Auto-NBA: Efficient and Effective Search Over the Joint Space of Networks, Bitwidths, and Accelerators

Yonggan Fu [1] **Yongan Zhang** [1] **Yang Zhang** [2] **David Cox** [2] **Yingyan Lin** [1]

Abstract

While maximizing deep neural networks' (DNNs') acceleration efficiency requires a joint search/design of three different yet highly coupled aspects, including the networks, bitwidths, and accelerators, the challenges associated with such a joint search have not yet been fully understood and addressed. The key challenges include (1) the dilemma of whether to explode the memory consumption due to the huge joint space or achieve sub-optimal designs, (2) the discrete nature of the accelerator design space that is coupled yet different from that of the networks and bitwidths, and (3) the chicken and egg problem associated with network-accelerator co-search, i.e., co-search requires operation-wise hardware cost, which is lacking during search as the optimal accelerator depending on the whole network is still unknown during search. To tackle these daunting challenges towards optimal and fast development of DNN accelerators, we propose a framework dubbed Auto-NBA to enable jointly searching for the **N**etworks, **B**itwidths, and **A**ccelerators, by efficiently localizing the optimal design within the huge joint design space for each target dataset and acceleration specification. Our Auto-NBA integrates a heterogeneous sampling strategy to achieve unbiased search with constant memory consumption, and a novel joint-search pipeline equipped with a generic differentiable accelerator search engine. Extensive experiments and ablation studies validate that both Auto-NBA generated networks and accelerators consistently outperform state-of-the-art designs (including co-search/exploration techniques, hardware-aware NAS methods, and DNN accelerators), in terms of search time, task accuracy, and accelerator efficiency. Our codes are available at: https://github.com/RICE-EIC/Auto-NBA.

[1]Department of Electrical and Computer Engineering, Rice University [2]MIT-IBM Watson AI Lab. Correspondence to: Yingyan Lin <yingyan.lin@rice.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

1. Introduction

The prohibitive complexity of deep neural networks (DNNs) have fueled a tremendous demand for efficient DNN accelerators which could boost the acceleration efficiency by orders-of-magnitude. In response, extensive research efforts have been devoted to developing DNN accelerators. Early works decouple the design of efficient DNN algorithms (Liu et al., 2018b; Wu et al., 2018b; You et al., 2020) and their accelerators (Du et al., 2015; Chen et al., 2017; Li et al., 2020a; Zhao et al., 2020). On the algorithms level, pruning, quantization, or neural architecture search (NAS) have been adopted; On the hardware level, various FPGA-/ASIC-based accelerators customize the *micro-architectures* (e.g., memory hierarchies/size and network-on-chip design) and algorithm-to-hardware *mapping methods* (e.g., loop tiling strategies and loop orders) to optimize the acceleration efficiency for given DNNs. Later, hardware-aware NAS (HA-NAS) was proposed to further improve DNNs' acceleration efficiency (Tan et al., 2019; Fu et al.).

It has been recently recognized that (1) optimal DNN accelerators require a joint consideration for three different yet coupled aspects: the network structure, network precision, and their accelerators, and (2) merely exploring a subset of these aspects will lead to sub-optimal hardware efficiency or task accuracy. For example, the optimal accelerators for DNNs with different structures (e.g., width/depth/kernel-size) can be very different; while the optimal networks and their bitwidths for different accelerators can differ a lot (Wu et al., 2019). However, the direction of jointly designing or searching for all three aspects has only been slightly touched on. For example, (Chen et al., 2018; Gong et al., 2019; Wang et al., 2020) proposed to jointly search for DNNs' structure and precision for a fixed target hardware; (Abdelfattah et al., 2020; Yang et al., 2020; Jiang et al., 2020a;c) proposed to jointly search for the networks and their accelerators, yet either their network or accelerator choices are limited, due to the prohibitive time cost required by their adopted reinforcement learning (RL) based methods; and EDD (Li et al., 2020b) formulated a differentiable joint search framework, which however only consider one accelerator parameter (i.e., parallel factor) and more importantly, has not yet solved the key challenges of efficient joint search.

Although differentiable search is promising in terms of search efficiency to explore the huge joint search space (see Sec. 4.2), plethora of challenges exist to achieve an effective, generic joint search for the above three aspects. First (*Challenge* 1), to co-search for a DNN and its precision, there exists a dilemma whether to activate all the paths during search. On one hand, the required memory can easily explode and thus constrain the search scalability to more complex tasks if all paths are activated; on the other hand, partially activating a subset of the paths require a sequential training of different precisions on the same weights, causing inaccurate accuracy ranking among different precisions (Jin et al., 2020). Second (*Challenge 2*), DNN accelerators' design factors are not differentiable, and it is non-trivial to abstract generic accelerator design spaces integrating all important factors, e.g., the number of memory hierarchies, loop orders/sizes, and processing array size/shape. Third (*Challenge 3*), there exists the chicken and egg problem associated with network-accelerator co-search, i.e., co-search requires operation-wise hardware costs, which is lacking during search as the optimal accelerator depending on the whole network is still unknown during search.

We aim to enable efficient and effective joint search for the mentioned three aspects, and make contributions as follows:

- We propose Auto-NBA that **for the first time** enables **Auto**mated joint search for the **N**etworks, **B**itwidths, and **A**ccelerators for efficiently exploring the huge joint design space which cannot be afforded by previous RL-based methods due to their required prohibitive search cost. Auto-NBA identifies and tackles the above *Challenges* 1-3 towards scalable joint search of the three for maximizing both the accuracy and efficiency.
- We propose a heterogeneous sampling strategy integrated by Auto-NBA for simultaneous updating the weights and network structures to (1) avoid sequentially training different precisions and (2) achieve unbiased search with constant memory consumptions, i.e., solving the above *Challenge* 1. We further develop a novel joint-search pipeline integrating a differentiable accelerator search engine to address *Challenges* 2-3.
- Extensive experiments and ablation studies validate the effectiveness and advantages of our Auto-NBA framework in terms of the resulting search time, task accuracy, and accelerator efficiency, when benchmarked over state-of-the-art (SOTA) co-search/exploration techniques, HA-NAS methods, and DNN accelerators, respectively. Furthermore, we visualize the searched accelerators by Auto-NBA to discuss insights towards efficient DNN accelerator design.
- Auto-NBA's searched algorithms and accelerators outperform both SOTA automatically searched and expert-designed DNNs and accelerators. Additionally, our Auto-NBA is general and allows users to easily plug-in both their own DNN search space and/or accelerator search space. Hence, we believe that Auto-NBA has made a nontrivial step to provide automated tools for expediting the development of DNN accelerators which falls far behind DNN algorithm advances.

