Co-Exploration of Neural Architectures and Heterogeneous ASIC Accelerator Designs Targeting Multiple Tasks

Lei Yang[1] Zheyu Yan[1] Meng Li[2] Hyoukjun Kwon[3] Liangzhen Lai[2] Tushar Krishna[3] Vikas Chandra[2]
Weiwen Jiang[1,*] Yiyu Shi[1]

[1] University of Notre Dame [2] Facebook [3] Georgia Institute of Technology

***Abstract*—Neural Architecture Search (NAS) has demonstrated its power on various AI accelerating platforms such as Field Programmable Gate Arrays (FPGAs) and Graphic Processing Units (GPUs). However, it remains an open problem how to integrate NAS with Application-Specific Integrated Circuits (ASICs), despite them being the most powerful AI accelerating platforms. The major bottleneck comes from the large design freedom associated with ASIC designs. Moreover, with the consideration that multiple DNNs will run in parallel for different workloads with diverse layer operations and sizes, integrating heterogeneous ASIC sub-accelerators for distinct DNNs in one design can significantly boost performance, and at the same time further complicate the design space. To address these challenges, in this paper we build ASIC template set based on existing successful designs, described by their unique dataflows, so that the design space is significantly reduced. Based on the templates, we further propose a framework, namely ASICNAS, which can simultaneously identify multiple DNN architectures and the associated heterogeneous ASIC accelerator design, such that the design specifications (specs) can be satisfied, while the accuracy can be maximized. Experimental results show that compared with successive NAS and ASIC design optimizations which lead to design spec violations, ASICNAS can guarantee the results to meet the design specs with 17.77%, 2.49×, and 2.32× reductions on latency, energy, and area and less than 1.6% accuracy loss. To the best of the authors' knowledge, this is the first work on neural architecture and ASIC accelerator design co-exploration.**

I. Introduction

Recently, Neural Architecture Search (NAS) [1]–[3] successfully opens up the design freedom to automatically identify neural architectures with the maximum accuracy; in addition, hardware-aware NAS [4]–[7] further enables hardware design space to jointly identify the best architecture and hardware designs to maximize network accuracy and hardware efficiency. Most of existing hardware-aware NAS focus on GPUs or Field Programmable Gate Arrays (FPGAs).

On the other hand, among all AI accelerating platforms, application-specific integrated circuits (ASICs), composed of processing elements (PEs) connected in different topologies, can provide incomparable energy efficiency, latency, and form factor [8], [9]. Most existing ASIC accelerators, however, target common neural architectures [8], [10], [11] and do not reap the power of NAS. Though seemingly straightforward, integrating NAS with ASIC designs is not a simple matter, as can be seen from image classification in Fig. 1. Neural architecture search space is formed by ResNet9 [12] with adjustable hyperparameters. Hardware design space is formed by ASICs with adjustable number of PEs and interconnections. Results are depicted in a three-dimensional space, where three axes represent different hardware metrics and each point represents a solution of paired neural architecture and ASIC design. We can see that when NAS and ASIC design are performed successively, all solutions (denoted by circles) violate user-defined hardware design specifications (design specs, denoted by diamond). When NAS is done in aware of a particular ASIC design, the resulting solution (denoted by triangle) has lower

* Weiwen Jiang is the corresponding author (wjiang2@nd.edu)

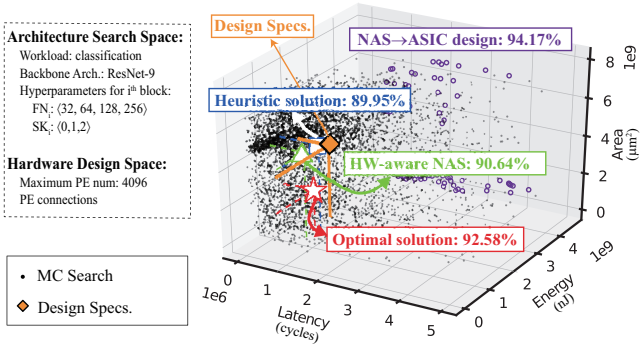

Figure 1: Neural architecture search space and hardware design space exploration: solutions from successive NAS and ASIC design; solution from NAS in aware of an ASIC design; the closest-to-spec solution; and the optimal solution from 10,000 Monte Carlo (MC) runs. (Best viewed in color)

accuracy compared with the optimal one (denoted by star) from 10,000 Monte Carlo runs, which uses a different ASIC design. A simple heuristic to pick a solution with latency, energy and area closest to the design specs (denoted by square) would also be sub-optimal. It is therefore imperative to jointly explore neural architecture search space and hardware design space to identify the optimal solution.

However, such a task is quite challenging, primarily due to the large design space of ASICs where a same set of PEs can constitute numerous topologies (and thus dataflows). Enumeration is simply out of the question. In addition, when ASIC accelerators are deployed on edge, they usually need to handle multiple tasks involving multiple DNNs. For instance, tasks like object detection, image segmentation, and classification can be triggered simultaneously on augmented reality (AR) glasses [13], each of which relies on one kind of DNN. Since DNNs for different tasks can have distinct architectures, one dataflow cannot fit all of them; meanwhile, multiple tasks need to be executed concurrently, which requires task-level parallelism. As such, it is best to integrate multiple heterogeneous sub-accelerators (corresponding to different dataflows) into one accelerator to improve performance and energy efficiency, which has been verified in [14]. Yet this further complicates the design space.

