Journal of Systems Architecture 135 (2023) 102810

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

Hardware-aware neural architecture search for stochastic computing-based neural networks on tiny devices

Yuhong Song [a], Edwin Hsing-Mean Sha [a,*], Qingfeng Zhuge [a], Rui Xu [a], Xiaowei Xu [b], Bingzhe Li [c], Lei Yang [d]

[a] Software/Hardware Co-design Engineering Research Center, Ministry of Education and the School of Computer Science and Technology, East China Normal University, Shanghai 200062, China
[b] Guangdong Cardiovascular Institute, Guangdong Provincial Key Laboratory of South China Structural Heart Disease, Guangdong Provincial People's Hospital, Guangdong Academy of Medical Science, Guangzhou 510080, China
[c] Department of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74074, USA
[d] Department of Information Sciences and Technology, George Mason University, Fairfax, VA 22030, USA

ARTICLE INFO

Keywords:
Stochastic computing
Hardware/software co-optimization
Neural architecture search

ABSTRACT

Along with the progress of artificial intelligence (AI) democratization, there is an increasing potential for the deployment of deep neural networks (DNNs) to tiny devices, such as implantable cardioverter defibrillators (ICD). However, tiny devices with extremely limited energy supply (e.g., battery) have high demands on low-power execution, while guaranteeing the model accuracy. Stochastic computing (SC) as a new promising paradigm significantly reduces the power consumption of DNNs by simplifying arithmetic circuits, but often sacrifices the model accuracy. To make up for the accuracy loss, previous works mainly focus on either only-hardware (only-HW) circuit design or software-to-hardware (SW→HW) sequential workflow, which leads to unilateral optimization. Therefore, as the first attempt, aiming at both the hardware (HW) and software (SW) performance, we innovatively propose an HW↔SW co-exploration framework for SC-based NNs, namely SC-NAS, which is the first to couple SC with neural architecture search (NAS) for HW/SW co-optimization. We redefine the optimization problem and show a complete workflow to intelligently search for a set of configurations of NNs with hardware consumption as low as possible and accuracy as high as possible. We comprehensively explore the influencing factors, which have impacts on both the HW and SW performance for SC-based NNs, to build a search space for NAS. Furthermore, in order to improve the search efficiency of NAS, we contract the search space and set an energy constraint to early terminate unnecessary model inference. Experiments show that SC-NAS achieves up to 7.0 × energy saving than FP-NAS, and exceeds pure SC methods by over 3.8% in accuracy. Meanwhile, SC-NAS obtains around 8.6 × 10^{19} × search efficiency improvement than exhaustive search using VGGNet.

1. Introduction

With the rapid development of artificial intelligence (AI), deep neural networks (DNNs) as typical machine learning (ML) models have been widely used on edge devices [1,2], such as smartphones and automated driving. Recently, there are increasing demands for the deployment of DNNs to tiny devices [3,4], such as surveillance cameras and healthcare sensors, which equip extremely limited resources. In particular, the energy supply (e.g., battery) for tiny devices is critical and strictly constrained, which requires low-power DNN implementations, while the inference accuracy of DNNs should be guaranteed. Stochastic computing (SC) [5–7] as a promising paradigm stands out in power reduction because of the particular data representation and computing scheme. It represents data using 0/1 bit streams and performs fundamental ML operations (i.e., multiply-accumulate, MAC) using only simple logic gates instead of complicated binary arithmetic circuits.

In the SC system, any real-valued number x ($0 \leq x \leq 1$) is represented by a sequence of random bits, each of which has probability p of being '1' and probability $1-p$ of being '0'. In this way, the expected frequency of '1' in the bit stream is p. For the 'unipolar' representation, the real-valued number x is equal to p, such as the data a and b in Fig. 1(a) and (c). There are four '1's in the data a and two '1's in the

* Corresponding author.
E-mail addresses: yhsong@stu.ecnu.edu.cn (Y. Song), edwinsha@cs.ecnu.edu.cn (E.H.-M. Sha), qfzhuge@cs.ecnu.edu.cn (Q. Zhuge), ruixu@stu.ecnu.edu.cn (R. Xu), xiao.wei.xu@foxmail.com (X. Xu), bingzhe.li@okstate.edu (B. Li), lyang29@gmu.edu (L. Yang).

https://doi.org/10.1016/j.sysarc.2022.102810
Received 16 September 2022; Received in revised form 7 December 2022; Accepted 17 December 2022
Available online 22 December 2022

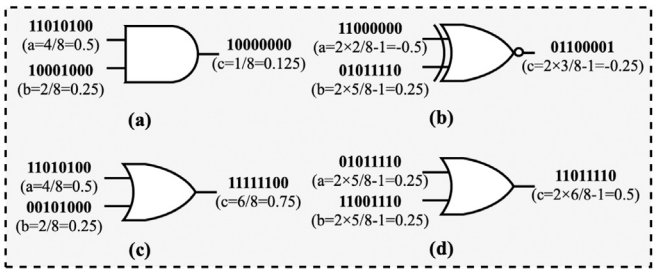

Fig. 1. Examples of SC arithmetic circuits: (a) AND gate for unipolar multiplication; (b) XNOR gate for bipolar multiplication; (c) OR gate for unipolar addition; and (d) OR gate for bipolar addition.

data b using 8-bit stream, so the real-valued number $a = 4/8 = 0.5$ and $b = 2/8 = 0.25$. Meanwhile, p can be transformed by $2p - 1$ to represent the data x range from -1 to 1, that is $x = 2p - 1$, this is called 'bipolar' data, such as the a and b in Fig. 1(b) and (d). The SC bit stream is generated using a stochastic number generator (SNG), which guarantees the randomness of the 0/1 bits. Conversely, the SC bit streams can be converted to floating-point (FP) data by counting the number of '1's, this operation can be implemented using simple gates (e.g., counter and shifters).

Using SC representation, the arithmetic operations of unipolar and bipolar data can be approximated using the theory of probability computing. In an SC system, the multiplication can be approximated using only an AND gate for two unipolar inputs (see Fig. 1(a)), and using an XNOR gate for two bipolar inputs (see Fig. 1(b)). Similarly, an OR gate can be used to approximate addition for both unipolar and bipolar data (see Fig. 1(c) and (d)). Obviously, the complexity of arithmetic circuits is significantly simplified, which greatly decreases the HW cost in terms of area and power consumption compared with traditional binary circuits (e.g., FP multiplier and adder). With the SC multipliers and adders, MAC operation can be accomplished. It also means that the DNNs can be implemented using these simple circuits to reduce HW area and power consumption for tiny devices.

Although SC brings HW area and power profits, it introduces huge accuracy loss in DNNs inference. The low data precision (i.e., the bit length of the bit stream) and the inaccuracy of arithmetic circuits (e.g., example in Fig. 1(b) cannot obtain the accurate result), etc. will cause computing errors. In order to improve the accuracy, existing works [8–12] mainly focus on either only-HW or SW→HW sequential designs. The only-HW means that the works [8–10] mainly focus on the circuit design and implementation. Correspondingly, the power, latency and resource usage of DNN inference can be obtained by HW deployment. Except for the circuit implementation, some works [11, 12] involve the SW process (e.g., data pre-treatment and model pre-training) before HW implementation. The methods are verified on local host for better accuracy, which are called SW→HW designs. Since these works pay more attention to the performance of accuracy, better HW/SW trade-offs are ignored. **Therefore, to achieve a better trade-off, we creatively propose an HW ↔ SW co-optimization framework for SC-based NNs, which does not only consider the performance of HW or SW, but co-design.** For better classifying, we view the power, latency and resource usage as the HW metrics, and accuracy as the SW metric.

First, we explore to identify the factors, which have impacts on both HW and SW performance for SC-based NNs. After exploring, we try to figure out a group of settings with HW and SW performance as high as possible for SC-based NNs. However, with our observations and experiments, there are diverse influencing factors. And for each factor, multiple options can be chosen, which significantly increases the combination space of factors. Therefore, how to make the best choices among these factors for both HW and SW consideration becomes a challenge. It is natural that exhaustive search (ES) can obtain the global optimal results. However, the time and resources overhead of ES is undoubtedly enormous. Fortunately, recent neural architecture search (NAS) performs well by utilizing automated ML (autoML), such as reinforcement learning (RL) and revolutionary algorithms, to efficiently search for a near-optimal DNNs architecture from the pre-built discrete search space [13–16]. The strategies utilize the best configurations after the search process as the final result. And they can converge to an optimized result than the intuitive setting. However, the techniques cannot be implemented directly for customized field. In this paper, as the first attempt, we couple SC with NAS to form the SC-NAS framework. We formulate the co-optimization problem and design a complete workflow to search for a group of settings of SC-based NNs with HW consumption as little as possible and inference accuracy as high as possible. Nevertheless, too huge a search space will weaken the search efficiency. Therefore, we first fully explore to build a contracted search space, and then early terminate the model inference to improve the overall efficiency.

The main contributions of this paper are listed as follows.

- First, we creatively propose the HW↔SW co-exploration and formulate the co-optimization problem for SC-based DNNs. In order to obtain a better HW/SW trade-off, as the first attempt, we propose an automated framework, namely SC-NAS, which couples SC to reduce the HW energy consumption with NAS to intelligently search for the excellent settings of SC-based NNs.
- Moreover, aiming to improve the search efficiency of SC-NAS, we explore the influencing factors of HW and SW metrics through comprehensive experiments, then build a contracted and integrated search space. Furthermore, we set an energy constraint for early termination to save the overall search time.
- Experiments show that SC-NAS achieves 7.0 × energy saving than FP-NAS, and the accuracy exceeds pure SC methods by 3.8%. Furthermore, the search time is saved up to $8.6 \times 10^{19}\times$ compared with ES for VGGNet.

This paper is organized as follows: In Section 2, we present the related works and motivations of this paper. Section 3 introduces our proposed architecture, SC-NAS. Experimental results are demonstrated in Section 4. Section 5 makes the conclusion of this paper.