2. Related works

Hardware-aware NAS. Hardware-aware NAS (HW-NAS) automates the design of efficient DNNs. Early works (Tan et al., 2019; Howard et al., 2019; Tan & Le, 2019) utilize RL-based NAS that requires a massive search time/cost, while recent works (Wu et al., 2019; Wan et al., 2020; Cai et al., 2018; Stamoulis et al., 2019) adopt differentiable search (Liu et al., 2018a) with much improved searching efficiency. Along another direction, one-shot NAS methods (Cai et al., 2019; Guo et al., 2020; Yu et al., 2020) pretrain the supernet and directly evaluate the performances of the sub-networks in a weight sharing manner as a proxy of their independently trained performances at the cost of a longer pretrain time. Additionally, NAS has been adopted to search for quantization strategies (Wang et al., 2019; Wu et al., 2018a; Cai & Vasconcelos, 2020; Elthakeb et al., 2020) to trimming down the complexity of DNNs. However, these works leave the hardware design space unexplored, which is a crucial enabler for DNN's acceleration efficiency.

DNN accelerators. Motivated by customized accelerators' large potential gains, SOTA accelerators (Du et al., 2015; Chen et al., 2017) innovate micro-architectures and mapping methods to optimize the acceleration efficiency, given a DNN and the hardware specifications. However, it is non-trivial to design an optimal accelerator as it requires cross-disciplinary knowledge in algorithm, micro-architecture, and circuit design. SOTA accelerator design relies on either experts' time-consuming manual design or design flow (Chen et al., 2005; 2009; Rupnow et al., 2011) and DNN accelerator design automation (Wang et al., 2016; Zhang et al., 2018a; Guan et al., 2017; Venkatesan et al., 2019; Wang et al., 2018a; Gao et al., 2017; Xu et al., 2020). As they merely explore the accelerator space, they can result in sub-optimal solutions as compared to SOTA co-search/exploration methods and our Auto-NBA.

Co-exploration/search techniques. Pioneering efforts have been made towards jointly searching of DNNs and their accelerators to some extent. For joint searching for DNNs and their precision, (Chen et al., 2018; Gong et al., 2019; Wang et al., 2020) adopt either differentiable or evolutionary algorithms yet without exploring their hardware accelerators. For joint searching for DNNs and their accelerators, (Abdelfattah et al., 2020; Yang et al., 2020; Jiang et al.,

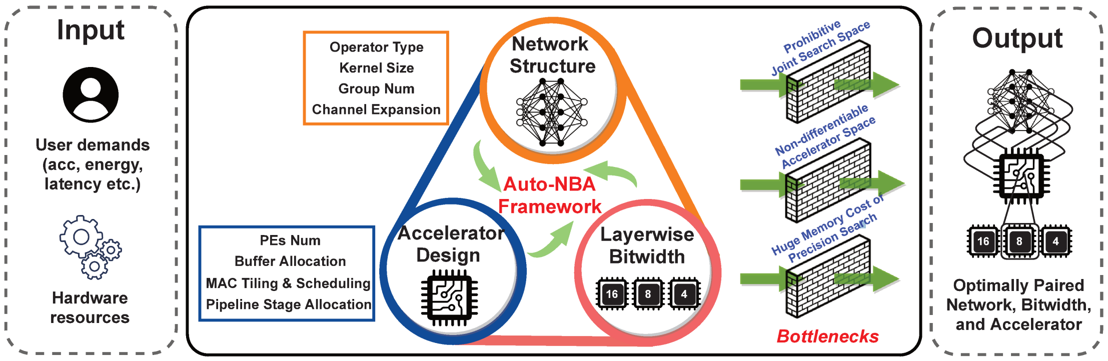
Input
User demands (acc, energy, latency etc.)
Hardware resources
Operator Type
Kernel Size
Group Num
Channel Expansion
Network Structure
Auto-NBA Framework
PEs Num
Buffer Allocation
MAC Tiling & Scheduling
Pipeline Stage Allocation
Accelerator Design
Layerwise Bitwidth
16
8
4
Prohibitive Joint Search Space
Non-differentiable Accelerator Space
Huge Memory Cost of Precision Search
Bottlenecks
Output
Optimally Paired Network, Bitwidth, and Accelerator

Figure 1. Illustrating our Auto-NBA framework: The middle part shows (1) an high-level view of Auto-NBA and (2) the technical challenges that Auto-NBA tackles for enabling a scalable, generic joint-search for the networks, bitwidths, and accelerators.

2020a;c;b) conduct RL-based search for the networks and some accelerator parameters/templates, where they strictly constrain the search space of the network or accelerator to achieve a practical RL search time, limiting their scalability and achievable efficiency. (Lin et al., 2020) attempts to co-design the newtork and accelerator in a sequential manner based on the fact that the accelerator's design cycle is longer than the networks. EDD (Li et al., 2020b) extends differentiable NAS to search for layer-wise precision and the accelerators' parallel factor, which is most relevant to our Auto-NBA. However, it has not yet solved the joint search challenges. First, it does not discuss or address the potentially explosive memory consumption issue of such a joint search; second, EDD's accelerator search space only includes one design parameter (i.e., the parallel factor), which is strictly limited to their accelerator template and cannot generalize to include common accelerator parameters such as the memory hierarchies and tiling strategies.

Auto-NBA targets a scalable, generic joint-search framework for boosting the search efficiency and effectiveness.

3. The Proposed Auto-NBA Framework

In this section, we describe our proposed techniques for enabling Auto-NBA. Sec. 3.1 introduces Auto-NBA's formulation, while Sec. 3.2 and Sec. 3.3 present Auto-NBA's technical enablers that address the key challenges of scalable joint search for the networks, bitwidths, and accelerators, and Sec. 3.4 unifies the enablers to realize Auto-NBA.