To address these challenges, in this paper, we establish a link between NAS and ASIC accelerator design. Instead of a full-blown exploration of the design space, we observe that there already exist a few great ASIC accelerator designs such as Shidiannao [10], NVDLA [11], and Eyeriss [8]. Each of these designs has its unique dataflow, and the accelerator is determined once the hardware resource associated with the dataflow is given. As such, we can create a set of ASIC templates, where each template corresponds to one specific dataflow, so that the design space can be significantly narrowed down to the selection of templates to form a heterogeneous accelerator, and the

978-1-7281-1085-1/20/$31.00 ©2020 IEEE

allocation of hardware resource (e.g., the number of PEs and NoC bandwidth) to the selected templates.

Based on the template concept, we then further propose a neural architecture and ASIC design co-exploration framework, namely ASICNAS, for edge devices with multiple tasks. The objective is to identify the best neural architectures for each task and ASIC design, such that all design specs can be met while the accuracy of neural architectures can be maximized. Specifically, we devise a novel controller that can simultaneously predict hyperparameters of multiple DNNs together with the parameters of hardware resource allocation for different template selections. Based on state-of-the-art cost model [15], we separately explore mapping and scheduling of neural architectures onto ASIC templates. Finally, a reward is generated to update the controller. To accelerate the search process, we apply the early pruning technique to remove neural architectures that cannot satisfy design specs without training. Experimental results on three workloads with different tasks show that compared with solutions generated by successive NAS and ASIC design optimization which cannot satisfy design specs, those from ASICNAS can guarantee to meet design specs with 17.77%, 2.49×, and 2.32× reductions in latency, energy, and area and less than 1.6% accuracy loss. Furthermore, compared with hardware-aware NAS for a fixed ASIC design, ASICNAS can achieve 3.65% higher accuracy. To the best of authors' knowledge, this is the first work on neural architecture and ASIC design co-exploration.

II. Background and Challenges

We are now witnessing the rapid growth of NAS. Since the very first work for NAS with reinforcement learning [1], there has been tremendous work to study efficient neural architecture search [2], [3]. Integrating hardware awareness in the search loop opens a new research direction, which attracts research efforts on hardware-aware NAS [4], [5]. Taking one step further, most recently, co-exploration of neural architecture and hardware design is proposed [6], [7]. Unlike the original NAS with mono-objective on maximizing accuracy, those hardware-aware NAS frameworks take inference latency into consideration, and push forward the deployment of DNNs on edge devices. NAS has been applied to GPUs and FPGAs but not ASICs, though they are the most efficient ones among all AI accelerating platforms. Two so-far-unseen but urgent-to-solve challenges exist.

Challenge 1: *How to enable the co-exploration of neural architectures and ASIC accelerator designs?*

The large design space of ASIC accelerators hinders the application of NAS to ASIC accelerators. Unlike GPUs with fixed hardware or FPGAs with well-structured hardware, ASIC designs grant the maximum flexibility to designers to determine the hardware organization. This enables to pursue the maximum efficiency; however, it significantly enlarges the design space. Fortunately, there exist extensive research works in designing ASIC AI accelerators [8], [10], [11], making it possible to shrink the design space on top of existing designs.

Among all ASIC accelerator designs, one of the key observations is that each design has a specific dataflow, such as Shidiannao [10], NVDLA [11] , and Eyeriss [8] styles. For instance, NVDLA [11] involves an adder-tree to calculate the partial sum of output feature maps. Inspired by this, we build a set of accelerator templates, each of which has a dataflow style, resulting in a fixed hardware structure. On top of it, we only need to allocate resource for templates, without changing hardware structures. Thus, design space can be significantly shrunk, and in turn, it enables co-exploration of neural architectures and ASIC designs by incorporating hardware allocation parameters.

Challenge 2: *Multiple neural architectures need to be identified under the unified design spec.*

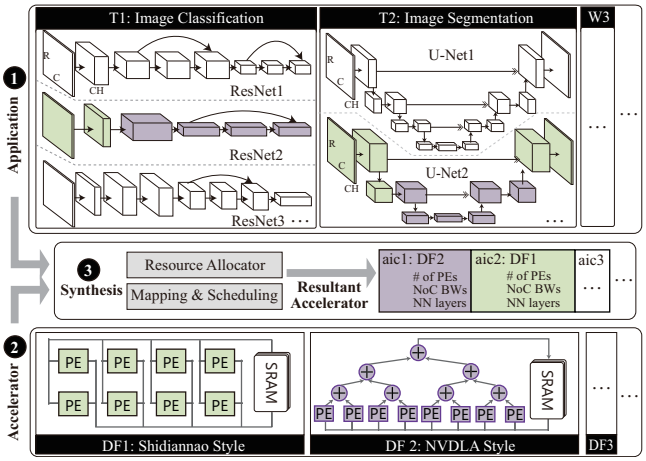

Figure 2: Overview: co-exploration with three layers of optimizations.

Another challenge is that realistic applications on edge devices require the collaboration of multiple tasks, which involves multiple DNNs. In addition, all these DNNs will be executed on the accelerator with unified design specs, including latency, energy, and area. In consequence, sequentially optimizing each DNN using hardware-aware NAS will not work; instead, the multiple neural architectures need to be simultaneously optimized under the unified design specs.