2. Related works and motivations

2.1. Related works

Pure SC methods. Pure SC represents the methods that just consider SC designs without the NAS involving. In order to improve the accuracy of SC-based NNs, existing works mainly focus on two perspectives: **(1) only-HW arithmetic circuit designs** [8–10] and **(2) following an SW→HW sequential workflow** [11,12]. Because MAC is the basic operation in ML, we mainly focus on the designs of SC multipliers and adders. Note that we only discuss the circuits designed for bipolar data, because all the data in DNNs are signed.

(1) [8] utilizes XNOR gate as the multiplier and proposes a MUX-tree adder (MUXADD) to approximate the addition. However, it meets severe correlation problem [17], which makes the preconditions of probability computing for circuits invalid and incurs large computation errors. [9] separates positive data POS and negative data NEG, it utilizes AND gate as the multiplier. Meanwhile, it proposes a new adder, which computes the POS and NEG separately, then utilizes $\frac{POS-NEG+1}{2}$ to transform the results. We call the adder SEPADD. When the number of inputs increases, SEPADD will encounter the overflow problem without data scaling. uGEMM [10] designs new multiplier (uMUL) and adder (uNSADD) for bipolar data, it utilizes the parallel counter (PC) to mitigate the correlation problem from inputs. But it will occur redundant or missing '1's problem in some special cases. [18] further proposes an intra-block adder based on accumulators to reduce

Table 1
Comparisons of SC works. Our work distinguishes from others with HW/SW co-optimization.

Method	Only-HW			SW-HW		Ours
	[8]	[9]	[10]	[11]	[12]	
Multiplier	XNOR	AND	uMUL	XNOR	AND	{XNOR,...}
Adder	MUXADD	SEPADD	uNSADD	MUXADD	APC	{MUXADD,...}
Eneg. (mJ)	11.4	13.0	38.1	–	–	14.5
MAE	1.67	1.43	0.72	–	–	0.21
SW Procs?	✗	✗	✗	✓	✓	✓
Co-optm?	✗	✗	✗	✗	✗	✓

the computing errors and designs an output revision scheme (OUR) to solve the redundant/missing '1's problem among blocks.

(2) From the aspect of SW→HW workflow, [11] concludes that the inaccuracy is maximal when the multiplication result is zero using the XNOR gate. So it first removes near-zero operands at the SW side, then implements to HW. [12] observes the correlation problem is mainly from the XNOR multiplier for bipolar through statistical analysis. So, it proposes a new data format to represent signed data, namely sign-magnitude (SM), which utilizes an additional sign bit in front of the unipolar absolute bit stream. Then it re-designs the circuits using AND gate as the multiplier. But its adder, accumulative PC (APC), can only compute one input in one adder, which is unfriendly for computing pipeline.

For better demonstration, we conclude these works in Table 1. Whether only-HW or SW→HW methods, they all do not consider HW and SW co-optimization. They aim to design high-accuracy circuits for SC-based NNs, ignoring the HW performance. Therefore, our work proposes the HW/SW co-exploration, which takes both HW and SW performance into consideration.

FP-NAS. NAS as a widely used model compression technology [19] can search a compact architecture and decrease the energy consumption of DNNs to a certain extent. Due to the vast architecture design space, automated NAS becomes popular. In existing automated NAS, the search algorithms are mainly based on RL[13–15,20,21] and evolutionary search [16,22], which provide the most competitive results. These two search algorithms are well suited for discrete search space. RL-based NAS trains an ML-based controller to intelligently search the best model architecture from the pre-built search space. A reward is calculated from the environment to update the RL controller to make better future actions, which greatly promotes the search process. Instead of training an ML-based controller, an evolutionary-based NAS maintains a population of model architectures. This population is updated through mutation and recombination. Because RL-based NAS makes the decision by interacting with the environment, it takes advantage of the details of individual behavioral interactions, which can be much more efficient than evolutionary-based NAS in many cases. Therefore, in this paper, we utilize RL algorithm as the automated NAS method. For traditional RL-based NAS, it produces compact DNNs computed by complex FP multiplier and adder. Coupling with SC, the HW circuits can be further simplified and energy is significantly saved. However, SC cannot be directly mapped to the architecture searched from NAS, because both the compact model and inaccuracy arithmetic circuits will cause huge accuracy loss. Meanwhile, the optimization problem and framework should be re-designed to face the specific SC problem.

2.2. Motivations

Motivation 1: Only-HW or SW→HW workflow cannot obtain a great HW and SW trade-off for SC-based NNs. Table 1 demonstrates the distinctions between our work and related works [8–12]. The energy consumption (i.e., power × latency) and the mean absolute error (MAE) of three only-HW arithmetic circuits under a 32-dimensions

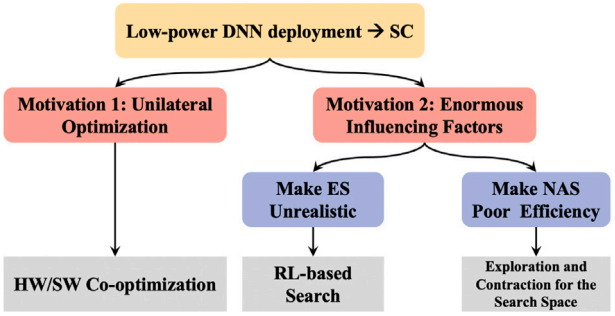

Fig. 2. Relationship of motivations and corresponding strategies.

MAC using 64 bits SC data are shown. Except the arithmetic circuit, other factors (e.g., the bit length of SC data) are set by default. For DNNs deployment, existing SC-based works mainly take advantage of low-power characteristic of SC and aim at the accuracy problem. However, these designs generally cannot obtain a great HW/SW trade-off, because the HW performance (e.g., power, latency and resource usage) in their works is not fully investigated, which leads to unilateral optimization. For example, work [10] owns the lowest computation error but the highest energy than other works. Even though some works obtain a better HW performance, they sacrifice computational accuracy. For example, work [8] holds the lowest energy consumption but the highest MAE. Focusing on only one side of optimization tends to sacrifice other performance. Without HW/SW co-design and problem formulation, the settings of SC-based NNs cannot be determined easily. Exploring and involving the influencing factors in a unified search space and searching for the final solutions are necessary. Because our work does not involve the pre-treatment methods on specific situation, the results of SW→HW are not shown here.

Motivation 2: Various factors make exhaustive search unrealistic and make neural architecture search poor efficiency. In our implementations, we observe that there are various factors that can influence both the HW and SW metrics for DNNs. Relying on intuition, we cannot easily tell which is the best setting. For conventional DNNs, the model architecture first influences the performance. For SC-based DNNs, the influencing factors further include the arithmetic circuits and bit length of SC data, etc, which additionally increase the search space. In each SC-based NNs inference, a set of model architecture and SC-related factors need to be determined. Then conduct the SC-based NNs' forward computation to obtain the model accuracy and computation energy, etc. However, there are tens of choices for each factor, which makes lots of configuration combinations. For example, for the number of neurons of each fully-connected layer, it can be set as 32, 64, 100, 128, 200, 256, 400, 512, 1024, 4096, etc. For the bit length of SC data, it can be set as 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, etc. Without exploration, the number of combinations can exceed 30 orders of magnitude. By using ES, each set of configurations is utilized to perform the SC-based NNs inference once, then select a set of factors with the best performance as the final result. It will take tens of thousands of years to exhaustively search for the optimal solution. The time and resource consumption are undoubtedly unacceptable. Besides, SC-based NNs possess longer inference latency than FP-based NNs under the same parallelism, which further makes ES unrealistic within acceptable time and resources. Therefore, efficient search algorithms are necessary.

Fortunately, RL-based NAS performs well in searching the near-optimal results intelligently from the discrete search space using fewer search episodes. Nevertheless, NAS cannot be implemented directly for the customized fields. The corresponding optimization problem needs to be defined, and the search space needs to be fully explored. Meanwhile, the environment and reward of RL need to be established. Though RL-based NAS can accelerate search, too huge discrete search

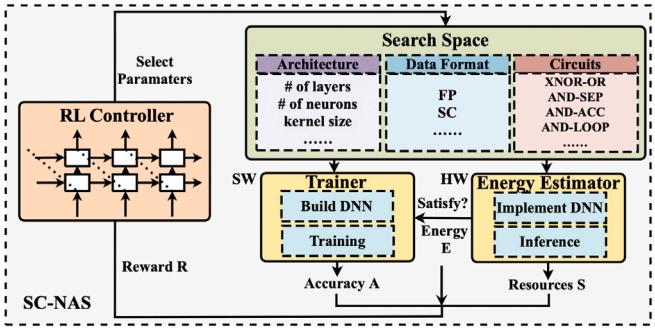

Fig. 3. Overview of SC-NAS framework.

space will also consume enormous time and resources. How to efficiently utilize RL-based NAS to search for the better model architecture and SC-based factors will be explored in this paper. The relationship of motivations and corresponding strategies are shown in Fig. 2 for better understanding.

3. Architecture of SC-NAS framework

3.1. Overview

In this work, to achieve the HW and SW co-optimization, we propose an RL-based framework, namely SC-NAS, to efficiently search a group of settings with tiny hardware consumption satisfying expected accuracy for SC-based NNs. As mentioned above, various factors affect both the HW and SW performance of model inference. The vast discrete search space will make poor search efficiency for RL-based NAS. Therefore, we contract the search space and set an energy constraint to early terminate SC-based NNs inference. The co-optimization problem of this paper is formulated as follows.