3.1. Auto-NBA: Problem Formulation

Fig. 1 shows an overview of Auto-NBA, which jointly searches for the networks (e.g., kernel-size/channel-expansion/group-number), precision (e.g., 4-/6-/8-/12-/16-bit), and the accelerators (e.g., memory size and tiling strategies of each memory) in a differentiable manner. Auto-NBA targets a *scalable* yet *generic* joint search framework, which we formulate as a bi-level optimization problem:

$$\min_{\alpha,\beta} \; L_{val}(\omega^*, net(\alpha), prec(\beta)) \tag{1}$$

$$s.t. \quad L_{cost}(hw(\gamma^*), net(\alpha), prec(\beta)) \leq E_{target}, \tag{2}$$

$$s.t. \quad \omega^* = \arg\min_{\omega} L_{train}(\omega, net(\alpha), prec(\beta)), \tag{3}$$

$$s.t. \quad \gamma^* = \arg\min_{\gamma} L_{cost}(hw(\gamma), net(\alpha), prec(\beta)) \tag{4}$$

where α, β, and γ are continuous variables parameterizing the probability of different choices for the network operators, precision bitwidths, and accelerator parameters as in (Liu et al., 2018a), respectively; ω is the supernet weights; L_{train}, L_{val}, and L_{cost} are the loss during training and validation, and the hardware-cost loss, respectively; E_{target} is the target hardware cost (e.g., energy or latency); and $net(\alpha)$, $prec(\beta)$, and $hw(\gamma)$ denote the network, precision, and accelerator characterized by α, β, and γ, respectively.

3.2. Auto-NBA Enabler 1: Heterogeneous Sampling for Scalable Network-Precision Joint-Search

As discussed in Sec. 1, there exists a dilemma (i.e., either memory explosion or biased search) whether to activate all the paths during precision search, for tackling which we propose a simple yet effective heterogeneous sampling strategy. Here we first use real experiments to illustrate the joint-search dilemma, and then introduce our heterogeneous sampling which effectively address the challenge.

Activating all choices → memory explosion and entangled correlation among choices. During precision search, activating all the precision choices as (Wu et al., 2018a; Gong et al., 2019) can easily explode the memory consumption especially when the precision is co-searched with the network structures. While composite convolutions (Cai & Vasconcelos, 2020) for mixed-precision search can potentially mitigate this memory explosion issue during search by shrinking the required computation, yet the large memory consumption issue would still exist during training when updating the precision parameters, i.e., β in Eq. (1). For example, as shown in Fig. 2 (a), the measured GPU memory consumption of co-searching for the network and precision on ImageNet grows linearly with the number of pre-

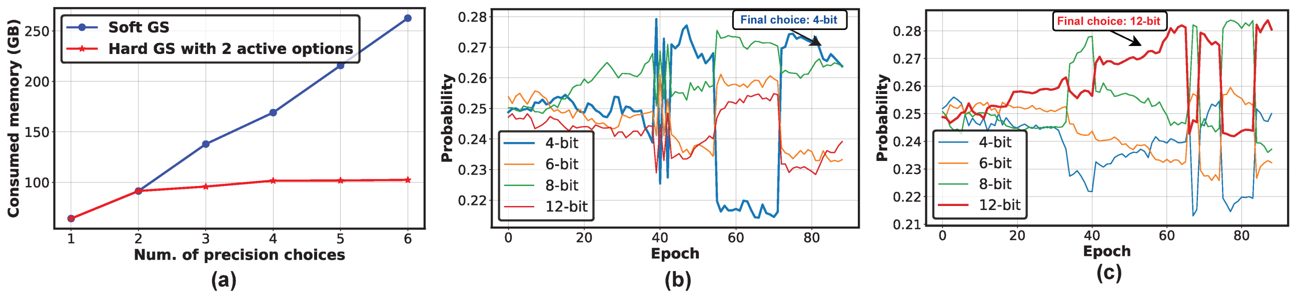

Figure 2. (a) GPU memory consumption comparison between soft Gumbel Softmax (GS) and hard GS sampling, which are two activating approaches for co-search for the network and precision; and the probability evolution of each precision choice during the search process in the 4-th block when searching with: (b) hard GS sampling for updating both the weights ω and precision choices β, which results in the lowest 4-bit, and (c) the proposed heterogeneous sampling for updating ω and β, which results in the highest 12-bit (desired).

Table 1. Comparing the accuracy when training a fixed network using different precision schedules, where high2low and low2high denote progressive training from high precision to low precision and the inverse case, respectively, following (Jin et al., 2020).

Strategy	Resulting Accuracy				
	4-bit (%)	8-bit (%)	12-bit (%)	16-bit (%)	32-bit (%)
Independent Train	63.52	67.44	67.56	67.65	68.21
high2low Train	59.29	45.09	45.45	45.15	65
low2high Train	4.36	26.55	43.58	63.3	63.5
joint Train	63.28	66.98	67.21	67.23	67.36

cision choices if activating all precision choices during search. Furthermore, the entangled correlation (e.g., co-adaptation (Hong et al., 2020), correlation (Li et al., 2019), and cooperation (Tian et al., 2020)) among different precision choices can lead to a large gap between the supernet during search and the final derived network, thus failing the joint search.

Activating only a subset of choices - Biased search. For addressing the above issues of memory explosion and correlation among choices, one natural approach is to adopt hard Gumbel Softmax (i.e., activating only one or a subset of paths as (Dong & Yang, 2019)) to constrain the memory consumption, which however can lead to a biased search and thus poor performance. Specifically, activating only a subset of the precision choices implies a sequential training of different precisions that can lead to inaccurate performance ranking. This is because a sequential training means different precision choices are applied on top of the same weights and activations. As a result, different precision choices can interfere with each other, and different training order would lead to a different result. For a better understanding, we next show two concrete experiments.

Co-search network and precision using hard Gumbel Softmax: Fig. 2 (b) shows the resulting precision probability evolution when co-searching for the network and precision on CIFAR-100 using hard Gumbel Softmax, which activates two precision choices, without imposing any hardware-cost constraints, thus the desired and effective precision choice would be the highest precision. However, as shown in Fig. 2 (b), the block co-searched using hard Gumbel Softmax collapses to the lowest precision (i.e., the highest probability towards the end of the search is the lowest precision choice 4-bit), indicating an ineffective search direction. Note that the fluctuation in the probability of different precision choices is caused by the intermittent activation of the block due to the hard Gumbel Softmax sampling.