Integrating multiple DNNs in one accelerator brings one further challenge. DNNs for different tasks have distinct architectures, yet one dataflow is not suitable for all architectures. For instance, NVDLA style [11] (DF_2 in Fig. 2) loads one pixel from each activation channel for one computation. In order to fully use the computation resource, it favors convolution layers with large activation channel but low activation resolution; while Shidiannao style [10] (DF_1 in Fig. 2) is on the opposite. As a result, NVDLA style works better for ResNets, while Shidiannao works better for U-Nets. As demonstrated in [14], we can integrate multiple heterogeneous sub-accelerators using a network-on-chip style through Network Interface Controller (NIC) in one AISC accelerator, which further complicates the design space.

In this work, we will address the above challenges.

III. Problem Definition

In this section we will first define multi-task workloads and heterogeneous accelerators, and then formulate the problem of neural architecture and ASIC design co-exploration.

Fig. 2 demonstrates an overview of the co-exploration problem, which involves three exploration layers: ❶ "Application", ❷ "Accelerator", and ❸ "Synthesis". The application layer determines the neural architectures to be applied, while the accelerator layer creates the ASIC template set based on the dataflow style of existing accelerator designs. Acting as the bridge, the synthesis layer allocates a template together with the resource to each sub-accelerator, and maps/schedules network layers to sub-accelerators. In the following text, we will define each exploration layer in detail.

❶ **Application.** The application workload considered in this work has multiple AI tasks which involve a DNN model for each task. A workload with m tasks is defined as $W = \langle T_1, T_2, \cdots, T_m \rangle$. Fig. 2 shows an example with two tasks (i.e., T_1 for classification and T_2 for segmentation). Task $T_i \in W$ corresponds to a DNN architecture D_i, which forms a set D with m DNN architectures. We define a DNN architecture as $D_i = \langle B_i, L_i, H_i, acc_i \rangle$, which is composed of a backbone architecture B_i, a set of layers L_i, a set of hyperparameters H_i, and an accuracy acc_i. For example, in Fig. 2, B_1 for classification

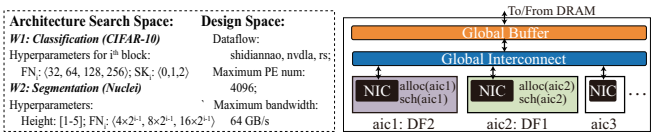

Figure 3: Left: search spaces for both NAS and ASIC accelerator designs. Right: the resultant heterogeneous ASIC accelerator.

task T_1 is ResNet9 [12], and its hyperparameters include the number of filter (FN) and the number of skip layers (SK) for each residual block, as shown in Fig. 3 (left); while for segmentation task T_2, the backbone architecture B_2 is U-Net [16] whose hyperparameters include the height ($Height$) and filter numbers (FN) for each layer.

Based on above definition, we define the neural architecture search function $H_i = nas(D_i)$, which determines hyperparameters H_i in DNN D_i to identify one neural architecture. Kindly note that NAS [1] is to determine $nas(D_i)$ with the mono-objective of maximizing accuracy acc_i. As shown in Fig. 2, each set of hyperparameters corresponds to one neural architecture, and we will determine $nas(D_i)$ to identify a specific neural architecture for task T_i (colored ones).

❷ **ASIC Accelerator.** A heterogeneous ASIC accelerator formed by multiple sub-accelerators connected in a Network-on-Chip (NoC) style through NIC is shown in Fig. 3 (right). Define $AIC = \langle aic_1, aic_2, \cdots aic_k \rangle$ to be a set of k sub-accelerators. A sub-accelerator $aic_i = \langle df_i, pe_i, bw_i \rangle$ has three properties: the dataflow style df_i, the number of processing elements pe_i, and the NoC bandwidth bw_i. With a set of predefined dataflow templates to choose from, as shown in Fig. 2, the ASIC design space is significantly narrowed down from choosing specific unrolling, mapping and data reuse patterns to allocating resources (one template with associated PEs and bandwidth) to each sub-accelerator. Kindly note that according to the template and mapped network layers, the memory sizes can be determined to support the full use of hardware, as in [15]. Therefore, memory sizes are not appeared in the search space.

❸ **Synthesis.** Based on the definition of applications and accelerators, next, we present the synthesis optimization.

Resource allocation. On the hardware side, we design each sub-accelerator in set $AIC = \langle aic_1, aic_2, \cdots aic_k \rangle$, given a set of dataflow templates $DF = \langle DF_1, DF_2, \cdots DF_q \rangle$, the maximum number of PEs (e.g., $NP = 4096$) and the maximum bandwidth (e.g., $BW64GB/s$. Note that since DF contains different dataflows, the resultant accelerator will be heterogeneous if more than one type of dataflows are mapped to AIC. By reducing the size of DF to one, the proposed techniques can be used for homogeneous designs.

We define an allocation function $alloc(aic_i)$ to determine the dataflow template from DF, and the PEs and bandwidth used for aic_i, such that $\sum_{i=1\cdots|AIC|}\{pe_i\} \leq NP$ and $\sum_{i=1\cdots|AIC|}\{bw_i\} \leq BW$. As an example, Fig. 2 illustrates two kinds of dataflow templates: shidiannao [10] and NVDLA [11]. The resultant accelerator (in Fig. 2 ❸) is composed of two heterogeneous sub-accelerators with different dataflow templates, PE numbers and bandwidth.

Mapper and scheduler. On the software side, we map network layers to sub-accelerators and determine their execution orders on each sub-accelerator. We define a map function $map(l_{i,j}) = aic_k$, which indicates the j^{th} network layer $l_{i,j}$ in the i^{th} DNN D_i to be mapped to the k^{th} sub-accelerator aic_k. Based on the mapping, we determine the execution order of network layers on sub-accelerator aic_k following a schedule function $sch(aic_k)$.

