supervised and semi-supervised learning models can also be combined [156]. Self-supervised learning, utilized with unlabeled datasets, and semi-supervised learning, employed with

labeled datasets, can effectively leverage the strengths of both types of datasets to enhance the model's capabilities and generalization. This combined approach offers an efficient solution in scenarios where the availability of labels is limited or obtaining them is expensive.

5.3.2. Model interpretability and implementation

Gaining a deeper understanding of the NoCs and their application scenarios enables the selection of more appropriate ML models for different problems. This understanding also sheds light on the mechanisms by which these models operate. When tackling specific issues, domain knowledge can assist designers in comprehending how ML models leverage training data to make decisions and aid them in explaining the rationale behind those decisions. The model itself should be the fount of knowledge, not the data. By improving the interpretability of ML models, we can gain better insight into the model's decision-making process and augment its implementation. Therefore, interpretability plays a pivotal role in enhancing the deployment of ML models in the context of NoCs. DiTomaso et al. [36] propose a fault tolerance scheme using active before faults affect the system. A dataset is constructed based on domain knowledge to train decision trees for predicting timing faults on the link. At the same time, we can analyze the effectiveness of the predictor on the test dataset. Yin et al. [154] use heat maps to visualize the average weights of the hidden layers of the neural network to analyze the importance of features, and designers explain why these features are essential. Finally, the critical insights from the analysis and interpretation are translated into a strategy for practical arbitration. ML models excel at prediction tasks and decision-making, which can demonstrate the potential of ML applications in the domain of NoCs. This highlights that ML is well-suited to tackle complex problems that are challenging for humans.

Aside from enhancing the interpretability of ML models to support designers, we can also utilize effective techniques and strategies to decrease the hardware overhead of model implementation. For instance, both DQN and supervised learning methods require weight storage, particularly for neural networks. One solution is to restrict the number of hidden layers in the network. Current studies implement a single hidden layer [162] to ease the complexity of the model and improve its implementability. Future work can endeavor to explore avenues such as software and hardware optimization or adopt a co-design approach to achieve better implementation of ML models.

5.4. Support for different application scenarios

ML-based techniques ought to be applied not only in 2D NoCs but also in other emerging NoCs to foster the development of NoCs. In NoCs design, it is customary to utilize ML techniques to optimize problems under homogeneous NoCs, offering accessible communication and efficient computation. However, researchers have begun experimenting with ML techniques to optimize problems under heterogeneous NoCs. This entails integrating multiple heterogeneous processor cores and other functional units onto a single chip to perform various computational tasks in the same system. For example, Zhang et al. [158] proposed an RL-based control strategy in a heterogeneous multi-core architecture capable of concurrently executing multiple applications to provide efficient communication support. In addition to driving the application to heterogeneous NoCs, we should also endeavor to use them for wireless and optical NoCs [144]. At the same time, we recommend integrating Chiplet architecture and RL to optimize the NoCs design space jointly. Chiplet is a modular chip design in which a large chip is divided into several smaller modules (called Chiplets), each of which can be designed and tested independently. Then these modules are combined to form a complete NoCs [149,153]. An agent of RL takes the best action by constantly interacting with the Chiplet to maximize the cumulative reward. Thus, we can strive to add an agent structure to each Chiplet to achieve dynamic control of each Chiplet module.

5.5. NoCs design automation

Although ML-based approaches have significantly diminished the evaluation costs during iterations of NoCs designs, thereby propelling advancements in hardware development, ML-based NoCs design still needs to progress, and the ultimate objective can automate the NoCs design. In recent developments, many researchers have explored how ML techniques can be leveraged to design and optimize NoCs across layers. Some issues still require attention, such as optimizing multiple components of the NoCs simultaneously while satisfying various constraints. We contend that a comprehensive framework for NoCs based on ML should be capable of harnessing information from different parts and characterizing the state behavior of the NoCs. Furthermore, this framework should be able to make decisions at varying levels of granularity, thus enabling precise and comprehensive control of the entire NoCs system. Hence, as more tasks are automated, it hastens the implementation of positive feedback loops between ML and NoCs. Simultaneously, implementing automated NoCs design also spurs NoCs hardware rapid design.

The fully automated design of the NoCs may be the ultimate goal, but the gradual realization of automated design is still essential research content for future work. In simple terms, the interplay between ML and the NoCs design will develop even more as more tasks in NoCs are automated. The introduction of ML techniques has accelerated the automation of NoCs design. For example, in the planning and designing router layout, Zhu et al. [163] proposed a new simulation automatic routing method based on a generative neural network, GeniusRoute. GeniusRoute does this by automatically learning human behavior in manual routing placement. GeniusRoute is a new method to automatically extract different network routing regions from manual layout and apply the learned knowledge to simulated routers. Therefore, one idea is to model the NoCs design hierarchically, from the underlying hardware to the overall NoCs behavioral design. The relevant features and potential features of the entire NoCs design are synthesized through the model to improve the accuracy of generating the actual NoCs. Another idea is to explore the automatic configuration of parameters in the NoCs. This learning model has the potential to incorporate existing parameter configurations and dynamically reconfigure parameters to suit the requirements of diverse applications, thereby advancing the progress of NoCs design automation.

5.6. NoCs-based hardware devices

This section introduces the current hardware devices based on NoCs; and then analyzes how these hardware devices utilize or apply ML technology.