Sequential training of a fixed network with multiple precision choices: As observed in (Jin et al., 2020), when training a fixed network with multiple precision choices, either ascending or descending the precision will incur an inferior convergence and thus chaotic accuracy ranking among different precision choices. For example, as shown in Tab. 1, we compare the accuracy of a fixed network (all blocks adopt the k3e1 (kernel size 3 and channel expansion 1) structure in (Wu et al., 2019)) under different precision choices, when being trained with different precision schedule strategies. We can see that only jointly training all the precision choices can maintain the ranking consistent with that of independently trained ones, while sequential training leads to both inferior accuracy and ranking.

Proposed solution - Heterogeneous sampling. To tackle both aspects of the aforementioned dilemma, we propose a heterogeneous sampling strategy as formulated below:

$$A^{l+1} = \bar{W}^l * \sigma(\bar{A}^l) = \sum_{j=1}^{J} \bar{\beta}_j^l W_j^l * \sigma(\sum_{j=1}^{J} \bar{\beta}_j^l A_j^l)$$

$$where\ \bar{\beta}_j^l = \begin{cases} GS(\beta_j^l) & \text{if updating weights} \\ GS_{hard}(\beta_j^l) & \text{if updating } \beta \end{cases} \tag{5}$$

where $\bar{W}^l$ / $\bar{A}^l$ are the composite weights / activations of the l-th layer as in (Cai & Vasconcelos, 2020) which are the weighted sum of weights / activations under different precision choices, e.g., W_j^l is the weights quantized to the j-th precision among the total J options for the l-th layer, and σ is the activation function.

Our heterogeneous sampling updates the weights in Eq. (3) by jointly updating the weights under all the precision choices weighted by their corresponding soft Gumbel Softmax $GS(\beta_j^l)$, where β_j^l parameterizes the probability of the j-th precision in the l-th layer, and the gradients is estimated by the straight through estimator (STE) (Zhou et al., 2016) as $\partial L_{train}/\partial A^l \approx \partial L_{train}/\partial \bar{A}^l$ so that no extra intermediate feature maps need to be stored during backward. For updating β, we adopt hard Gumbel Softmax (Jang et al., 2016) with one-hot outputs $GS_{hard}(\beta_j^l)$ to save memory computation while reducing the correlation among precision choices. Under the same co-search setting as Fig. 2 (b), all the blocks searched using our heterogeneous sampling converge to the highest precision towards the end of the search (see Fig. 2 (c)), indicating an effective search as further validated in Sec. 4.

3.3. Auto-NBA Enabler 2: Differentiable Accelerator Search Engine

Motivation. Although EDD (Li et al., 2020b) also co-searches the accelerator with the network, their search space is limited to include merely one accelerator parameter (i.e., the parallel factor within their template) which can be fused into their computational cost, whereas this is not always applicable to other naturally non-differentiable accelerator design parameters such as memory hierarchies and tiling strategies. Hence, a more general and efficient search engine is needed towards generic differentiable accelerator search.

Search algorithm. We propose a differentiable search engine to efficiently search for the optimal accelerator (including the micro-architectures and mapping methods) given a DNN and its precision using single-path sampling as discussed in Sec. 3.4. Specifically, we solve Eq. (4) as follows:

$$\arg\min_{\gamma} \sum_{m=1}^{M} GS_{hard}(\gamma^m)\, L_{cost}(hw(\{GS_{hard}(\gamma^m)\}), net(\{O_{fw}^l\}), prec(\{B_{fw}^l\})) \tag{6}$$

where M is the total number of accelerator design parameters. Given the network $net(\{O_{fw}^l\})$ and precision $prec(\{B_{fw}^l\})$, where $O_{fw}^l \in \alpha$ and $B_{fw}^l \in \beta$ are the activated forward operator and precision for each layer as discussed in Sec. 3.4. Our search engine utilizes hard Gumbel Softmax GS_{hard} sampling on each design parameter γ^m to build an accelerator $hw(\{GS_{hard}(\gamma^m)\})$ and penalize each sampled accelerator parameter with the overall hardware-cost L_{cost} through relaxation in a gradient manner.

Hardware template. We adopt a unified template for both the FPGA and ASIC accelerators, which is a parameterized chunk-based pipeline micro-architecture inspired by (Shen et al., 2017). As elaborated in Sec. 4.1, the hardware/micro-architecture template comprises multiple sub-accelerators (i.e., chunks) and executes DNNs in a pipeline fashion. Each chunk is assigned with multiple but not necessarily consecutive layers which are executed sequentially within the chunk. Similar to Eyeriss, each chunk consists of several levels of memories (e.g., on-chip buffer and local register files) and processing elements (PEs) to facilitate data reuses and parallelism with searchable design knobs, such as PE interconnections (i.e., Network-on-chip), allocated buffer sizes, multiply-and-accumulate (MAC) operations' scheduling and tiling (i.e., dataflows), and so on.

General applicability. As shown in Eq. (6), our accelerator search engine is general and does not hold any prior assumptions about the adopted accelerators. Hence, it is applicable to different accelerator architectures and mapping methods. Specifically, for a given target accelerator architecture or template, such as TPU-like (Jouppi et al., 2017) or other accelerators (Chen et al., 2016; Li et al., 2020a; Zhao et al., 2020), our search engine can be directly applied once given (1) a simulator to estimate the hardware cost, and (2) a set of user-defined searchable accelerator design knobs abstracted from the target accelerator template.

3.4. Auto-NBA: The Overall Joint-Search Framework

Objective and challenges. The key objective of Auto-NBA is formulated in Eq. (1) involving all the three major aspects towards efficient DNN accelerators. The key challenges for joint-search of the three include **(1)** the prohibitively large joint space (e.g., **2.3E+21** in this work) which, if not addressed, will limit the search scalability to practical yet complex tasks; **(2)** the entangled co-adaptation (Hong et al., 2020), correlation (Li et al., 2019), and cooperation (Tian et al., 2020) issues among different network and precision choices can enlarge the gap between the supernet during search and the final derived network, thus failing the joint search; and **(3)** the chicken and egg problem associated with network-accelerator co-search, i.e., co-search requires operation-wise hardware cost, which is lacking during search as the optimal accelerator depending on the whole network is still unknown during search.