The synthesis results can be evaluated via four metrics, including accuracy, latency, energy, and area. In this work, we aim to maximize the accuracy of DNNs under the given design specs on latency (LS), energy (ES) and area (AS).

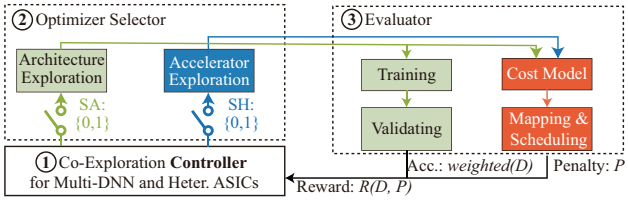

Figure 4: ASICNAS: parameters for neural architecture and accelerator are first determined by controller; then the identified neural architecture and accelerator will be evaluated; finally, a reward will be generated by the evaluation results to feedback and update the controller.

Problem Definition. Based on all the above definitions, we formally define the optimization problem as follows: given a multi-task workload W, the backbone neural architecture for each DNN in set D, a set of sub-accelerators AIC, a set of dataflow templates DF, the maximum number of PEs and bandwidth, and design specs (LS, ES, AS), we will determine:

- $nas(D_i)$: the neural architecture of each DNN $D_i \in D$;
- $alloc(aic_k)$: the dataflow and resource allocation for each sub-accelerator $aic_k \in AIC$;
- $map(l_{i,j})$ and $sch(aic_k)$: the mapping of network layers to sub-accelerators and their schedule orders;

such that the maximum accuracy of DNNs can be achieved while all design specs and resource constraints can be met; i.e., $max = weighted(D)$, $s.t.$, $rl \leq LS$, $re \leq ES$, $ra \leq AS$ $\sum_{i=1\cdots|AIC|}\{pe_i\} \leq NP$, $\sum_{i=1\cdots|AIC|}\{bw_i\} \leq BW$, where rl, re, ra represent latency, energy, and area of the resultant accelerator, and a $weighted$ function defined in next section is to get the accuracy of all networks, which can be functions like avg (maximize the average accuracy) or min (maximize the minimum accuracy).

IV. Proposed Co-Exploration Framework: ASICNAS

This section will present the details of ASICNAS that addresses the problem formulated in Section III. Fig. 4 demonstrates the overview of ASICNAS. It contains three components, ① controller, ② optimizer selector, and ③ evaluator. In general, the controller samples neural architectures and hardware resource allocation in each episode (aka. iteration). Then the predicted sample goes through the optimizer selector and evaluator to generate accuracy and hardware cost. Finally, a reward is generated to update controller. All components work together to generate solutions with high weighted accuracy and to meet all design specs. To illustrate ASICNAS, we apply reinforcement learning approach in this paper. Based on the formulated reward function, other optimization approaches, such as evolution algorithms, can also be applied. Note that since hardware constraints are non-differentiable, differentiable neural architecture search (DARTS) cannot be applied. Then, we will introduce each component in detail.

① **Multi-Task Co-Exploration Controller.** The controller is the key component in ASICNAS. Driven by the requirement of multi-task in one application workload, we propose a novel reinforcement-learning based Recurrent Neural Network (RNN) controller to simultaneously predict multiple neural architectures. In addition, we integrate accelerator design parameters into the controller to realize a genuine co-exploration of neural architectures and hardware designs.

Fig. 5 demonstrates the proposed controller. It is composed of N segments, where N is the sum of task number in workload $W = \{T_1, T_2, \cdots, T_m\}$ and sub-accelerator number in set $AIC = \{aic_1, aic_2, \cdots, aic_k\}$; i.e., $N = m + k$. The first m segments correspond to m DNNs, while the remaining segments correspond

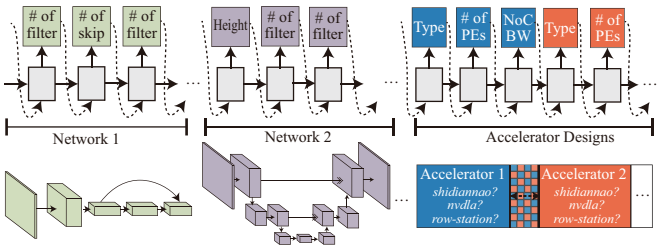

Figure 5: Co-exploration controller for multiple tasks: determine neural architecture hyperparameters, and hardware design parameters.

to k sub-accelerators. For the segment associated with a DNN, say $D_i\rangle$, its outputs determine D_i's hyperparameters, i.e., the $nas(D_i)$ function. For instance, in Fig. 5, the first segment predicts the filter numbers (FN) and skip layers (SK). Similarly, the segment for sub-accelerator aic_k determines its hardware design parameters, i.e., the $alloc(aic_k)$ function, as shown in the right part of Fig. 5.

We employ reinforcement learning method to update the controller and predict new samples. Specifically, in each episode, the controller first predicts a sample, and gets its reward R based on the evaluation results form components ③ and ④. Then, we employ the Monte Carlo policy gradient algorithm [17] to update the controller:

$$\nabla J(\theta) = \frac{1}{m}\sum_{k=1}^{m}\sum_{t=1}^{T}\gamma^{T-t}\nabla_\theta \log \pi_\theta(a_t|a_{(t-1):1})(R_k - b) \quad (1)$$

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 γ and the baseline b is the average exponential moving of rewards.