Currently, commonly used NoCs-based hardware devices include GPU (Graphics Processing Unit), TPU (Tensor Processing Unit), FPGA (Field-Programmable Gate Array), ASIC (Application-Specific Integrated Circuit), NPU (Neural Processing Unit) and DPU (DL Processing Unit). GPU is a hardware accelerator dedicated to graphics processing. However, it is widely used to speed up DL and other ML tasks due to its parallel processing capabilities. TPU is a hardware specially developed by Google to accelerate ANN calculations and is especially good at matrix operations and tensor calculations. FPGA can be reprogrammed as needed to adapt to different ML tasks and, in some cases, can provide high-performance acceleration. ASIC is a dedicated integrated circuit optimized for a specific ML algorithm or task to provide highly customized acceleration. NPU is a kind of hardware specially designed to accelerate the calculation of neural networks, usually used in mobile devices and embedded systems. DPU is a dedicated hardware accelerator for deep learning computing, typically including features optimized for tensor calculations and model inference. In addition to these commonly used hardware devices, there are other hardware devices. Esmaeilzadeh et al. [44] proposed a hardware NPU that transforms approximate program functions with an approximation algorithm. The Diannao family

of accelerators includes Diannao [14], DaDiannao [97], and Shidiannao [38], which use neural functional units for neuron and synapse computations. Yang et al. [150] designed an accelerator based on fused kernel convolution, Fusekna, which fully uses activation value sparsity and weight bit sparsity.

Subsequently, we present an exposition on how hardware devices based on NoCs leverage and apply ML techniques. Choudhury et al. [19] proposed a software/hardware co-design framework for accelerated graph computing. The method combines architecture-aware vertex reordering with priority-based task assignment techniques. By leveraging emerging 3D integration techniques, they again propose a small-world NoCs-enabled multi-core GPU architecture, in which a link layout connecting streaming multiprocessors and memory controllers follows a power law distribution. Dong et al. [37] proposed a new FPGA ANN structure combining layer multiplexing and pipeline structure. This architecture achieves higher performance by improving the resource usage efficiency of the same FPGA board. Bui et al. [6] proposed a NoCs-based feed-forward neural network implementation method, which utilizes NoCs on FPGAs. A dedicated NoCs is implemented to handle the many interconnections between neurons. The design uses multiple processing units to exploit the parallelism available in the neural network layers best to accommodate the large-scale communication requirements of the network. The development of DL-based applications and the complexity of ML techniques have increased the depth of ANNs. Increasing a neural network's depth is challenging regarding latency, energy consumption, learning, and inference speed. Mirmahaleh et al. [102] proposed two Booth and Matyas standard generator function training DNN gradient descent methods. Methods for pruning weights, neurons, and layers of DNN based on the minimum distance error before and after pruning. The method employs a novel elastic data flow and DNN mapping on a mesh topology to reduce latency and energy consumption. Hojabr et al. [59] proposed a custom Clos topology CLOSNN for neural networks. It customizes the baseline folded Clos for neural networks. These customizations include increasing its bandwidth and eliminating unnecessary connections to broadcast-based neural network traffic. Khawaja et al. [75] proposed a simplified multi-core-based scalable hardware architecture to implement K-means. The mean and fitness modules in this architecture are further expanded to increase the speed of the K-means clustering algorithm. This architecture connects cores through NoCs to improve messaging bottlenecks and scalability. Mahajan et al. [99] proposed a framework for generating accelerators for a class of ML, Tabla, rather than designing accelerators for ML. The key is to identify the commonalities of various ML algorithms and use these commonalities to provide programmers with a high level of abstraction. This framework leverages the notion that many ML algorithms can be expressed as stochastic optimization problems. Tabla offers a template-based framework for accelerating such ML algorithms. Song et al. [127] proposed the first holistic solution to accelerate GAN-based unsupervised DL. The scheme optimizes the training process to reduce NoCs memory consumption. They again offer a novel time-division multiplexing design to map rich computation stages to the microarchitecture efficiently.

In summary, with GPU accelerators' powerful computing expansion capabilities, if multiple ML models are co-located on the same GPU to achieve multi-tenant deep learning inference, it can improve resource utilization, increase service throughput, and reduce energy consumption. However, performing efficient multi-tenant ML inference is challenging and requires thorough optimization of full-stack systems [155]. Accelerators for ANNs are usually implemented as multi-core chips and rely on NoCs to manage large amounts of interneuron communication [59]. For different NoCs hardware devices, future research directions can adopt novel NoCs topologies and efficient routing algorithms to limit the communication further overhead, thereby improving the performance of accelerators. However, when the same ML technology is applied to different hardware accelerators, multiple optimization goals cannot be satisfied simultaneously. For example, SNNs are deployed on different platforms, such as central CPU, GPU, and ASIC. However, due to the memory bandwidth limitation, SNNs on CPU/GPU consume high power overhead and limited throughput [62,65]. For optimal performance and energy efficiency, many researchers have focused on building custom ASIC to accelerate network inference workloads. Although ASIC is an attractive solution, it does not provide enough flexibility to accommodate the rapid evolution of neural network models [64].

6. Conclusion

The expeditious progress of ML is bolstered by the implementation of robust NoCs systems that are instrumental in facilitating the training and implementation of ML models. Concurrently, ML can potentially alter the approach to NoCs design. This has enabled a tight collaboration between ML and NoCs research, offering new perspectives and optimization strategies to advance NoCs design. The current research about the application of ML to NoCs can be broadly categorized into two domains: performance modeling and prediction and direct utilization of ML to design NoCs. We advocate for a symbiotic relationship between ML and NoCs, wherein ML-based methods can function effectively on NoCs. However, there are still challenges and deficiencies in the current works on ML-based NoCs design. We anticipate that ML will stimulate novel design concepts and perspectives for NoCs, which can be seamlessly incorporated for deployment onto hardware systems.