Auto-NBA implementation. Auto-NBA integrates the two enablers in Sec. 3.2 and Sec. 3.3 to develop a unified joint-search pipeline. Specifically, Auto-NBA search starts from updating both the supernet weights ω and accelerator parameters γ (based on Enablers 1-2 in Sec. 3.2 and Sec. 3.3, respectively), given the current network $net(\alpha)$ quantized using precision $prec(\beta)$, and then updates α and β based on the derived optimal weights ω^* and accelerator $hw(\gamma^*)$ resulting from the previous step.

During joint-search, Auto-NBA updates α and β as follows (see Eq. (7)-Eq. (9)) to solve Eq. (1), where only the update for α is shown for simplicity as it is similarly applicable to update β. Note that here we define *path* to be one of the parallelled candidate operators between the layer input and layer output within one searchable layer, which can be viewed as a coarse-grained (layer-wise) version of the path

Table 2. Benchmark Auto-NBA's Search efficiency over SOTA co-search/exploration works and one-shot NAS methods.

Method	Dataset	Network Space	Accelerator Space	Precision Space	Joint Space	Search Time (GPU hours)
HS-Co-Opt (Jiang et al., 2020c)	CIFAR-10	1.15E+18	-	-	1.15E+18	103.9
Auto-NBA	CIFAR-10	9.85E+20	2.24E+27	2.40E+15	**5.30E+63**	**6**
BSW (Abdelfattah et al., 2020)	CIFAR-100	4.20E+05	8.64E+03	-	3.63E+09	5184
Auto-NBA	CIFAR-100	9.85E+20	2.24E+27	2.40E+15	**5.30E+63**	**12**
HS-Co-Opt (Jiang et al., 2020c)	ImageNet	2.22E+18	-	-	2.22E+18	266.8
Once-For-All (Cai et al., 2019)	ImageNet	2.00E+19	-	-	2.00E+19	1200
APQ (Wang et al., 2020)	ImageNet	1.00E+35	-	1.00E+10	1.00E+45	2400
Single One-shot (Guo et al., 2020)	ImageNet	7.00E+21	-	-	7.00E+21	288
Auto-NBA	ImageNet	9.85E+20	2.24E+27	2.40E+15	**5.30E+63**	**80**

definition in (Wang et al., 2018b; Qiu et al., 2019).

Single-path forward: For updating both α (see Eq. (7)) and β during forward, Auto-NBA adopts hard Gumbel Softmax sampling (Hu et al., 2020a), i.e., only the choice with the highest probability will be activated to narrow the gap between the search and evaluation, leveraging the single-path property of hard Gumbel Softmax sampling. In Eq. (7), A^l and A^{l+1} denote the feature maps of the l-th and $(l+1)$-th layer, respectively, N is the total number of operator choices, O_i^l is the i-th operator in the l-th layer parameterized by α_i^l, and O_{fw}^l is the activated operator during forward.

$$Forward: A^{l+1} = \sum_{i=1}^{N} GS_{hard}(\alpha_i^l)\, O_i(A^l) = O_{fw}^l(A^l) \tag{7}$$

$$\begin{aligned} Backward: \frac{\partial L_{val}}{\partial \alpha_i^l} &= \sum_{k=1}^{K} \frac{\partial L_{val}}{\partial GS(\alpha_k^l)} \frac{\partial GS(\alpha_k^l)}{\partial \alpha_i^l} \\ &= \frac{\partial L_{val}}{\partial A^{l+1}} \sum_{k=1}^{K} O_k^l(A^l) \frac{\partial GS(\alpha_k^l)}{\partial \alpha_i^l} \end{aligned} \tag{8}$$

$$\frac{\partial L_{cost}}{\partial \alpha_i^l} = 1(GS_{hard}(\alpha_i^l) = 1)\, L_{cost}^{\alpha_i^l}(hw(\gamma^*), net(\alpha_i^l), prec(\beta)) \tag{9}$$

Multi-path backward: For updating both α (see Eq. (8)) and β during backward, Auto-NBA activates multiple paths to calculate the gradients of α and β through Gumbel Softmax relaxation in order to balance the search efficiency and stability inspired by (Cai et al., 2018; Hu et al., 2020b), where α_i^l's gradients are calculated using Eq. (8), with $K \in (1, N)$ being the number of activated choices with the top K Gumbel Softmax probability and controlling the search cost.

Hardware-cost penalty: The network search in Eq. (1) is performed in a layer/block-wise manner as in (Liu et al., 2018a), thus requiring layer/block-wise hardware-cost penalty which is determined by both the layer/block-to-accelerator *mapping method* and the corresponding layer/block execution cost on the optimal accelerator $hw(\gamma^*)$. The optimal mapping method is yet determined by the whole network. To handle this gap, we derive the layer/block-wise hardware-cost assuming that the single-path network derived from the current forward would be the final derived network, as this single-path network has a higher if not the highest probability to be finally derived. In Eq. (9), $1(\cdot)$ is an indicator denoting whether α_i^l (i.e., the i-th operator in the l-th layer) is activated during forward.

4. Experiment Results

4.1. Experiment Setup

Software settings. Search space and hyper-params. We adopt the same search space as (Wu et al., 2019) for the ImageNet experiments, from which we disable the first two down sampling operations for the CIFAR-10/100 experiments. We use [4, 6, 8, 12, 16] as the candidate precision set, where the precisions of the first and last blocks are fixed to 8-bit, and each block shares the same precision for both the weights and activations for more hardware friendly implementation. We activate two paths during backward, i.e., $K = 2$ in Eq. (8), for search efficiency. For L_{cost} in Eq. (4), we use the acceleration latency, i.e., Frame-Per-Second (FPS), and Energy-Delay-Product (EDP) for FPGA- and ASIC-based accelerators, respectively.

Search settings. We adopt standard search settings used in SOTA hardware-aware NAS works (Wu et al., 2019). Specifically, for searching on the CIFAR-10/100 datasets, we use half of the dataset for updating supernet weights ω and the other half for updating the network and precision parameter α and β, and search for 90 epochs with an initial gumbel softmax temperature of 5 decayed by a factor of 0.975 every epoch; For searching on ImageNet, we randomly sample 100 classes as a proxy search dataset from which we use 80% for updating ω and the other 20% for updating α and β, pretrain the supernet for 30 epochs without updating the network architecture and precision, and then search for 90 epochs with an initial temperature of 5 decayed by a factor of 0.956 every epoch, following (Wu et al., 2019). For both CIFAR-10/100 and ImageNet, we use an initial learning rate of 0.1 and an annealing cosine learning rate.