② **Optimizer Selector.** We integrate an optimizer selector in ASICNAS to accelerate the search process. This is based on the observation that the speed of hardware evaluation is much faster than the training process. Specifically, as shown in Fig. 4, we add two switches (SA for neural architecture exploration and SH for hardware design exploration). In terms of the status of switches, the framework can perform different functions listed as follows:

- $SA = 1, SH = 0$, it performs conventional NAS, like [1].
- $SA = 0, SH = 1$, it uses the previous neural architecture and explore hardware designs only. In this case, we aim to obtain valid accelerator design for the neural architecture, and therefore, we do not consider the accuracy in reward.
- $SA = 1, SH = 1$, it predicts new neural architectures and hardware designs.

ASICNAS repeatedly conducts the following two steps β times: (1) both SA and SH are closed for 1 step, aiming to obtain new neural architecture and hardware design; (2) the switch SA is opened for ϕ steps, aiming to explore the best hardware for a previous identified neural architecture. Kindly note that the first step is carried out in a non-blocking scheme, such that one training and β times hardware exploration can be conducted in parallel. Once all hardware explorations are completed and no feasible hardware design is found, it will terminate the training process to accelerate the search process.

③ **Evaluator.** The evaluator contains two paths: (1) via the training and validating to obtain networks' accuracy; (2) via cost modeling, mapping and scheduling to generate penalty in terms of design specs.

Training and validating In this path, we are given the hyperparameters H_i for DNN architecture D_i. For each DNN $D_i \in D$, we train it from scratch and obtain its accuracy acc_i on a held-out validation dataset. Based on the accuracy, we obtain the weighted accuracy $weighted(D)$ for calculating the reward R as follows:

$$weighted(D) = \sum\nolimits_{i=1,2,\cdots,|W|}\{\alpha_i \times acc_i\} \quad (2)$$

where $|W|$ is the total number of tasks in the given workload, and α_i is a weight ranging from 0 to 1, such that $\sum_{i=1,2,\cdots,|W|}\{\alpha_i\} = 1$.

Mapping and scheduling On this path, we are given a set of identified DNN architectures D and a set of determined sub-accelerator AIC. We need to get the hardware metrics including latency rl, energy re, and area ra. ASICNAS incorporates the state-of-the-art cost model, MAESTRO [15], and a mapping and scheduling algorithm to obtain the above metrics. For area ra, we can directly obtain it from MAESTRO with the given sub-accelerator AIC. The latency rl and energy re are determined by the mapping and scheduling. To develop an algorithm for mapping and scheduling, we need to obtain the latency and energy of each layer on different sub-accelerators. Let $L = \bigcup_{D_k \in D}\{L_k\}$ be the layer set. For a pair of network layer $\forall l_i \in L$ and sub-accelerator $aic_j \in AIC$, we can input them to MAESTRO to obtain the latency $l_{i,j}$ and energy $e_{i,j}$.

The problem can be proved to be equivalent to the traditional heterogeneous assignment problem [18], [19]: given the latency $l_{i,j}$ and energy cost $e_{i,j}$ for each layer i on sub-accelerator j, the dependency among layers, and a timing constraint LS, we are going to determine the mapping and scheduling order of each layer on one sub-accelerator, such that the energy cost re is minimized while latency $rs \leq LS$. We denote HAP to be an optimal solver, i.e., $re = HAP(D, AIC, LS)$. Then, we have the following theorem.

Theorem *Given a layer set D, a sub-accelerator set AIC, and design specs on latency LS and energy ES, the design specs can be met if and only if $re = HAP(D, AIC, LS) \leq ES$.*

The above theorem can be proved using contradiction. Due to the space limitation, the detailed proof is omitted. Based on this theorem, we can obtain latency rl and energy re by the solver HAP, which can be instantiated by Integer-Linear Programming (ILP) for the optimal solution; however, since ILP is time-consuming, this paper applies a heuristic approach in [19] to accelerate the search process. On top of the obtained hardware metrics and the given design specs, we formulate a penalty function. Penalty is determined in terms of the degree that the solution beyond the design specs, and no penalty if all design specs are met, which is formulated as follows:

$$P = \frac{\max(rl - LS, 0)}{(bl - LS)} + \frac{\max(re - ES, 0)}{(be - ES)} + \frac{\max(ra - AS, 0)}{(ba - AS)} \quad (3)$$

where bl, be, ba is the upper bound for each metric, which can be obtained by exploring the hardware design space using the neural architecture identified by NAS, as the circles in Fig. 1.

Finally, based on all the above evaluation results, we calculate the reward with a scaling variable ρ, listed as follows:

$$R(D, P) = weighted(D) - \rho \times P \quad (4)$$

V. Experimental Evaluation

We evaluate the efficacy of the proposed framework, ASICNAS, using different application workloads and hardware configurations. Results reported in this section demonstrate that ASICNAS can efficiently identify accurate neural architectures together with AISC accelerator designs that are guaranteed to meet the given design specs, while achieving high accuracy for multiple AI tasks.

A. Evaluation Environment

Application workloads: We use typical workloads on AR glasses to demonstrate the efficacy of ASICNAS. In these workloads, the core tasks involve classification and segmentation, where representative datasets such as CIFAR-10, STL-10, and Nuclei are commonly employed, along with light-weight neural architectures. We synthesize the following three workloads.