Definition 1. Given a target hardware platform, a training dataset, a discrete contracted search space $SS = \{MASS, DFSS, ACSS\}$ and an energy constraint E_c, our SC-NAS is to determine: a set of model configurations $CF = \{mass, dfss, acss\}$. The $MASS$ is the given model architecture search space, $DFSS$ is the given data format search space and $ACSS$ is the given arithmetic circuit search space. The resulting configuration CF includes the resulting factors $mass$, $dfss$, $acss$ from $MASS$, $DFSS$ and $ACSS$, respectively. The objective of SC-NAS is to find a set of configuration CF, which possesses the best HW and SW performance trade-off. Specifically, we want an SC-based NNs implementation on tiny devices with the energy and resource usage as little as possible, and the inference accuracy as much as possible simultaneously. The energy constraint E_c should be satisfied. The objective function of SC-NAS can be formulated as:

$$\begin{aligned} &Maximum \quad \alpha \cdot Eng(CF) + \beta \cdot Res(CF) + \gamma \cdot Acc(CF) \\ &s.t. \quad pre-set\ energy\ constraint\ E_c\ \ is\ satisfied \end{aligned} \tag{1}$$

The $Eng(CF)$, $Res(CF)$ and $Acc(CF)$ identify the transformation functions of obtained energy consumption, resource usage and accuracy performance using selected configuration CF, respectively. The functions utilize normalization operation to standardize all metrics to the same specification (i.e., the range [0, 1]), which eliminates the influences of specific values being too large or too small. Meanwhile, all the values are transformed to the larger the better. The α, β and γ are the corresponding coefficients for each performance metric to point out the importance of this metric. The weighted sum of these transformed metrics reflects a performance trade-off. Note that the coefficients α, β and γ can be adjusted according to the requirements of different application scenarios. In this paper, we consider the energy consumption (i.e., power × latency) and resource (i.e., LUT and LUTRAM) usage of SC-based NNs as the HW metrics and the accuracy as the SW metric. The specific calculation formulas of $Eng(CF)$, $Res(CF)$ and $Acc(CF)$ are shown in Section 3.5. An expected accuracy is set to formulate the reward of SC-NAS.

Table 2
State space of RL framework.

State	Description
E	Energy consumption of model inference for selected action
S	Resource usage of model inference for selected action
A	Accuracy of model inference for selected action

Next, we demonstrate the complete workflow of our proposed framework. The overview of SC-NAS is shown in Fig. 3. First, an RL controller based on recurrent neural network (RNN) (see Section 3.2) is built to select parameters from the pre-built discrete search space. The search space is established by fully exploring the influencing factors and is contracted from a huge space (see Section 3.3). When a set of parameters is selected by the RL controller from the contracted search space, the corresponding DNN is modeled based on the parameters. When the inference energy consumption E, which calculates by the energy estimator (see Section 3.5), satisfies our pre-set energy constraint E_c, the model trainer is launched to train and infer for accuracy A (see Section 3.4). Meanwhile, the resource usage S is obtained by NNs inference. Finally, after DNN training and HW implementation, the reward is computed to update the RNN-based controller and guide the next selection (see Section 3.5). We call such one round selection an episode. Details are introduced as follows.

3.2. Reinforcement learning controller

Driven by various discrete influencing factors, we implement RL-based SC-NAS to intelligently predict a better result. RL controller is the key component of SC-NAS. The controller based on deep learning (DL) has great advantages to converge to the global optimal solutions. It can automatically learn from the environment and update itself to make a better decision in future episodes. In this paper, the RL controller is the agent of RL field. The agent predicts an action in the action space by observing the state of system environment. Based on selected action, the environment will feedback a reward to indicate whether the action is good or bad for the environment. A higher reward means a better action. The objective of RL is to maximize the reward.

State space. In this paper, we utilize the energy consumption E, resource usage S and accuracy A for selected action (i.e., a set of configurations for SC-based NNs) in each episode to depict the state of current environment. The state space is shown in Table 2. RL controller updates itself by learning past experience based on previous actions and corresponding reward values. By training RL controller, it can predict a better set of configurations from action space (i.e., search space). The state space comprehensively reflects HW and SW performance of SC-based NNs inference, which makes RL controller can better understand the complex system environment and timely capture the dynamic changes of the environment.

Action space. In this paper, the action space includes three parts: model architecture search space (MASS), which consists of the configurations of model architecture; data format search space (DFSS), which consists of the data format and bit length of SC data; arithmetic circuit search space (ACSS), which consists of a SC circuit library. Details are shown in Section 3.3.

Agent. In our SC-NAS framework, the RL controller is implemented based on an RNN, which is similar to [13]. RNN-based NNs can commendably extract the context relationship and retain information for sequence inputs. So, using the RNN-based NNs as the RL controller is conducive to make better predictions by learning the experience in previous actions. In this paper, we utilize LSTM as the RL controller, which

Table 3
Integrated search space includes MASS, DFSS and ACSS.

MASS	# of layers: {1, 2, 3,...}; # of neurons for each layer: {32, 64, 100, 128,...} kernel size for convolutional and pooling layers: {2, 3, 5, 7,...} stride for convolutional and pooling layers: {1, 2,...}
DFSS	Data format: {FP, SC} for each layer; Bit length of SC data: {4, 8, 16, 32,...}
ACSS	Arithmetic circuits library: {XNOR-MUX, AND-SEP, uGEMM, AND-ACC, AND-LOOP,...}

can well solve the problem of gradient disappearance and explosion in the training process.

First, RL controller predicts a set of parameters from the search space that we built (in Section 3.3) based on a softmax classifier. The selected parameters by the RL controller are applied to build corresponding DNNs. After SW DNNs training on the local host and HW board implementation, the energy consumption E, resource usage S and accuracy A are obtained, which are used to formulate the reward R (in Section 3.5). Then, based on the reward R, a Monte Carlo policy gradient algorithm [23] is applied to update the controller:

$$\nabla J\left(\theta_c\right) = \frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T}\zeta^{T-t}\nabla_{\theta} log\left(a_t|a_{(t-1):1},\theta_c\right)\left(R_b - b\right)$$

where m is the batch size and T is the number of steps in each episode. Rewards are discounted at every step by an exponential factor ζ and the baseline b is the average exponential moving of rewards. The parameter θ_c of RNN is updated to make the RL controller predict a better set of parameters in the future. This search process is executed EP episodes for one-time SC-NAS. For stable results [24], we run the SC-NAS multiple times and report the setting with the best reward R as our final solution. The time overhead for SC-NAS is calculated by average.

3.3. Search space and arithmetic circuits

In order to obtain the HW and SW trade-off, we need to know what factors affect both HW and SW performance, then select a group of settings with the best HW/SW trade-off (i.e., the optimization objective). Therefore, we comprehensively explore the influencing factors and finally divide them into three parts (see Table 3): MASS from NAS, DFSS, and ACSS from SC.

First, we find that the model architecture needs to be determined before model implementation. Meanwhile, we find that generally the more complex the model (i.e., larger model size), the higher the accuracy but the higher the power consumption and latency. In DNNs, the number of layers and the number of neurons in each layer are the main factors to determine the model architecture. For convolutional and pooling layers, the kernel size and stride also need to be determined. Therefore, we involve them in the MASS, which affects the HW and SW performance by tuning the model size.

After determining the model architecture, we consider that if all layers utilize SC data, the accuracy will be limited because of the inaccuracy of SC arithmetic circuits. So we think about hybrid layers with different data representations for DNNs implementations to obtain a better trade-off. We distinguish the data representation of traditional FP and SC bit streams into 'FP' and 'SC'. As far as we know, FP data format coupling with traditional FP multiplier and adder can obtain accurate computing results instead of approximate results in SC. But FP arithmetic circuits consume larger power and area than SC. Therefore, for the energy-accuracy trade-off, we consider these two data formats into DFSS so that the SC-NAS can find the customized data format for each layer. Besides, we involve the data precision (i.e., the bit length of SC bit streams) into the DFSS, which will be utilized when the SC format is chosen. Setting it manually cannot guarantee a better result

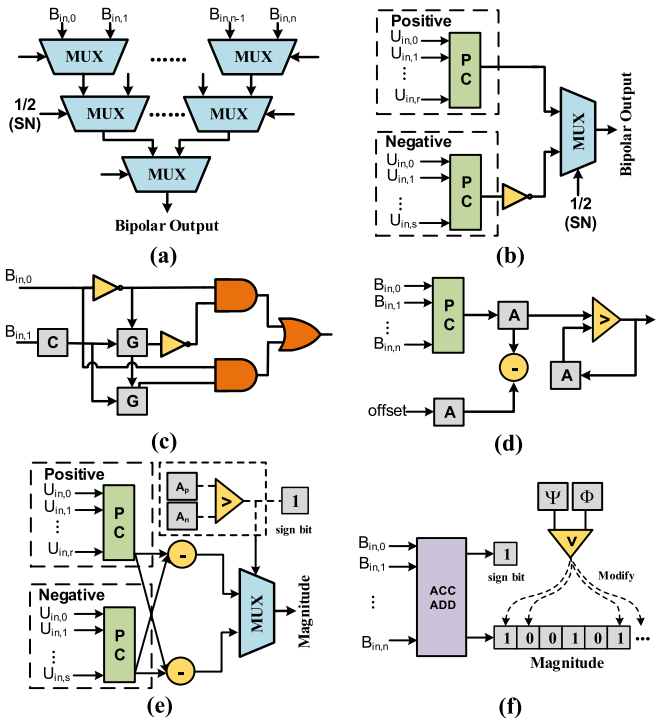

Fig. 4. Five arithmetic circuits in search space ACSS. (a) MUX-tree adder (MUX-ADD) [8]; (b) separated adder (SEPADD) [9]; (c) multiplier proposed in [10] (uMUL); (d) adder proposed in [10] (uNSADD); (e) accumulator-based adder (ACCADD) [18]; (f) loop-based adder (LOOPADD) [18].

because it will influence both the inference latency and accuracy. In general, under the same parallelism, the longer the bit length, the less error between FP-to-SC transformation, but longer latency.