Training settings. For CIFAR-10/100, we train the derived networks for 600 epochs using a batch size of 256 with an initial learning rate of 0.1 and an annealing cosine learning rate on a single NVIDIA RTX-2080Ti GPU following (Liu et al., 2018a). For ImageNet, we follow the training recipe in (Wu et al., 2019) on four NVIDIA Tesla V100 GPUs.

Baselines. We benchmark against four kinds of SOTA baselines: (1) the most relevant baseline EDD (Li et al., 2020b) which co-searches for networks, precisions, and one accelerator paremeters, (2) SOTA methods co-exploring networks and accelerators including HS-Co-Opt (Jiang et al., 2020c), NASAIC (Yang et al., 2020), BSW (Abdelfattah et al., 2020), and NHAS (Lin et al., 2020), (3) SOTA methods co-searching for the network and precision including APQ (Wang et al., 2020) and MP-NAS (Gong et al., 2019), and (4) hardware-aware NAS with uniform precision, including FBNet (Wu et al., 2019), ProxylessNAS (Cai et al., 2018), and Single-Path NAS (Stamoulis et al., 2019).

Hardware settings. Search space. Our accelerator search space is inspired by a SOTA accelerator architecture (Shen et al., 2017; Zhang et al., 2018b) and adopts a chunk-wise pipelined architecture, aiming to more efficiently accelerate more recent networks which have diverse network structures. Specifically, our accelerator search space is a parameterized chunk-wise pipelined architecture (Shen et al., 2017; Zhang et al., 2018b), in which the following parameters are searchable: **(1)** the parallel PE array, i.e., the number and the inter-connections of the PEs, **(2)** the on-chip buffers, i.e., allocated lower-level memories for the inputs, weights, and outputs, **(3)** the tiling and scheduling for the MAC computations, and **(4)** the network layer allocation, i.e., how to assign each layer to be processed by different chunks within the chunk-wise pipelined architecture, with all being critical accelerator parameters as pointed out by SOTA accelerator works (Chen et al., 2017; Zhang et al., 2015; Yang et al., 2016). To facilitate automated search, all the choices for the aforementioned accelerator parameters are formatted and maintained using vectors so that they can be compatible with the optimization formulation in Sec. 3.3. Note that users of our proposed Auto-NBA can easily plug in their preferred accelerator search space as discussed in Sec. 3.3.

Platforms. To evaluate the generated network and accelerator designs, for FPGA accelerators, we adopt the standard Vivado HLS (Xilinx Inc., a) design flow on the target Xilinx ZC706 development board (Xilinx Inc., b), which has a total 900 DSPs (Digital Signal Processors) and 19.1Mb BRAM (Block RAM); for ASIC accelerators, we use the SOTA energy estimation tool Timeloop (Parashar et al., 2019) and Accelergy, (Wu et al., 2019), to validate our generated design's performance, with CACTI7 (Balasubramonian et al., 2017) and Aladdin (Shao et al., 2014) at a 32nm CMOS technology as unit energy and timing cost plugins.

4.2. Auto-NBA vs. SOTA in Search Efficiency

To evaluate the superiority of Auto-NBA in terms of search efficiency, we compare the search space size and search time of Auto-NBA with both RL-based co-search/exploration works and one-shot NAS methods using the reported data from the baselines' original papers as shown in Tab. 2. We can see that Auto-NBA consistently require a notably less search time while handling the largest joint search space on all the considered tasks. In particular, compared with the one-shot NAS methods (Guo et al., 2020; Cai et al., 2019) which can be potentially extended to implement co-search yet can suffer from a large pretraining cost, Auto-NBA achieves a 3.6× ~ 30× less search time on ImageNet, justifying our choice of differentiable co-search.

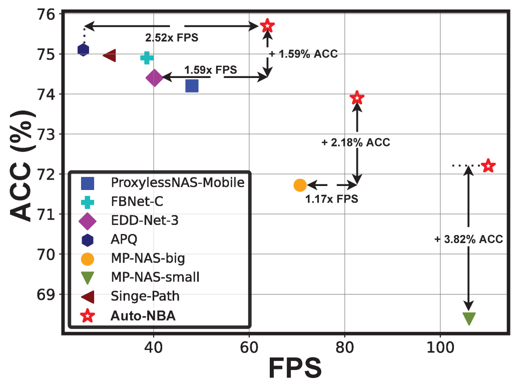

Figure 3. Accuracy vs. FPS trade-off of Auto-NBA against SOTA efficient DNN solutions on ImageNet.

4.3. Auto-NBA vs. SOTA in Searched Accelerators

Co-exploration of networks, precision, and accelerators. Here we benchmark Auto-NBA with SOTA automatically searched, expert designed, and co-searched/co-explored DNN algorithms/accelerators on ImageNet, considering FPGA-based accelerators as shown in Fig. 3 which include four Auto-NBA searched results for a fair comparison. We can observe that (1) the searched networks by our Auto-NBA consistently push forward the frontier of accuracy-FPS trade-offs, compared to all SOTA baselines, and (2) compared with the most relevant baseline EDD, we achieve a +1.3% higher accuracy together with a 1.59× better FPS. The consistently large improvement of Auto-NBA over SOTA methods in co-design/co-exploration validate the necessity and effectiveness of Auto-NBA joint-search for all the three aspects towards efficient DNN accelerators.

Note that we use EDD's reported results, and search for the optimal accelerator based on our accelerator space for APQ, MP-NAS, and SOTA hardware-aware NAS methods; the ProxylessNAS-8bit result is reported in APQ (Wang et al., 2020); and the other baselines are all quantized to 8-bit for hardware measurement and the accuracies are from the original papers without considering their accuracy degradation due to quantization effects. All methods consider a 450 DSP limit in FPGA for a fair comparison.

Co-exploration of networks and accelerators. Software-Hardware co-design is a significant property of our Auto-NBA framework, so we further benchmark Auto-NBA with both searched precision and fixed-precision over SOTA network/accelerator co-search/exploration works.