CRediT authorship contribution statement

Xiaoyun Zhang: Investigation, Visualization, Writing – original draft, Writing – review & editing. **Dezun Dong:** Conceptualization, Funding acquisition, Methodology. **Cunlu Li:** Writing – review & editing. **Shaocong Wang:** Writing – review & editing. **Liquan Xiao:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The authors would like to express their sincere gratitude to the anonymous reviewers for their invaluable comments and suggestions. This work is supported by the National Key Research and Development Program of China under Grant No.2022YFB4501702, the Natural Science Foundation of China (NSFC) under Grant No.62002368, and the Excellent Youth Foundation of Hunan Province under Grant No.2021JJ10050.

References

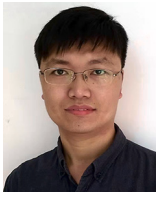
[1] M. Alawieh, F. Wang, X. Li, Efficient hierarchical performance modeling for integrated circuits via Bayesian co-learning, in: Design Automation Conference, 2017, pp. 1–6.
[2] Bahareh Bahrami, M.A.J. Jamali, N. Bagherzadeh, L.G.G. Morales, Wireless network-on-chip architectures: a comprehensive review, in: Horizons in Computer Science Research, vol. 16, 2018.
[3] J.A. Boyan, M.L. Littman, Packet routing in dynamically changing networks: a reinforcement learning approach, in: International Conference on Neural Information Processing Systems, 1993.
[4] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and regression trees (cart), Biometrics 40 (1984) 358.
[5] L. Breiman, J.H. Friedman, R.A. Olshen, C. Stone, Classification and Regression Trees, Routledge, 2017.