Finally, we collect various SC arithmetic circuits proposed in other works into ACSS. These circuits produce different power consumption, calculation latency and accuracy (see example in Table 1). When the SC format is selected for one layer, then we hope to find the most suitable circuits. It is worth noting that our ACSS allows to expediently add any types of circuits proposed from other works. In our work, we implement five SC circuits (i.e., XNOR-MUX, AND-SEP, uGEMM, AND-ACC, and AND-LOOP) into ACSS. XNOR-MUX implements a circuit using an XNOR gate (see Fig. 1(b)) as the multiplier and a MUX-tree adder, namely MUXADD (see Fig. 4(a)), designed from [8]. MUXADD finally produces a scaled summation for all bipolar inputs. AND-SEP separates data into positive part and negative part through the first sign bit. Then it utilizes the AND gate (see Fig. 1(a)) to approximate multiplication, and designs an adder SEPADD (see Fig. 4 (b)) [9], which masterly transforms the result to bipolar format by only a MUX gate. Fig. 4(c) and (d) demonstrate the bipolar multiplier uMUL and n-input adder uNSADD proposed in uGEMM [10], which eliminates the correlation problem using the PC. AND-ACC and AND-LOOP also utilize AND gate for the multiplier and design optimized adder. Fig. 4(e) shows an accumulator-based adder (ACCADD) and (f) is a loop-based adder (LOOPADD), which is optimized based on ACCADD [18]. In ACCADD, it accumulates the real number of '1's by integrating positive and negative data not separately. It mitigates the computing overflow problem in SEPADD. Also, LOOPADD solves the assembling problem, which brings the redundant number of '1's or missing number of '1's in the results of ACCADD.

In this paper, we collect all the above sub-search space into an integrated search space, which is shown in Table 3. However, because every sub-search space can involve dozens or even hundreds of choices, which leads to abundant combinations, we explore the factors through experiments in Section 4.2 to contract the search space. Note that our

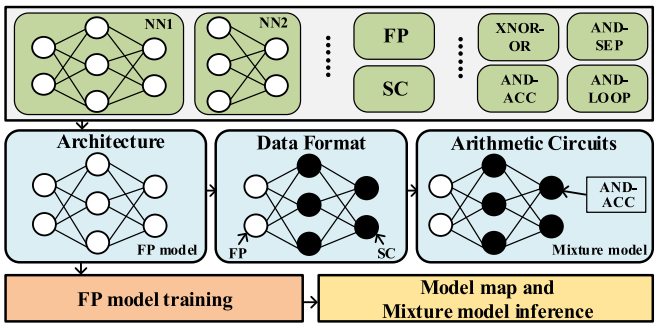

Fig. 5. The architecture of model trainer. The top box represents the integrated search space.

SC-NAS is a re-configurable framework, it allows other factors to be involved. And the values of search space can be adjusted for different requirements and scenarios.

3.4. Model trainer

When a set of parameters is selected from the integrated search space, a model trainer is next launched. If the energy constraint E_c is verified to be satisfied (see Section 3.5), the model is built, trained, and inferred to obtain the accuracy A on the local host. Fig. 5 shows the model building, training and inference process. For the selected parameters, the model architecture is first resolved, and the corresponding FP model with the selected number of layers, neurons, kernel size and stride is established. Then, the data format is determined for each layer. According to the selected data format and arithmetic circuits, we build a mixture model with specific data formats and circuits for each layer. Next, the FP model is trained to a high accuracy followed by the model mapping and mixture model inference. In order to obtain the inference accuracy A of the mixture model, we map the weights and biases of the trained FP model onto the mixture model instead of training the mixture model from scratch. If the corresponding layer utilizes SC to represent data, an SNG is used to transform FP weights/biases to SC bit streams for mixture model initialization. After the model map, the mixture model is evaluated forward. When the corresponding layer is represented as SC format, the selected arithmetic circuits and bit length are used to compute. Similarly, when the layer is represented using FP, the traditional FP multiplier and adder are exploited. Besides, to improve the NAS efficiency, we set an energy constraint E_c. If the inference energy E is higher than E_c, the inference process in this episode is early terminated, which greatly improves the efficiency of SC-NAS.

3.5. Energy estimator and reward generator

In order to obtain the energy consumption E and resource usage S of mixture model inference, an energy estimator is designed. We simulate the multiplier and adder in high-level synthesis tools, and gain the power consumption in unit time, the computation latency of one operation, and the LUT and LUTRAM usage for each circuit. Then we calculate the total energy consumption and resource usage based on the number of operations in each model. In this way, the energy consumption E (i.e., power × latency) and resource usage S are evaluated.

After the model trainer and energy estimator, the energy E, resource usage S and accuracy A under a set of factors are obtained. Then the feedback (i.e., reward) from the environment to RL controller can be calculated in this episode. In SC-NAS, the reward R is used to update the RL controller and guide the search process. The reward represents our optimization objective, which is to find the mixture model, which possesses both high accuracy, low energy consumption and less resource usage as much as possible. Our reward is divided into two parts: HW reward part and SW reward part. The HW reward part consists of the transformed energy and resource usage. The SW reward part refers to the transformed accuracy. The reward R is formulated as follows:

$$R = \begin{cases} \frac{1}{2}\left(\frac{E_i - E_l}{E_u - E_l} + \frac{S_i - S_l}{S_u - S_l}\right) + penalty & E_i > E_c \\ \frac{1}{2}\left(\frac{E_i - E_l}{E_u - E_l} + \frac{S_i - S_l}{S_u - S_l}\right) + \frac{A_i - A_l}{A_u - A_l} + award & E_i \le E_c \ \&\ A_i > A_c \\ \frac{1}{2}\left(\frac{E_i - E_l}{E_u - E_l} + \frac{S_i - S_l}{S_u - S_l}\right) + \frac{A_i - A_l}{A_u - A_l} & otherwise \end{cases} \quad (2)$$

There are three cases in calculating reward R: (1) if $E_i > E_c$ indicating the pre-set energy constraint E_c cannot be satisfied, we directly set the accuracy reward as a negative value *penalty* to punish the reward and terminate the mixture model inference in this episode; (2) if $E_i \le E_c$ & $A_i > A_c$ indicating the energy constraint E_c can be satisfied and the accuracy A_i is higher than our pre-set expected accuracy A_c, we give an extra *award* for this case; otherwise, (3) the transformed energy, resource usage and accuracy are summed to calculate the reward R. The transformed formulas also refer to the $Eng(CF)$, $Res(CF)$ and $Acc(CF)$ in formula (1). In this paper, the $\alpha = \frac{1}{2}$, $\beta = \frac{1}{2}$ and $\gamma = 1$ for the fairness of HW and SW performance. The α, β and γ can be set as other values for different requirements. In the formula (2), E_i, S_i and A_i represent the energy consumption, resource usage and accuracy in i th episode. E_u, S_u and A_u indicate the pre-set upper bound of energy, resources and accuracy. And E_l, S_l and A_l refer to the lower bound of energy, resources and accuracy, respectively. The upper bound and lower bound can be set according to different requirements and scenarios. Therefore, our SC-NAS can automatically find the suitable configurations adapting to different applications.

4. Experiments and comparison

4.1. Experimental setup

Methods Comparison. We first explore the influencing factors comprehensively through a series of experiments and build the contracted search space. The HW and SW effects of these factors are both demonstrated for exploration. And the values of influencing factors are shrunk from a wide scope to a reasonable range to contract the search space. Next, in order to prove the validity and efficiency of our SC-NAS, we compare the results of SC-NAS with state-of-the-art pure SC methods and FP-NAS method. Pure SC means the data in DNNs are represented using only SC format and the setting of influencing factors are selected for comparison or set by intuition. Without NAS, a better model configuration cannot be determined directly and reasonably. FP-NAS means the DNNs are implemented using only FP data without SC involving and the architecture is determined through NAS. Without SC implementation, the MAC operations are finished using FP multiplier and adder, which leads to high energy consumption for FP-NAS. Meanwhile, the solutions obtained from SC-NAS are visualized, which include the neural architectures and the corresponding hardware circuits. Finally, we plot the Pareto frontier of our SC-NAS to demonstrate the search process. The solution with the best HW/SW trade-offs in our search process is selected as our final result.

Evaluation Platforms. We evaluate our method on multiple multi-layer perceptrons (MLPs) and LeNet models using MNIST dataset, and VGGNet using Cifar10 dataset. The VGG11 model is implemented. The MASS, DFSS, and ACSS are all integrated into a unified search space for SC-NAS. For HW evaluation, FP and SC arithmetic circuits are implemented by Bluespec SystemVerilog, then compile to Verilog to evaluate for the power consumption, latency and resource usage on Vivado v2020.2, which is the commonly used high-level synthesis (HLS) tool. This tool supports implementing the accelerator with C++ languages and exports the RTL as a Vivado's IP core. For the SW

Table 4
The specific search space settings of MLP, LeNet and VGGNet.

MLP	MASS	# of layers: {1, 2, 3} # of neurons for 3-linear layers: {32, 64, 100, 128, 200, 256, 400, 512, 1024}
	DFSS	Data format for each layer: {FP, SC} Bit length of SC data: {4, 8, 16, 32, 64, 128, 256}
	ACSS	Arithmetic circuit library: {XNOR-MUX, AND-SEP, uGEMM, AND-ACC, AND-LOOP}
	* number of configurations: 68 040 ($\approx 6.8 \times 10^4$)	
LeNet	MASS	Stride for 2-convolutional layers and 2-max-pooling layers: {1, 2} Kernel size for 2-convolutional layers: {3, 5, 7} Kernel size for 2-max-pooling layers: {1, 2} Output channel for 2-convolutional layers: {6, 16, 32, 64} # of neurons for 3-linear layers: {32, 64, 84, 100, 120, 128, 200, 400, 512}
	DFSS	Data format for each layer: {FP, SC} Bit length of SC data: {4, 8, 16, 32, 64, 128, 256}
	ACSS	Arithmetic circuit library: {XNOR-MUX, AND-SEP, uGEMM, AND-ACC, AND-LOOP}
	* number of configurations: 3344302080 ($\approx 3.3 \times 10^9$)	
VGGNet	MASS	Stride for 8-convolutional layers and 5-max-pooling layers: {1, 2} Kernel size for 8-convolutional layers: {1, 3, 5} Kernel size for 5-max-pooling layers: {1, 2} Output channel for 8-convolutional layers: {32, 64, 256, 512} # of neurons for 3-linear layers: {32, 64, 100, 128, 200, 256, 400, 512}
	DFSS	Data format for each layer: {FP, SC} Bit length of SC data: {4, 8, 16, 32, 64, 128, 256}
	ACSS	Arithmetic circuit library: {XNOR-MUX, AND-SEP, uGEMM, AND-ACC, AND-LOOP}
	* number of configurations: 16546945606899581583360 ($\approx 1.7 \times 10^{22}$)	

Table 5
Energy consumption and accuracy results under different MLP architectures.