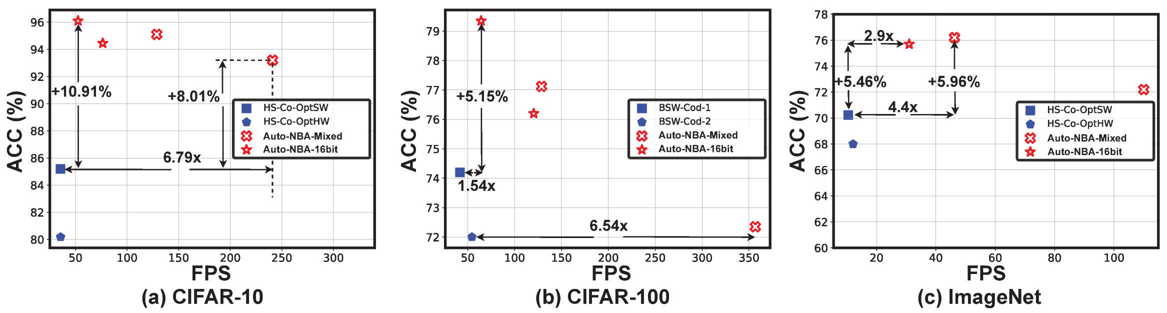

Figure 4. Benchmark Auto-NBA w/ and w/o precision search (denoted as Auto-NBA-Mixed and Auto-NBA-16bit, respectively) with SOTA network/accelerator co-exploration methods (Jiang et al., 2020c; Abdelfattah et al., 2020) on CIFAR-10/100/ImageNet.

Table 3. Comparing the accuracy and ASIC efficiency (i.e., EDP and area) of Auto-NBA and SOTA co-exploration ASIC works (Yang et al., 2020).

Optimization Methods	Accuracy (%)	EDP (J*clock-cycle)	Area (um^2)
NAS → ASIC	94.17	3.30E+06	4.83E+09
ASIC → HW-NAS	92.53	2.81E+06	3.86E+09
NASAIC	92.62	1.62E+06	3.34E+09
Auto-NBA	**94.34**	**4.36E+03**	**5.92E+05**

Co-search on FPGA. We benchmark with HS-Co-Opt (Jiang et al., 2020c) and BSW (Abdelfattah et al., 2020) on ZC706, under the same DSP limits as the baselines on CIFAR-10/100/ImageNet. Since all the baselines adopt a 16-bit fixed-point design, here we provide Auto-NBA with both fixed 16-bit and searched precision for a fair comparison. From Fig. 4, we can see that (1) on both CIFAR-10/100, Auto-NBA with fixed 16-bit consistently achieves a better accuracy (up to 10.91% and 5.15%, respectively) and a higher FPS (up to 2.21× and 2.15×, respectively) under the same DSP constraint, and (2) when co-searching for the precision, Auto-NBA can more aggressively push forward the FPS improvement (up to 6.79× and 6.54×, respectively on CIFAR-10/100), implying the importance of co-exploring the precision dimension in addition to network and accelerator co-explorations. Specifically, Auto-NBA with searched precision achieves a +5.96% higher accuracy and 4.4× FPS improvement on ImageNet over (Jiang et al., 2020c).

Co-search on ASIC. Here we evaluate Auto-NBA against three SOTA co-search methods for ASIC-based accelerators. In Tab. 3, we benchmark Auto-NBA with NASAIC (Yang et al., 2020) on CIFAR-10, which is the first exploration towards network/accelerator co-search targeting ASIC accelerators, considering both their reported co-search, sequential

Table 4. Benchmark Auto-NBA over NHAS (Lin et al., 2020) under the same precision setting.

Co-search Methods	**Accuracy (%)**	**Latency (ms)**	**Area (mm^2)**
NHAS (Lin et al., 2020)	70.74	1.58	5.87
Auto-NBA	**71.70**	**1.25**	**5.50**

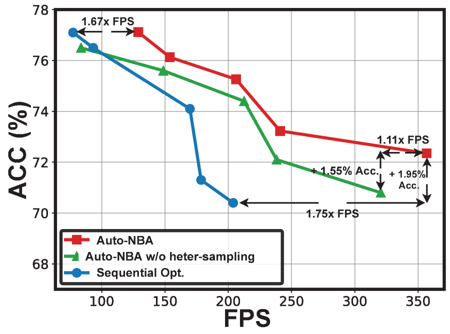

Figure 5. Accuracy vs. FPS trade-off of Auto-NBA, Auto-NBA w/o heterogeneous sampling, and the sequential optimization baseline on CIFAR-100, under an FPGA DSP limit of 512.

optimization, and hardware-aware optimization results. We can observe that compared with both co-search, sequential optimization, and hardware-aware optimization methods for exploring the ASIC-based accelerator design space, our Auto-NBA consistently achieves notably improved trade-offs between accuracy and EDP, which is equal to the acceleration energy cost multiplied with the acceleration latency (a commonly used metric for ASIC-based accelerators). In particular, Auto-NBA achieves a +0.17% ~ +1.81% higher accuracy together with a 371.56× ~ 756.88× reduction in EDP. In the baseline implementations (Yang et al., 2020), most of the area is occupied by the support for heterogeneous functionalities, which leads to a severely low utilization of the PE arrays when executing one task, thus leading to the surprisingly higher area and energy consumption.

We further benchmark Auto-NBA over another co-search baseline for ASIC-based accelerators, i.e., NHAS (Lin et al., 2020). In particular, we fix the precision of Auto-NBA to be 4-bit for a fair comparison. As shown in Tab. 4, Auto-NBA achieves a 0.96% higher accuracy and a 20.9% reduction in latency under a comparable area consumption compared with NHAS, verifying the superiority of our Auto-NBA.