[6] T.T.T. Bui, B. Phillips, A scalable network-on-chip based neural network implementation on fpgas, in: 2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF), IEEE, 2019, pp. 1–6.
[7] S. Carrillo, J. Harkin, L. Mcdaid, S. Pande, F. Morgan, Adaptive routing strategies for large scale spiking neural network hardware implementations, in: International Conference on Artificial Neural Networks, 2011.
[8] Y.Y. Chang, S.C. Huang, M. Poremba, V. Narayanan, C.T. King, Ts-router: on maximizing the quality-of-allocation in the on-chip network, in: IEEE International Symposium on High Performance Computer Architecture, 2013.
[9] S. Charles, P. Mishra, A survey of network-on-chip security attacks and countermeasures, ACM Comput. Surv. 54 (2021) 1–36.
[10] S. Charles, P. Mishra, A survey of network-on-chip security attacks and countermeasures, ACM Comput. Surv. 54 (2021) 1–36.
[11] K. Chen, Y.H. Liao, Online machine learning-based temperature prediction for thermal-aware noc system, in: 2019 International SoC Design Conference (ISOCC), 2019.
[12] K.-C.J. Chen, Y.-H. Liao, Adaptive machine learning-based temperature prediction scheme for thermal-aware noc system, in: 2020 IEEE International Symposium on Circuits and Systems (ISCAS), 2020, pp. 1–4.
[13] L. Chen, Z. Di, M. Pedram, T.M. Pinkston, Power punch: towards non-blocking power-gating of noc routers, in: 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), 2015.
[14] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, O. Temam, Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning, ACM SIGARCH Comput. Archit. News 42 (2014) 269–284.
[15] Y. Chen, A. Louri, Learning-based quality management for approximate communication in network-on-chips, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 39 (2020) 3724–3735.
[16] K. Cho, B.V. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, Comput. Sci. (2014).
[17] S. Choi, D.Y. Yeung, Predictive q-routing: a memory-based reinforcement learning approach to adaptive traffic control, in: Neural Information Processing Systems, 1995.
[18] F. Chollet, Xception: deep learning with depthwise separable convolutions, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[19] D. Choudhury, R. Barik, A.S. Rajam, A. Kalyanaraman, P.P. Pande, Software/hardware co-design of 3d noc-based gpu architectures for accelerated graph computations, ACM Trans. Des. Autom. Electron. Syst. (TODAES) 27 (2022) 1–22.
[20] T.V. Chu, K. Kise, Lef: an effective routing algorithm for two-dimensional meshes, IEICE Trans. Inf. Syst. E102D (2019) 1925–1941.
[21] M. Clark, R. Bunescu, A. Kodi, A. Louri, Lead: learning-enabled energy-aware dynamic voltage/frequency scaling in nocs, in: ACM/ESDA/IEEE Design Automation Conference, 2018.
[22] M. Clark, Y. Chen, A. Karanth, B. Ma, A. Louri, Dozznoc: reducing static and dynamic energy in nocs with low-latency voltage regulators using machine learning, in: 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2020.
[23] Gross Coskun Rosing, Temperature management in multiprocessor socs using online learning, in: ACM/IEEE Design Automation Conference, 2008.
[24] I. Dagan, S.P. Engelson, Committee-based sampling for training probabilistic classifiers, in: Machine Learning Proceedings, 1995, pp. 150–157.
[25] W.J. Dally, C.L. Seitz, The torus routing chip, Distrib. Comput. 1 (1986) 187–196.
[26] W.J. Dally, B.P. Towles, Principles & practices of interconnection networks, 2004.
[27] A.K. Das, R.A. Shafik, G.V. Merrett, B.M. Al-Hashimi, B. Veeravalli, Reinforcement learning-based inter- and intra-application thermal optimization for lifetime improvement of multicore systems, in: Design Automation Conference, 2014.
[28] R. Das, S. Narayanasamy, S.K. Satpathy, R.G. Dreslinski, Catnap: energy proportional multiple network-on-chip, in: Proceedings of the 40th Annual International Symposium on Computer Architecture, 2013.
[29] S. Das, J.R. Doppa, D. Kim, P.P. Pande, K. Chakrabary, Optimizing 3d noc design for energy efficiency: a machine learning approach, in: IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2015, 2015.
[30] S. Das, J.R. Doppa, D.H. Kim, P.P. Pande, K. Chakrabarty, Optimizing 3d noc design for energy efficiency: a machine learning approach, in: 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2015, pp. 705–712.
[31] S. Das, J.R. Doppa, P.P. Pande, K. Chakrabarty, Energy-efficient and reliable 3d network-on-chip (noc): architectures and optimization algorithms, in: The 35th International Conference, 2016.
[32] S. Das, J.R. Doppa, P.P. Pande, K. Chakrabarty, Design-space exploration and optimization of an energy-efficient and reliable 3-d small-world network-on-chip, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 36 (2017) 719–732.
[33] B.K. Daya, L.S. Peh, A.P. Chandrakasan, Quest for high-performance bufferless nocs with single-cycle express paths and self-learning throttling, in: The 53rd Annual Design Automation Conference, 2016, pp. 1–6.
[34] Y. Ding, N. Mishra, H. Hoffmann, Generative and multi-phase learning for computer systems optimization, in: The 46th International Symposium, 2019.
[35] D. Ditomaso, T. Boraten, A. Kodi, A. Louri, Dynamic error mitigation in nocs using intelligent prediction techniques, in: 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2016.
[36] D. Ditomaso, A. Sikder, A. Kodi, A. Louri, Machine learning enabled power-aware network-on-chip design, in: Design, Automation & Test in Europe Conference & Exhibition, 2017.
[37] Y. Dong, A Study on Hardware Design for High Performance Artificial Neural Network by using FPGA and NoC, Ph.D. thesis, Waseda University, 2011.
[38] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, O. Temam, Shidiannao: shifting vision processing closer to the sensor, in: Proceedings of the 42nd Annual International Symposium on Computer Architecture, 2015, pp. 92–104.
[39] G.H. Dunteman, Principal Components Analysis, vol. 69, Sage, 1989.
[40] T. Ebi, D. Kramer, W. Karl, J. Henkel, Economic learning for thermal-aware power budgeting in many-core architectures, in: Proceedings of the 9th International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2011, Part of ESWeek '11 Seventh Embedded Systems Week, Taipei, Taiwan, 9–14 October, 2011, 2011.
[41] M. Ebrahimi, M. Daneshtalab, F. Farahnakian, J. Plosila, H. Tenhunen Haraq, Congestion-Aware Learning Model for Highly Adaptive Routing Algorithm in on-Chip Networks, ACM, 2012.
[42] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, C. Williamson, Offline/realtime traffic classification using semi-supervised learning, Perform. Eval. 64 (2007) 1194–1213.
[43] H. Esmaeilzadeh, A. Sampson, L. Ceze, D. Burger, Neural acceleration for general-purpose approximate programs, in: 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture, 2012, pp. 449–460.
[44] H. Esmaeilzadeh, A. Sampson, L. Ceze, D. Burger, Neural acceleration for general-purpose approximate programs, in: 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture, IEEE, 2012, pp. 449–460.
[45] J. Fan, Z. Wang, Y. Xie, Z. Yang, A theoretical analysis of deep q-learning, in: Learning for Dynamics and Control, PMLR, 2020, pp. 486–489.
[46] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, J. Plosila, P. Liljeberg, Optimized q-learning model for distributing traffic in on-chip networks, in: IEEE International Conference on Networked Embedded Systems for Every Application, 2013.
[47] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, J. Plosila, P. Liljeberg, Adaptive reinforcement learning method for networks-on-chip, in: International Conference on Embedded Computer Systems, 2013.
[48] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, Adaptive load balancing in learning-based approaches for many-core embedded systems, J. Supercomput. 68 (2014) 1214–1234.
[49] F. Farahnakian, M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, Bi-lcq: a low-weight clustering-based q-learning approach for nocs, Microprocess. Microsyst. 38 (2014) 64–75.
[50] C. Feng, Z. Lu, A. Jantsch, J. Li, M. Zhang, A reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning for network-on-chip, 2010.
[51] Q. Fettes, M. Clark, R. Bunescu, A. Karanth, A. Louri, Dynamic voltage and frequency scaling in NoCs with supervised and reinforcement learning techniques, IEEE Trans. Comput. (2018) 1.
[52] D. Fick, A. Deorio, G.K. Chen, V. Bertacco, D. Blaauw, A highly resilient routing algorithm for fault-tolerant nocs, in: Design, Automation & Test in Europe Conference & Exhibition, 2009.
[53] B. Fu, Footprint: regulating routing adaptiveness in networks-on-chip, Comput. Archit. News 45 (2017) 691–702.
[54] D.E. Goldberg, Genetic Algorithm in Search, Optimization, and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., 75 Arlington Street, Suite 300 Boston, MA, United States, 1989.
[55] P. Gratz, B. Grot, S.W. Keckler, Regional congestion awareness for load balance in networks-on-chip, in: 2008 IEEE 14th International Symposium on High Performance Computer Architecture, 2008, pp. 203–214.
[56] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al., Recent advances in convolutional neural networks, Pattern Recognit. 77 (2018) 354–377.
[57] N. Gupta, M. Kumar, A. Sharma, M.S. Gaur, M. Ebrahimi, Improved route selection approaches using q-learning framework for 2d nocs, in: The 3rd International Workshop, 2015.
[58] I.M. Guyon Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. (2003).
[59] R. Hojabr, M. Modarressi, M. Daneshtalab, A. Yasoubi, A. Khonsari, Customizing clos network-on-chip for neural networks, IEEE Trans. Comput. 66 (2017) 1865–1877.
[60] J. Hou, Q. Han, M. Radetzki, A machine learning enabled long-term performance evaluation framework for nocs, in: 2019 IEEE 13th International Symposium on Embedded Multicore/Many-Core Systems-on-Chip (MCSoC), 2019.
[61] Yong Hu, Marcel Mettler, Daniel Mueller-Gritschneder, Thomas Wild, Andreas Herkersdorf, Ulf Schlichtmann, Machine learning approaches for efficient design space exploration of application-specific nocs, ACM Trans. Des. Autom. Electron. Syst. (TODAES) (2020).
[62] P.K. Huynh, M.L. Varshika, A. Paul, M. Isik, A. Balaji, A. Das, Implementing spiking neural networks on neuromorphic architectures: a review, arXiv preprint, arXiv: 2202.08897, 2022.
[63] A. Iranfar, S.N. Shahsavani, M. Kamal, A. Afzali-Kusha, A heuristic machine learning-based algorithm for power and thermal management of heterogeneous mpsocs, in: IEEE/ACM International Symposium on Low Power Electronics & Design, 2015, pp. 291–296.