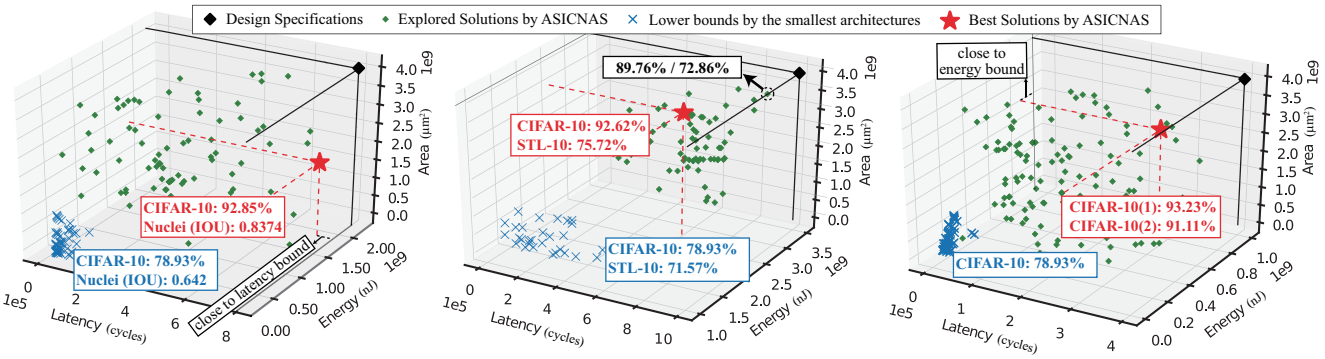

Figure 6: Exploration results obtained by ASICNAS for three different workloads under design specs: (left) $W1$ with CIFAR-10 and STL-10 datasets; (middle) $W2$ with CIFAR-10 and Nuclei; (right) $W3$ with CIFAR-10 dataset. (Best viewed in color)

- W1: Tasks on one classification dataset (CIFAR-10) and one segmentation dataset (Nuclei).
- W2: Tasks on two classification datasets (CIFAR-10, STL-10).
- W3: Tasks on the same classification dataset (CIFAR-10).

The backbone architectures and search space for above tasks are defined as follows. For classification tasks, we select ResNet9 [12], which contains multiple residual blocks, as architecture backbone. During NAS, the number of convolution layer and the number of filter channels for each residual block are searched and then determined. For CIFAR-10, we employ 3 residual blocks, and parameter options for each block are depicted in Fig. 1(a); while for STL-10, considering its input images have higher resolution (i.e., 96×96 pixels), we deepen network to 5 residual blocks, and increase the maximum number of convolution layers in each residual block to 3 and the maximum number of filter channel to 512 for each block. For segmentation tasks, we use U-Net [16] as architecture backbone. Search space for this backbone architecture includes the number of height and filter channel in each layer, as shown in Fig. 1. Note that we follow the standard NAS [1] to hold out a part of data from training images to be the validation set, and the training parameters (e.g., batch size, learning rate, and etc.) follow ResNet9 [12] and U-Net [16].

Hardware configuration: Accelerator design includes hardware resource allocation to sub-accelerators, and dataflow selection for each sub-accelerator. For resource allocation, we set the maximum number of PEs as 4096 and the maximum NoC bandwidth as 64GB/s, in accordance to [14]. Note that, ASICNAS can support arbitrary number of sub-accelerators; for simple demonstration, we make a case study by integrating two sub-accelerators. Specifically, each sub-accelerator uses one of the following dataflows: Shidiannao (abbr. shi) [10], NVDLA (abbr. dla) [11], and row-stationary [8] style. In the case where one sub-accelerator has no resource allocation, the design degenerates to a single large accelerator; for another one, sub-accelerators have exactly the same allocation to degenerate homogeneous accelerators.

Hardware constraints on latency, energy and area will be set by designers (users), according to their own use cases. To evaluate the effectiveness of ASICNAS, we set distinct and strict design specs, including **L**atency ($cycles$), **E**nergy (nJ), **A**rea (μm^2), for each application workload as follows: $\langle 8e5, 2e9, 4e9 \rangle$ for $W1$; $\langle 1e6, 3.5e9, 4e9 \rangle$ for $W2$; $\langle 4e5, 1e9, 4e9 \rangle$ for $W3$.

ASICNAS setting: For exploration parameters, we set $\beta = 500$ and $\phi = 10$, indicating that we explore the search space for 500 episodes and explore 10 accelerator designs in each episode. For reward calculation parameters, we set $\alpha_1 = \alpha_2 = 0.5$ to calculate the weighted accuracy, and $\rho = 10$. Controller RNN is trained using RMSProp optimization, with the initial learning rate of 0.99 and exponential decay of 0.5 for 50 steps. All experiments are conducted on a server with a 48-thread Intel Xeon CPU and one P100 GPU. ASICNAS only takes 3.5 GPU Hours to complete the exploration for each workload, which mainly benefits from the early pruning from optimizer selector component in ASICNAS (See Section IV ②).

B. Design Space Exploration

Fig. 6 demonstrates the exploration results of ASICNAS on three application workloads. In this figure, the x-axis, y-axis, and z-axis represent latency, energy and area, respectively. The black diamond indicates the design specs (upper bound); each green diamond is a solution (neural architecture-ASIC design pair) explored by ASICNAS; each blue cross is a solution based on the smallest neural network in the search space combined with different ASIC designs (lower bound); and the red star refers to the best solution in terms of the average accuracy explored by ASICNAS. The numbers in the rectangles with blue, green, and red colors represent the accuracy of the smallest network, the inferior solutions, and our best solutions, respectively.