Architecture		FP-MLP	Stochastic MLP				
			4	8	16	32	64
784-512-512-10	Eneg. (mJ)	29 606.7	1028.5	1328.5	1965.7	3182.4	5579.5
	Acc.	98.0%	24.4%	39.5%	80.5%	93.7%	96.3%
784-100-200-10	Eneg. (mJ)	2664.5	96.3	124.4	184.1	298.1	522.7
	Acc.	97.4%	32.3%	62.2%	88.1%	95.4%	96.3%
784-32-64-10	Eneg. (mJ)	598.3	24.0	30.9	45.8	74.1	130.0
	Acc.	96.1%	38.3%	70.0%	88.3%	91.8%	94.1%
784-32-10	Eneg. (mJ)	507.6	19.1	24.7	36.6	59.2	103.8
	Acc.	93.0%	68.2%	88.1%	90.5%	92.3%	92.2%
784-10	Eneg. (mJ)	150.5	5.2	6.8	10.0	16.2	28.4
	Acc.	90.3%	71.8%	78.3%	86.3%	88.9%	89.8%

metric evaluation, we conduct the inference of mixture model with RL optimization on a 2× NVIDIA Tesla P100 GPU server (16 GB GPU memory). Experiments are performed on Python 3.6.0, GCC 9.3.0, PyTorch 1.5.1, and CUDA 10.1.

In our SC implementations, Sobol sequence [25,26] is utilized as the SNG to generate a random number for FP-to-SC transformation. The RL controller is implemented using LSTM model, the input batch size of LSTM $m = 35$, the number of steps in each episode $T = 50$, and the learning rate $lr = 0.99$. The energy constraint E_c is set as 500 mJ, 3000 mJ and 2300 J for MLP, LeNet and VGGNet, respectively. For the expected accuracy A_c, it is set as 94% for three models. The specific search space settings of MLP, LeNet and VGGNet are shown in Table 4.

4.2. Search space exploration

① *Model Architecture is the first factor that should be determined.*

Before model training and inference, we need to know the model architecture to build corresponding models. Model architecture (including the number of layers and neurons, stride in each layer, etc.) refers to the size of model, which can affect the power consumption, latency, resource usage and accuracy. Generally, deeper and wider models hold higher power consumption and longer latency but higher accuracy. We implement MLPs with different architectures in FP and SC formats to show the effects. These architectures listed in Table 5 are often discussed in other works [27]. Table 5 demonstrates the energy consumption of one-time inference of MLPs for FP and SC representation. Moreover, the accuracy of model inference under different architectures and bit lengths are shown. For SC inference, the weights and biases are mapped from the trained FP models with the same architecture. Then AND-ACC circuit is utilized as the basic arithmetic circuit to finish the model computation.

We analyze the results in Table 5 from HW and SW sides. From the aspect of HW efficiency, larger architecture with more operations consumes more energy. With the same number of operations, the SC implementations reduce up to 28.9 × energy than FP. This fully demonstrates the superiority of SC in reducing energy consumption, which is very advantageous for tiny devices with strictly limited energy. For the SW performance, Table 5 shows that more complex models can finally reach higher accuracy whether in a conventional FP system or in SC system with the bit length of 64. However, there is still more than 1.7% accuracy gap between FP and SC systems. In SC system, with the bit length increases, the accuracy shows an upward trend. Besides, we have an interesting observation. When the data precision is low, such as 4 bits, the simpler the model, the higher the model accuracy. This is because lower data precision will produce more errors in each operation. More complex architectures, which possess more operations, cause more errors to be accumulated. However, the errors decrease for each operation with the increasing of bit length, simple models cannot reach higher accuracy. Even though the above architectures are commonly used in other work, we also cannot determine the best architecture directly. Because there are lots of neuron combinations for different layers. But we can involve them in the search space and search for a better combination. Based on above experiments, we set the MASS of MLP, LeNet and VGGNet as shown in Table 4.

② *Which format should be applied for each layer: SC or FP?*

When the model architecture is fixed, the data format for each layer then needs to be determined. Because of the low data precision and the inaccuracy of arithmetic circuits, SC gets a higher accuracy loss than FP. Table 5 also shows this phenomenon. Therefore, if the hybrid data format can be used to obtain a better trade-off? We do experiments to demonstrate the results. We implement two MLPs with different architectures (784-100-200-10 and 784-32-64-10). In Table 6,

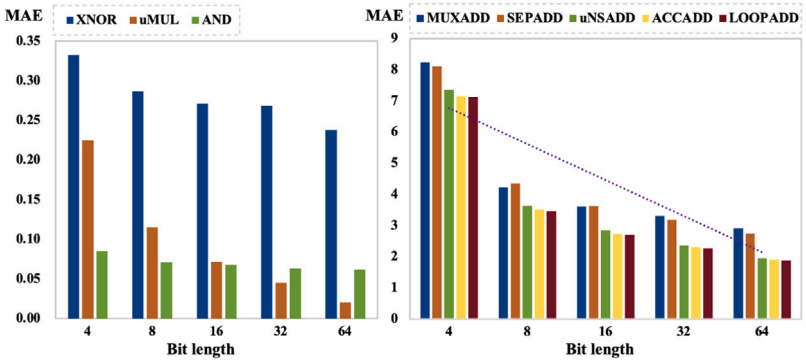

Fig. 6. The mean absolute error (MAE) of (a) multiplier and (b) adder with different bit lengths.

Table 6
Energy consumption and accuracy results of MLPs using pure SC representation and hybrid FP-SC representation. The first two layers of the MLPs in 'Hybrid' column utilize SC representation and the last layer utilizes FP data.

Architecture	Data format	Metric	Bit length				
			4	8	16	32	64
784-100-200–10	SC	Eneg. (mJ)	96.3	124.4	184.1	298.1	522.7
		Acc.	32.3%	62.2%	88.1%	95.4%	96.3%
	Hybrid	Eneg. (mJ)	1723.4	2223.1	3284.6	5311.4	9304.3
		Acc.	35.1%↑	80.5%↑	92.6%↑	96.5%↑	96.9%↑
784-32-64–10	SC	Eneg. (mJ)	24.0	30.9	45.8	74.1	130.0
		Acc.	38.3%	70.0%	88.3%	91.8%	94.1%
	Hybrid	Eneg. (mJ)	165.5	213.4	315.3	509.7	892.7
		Acc.	58.7%↑	84.4%↑	90.8%↑	94.2%↑	94.3%↑

Table 7
Energy consumption and accuracy results of MLP (784-100-200-10) under different SC arithmetic circuits.

Circuits	Metric	Bit length				
		4	8	16	32	64
XNOR-OR [8]	Eneg. (mJ)	184.0	186.7	213.7	266.3	369.4
	Acc.	9.1%	9.7%	9.1%	10.4%	9.9%
AND-SEP [9]	Eneg. (mJ)	32.7	37.7	49.7	72.8	119.0
	Acc.	21.3%	32.3%	32.1%	37.0%	32.0%
uGEMM [10]	Eneg. (mJ)	49.1	82.4	145.0	267.4	511.1
	Acc.	9.90%	11.90%	17.50%	63.50%	91.10%
AND-ACC [18]	Eneg. (mJ)	24.0	30.9	45.8	74.1	130.0
	Acc.	32.3%	62.2%	88.1%	95.4%	96.3%
AND-LOOP [18]	Eneg. (mJ)	39.9	50.1	74.5	120.5	208.7
	Acc.	38.3%	81.9%	95.0%	96.8%	97.4%

the 'SC' column identifies that all the data in the model are represented as SC format. The 'Hybrid' column represents that the first two layers utilize SC representation and the last layer utilizes FP data. We use this assignment just as an example. Similarly, all the SC data in models are calculated by AND-ACC circuit. The bit lengths of SC data from 4 to 64 are explored.

Results show that the energy consumption of 'Hybrid' is up to 17.9 × higher than 'SC' under the same architecture. Meanwhile, it is up to 3.6 × lower than the case of pure FP in Table 5. It seems that, under the same architecture, hybrid data representation makes a better trade-off in the energy aspect. On the other hand, it is obvious that the accuracy in all bit lengths of 'Hybrid' is higher than 'SC' with the same architecture, because FP can obtain accurate computation results. It mitigates the errors affected by approximate computing. But it also has over 0.5% accuracy loss than the pure FP model. Therefore, with the hybrid FP-SC format, SC is used to significantly reduce energy, and the FP is used to make up for the accuracy loss. In a nutshell, 'Hybrid' is an energy-accuracy trade-off compared with pure SC and pure FP under the same architecture in our examples. However, DNNs generally consist of multiple layers. Without NAS, the data format for each layer with a better trade-off cannot be determined manually. Therefore, we provide two data formats {FP, SC} into our DFSS and try to find out the optimized and customized data format for each layer.