[64] M. Isik, A survey of spiking neural network accelerator on fpga, arXiv preprint, arXiv:2307.03910, 2023.
[65] M. Isik, K. Inadagbo, H. Aktas, Design optimization for high-performance computing using fpga, arXiv preprint, arXiv:2304.12474, 2023.
[66] K. Jeong, A.B. Kahng, B. Lin, K. Samadi, Accurate machine-learning-based on-chip router modeling, IEEE Embed. Syst. Lett. 2 (2010) 62–66.
[67] B.K. Joardar, R.G. Kim, J.R. Doppa, P.P. Pande, D. Marculescu, R. Marculescu, Learning-based application-agnostic 3d noc design for heterogeneous manycore systems, 2018.
[68] D.C. Juan, D. Marculescu, Power-aware performance increase via core/uncore reinforcement control for chip-multiprocessors, in: ACM/IEEE International Symposium on Low Power Electronics & Design, 2017.
[69] D.C. Juan, H. Zhou, D. Marculescu, L. Xin, A learning-based autoregressive model for fast transient thermal analysis of chip-multiprocessors, in: Asia & South Pacific Design Automation Conference, 2012.
[70] H. Jung, M. Pedram, Supervised learning based power management for multicore processors, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 29 (2010) 1395–1408.
[71] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: a survey, J. Artif. Intell. Res. 4 (1996) 237–285.
[72] E. Kakoulli, V. Soteriou, T. Theocharides, Intelligent hotspot prediction for network-on-chip-based multicore systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 31 (2012) 418–431.
[73] S.C. Kao, C. Yang, P.Y. Chen, X. Ma, T. Krishna, Reinforcement learning based interconnection routing for adaptive traffic optimization, 2019.
[74] P. Kermani, L. Kleinrock, Virtual cut-through: a new computer communication switching technique, Comput. Netw. 3 (1979) 267–286.
[75] S.G. Khawaja, M.U. Akram, S.A. Khan, A. Shaukat, S. Rehman, Network-on-chip based mpsoc architecture for k-mean clustering algorithm, Microprocess. Microsyst. 46 (2016) 1–10.
[76] A.E. Kiasari, Z. Lu, A. Jantsch, An analytical latency model for networks-on-chip, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 21 (2013) 113–123.
[77] J. Kim, J. Balfour, W.J. Dally, Flattened butterfly topology for on-chip networks, in: IEEE/ACM International Symposium on Microarchitecture, 2007, pp. 37–40.
[78] R.G. Kim, Learning-enabled noc design for heterogeneous manycore systems, in: 2020 21st International Symposium on Quality Electronic Design (ISQED), 2020.
[79] R.G. Kim, J.R. Doppa, P.P. Pande, Machine learning for design space exploration and optimization of manycore systems, in: 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), IEEE, 2018, pp. 1–6.
[80] W.J. Kim, J.W. Song, K.S. Chung, On-line learning based dynamic thermal management for multicore systems, in: Isocc, 2008, I–391 – I–394.
[81] M.A. Kinsy, S. Khadka, M. Isakov, Prenoc: neural network based predictive routing for network-on-chip architectures, in: the, 2017.
[82] K. Kira, L.A. Rendell, The feature selection problem: traditional methods and a new algorithm, in: Tenth National Conference on Artificial Intelligence, 1992.
[83] A. Kumar, B. Talawar, Machine learning based framework to predict performance evaluation of on-chip networks, in: 2018 Eleventh International Conference on Contemporary Computing (IC3), 2018.
[84] A. Kumar, B. Talawar, A support vector regression-based approach to predict the performance of 2d & 3d on-chip communication architectures, in: 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), 2019.
[85] G.R. Kumar, Machine learning based resource utilization and pre-estimation for network on chip (noc) communication, Wirel. Pers. Commun., Int. J. 102 (2018).
[86] S. Kumar, R. Miikkulainen, Dual reinforcement q-routing: an on-line adaptive routing algorithm, 1997.
[87] M.C. Lai, G. Lei, X. Nong, Z. Wang, An accurate and efficient performance analysis approach based on queuing model for network on chip, in: IEEE/ACM International Conference on Computer-Aided Design, 2009.
[88] T.T. Le, Z. Dan, M. Bayoumi, Efficient reconfigurable global network-on-chip designs towards heterogeneous cpu-gpu systems: an application-aware approach, in: 2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2017.
[89] Y. Lei, H. Liu, Feature selection for high-dimensional data: a fast correlation-based filter solution, in: Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), Washington, DC, USA, August 21–24, 2003, 2003.
[90] C. Lewis-Beck, M. Lewis-Beck, Applied Regression: An Introduction, vol. 22, Sage Publications, 2015.
[91] C. Li, D. Dong, Z. Lu, X. Liao, Rob-router: a reorder buffer enabled low latency network-on-chip router, IEEE Trans. Parallel Distrib. Syst. (2018) 1.
[92] J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, Feature selection: a data perspective, 2016.
[93] L. Li, Y. Duan, A GA-based feature selection and parameters optimization for support vector regression, in: International Conference on Natural Computation, 2011.
[94] M. Li, Q. an Zeng, W. ben Jone, Dyxy: a proximity congestionaware deadlock-free dynamic routing method for network on chip, in: Proceedings of the 43rd Annual Design Automation Conference, 2006, pp. 849–852.
[95] T.R. Lin, D. Penney, M. Pedram, L. Chen, A deep reinforcement learning framework for architectural exploration: a routerless noc case study, in: 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2020.