We have several observations from Fig. 6. First, ASICNAS can guarantee that all the explored solutions meet design specs. Second, the identified solutions have high accuracy. The accuracy on CIFAR-10 for the four solutions are 92.85%, 92.62%, 93.23%, and 91.11%, while the accuracy lower bounds from the smallest network is 78.93%. Similarly, for STL-10, the accuracy is 75.72% compared with the lower bound of 71.57%. For Nuclei, the IOU (Intersection Over Union) is 0.8374 compared with the lower bound of 0.6462. Third, we observe that the best solutions of $W1$ and $W3$ identified by ASICNAS are quite close to the boundary defined by one of the three design specs, which indicates that in these cases the accuracy is bounded by resources. For $W1$, the energy of the identified solution is 97.12% of the spec; while for $W3$, the latency of the identified solution is 93.4%. This gives designers insights on if/where hardware bottleneck is that prevents the accelerator from getting higher accuracy, and thus they can loose such constraint to increase accuracy if necessary. On the other hand, for $W2$ (middle of Fig. 6), our best solution is farther away from the specs compared with solution S pointed out by the arrow (S is one of the explored solutions by ASICNAS). However, the accuracy of S for CIFAR-10 and STL-10 are 2.86% and 2.91% lower than the best solution. This reflects that the best solution may not always be the one closest to the specs, and therefore, heuristics that select the solution that is closest to the specs cannot work.

C. Results on Multiple Tasks for Multiple Datasets

Table I reports the comparison results on multi-dataset workloads. We implement two additional approaches. 1) "NAS→ASIC" indicates successive NAS [1] and brute-force hardware exploration. 2) in "ASIC→HW-NAS", a Monte Carlo search with 10,000 runs will

Table I: Comparison between successive NAS and ASIC design (NAS→ASIC), ASIC design followed by hardware-aware NAS (ASIC→HW-NAS), and ASICNAS.

Work.	Approach	Hardware	Dataset	Accuracy	L /$_{cycles}$	E /$_{nJ}$	A /$_{\mu m^2}$
W1	NAS→ASIC	$\langle dla, 2112, 48\rangle$	CIFAR-10	94.17%	9.45e5	3.56e9	4.71e9
		$\langle shi, 1984, 16\rangle$	Nuclei	83.94%	×	×	×
	ASIC→ HW-NAS	$\langle dla, 1088, 24\rangle$	CIFAR-10	91.98%	5.8e5	1.94e9	3.82e9
		$\langle shi, 2368, 40\rangle$	Nuclei	83.72%	✓	✓	✓
	ASICNAS	$\langle dla, 576, 56\rangle$	CIFAR-10	92.85%	**7.77e5**	**1.43e9**	**2.03e9**
		$\langle shi, 1792, 8\rangle$	Nuclei	83.74%	✓	✓	✓
W2	NAS→ASIC	$\langle dla, 2368, 56\rangle$	CIFAR-10	94.17%	9.31e5	3.55e9	4.83e9
		$\langle shi, 1728, 8\rangle$	STL-10	76.50%	✓	×	×
	ASIC→ HW-NAS	$\langle dla, 2112, 24\rangle$	CIFAR-10	92.53%	9.69e5	2.90e9	3.86e9
		$\langle shi, 1536, 40\rangle$	STL-10	72.07.%	✓	✓	✓
	ASICNAS	$\langle dla, 2112, 40\rangle$	CIFAR-10	92.62%	**6.48e5**	**2.50e9**	**3.34e9**
		$\langle shi, 1184, 24\rangle$	STL-10	75.72%	✓	✓	✓

×: violate design specs; ✓ : meet design specs.

first be conducted to obtain the ASIC design closest to the design specs. Then, for that specific ASIC design, we extend hardware-aware NAS [20] to identify the best neural architecture under design specs.

Results in Table I demonstrate that for the neural architectures identified by NAS, none of the accelerator designs explored by the brute-force approach can provide a legal solution that satisfies all design specs. On the contrary, for both workloads, ASICNAS can guarantee the solutions to meet all specs with accuracy loss less than 1.6%. For workload $W1$, ASICNAS achieves 17.77%, 2.49×, and 2.32× reduction on latency, energy, and area, respectively, against NAS→ASIC. For workload $W2$, the reduction numbers are 30.39%, 29.58%, and 30.85%. When comparing ASICNAS with ASIC→HW-NAS, even though the solution of the latter is closer to the design specs, for W1 ASICNAS achieves 0.87% higher accuracy for CIFAR10 and similar accuracy for Nuclei; for W2 3.65% higher accuracy for STL-10 and similar accuracy for CIFAR-10.

All the above results have revealed the necessity and underscored the importance of co-exploring neural architectures and ASIC designs.

D. From Single and Homogeneous to Heterogeneous ASIC Accelerator

The benefits of heterogeneous accelerators under heterogeneous workloads are evident. Table II reports the comparison results of different accelerator configurations under the homogeneous workload CIFAR-10 (W3). In these approaches, "NAS" explores neural architectures without hardware awareness and the corresponding ASIC applies the maximum hardware resource; "Single Acc.", "Homo. Acc.", "Hetero. Acc" are ASICNAS with single accelerator design, two homogeneous sub-acclerators, and two heterogeneous sub-accelerators. Kindly note that, as discussed in Section V-A, ASICNAS can support the exploration of a single accelerator. We set hardware configurations as follows to guarantee single and homongeneous solutions to meet design specs. For Single Acc., the network will be sequentially executed twice, which indicates that the constraint on latency and energy should be halved. For Homo. Acc., two homogeneous sub-accelerators will run a same network simultaneously, which indicates that the energy and area for each accelerator should be halved.