③ *Multipliers/Adders Affect Results.*

Finally, for the layers represented by SC format, the arithmetic circuits should be determined. In the FP system, the deterministic FP multiplier and adder are utilized. For SC, we implement five circuits for MLPs inference, including XNOR-MUX [8], AND-SEP [9], uGEMM [10], AND-ACC and AND-LOOP [18]. Detailed circuits have been introduced in Section 3.3.

We first evaluate the MAE of different multipliers and adders. Results are demonstrated in Fig. 6. The multipliers are evaluated for 2-inputs range from −1 to 1, because the MAC operations only involve 2-inputs multiplications. For adders, we implement circuits for 64 inputs as a test example. In our experiments, the arithmetic circuits include three types of multipliers (i.e., XNOR gate, uMUL and AND gate) and five types of adders (i.e., MUXADD, SEPADD, uNSADD, ACCADD and LOOPADD). As the bit length increases, the MAE of each circuit shows a decreasing trend. Because with the data precision increasing, the error produced by FP-to-SC transformation is mitigated. For multipliers, XNOR gate produces the highest MAE, and with the data precision increasing, uMUL becomes the best multiplier with the lowest MAE. For adders, the MUXADD produces the highest MAE because of the correlation problem, and the LOOPADD possesses the best SW performance. In Fig. 6, SEPADD also keeps a high MAE because of its overflow problem.

In addition to testing each multiplier and adder, we also evaluate the performance of the circuits in the MLPs. Table 7 demonstrates the energy consumption and accuracy of MLP (784-100-200-10) among five SC arithmetic circuits. As we can see, different computation circuits possess different performances in energy consumption and accuracy. The XNOR-MUX circuit has the lowest energy consumption and the AND-LOOP circuit has the highest accuracy. This is consistent with the results of the single multiplier/adder evaluation in Fig. 6. Therefore, a circuit with a better HW/SW trade-off needs to be determined by our SC-NAS framework coupling with other factors. In this paper, we involve {XNOR-MUX, uGEMM, AND-SEP, AND-ACC, AND-LOOP} into the ACSS. Note that our search space is re-configurable, the arithmetic circuits proposed in other works can also be considered into the ACSS.

In a conclusion, we add MASS, DFSS and ACSS into an integrated search space. Without exploration, the search space can consist of arbitrary values. After our exploration, the search space is contracted to a reasonable range and it can include high accuracy and low energy consumption situations. LeNet and VGGNet also have a similar phenomenon, here we do not make additional descriptions. Based on this huge discrete search space, RL algorithm is necessary to be applied to search a set of parameters for efficient HW and SW co-optimization.

Table 8
Evaluation results of SC-NAS compared with FP-NAS and pure SC methods.

Methods		FP-NAS	Pure SC methods					SC-NAS
			[8]	[9]	[10]	[18]	[18]	
	Model architecture	784-32-32-10	784-128-10	784-128-10	784-128-10	784-128-10	784-128-10	784-128-10
	Data format	Pure FP	Pure SC	Pure SC	Pure SC	Pure SC	Pure SC	SC-SC
	Arithmetic circuits	FP	XNOR-MUX	AND-SEP	uGEMM	AND-ACC	AND-LOOP	AND-LOOP
	Data precision	32	32	32	32	32	32	32
MLP	Accuracy	94.2%	11.0%	53.9%	56.8%	94.8%	96.2%	96.2%
	Acc. satisfy?	✓	✗	✗	✗	✓	✓	✓
E_c: 500 mJ	Energy (mJ)	548.6	1013.7	252.6	974.2	252.6	400.8	400.8
A_c: 94.0%	Energy Satisfy?	✗	✗	✓	✗	✓	✓	✓
	LUT usage	676 966	1918	3814	1898	3968	4032	4032
	LUTRAM usage	1 282 868	0	344	1904	284	288	288
	Search/Infer. time	0.3 h/0.4 ms	140.5 ms	18.0 ms	70.0 ms	16.6 ms	17.3 ms	1.4 h/17.3 ms (w/o $E_c \approx$ 1.7 h)
	ES time	0.4 h	-	-	-	-	-	≈373.5 h
	Model architecture	ARCH1	ARCH2	ARCH2	ARCH2	ARCH2	ARCH2	ARCH2
	Data format	Pure FP	Pure SC	Pure SC	Pure SC	Pure SC	Pure SC	FP-FP-FP-FP-SC-SC-SC
	Arithmetic circuits	FP	XNOR-MUX	AND-SEP	uGEMM	AND-ACC	AND-LOOP	AND-ACC
	Data precision	32	32	32	32	32	32	32
	Accuracy	97.5%	9.0%	28.7%	47.8%	89.0%	92.5%	95.2%
LeNet	Acc. satisfy?	✓	✗	✗	✗	✗	✗	✓
	Energy (mJ)	10 098.3	4079.0	1233.9	4299.6	1288.3	2106.2	1438.9
E_c: 3000 mJ	Energy satisfy?	✗	✗	✓	✗	✓	✓	✓
A_c: 94.0%	LUT usage	276 010	3484	4928	3434	5298	5458	48 186
	LUTRAM usage	523 068	0	900	1444	720	720	86 560
	Search/Infer. time	4.7 h/18.5 ms	828.1 ms	106.2 ms	412.5 ms	97.5 ms	98.4 ms	5.3 h/14.2 ms (w/o $E_c \approx$ 9.3 h)
	ES time	≈17 900.7 h	-	-	-	-	-	≈1.1 × 10^8 h
	* ARCH1: Conv1(k = 3, s = 1, c = 64)-Pooling1(k = 1, s = 2)-Conv2(k = 5, s = 1, c = 6)-Pooling2(k = 1, s = 2)-FC1(120)-FC2(64)-FC3(10)							
	* ARCH2: Conv1(k = 7, s = 1, c = 16)-Pooling1(k = 1, s = 2)-Conv2(k = 3, s = 1, c = 16)-Pooling2(k = 1, s = 2)-FC1(64)-FC2(100)-FC3(10)							
	Model architecture	ARCH1	ARCH2	ARCH2	ARCH2	ARCH2	ARCH2	ARCH2
	Data format	Pure FP	Pure SC	Pure SC	Pure SC	Pure SC	Pure SC	Hybrid
	Arithmetic circuits	FP	XNOR-MUX	AND-SEP	uGEMM	AND-ACC	AND-LOOP	AND-LOOP
	Data precision	32	64	64	64	64	64	64
	Accuracy	97.5%	10.0%	46.8%	52.3%	90.3%	92.7%	96.5%
	Acc. satisfy?	✓	✗	✗	✗	✗	✗	✓
	Energy (J)	5256.3	1097.8	426.7	1660.6	475.8	802.8	2203.4
VGGNet	Energy satisfy?	✗	✓	✓	✓	✓	✓	✓
E_c: 2300 J	LUT usage	436 342	5760	7432	5650	8246	8598	417 794
A_c: 94.0%	LUTRAM usage	826 932	0	1980	1672	1584	1584	790 176
	Search/Infer. time	11.2 h /6.1 s	223.2 s	33.7 s	153.2 s	32.2 s	33.2 s	41.7 h/7.7 s (w/o $E_c \approx$ 63.8 h)
	ES time	≈6.5 × 10^{20} h	-	-	-	-	-	≈3.6 × 10^{21} h
	*ARCH1: Conv1(k = 3, s = 1, c = 512)-Pooling1(k = 1, s = 1)-Conv2(k = 1, s = 1, c = 64)-Pooling2(k = 1, s = 1)-Conv3(k = 3, s = 1, c = 256)-Conv4(k = 5, s = 2, c = 32)-Pooling3(k = 1, s = 1)-Conv5(k = 1, s = 2, c = 512)-Conv6(k = 1, s = 1, c = 512)-Pooling4(k = 1, s = 1)-Conv7(k = 5, s = 2, c = 256)-Conv8(k = 5, s = 2, c = 32)-Pooling5(k = 1, s = 1)-FC1(200)-FC2(128)-FC3(10)							
	*ARCH2: Conv1(k = 5, s = 2, c = 256)-Pooling1(k = 1, s = 1)-Conv2(k = 3, s = 2, c = 512)-Pooling2(k = 1, s = 1)-Conv3(k = 3, s = 1, c = 64)-Conv4(k = 1, s = 1, c = 64)-Pooling3(k = 1, s = 1)-Conv5(k = 5, s = 2, c = 512)-Conv6(k = 3, s = 1, c = 32)-Pooling4(k = 1, s = 1)-Conv7(k = 3, s = 2, c = 512)-Conv8(k = 3, s = 2, c = 256)-Pooling5(k = 1, s = 1)-FC1(64)-FC2(200)-FC3(10)							
	*Hybrid: FP-FP-FP-FP-SC-FP-FP-FP-SC-FP-FP-SC-FP-FP-SC-FP							

4.3. Evaluation of SC-NAS

This section evaluates the validity and efficiency of SC-NAS using MLP and LeNet on the MNIST dataset, and VGGNet on the Cifar10 dataset. The results of SC-NAS are compared with the FP-NAS and state-of-the-art pure SC methods. FP-NAS refers to the method which only utilizes FP as the data format, and the model architecture is searched based on RL technology. The search space only involves MASS. Meanwhile, the FP-NAS also takes energy, resource usage and accuracy into consideration. The reward of FP-NAS is set the same as our SC-NAS. Pure SC indicates the methods which only utilize SC as the data format and utilize the SC arithmetic circuits to approximate computation. The model architecture is selected based on the resulting architecture of SC-NAS for a fair comparison. The evaluation results of FP-NAS, state-of-the-art pure SC methods and SC-NAS are demonstrated in Table 8. For SC-NAS and FP-NAS methods, the search episode EP is set as 300. And they are executed 3 times for stable results and average time overhead. For tiny devices and SC-NAS method, the energy constraints E_c are set as 500 mJ for MLP and 3000 mJ for LeNet. Because the VGGNet is huge, the energy constraint E_c is set as 2300 J. For the FP-NAS, the energy constraints are set as 6000 J, 14 000 J and 160 000 J for MLP, LeNet and VGGNet, respectively. The constraints are higher than SC-NAS method because the FP model inference is hard to satisfy such low energy constraint with an acceptable accuracy. The 'Energy' and 'Infer. Time' in Table 8 refer to the result for one-picture inference. The model inference is implemented with parallelism $p = 2$ for SC data to accelerate the computation. Besides, the expected accuracy A_c is set as 94.0% for these three models.