[96] T.R. Lin, D. Penney, M. Pedram, L. Chen, Optimizing routerless network-on-chip designs: an innovative learning-based framework, 2019.
[97] T. Luo, S. Liu, L. Li, Y. Wang, S. Zhang, T. Chen, Z. Xu, O. Temam, Y. Chen, Dadiannao: a neural network supercomputer, IEEE Trans. Comput. 66 (2016) 73–88.
[98] Sheng Ma, N. Jerger, Z. Wang, Dbar: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip, in: ACM SIGARCH Computer Architecture News, vol. 39, 2011, pp. 413–424.
[99] D. Mahajan, J. Park, E. Amaro, H. Sharma, A. Yazdanbakhsh, J.K. Kim, H. Esmaeilzadeh, Tabla: a unified template-based framework for accelerating statistical machine learning, in: 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2016, pp. 14–26.
[100] H. Matsutani, M. Koibuchi, D. Ikebuchi, K. Usami, H. Amano, Ultra fine-grained run-time power gating of on-chip routers for cmps, in: Fourth Acm/IEEE International Symposium on Networks-on-Chip, 2010.
[101] L.R. Medsker, L. Jain, Recurrent neural networks, Des. Appl. 5 (2001) 64–67.
[102] S.Y.H. Mirmahaleh, A.M. Rahmani, Dnn pruning and mapping on noc-based communication infrastructure, Microelectron. J. 94 (2019) 104655.
[103] R. Morris, A.K. Kodi, A. Louri, 3d-noc: reconfigurable 3d photonic on-chip interconnect for multicores, in: 2012 IEEE 30th International Conference on Computer Design (ICCD), IEEE, 2012, pp. 413–418.
[104] M.G. Omran, A.P. Engelbrecht, A. Salman, An overview of clustering methods, Intell. Data Anal. 11 (2007) 583–605.
[105] M. Otoom, P. Trancoso, H. Almasaeid, M. Alzubaidi, Scalable and Dynamic Global Power Management for Multicore Chips, ACM, 2015, pp. 25–30.
[106] S. Pagani, D. Smp, A. Jantsch, J. Henkel, Machine learning for power, energy, and thermal management on multi-core processors: a survey, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2018) 1.
[107] D.D. Penney, L. Chen, A survey of machine learning applied to computer architecture design, 2019.
[108] M.K. Puthal, V. Singh, M.S. Gaur, V. Laxmi, C-routing: an adaptive hierarchical noc routing methodology, in: IEEE/IFIP International Conference on Vlsi & System-on-Chip, 2011.
[109] Z. Qian, D.C. Juan, P. Bogdan, C.Y. Tsui, R. Marculescu, Svr-noc: a performance analysis tool for network-on-chips using learning-based support vector regression model, in: Design, Automation & Test in Europe Conference & Exhibition, 2013.
[110] Z.L. Qian, D.C. Juan, P. Bogdan, C.Y. Tsui, D. Marculescu, R. Marculescu, A support vector regression (svr)-based latency model for network-on-chip (noc) architectures, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 35 (2016) 471–484.
[111] J.R. Quinlan, C4.5: programs for machine learning, 1993.
[112] M. Ramakrishna, P.V. Gratz, A. Sprintson Gca, Global congestion awareness for load balance in networks-on-chip, in: 2013 Seventh IEEE/ACM International Symposium on Networks-on-Chip (NoCS), 2013, pp. 1–8.
[113] N. Rao, A. Ramachandran, A. Shah, Mlnoc: a machine learning based approach to noc design, in: International Symposium on Computer Architecture and High Performance Computing, 2018.
[114] M.F. Reza, Reinforcement learning based dynamic link configuration for energy-efficient noc, in: 63rd IEEE International Midwest Symposium on Circuits and Systems, 2020.
[115] M.F. Reza, Deep reinforcement learning for self-configurable noc, in: 33rd IEEE International System-on-Chip Conference (SOCC), 2020.
[116] M.F. Reza, T.T. Le, Reinforcement learning enabled routing for high-performance networks-on-chip, in: 54th IEEE International Symposium on Circuits and Systems (ISCAS), 2021.
[117] M.F. Reza, T.T. Le, B. Dey, M. Bayoumi, Z. Dan, Neuro-noc: energy optimization in heterogeneous many-core noc using neural networks in dark silicon era, in: IEEE International Symposium on Circuits and Systems (ISCAS), 2018.
[118] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, Technical Report, California Univ San Diego la Jolla Inst for Cognitive Science, 1985.
[119] J. Samala, H. Takawale, Y. Chokhani, P.V. Bhanu, J. Soumya, Fault-tolerant routing algorithm for mesh based noc using reinforcement learning, in: 2020 24th International Symposium on VLSI Design and Test (VDAT), 2020.
[120] K. Sangaiah, M. Hempstead, B. Taskin, Uncore rpd: rapid design space exploration of the uncore via regression modeling, in: 2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2015.
[121] A.G. Savva, T. Theocharides, V. Soteriou, Intelligent on/off dynamic link management for on-chip networks, J. Electr. Comput. Eng. 2012 (2012) 343–344.
[122] B. Schölkopf, A.J. Smola, F. Bach, et al., Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, 2002.
[123] G.A. Seber, A.J. Lee, Linear Regression Analysis, vol. 330, John Wiley & Sons, 2003.
[124] D. Seo, A. Ali, W.T. Lim, N. Rafique, M. Thottethodi, Near-optimal worst-case throughput routing for two-dimensional mesh networks, in: ACM SIGARCH Computer Architecture News, 2005.
[125] Y. Sheng, R.A. Shafik, G.V. Merrett, E. Stott, Adaptive energy minimization of embedded heterogeneous systems using regression-based learning, in: 25th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS 2015), 2015.
[126] J. Silva, M. Kreutz, M. Pereira, M. Da Costa-Abreu, An investigation of latency prediction for noc-based communication architectures using machine learning techniques, J. Supercomput. 75 (2019) 7573–7591.