From the results in Table II, we observe that although NAS can successfully identify the neural architectures with the highest accuracy (94.17%), they cannot satisfy the specs even though all hardware resources are used. In comparison, Single Acc. identifies a relatively smaller neural architecture with less hardware resource, but can meet the specs with the accuracy of 91.45%. Without exploring parallelism, Single Acc. cannot further improve accuracy since it is bounded by latency. After boosting performance, Homo. Acc. identifies the neural architecture with 92.00% accuracy. Exploring the heterogeneous accelerators by ASICNAS, two distinct networks can be generated: one is with accuracy of 93.23%, close to the best result identified by NAS; and the other one with slightly lower accuracy of 91.11% is comparable with that of Single Acc.. This solution will be useful in Ensemble learning [21], and can provide more choices for designers.

Table II: On CIFAR-10 (W3), comparison results of architectures and accelerator designs obtained by different accelerator configurations.

Approach	Hardware	Architecture	Accuracy	Sat.
NAS	$\langle dla, 4096, 64\rangle$	$\langle 32, 128, 2, 256, 2, 256, 2\rangle$	94.17%	×
Single Acc.	$\langle dla, 3104, 24\rangle$	$\langle 8, 32, 2, 128, 1, 256, 1\rangle$	91.45%	✓
Homo. Acc.	$2 \times \langle dla, 1408, 32\rangle$	$2 \times \langle 32, 32, 1, 128, 1, 256, 1\rangle$	92.00%	✓
Hetero. Acc.	$\langle dla, 1760, 56\rangle$	$\langle 8, 64, 2, 256, 2, 256, 2\rangle$	**93.23%**	✓
(ASICNAS)	$\langle shi, 1152, 8\rangle$	$\langle 8, 32, 2, 128, 2, 128, 1\rangle$	91.11%	

$\langle FN_0, FN_1, Sk_1, FN_2, Sk_2, FN_3, SK_3\rangle$: For the i^{th} block, FN_i is filter numbers, SK_i is skip layer numbers. Block 0 is a standard conv instead of residual.

VI. Conclusion

In this work, we have proposed a framework, namely ASICNAS, to co-explore neural architectures and ASIC accelerator designs targeting multiple AI tasks on edges devices. ASICNAS has filled the missing link between NAS and ASIC by creating an accelerator template set in terms of the dataflow style. In addition, a novel multi-task oriented RNN controller has been developed to simultaneously determine multiple neural architectures under a unified design spec. The efficacy of ASICNAS is verified through a set of comprehensive experiments.

Acknowledgement

This work is partially supported by National Science Foundation (NSF) under grants CNS-1822099, SPX-1919167, and OAC-1909900.

References

[1] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *Proc. of ICLR*, 2017.

[2] Hanxiao Liu et al. Darts: Differentiable architecture search. In *Proc. of ICLR*, 2019.

[3] Hieu Pham et al. Efficient neural architecture search via parameter sharing. In *Proc. of ICML*, pages 4092–4101, 2018.

[4] Bichen Wu et al. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proc. of CVPR*, pages 10734–10742, 2019.

[5] Han Cai et al. Proxylessnas: Direct neural architecture search on target task and hardware. In *Proc. of ICLR*, 2019.

[6] Weiwen Jiang et al. Accuracy vs. efficiency: Achieving both through fpga-implementation aware neural architecture search. In *Proc. of DAC*, 2019.

[7] Cong Hao et al. Fpga/dnn co-design: An efficient design methodology for iot intelligence on the edge. In *Proc. of DAC*, 2019.

[8] Yu-Hsin Chen et al. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *Proc. of ISCA*, pages 367–379, 2016.

[9] Angshuman Parashar et al. Scnn: An accelerator for compressed-sparse convolutional neural networks. In *Proc. of ISCA*, pages 27–40, 2017.

[10] Zidong Du et al. Shidiannao: Shifting vision processing closer to the sensor. In *Proc. of ISCA*, pages 92–104, 2015.

[11] NVIDIA. Nvdla deep learning accelerator. *http:// nvdla.org*, 2017.

[12] Chuan Li. https://lambdalabs.com/blog/resnet9-train-to-94-cifar10-accuracy-in-100-seconds. 2019. Accessed: 2019-11-24.

[13] Michael Abrash. https://www.oculus.com/blog/inventing-the-future/. 2019. Accessed: 2019-11-26.

[14] Hyoukjun Kwon et al. Herald: Optimizing heterogeneous dnn accelerators for edge devices. *arXiv preprint arXiv:1909.07437*, 2019.

[15] Hyoukjun Kwon et al. Understanding reuse, performance, and hardware cost of dnn dataflow: A data-centric approach. In *Proc. of MICRO*, pages 754–768, 2019.

[16] Olaf Ronneberger et al. U-net: Convolutional networks for biomedical image segmentation. In *Proc. of MICCAI*, pages 234–241, 2015.

[17] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[18] K. Ito et al. Ilp-based cost-optimal dsp synthesis with module selection and data format conversion. *IEEE Trans. TVLSI*, 6(4):582–594, 1998.

[19] Zili Shao et al. Efficient assignment and scheduling for heterogeneous dsp systems. *IEEE Trans. TPDS*, 16(6):516–525, 2005.

[20] Mingxing an et al. Mnasnet: Platform-aware neural architecture search for mobile. In *Proc. of CVPR*, pages 2820–2828, 2019.

[21] Michael P Perrone and Leon N Cooper. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, 1992.