First, the comparison of FP-NAS and SC-NAS is discussed. For FP-NAS, because it utilizes an accurate FP data format and suitable architecture, the expected accuracy of MLP, LeNet and VGGNet are all achieved. But the energy constraint for resulting models cannot be satisfied because of the high-power FP multiplier and adder. However, our SC-NAS can satisfy both the requirements and obtain a better HW and SW trade-off. From the search results of FP-NAS and SC-NAS, we have an interesting observation that the data format for SC-NAS

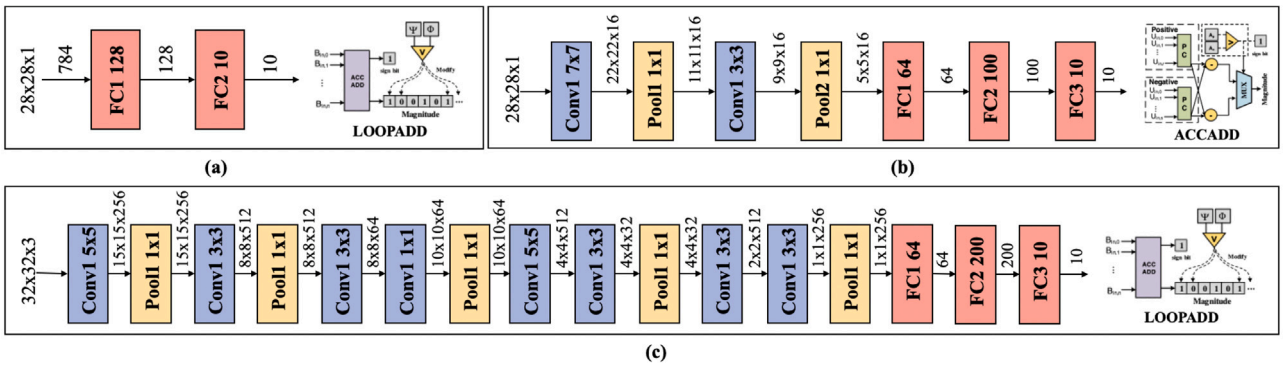

Fig. 7. Neural architectures and SC circuits visualization of SC-NAS for (a) MLP, (b) LeNet and (c) VGGNet models.

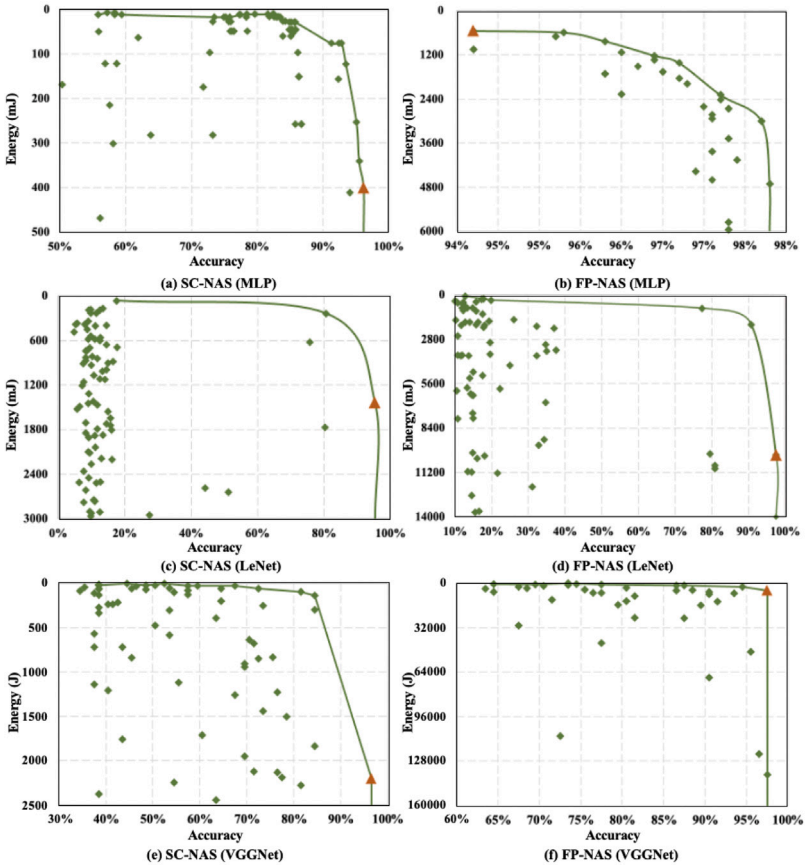

Fig. 8. The Pareto frontier of SC-NAS and FP-NAS for MLP, LeNet and VGGNet.

result of MLP model is SC for each layer instead of expected hybrid data format. We analyze this because the AND-LOOP circuit approximates the linear layer well, which makes high inference accuracy for MLP. Meanwhile, there is a low energy consumption for AND-LOOP implementation, so pure SC implementation wins the search for MLP model. However, pure SC circuits do not perform well in LeNet and VGGNet, because they consist of extra convolutional and pooling layers. It makes LeNet and VGGNet need FP format to guarantee the inference accuracy, which leads to a hybrid data format. For MLP model, our SC-NAS with pure SC format even obtains 2.0% higher accuracy than FP-NAS with the pure FP data format. That is because the FP-NAS also aims at searching the best HW/SW trade-off. The architecture with higher accuracy may consume unacceptable energy and resources. So, a better result with HW/SW trade-off is searched. Besides, our SC-NAS respectively achieves 1.4 ×, 7.0 × and 2.4 × energy saving than FP-NAS for MLP, LeNet and VGGNet. The LUT usage of FP-NAS is 167.9 ×, 5.7 × and 1.1 × more than our SC-NAS for three models. And the LUTRAM usage of FP-NAS is 4454.4 ×, 6.0 × and 1.1 × more than our SC-NAS for three models. Our SC-NAS demonstrates the advantages of SC in low-power execution.

Then, five state-of-the-art pure SC methods are compared with our SC-NAS (shown in Table 8). The architectures and bit length for SC methods are set the same with our SC-NAS solution for a fair comparison. It is obvious that most of the SC methods cannot meet both the energy constraint and the expected accuracy at the same time. In most cases, SC methods can achieve better energy consumption, but produce a high accuracy loss. Among these methods, the AND-LOOP circuit achieves the best accuracy than others, and the AND-SEP circuit possesses minimal energy consumption. Besides, the AND-ACC circuit takes the least amount of time for NNs inference. The neural architectures and corresponding SC circuits for the resulting MLP, LeNet and VGGNet models of SC-NAS are demonstrated in Fig. 7 for better

visualization. Finally, for SC-NAS, the ES methods under the same contracted search space as SC-NAS will consume about 746.9 h (≈31 days) for MLP model, 2.1×10^8 h (≈23 924 years) for LeNet model, and 7.0×10^{21} h for VGGNet model, which are obviously unacceptable time overhead. Correspondingly, our SC-NAS achieves up to 327.6×, $2.0 \times 10^7 \times$ and 8.6×10^{19} speedup than ES, respectively. Meanwhile, with our energy constraint E_c, some episodes can be early terminated, it achieves 1.2 ×, 1.8 × and 1.5 × efficiency improvement on MLP, LeNet and VGGNet models, respectively.

Except for the experiments shown in Table 8, we also do experiments to verify that the RL algorithm can obtain near-optimal results using MLP model. We exhaustively search the best FP model architecture with the highest trade-off performance, namely FP-ES. The difference of results between FP-ES and FP-NAS methods is so tiny. In our statistical results, we find that the solution of FP-NAS is exactly the second optimal result of FP-ES, there is only 0.4% difference on the normalized optimization metrics (i.e., the reward R in formula (2)). In a nutshell, NAS can obtain nearly global optimal results, and SC shows great advantages in reducing energy consumption.

4.4. Search Pareto frontier

Finally, we collect the explored solutions from RL to form the results of search space for SC-NAS and FP-NAS in Fig. 8. In the figure, the x-axis and y-axis stand for the accuracy and energy consumption. The energy constraints are set as 500 mJ and 6000 mJ for SC-NAS and FP-NAS in MLP models. The energy constraints of SC-NAS and FP-NAS methods for LeNet are 3000 mJ and 14 000 mJ. And the energy constraints of SC-NAS and FP-NAS for VGGNet are 2300 J and 160 000 J. One point in the figure means a set of selected parameters in an episode. We remove some points on the boundary with very low inference accuracy. The figure also shows the Pareto frontier of these six search processes. In these Pareto frontiers, we select the ones with the highest energy-accuracy trade-off as the final results, which are marked as the triangle in the figure. Even if resource usage is not shown in the Figure, it has properties similar to the energy consumption. As we can see, the solution of FP-NAS produces lower accuracy than SC-NAS in MLP, because the FP circuits are power-consumed. It forces FP-NAS to select the solution with a smaller model size, which naturally leads to relatively lower accuracy. Benefiting from the low-power SC circuits, SC-NAS obtains a larger model with relatively higher accuracy and also low energy consumption for MLP model. For LeNet, the FP-NAS obtains a higher accuracy but higher energy consumption than SC-NAS, this is because SC circuit does not perform high accuracy in convolutional neural networks as the MLP model with pure linear layers. The results of VGGNet are similar with the LeNet because of the similar types of layers.