[127] M. Song, J. Zhang, H. Chen, T. Li, Towards efficient microarchitectural design for accelerating unsupervised gan-based deep learning, in: 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2018, pp. 66–77.
[128] V. Soteriou, E. Kakoulli, T. Theocharides, Hpra: a Pro-Active Hotspot-Preventive High-Performance Routing Algorithm for Networks-on-Chips, IEEE, 2012.
[129] V. Soteriou, T. Theocharides, E. Kakoulli, A holistic approach towards intelligent hotspot prevention in network-on-chip-based multicores, IEEE Trans. Comput. 65 (2016) 819–833.
[130] C. Sudusinghe, S. Charles, P. Mishra, Denial-of-service attack detection using machine learning in network-on-chip architectures, in: 2021 15th IEEE/ACM International Symposium on Networks-on-Chip (NOCS), 2021, pp. 35–40.
[131] H. Sullivan, T.R. Bashkow, A large scale, homogeneous, fully distributed parallel machine, I, in: Acm Sigarch Computer Architecture News, 1977, pp. 105–117.
[132] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[133] J.R. uinlan, Induction of decision trees, Mach. Learn. 1 (1986) 81–106.
[134] J.E. Van Engelen, H.H. Hoos, A survey on semi-supervised learning, Mach. Learn. 109 (2020) 373–440.
[135] M.M. Waldrop, The chips are down for Moore's law, Nature 530 (2016).
[136] B. Wang, Z. Lu, S. Chen, Ann based admission control for on-chip networks, in: Proceedings of the 56th Annual Design Automation Conference 2019, 2019, pp. 1–6.
[137] C. Wang, D. Dong, Z. Wang, X. Zhang, Z. Zhao, Relar: a reinforcement learning framework for adaptive routing in network-on-chips, in: 2021 IEEE International Conference on Cluster Computing (CLUSTER), 2021.
[138] C. Wang, Z. Wang, D. Dong, X. Zhang, Z. Zhao, A novel reinforcement learning framework for adaptive routing in network-on-chips, in: 2021 IEEE 23rd Int Conf on High Performance Computing (HPCC), Haikou, Hainan, China, December 20–22, 2021, IEEE, 2021, pp. 336–344.
[139] K. Wang, A. Louri Cure, A high-performance, low-power, and reliable network-on-chip design using reinforcement learning, IEEE Trans. Parallel Distrib. Syst. (2020) 1.
[140] K. Wang, A. Louri, A. Karanth, R. Bunescu, High-performance, energy-efficient, fault-tolerant network-on-chip design using reinforcement learnin, in: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2019.
[141] K. Wang, A. Louri, A. Karanth, R. Bunescu, Intellinoc: a holistic design framework for energy-efficient and reliable on-chip communication for manycores, in: International Symposium on Computer Architecture, 2019.
[142] K. Wang, H. Zheng, A. Louri, Tsa-noc: learning-based threat detection and mitigation for secure network-on-chip architecture, IEEE MICRO (2020) 1.
[143] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (1992) 279–292.
[144] S. Werner, J. Navaridas, Mikel Luján, A survey on optical network-on-chip architectures, ACM Comput. Surv. (CSUR) (2017).
[145] S.V. Winkle, A.K. Kodi, R. Bunescu, A. Louri, Extending the power-efficiency and performance of photonic interconnects for heterogeneous multicores with machine learning, in: 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2018.
[146] J.-Y. Won, X. Chen, P. Gratz, J. Hu, V. Soteriou, Up by their bootstraps: online learning in artificial neural networks for cmp uncore power management, in: 2014 IEEE 20th International Symposium on High-Performance Computer Architecture (HPCA), 2014, pp. 308–319.
[147] J. Wu, Fault-tolerant adaptive and minimal routing in mesh-connected multicomputers using extended safety levels, in: International Conference on Distributed Computing Systems, 1998.
[148] N. Wu, Y. Xie, A survey of machine learning for computer architecture and systems, ACM Comput. Surv. (CSUR) (2021).
[149] Y. Wu, L. Wang, X. Wang, J. Han, J. Zhu, H. Jiang, S. Yin, S. Wei, L. Liu, Upward packet popup for deadlock freedom in modular chiplet-based systems, in: 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2022, pp. 986–1000.
[150] J. Yang, Z. Zhang, Z. Liu, J. Zhou, L. Liu, S. Wei, S. Yin, Fusekna: fused kernel convolution based accelerator for deep neural networks, in: 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021, pp. 894–907.
[151] Y. Ye, W. Zhang, W. Liu, Thermal-aware design and simulation approach for optical nocs, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 39 (2019).
[152] J. Yin, Y. Eckert, S. Che, M. Oskin, G.H. Loh, Toward more efficient noc arbitration: a deep reinforcement learning approach, in: Proc. IEEE 1st Int. Workshop AI-Assisted Des. Architecture, vol. 128, 2018.
[153] J. Yin, Z. Lin, O. Kayiran, M. Poremba, M. Shoaib, Bin Altaf, N. Enright Jerger, G.H. Loh, Modular routing design for chiplet-based systems, in: 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), 2018, pp. 726–738.
[154] J. Yin, S. Sethumurugan, Y. Eckert, C. Patel, A. Smith, E. Morton, M. Oskin, N.E. Jerger, G.H. Loh, Experiences with ml-driven design: a noc case study, in: 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA), IEEE, 2020, pp. 637–648.
[155] F. Yu, D. Wang, L. Shangguan, M. Zhang, C. Liu, X. Chen, A survey of multi-tenant deep learning inference on gpu, arXiv preprint, arXiv:2203.09040, 2022.
[156] X. Zhai, A. Oliver, A. Kolesnikov, L. Beyer, S4l: self-supervised semi-supervised learning, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 1476–1485.
[157] K. Zhang, Z. Yang, T. Baar, Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms, Springer, Cham, 2021.
[158] W. Zhang, Y. Ye, A table-free approximate q-learning-based thermal-aware adaptive routing for optical nocs, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 40 (2020).
[159] Z. Zhang, A. Greiner, S. Taktak, A reconfigurable routing algorithm for a fault-tolerant 2d-mesh network-on-chip, in: Design Automation Conference, DAC 2008, 45th ACM/IEEE, 2008, 2008.
[160] H. Zheng, A. Louri, An energy-efficient network-on-chip design using reinforcement learning, in: 2019 56th ACM/IEEE Design Automation Conference (DAC), 2019.
[161] H. Zheng, K. Wang, A. Louri, Adapt-noc: a flexible network-on-chip design for heterogeneous manycore architectures, in: 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021.
[162] Y. Zhou, H. Wang, J. Yin, Z. Zhang, Distilling arbitration logic from traces using machine learning: a case study on noc, in: 2021 58th ACM/IEEE Design Automation Conference (DAC), IEEE, 2021, pp. 55–60.
[163] K. Zhu, M. Liu, Y. Lin, B. Xu, S. Li, X. Tang, N. Sun, D.Z. Pan Geniusroute, A new analog routing paradigm using generative neural network guidance, in: 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2019, pp. 1–8.
[164] C. Zhuo, D. Marculescu, Distributed reinforcement learning for power limited many-core system performance optimization, in: Design, Automation & Test in Europe Conference & Exhibition, 2015.

Xiaoyun Zhang received her B.S. degree in computer science and technology from Zhengzhou University, Zhengzhou, China, in 2016. She received her M.S. degree in software engineering from Xi'an University of Science and Technology, Xi'an, China, in 2019. She is currently a doctor candidate of Computer Science in the National University of Defense Technology. Her research interests include high performance computing and network-on-chips.



Dezun Dong received his B.S., M.S., and Ph.D. degrees from the National University of Defense Technology (NUDT), Changsha, China, in 2002, 2004, and 2010, respectively. He is a professor in the College of Computer, NUDT, where he leads the research group of high-performance network and architecture (HiNA) and serves as the deputy director designer of the Tianhe supercomputer. His research interests focus on high-performance networks and architecture for supercomputers, datacenter, and deep learning systems. He has published over 100 peer-reviewed papers in reputed international journals and conferences.

Cunlu Li received his B.S. degree in computer science and technology from Tsinghua University, Beijing, China, in 2012. He received his M.S. and Ph.D. degrees at the National University of Defense Technology (NUDT), China, in 2015, and 2019, respectively. He is currently a research associate at the College of Computer, NUDT, China. His current research interests are high performance computing, network-on-chips, and distributed computing.

Shaocong Wang received his B.S. degree in Pharmaceutics from the Wuhan Institute of Technology, Wuhan, China, in 2020. He is currently pursuing a master's degree in the College of Computer at the National University of Defense Technology. His research interests include high performance computing and network-on-chips.

Liquan Xiao received his M.S. and Ph.D. degrees in computer science and technology from the National University of Defense Technology (NUDT), Changsha, China, in 1994, and 1997, respectively. Currently, he is a professor at NUDT. His research interests include architecture of high performance computing, high speed interconnect networks, system integration, and power management.