5. Conclusion

This work presents an RL-based framework, namely SC-NAS, which is the first to couple SC with NAS to achieve a better HW and SW co-optimization for tiny devices. With the challenge of lots of discrete influencing factors, which affect both HW and SW performance, we propose to utilize NAS to intelligently and automatically search a set of parameters for an energy-resources-accuracy trade-off. On the other hand, the huge search space still impedes the search efficiency of NAS. Therefore, we first contract the search space through comprehensive experiments to improve the search efficiency. Furthermore, we set an energy constraint to early terminate the mixture model inference to save search time. Experiments show that RL-based SC-NAS method can produce near-optimal solutions. And our SC-NAS can achieve up to 7.0 × energy saving and 4454.4 × resources reduction than FP-NAS and obtain over 3.8% accuracy gains than pure SC methods. Meanwhile, with NAS and energy constraint, we save up to $8.6 \times 10^{19}\times$ search time than ES using VGGNet.

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

Data will be made available on request.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC) 61972154 and Shanghai Science and Technology Commission Project, China 20511101600.

References

[1] C. Garvey, A framework for evaluating barriers to the democratization of artificial intelligence, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[2] H. Wang, Z. Wu, Z. Liu, H. Cai, L. Zhu, C. Gan, S. Han, Hat: Hardware-aware transformers for efficient natural language processing, 2020, arXiv preprint arXiv:2005.14187.
[3] A. Kumar, S. Goyal, M. Varma, Resource-efficient machine learning in 2 kb ram for the internet of things, in: International Conference on Machine Learning (ICML), 2017, pp. 1935–1944.
[4] J. Lin, W.-M. Chen, Y. Lin, C. Gan, S. Han, et al., Mcunet: Tiny deep learning on iot devices, Adv. Neural Inf. Process. Syst. 33 (2020) 11711–11722.
[5] B.R. Gaines, Stochastic computing systems, in: Advances in Information Systems Science, Springer, 1969, pp. 37–172.
[6] P. Jeavons, D.A. Cohen, J. Shawe-Taylor, Generating binary sequences for stochastic computing, IEEE Trans. Inform. Theory 40 (3) (1994) 716–720.
[7] W. Qian, X. Li, M.D. Riedel, K. Bazargan, D.J. Lilja, An architecture for fault-tolerant computation with stochastic logic, IEEE Trans. Comput. 60 (1) (2010) 93–105.
[8] P. Li, D.J. Lilja, W. Qian, K. Bazargan, M.D. Riedel, Computation on stochastic bit streams digital image processing case studies, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 22 (3) (2013) 449–462.
[9] B. Li, Y. Qin, B. Yuan, D.J. Lilja, Neural network classifiers using stochastic computing with a hardware-oriented approximate activation function, in: 2017 IEEE International Conference on Computer Design (ICCD), 2017.
[10] D. Wu, J. Li, R. Yin, H. Hsiao, Y. Kim, J. San Miguel, UGEMM: Unary computing architecture for GEMM applications, in: 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), IEEE, 2020.
[11] K. Kim, J. Kim, J. Yu, J. Seo, J. Lee, K. Choi, Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks, in: 2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC), IEEE, 2016.
[12] A. Zhakatayev, S. Lee, H. Sim, J. Lee, Sign-magnitude sc: getting 10x accuracy for free in stochastic computing for deep neural networks, in: 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), IEEE, 2018, pp. 1–6.
[13] W. Jiang, X. Zhang, E.H.-M. Sha, L. Yang, Q. Zhuge, Y. Shi, J. Hu, Accuracy vs. efficiency: Achieving both through fpga-implementation aware neural architecture search, in: Proceedings of the 56th Annual Design Automation Conference (DAC) 2019, 2019, pp. 1–6.
[14] W. Jiang, L. Yang, E.H.-M. Sha, Q. Zhuge, S. Gu, S. Dasgupta, Y. Shi, J. Hu, Hardware/software co-exploration of neural architectures, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 39 (12) (2020) 4805–4815.
[15] M.S. Abdelfattah, Ł. Dudziak, T. Chau, R. Lee, H. Kim, N.D. Lane, Best of both worlds: Automl codesign of a cnn and its hardware accelerator, in: 2020 57th ACM/IEEE Design Automation Conference (DAC), IEEE, 2020, pp. 1–6.
[16] H. Cai, L. Zhu, S. Han, Proxylessnas: Direct neural architecture search on target task and hardware, 2018, arXiv preprint arXiv:1812.00332.
[17] V.T. Lee, A. Alaghi, L. Ceze, Correlation manipulating circuits for stochastic computing, in: 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2018, pp. 1417–1422.
[18] Y. Song, E.H.-M. Sha, Q. Zhuge, R. Xu, Y. Zhang, B. Li, L. Yang, BSC: Block-based stochastic computing to enable accurate and efficient TinyML, 2021, arXiv:2111.06686.
[19] L. Deng, G. Li, S. Han, L. Shi, Y. Xie, Model compression and hardware acceleration for neural networks: A comprehensive survey, Proc. IEEE 108 (4) (2020) 485–532.
[20] B. Zoph, Q.V. Le, Neural architecture search with reinforcement learning, 2016, arXiv preprint arXiv:1611.01578.
[21] A. Ashok, N. Rhinehart, F. Beainy, K.M. Kitani, N2n learning: Network to network compression via policy gradient reinforcement learning, 2017, arXiv preprint arXiv:1709.06030.

[22] E. Real, A. Aggarwal, Y. Huang, Q.V. Le, Regularized evolution for image classifier architecture search, in: Proceedings of the Aaai Conference on Artificial Intelligence, Vol. 33, 2019, pp. 4780–4789.
[23] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Mach. Learn. 8 (3) (1992) 229–256.
[24] X. Dong, Y. Yang, Nas-bench-201: Extending the scope of reproducible neural architecture search, 2020, arXiv preprint arXiv:2001.00326.
[25] S. Liu, J. Han, Energy efficient stochastic computing with Sobol sequences, in: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, IEEE, 2017, pp. 650–653.
[26] M.H. Najafi, D.J. Lilja, M. Riedel, Deterministic methods for stochastic computing using low-discrepancy sequences, in: 2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), IEEE, 2018, pp. 1–8.
[27] Y. Liu, S. Liu, Y. Wang, F. Lombardi, J. Han, A survey of stochastic computing neural networks for machine learning applications, IEEE Trans. Neural Netw. Learn. Syst. (2020).

Yuhong Song received the BS degree from the Department of Computer Science, Nanjing Agricultural University, Nanjing, China, in 2019. She is currently working towards the Ph.D. degree in the Department of Computer Science and Technology, East China Normal University, Shanghai, China. Her current research interests include embedded system, software-hardware co-design, automated machine learning, stochastic computing and quantum computing.

Edwin Hsing-Mean Sha received the Ph.D. degree from the Department of Computer Science, Princeton University, Princeton, NJ, USA, in 1992. From 1992 to 2000, he was with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA. Since 2000, he has been a Tenured Full Professor with the Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA. Since 2012, he has been serving as the Dean of the College of Computer Science, Chongqing University, Chongqing, China. Since 2018, he has been a distinguished professor of East China Normal University. His research interests include big data systems, high-performance intelligent computing, advanced storage, parallel/distributed systems and computing, embedded systems, scheduling and optimization, resource allocation and quantum computing.

Qingfeng Zhuge received the BS and MS degrees in electronics engineering from Fudan University, Shanghai, China and the Ph.D degree from the Department of Computer Science, University of Texas at Dallas, in 2003. She is currently a professor at East China Normal University, China. She received Best Ph.D. Dissertation Award in 2003. She has published more than 90 research articles in premier journals and conferences. Her research interests include parallel architectures, embedded systems, supply-chain management, real-time systems, optimization algorithms, compilers, and scheduling. She is a member of the IEEE.

Rui Xu received the BS degree from Faculty Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China, in 2018. She is now a Ph.D. candidate in Computer Science and Technology from East China Normal University, Shanghai, China. Her research interests include non-volatile memory, optimization algorithms and computer architecture.

Xiaowei Xu is currently an associate professor at Guangdong Cardiovascular Institute, Guangdong Provincial People's Hospital, Guangzhou, China. He received the BS and Ph.D. degrees in electronic science and technology from Huazhong University of Science and Technology, Wuhan, China, in 2011 and 2016 respectively. He worked as a postdoc researcher at University of Notre Dame, IN, USA from 2016 to 2019. His research interests include deep learning, and medical image segmentation. He was a recipient of DAC system design contest special service recognition reward in 2018 and outstanding contribution in reviewing, Integration, the VLSI journal in 2017. He has served as TPC members in ICCD, ICCAD, ISVLSI and ISQED.

Bingzhe Li received the Ph.D. degree in electrical and computer engineering from the University of Minnesota, Twin Cities in 2018. After that, he worked as a postdoctoral associate in Department of Computer Science and Engineering, University of Minnesota, Twin Cities. He is currently an assistant professor of electrical and computer engineering at Oklahoma State University. His research interests focus on memory and storage systems, DNA storage, emerging storage systems for big data applications including machine learning, blockchain, key–value store, etc., and low-cost computing architecture.

Lei Yang is currently an Assistant Professor in the Department of Information Sciences and Technology at George Mason University. She received her Ph.D. degree and B.E. degree in 2019 and 2013, respectively, from Chongqing University, China. Her primary research interests lie in the joint area of Hardware/Software Co-Exploration for Neural Network Architectures, Embedded Systems, and High-Performance Computing. She is passionate about Automated Machine Learning, and System-Level Design and Optimization for Applied Machine Learning. She has authored and co-authored more than 50 research articles in refereed international conferences and premier journals, including DAC, CODES+ISSS, ASP-DAC, ICCD, HPCC, RTCSA, IEEE Transactions (TC, TPDS, TVLSI, TCAD), ACM Transactions (TECS), and FGCS. She has received IEEE Transactions on Computer-Aided Design (TCAD) for the 2021 Donald O. Pederson Best Paper Award and Best Paper Award in ICCD 2017, and five Best Paper Nominations in ASP-DAC 2020, CODES+ISSS 2019, DAC 2019, ASP-DAC 2019, and ASP-DAC 2016.