4.4. Auto-NBA: Ablation Studies

Scalability under the same DSP. Fig. 5 shows the pareto frontier achieved by Auto-NBA under the same DSP constraint with different accuracy and FPS trade-offs on CIFAR-100, which indicates that Auto-NBA can handle and is scal-

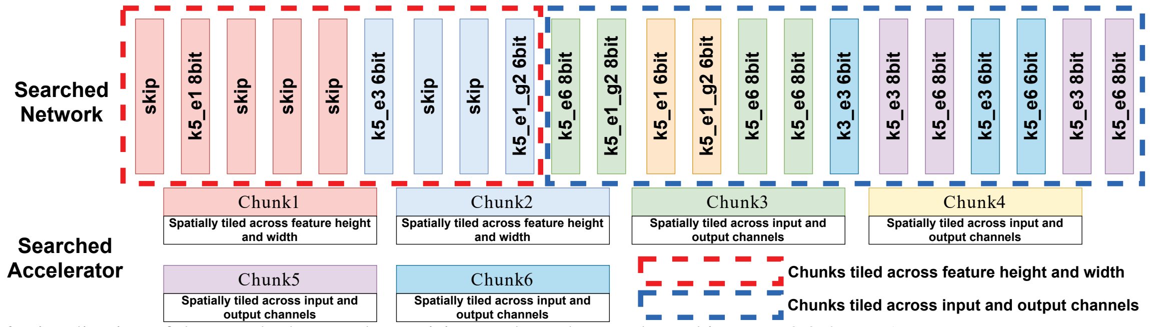
Searched Network
skip
k5_e1 8bit
skip
skip
skip
k5_e3 6bit
skip
skip
k5_e1_g2 6bit
k5_e6 8bit
k5_e1_g2 8bit
k5_e1 6bit
k5_e1_g2 6bit
k5_e6 8bit
k5_e6 8bit
k3_e3 6bit
k5_e3 8bit
k5_e6 8bit
k5_e3 6bit
k5_e6 6bit
k5_e3 8bit
k5_e6 8bit
Searched Accelerator
Chunk1
Spatially tiled across feature height and width
Chunk2
Spatially tiled across feature height and width
Chunk3
Spatially tiled across input and output channels
Chunk4
Spatially tiled across input and output channels
Chunk5
Spatially tiled across input and output channels
Chunk6
Spatially tiled across input and output channels
Chunks tiled across feature height and width
Chunks tiled across input and output channels

Figure 6. Visualization of the searched network, precision, and accelerator that achieves a 72.2% top -1 accuracy on ImageNet and an FPS of 110 on ZC706 FPGA, where the block definition follows (Wu et al., 2019).

able to a large range of required acceleration efficiency.

Effectiveness of heterogeneous sampling. In addition to the example and analysis in Sec. 3.2, we further validate the effectiveness of the proposed *heterogeneous* sampling strategy by benchmarking Auto-NBA w/ and w/o *homogeneous* sampling that adopts hard GS sampling ($K = 2$) for updating both the weights ω and precision choices β as that in Fig. 2 (b), the latter of which is termed as Auto-NBA w/o h-sampling. The achieved trade-offs between the task accuracy and acceleration FPS in Fig. 5 show that Auto-NBA w/o h-sampling tends to select lower precision choices which hurt the achieved accuracy, and is consistently inferior than Auto-NBA with heterogeneous sampling, due to its inaccurate estimation for different precision ranking.

Comparison with sequential optimization. Considering the great flexibility on both DNNs' structure and accelerator sides, a natural baseline of Auto-NBA is to search the network and precision based on theoretical efficiency metrics (e.g., total bit operations), and then search for the optimal accelerator given the searched network and precision from the first search. We benchmark Auto-NBA over the aforementioned sequential search in Fig. 5 on CIFAR-100, which shows that Auto-NBA consistently outperforms the sequential optimization baseline, e.g., a 1.95% higher accuracy with a 1.75× better FPS, indicating the poor correlation between theoretical efficiency and real hardware efficiency and thus motivating the necessity of joint-search.

4.5. Visualization of the searched network, precision, and accelerator

Fig. 6 visualizes Auto-NBA' searched network, precision, and accelerator, from which we discuss our extracted insights below.

Insights for the searched networks of Auto-NBA. The automatically searched network of Auto-NBA is shown in Fig. 6 and we can find that wide-shallow networks tend to better favor real-device efficiency on the ZC706 FPGA board while achieving a similar accuracy. We conjecture the reason is that wider networks offer more opportunities for making use of feature/channel-wise parallelism for a given batch size, thus leading to a higher resource utilization rate and thus an overall higher throughput.

Insights for the searched accelerators of Auto-NBA. As shown in Fig. 6, we can observe that the whole network is partitioned into multiple pipelined chunks to maximize the acceleration throughput, with each chunk being highlighted using a different color. As (Shen et al., 2017) points out, such multi-chunk accelerator architectures can boost the overall utilization of the PE arrays via 1) optimizing each accelerator chunk (i.e., sub-accelerator) for a cluster of layers which have similar patterns/workloads and 2) pipelining all the chunks to process different network inputs and process non-consecutive layers. Additionally, the chunks which are assigned with the early layers of the network prefer spatially tiling the feature map height and width as this offers more parallelism, while the chunks handling the deeper layers of the network tend to tile the channel dimension as the parallelism opportunity is more prominent along channel dimensions at the deeper layers.

An ablation study for Auto-NBA's accelerator search engine is provided in the Appendix.

5. Conclusion

In this work, we present Auto-NBA, which is the first to identify and tackle the prohibitive challenges of jointly search for the networks, bitwidths, and accelerators for maximizing the task accuracy and acceleration efficiency. When benchmarking with a comprehensive set of SOTA efficient DNN algorithms, accelerators, and co-explored/co-searched works, Auto-NBA consistently achieves large improvements, outperforming both SOTA automatically searched and expert-designed DNNs and accelerators. Auto-NBA promises to expedite the development of DNN accelerators which falls far behind DNN algorithm advances.

Acknowledgements

The work is supported by the National Science Foundation (NSF) CAREER Program (Award number: 2048183).

References

Abdelfattah, M. S., Dudziak, Ł., Chau, T., Lee, R., Kim, H., and Lane, N. D. Best of both worlds: Automl codesign of a cnn and its hardware accelerator. *arXiv preprint arXiv:2002.05022*, 2020.

Balasubramonian, R., Kahng, A. B., Muralimanohar, N., Shafiee, A., and Srinivas, V. Cacti 7: New tools for interconnect exploration in innovative off-chip memories. *ACM Trans. Archit. Code Optim.*, 14(2), June 2017. ISSN 1544-3566. doi: 10.1145/3085572. URL https://doi.org/10.1145/3085572.

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

Cai, H., Gan, C., Wang, T., Zhang, Z., and Han, S. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

Cai, Z. and Vasconcelos, N. Rethinking differentiable search for mixed-precision neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2349–2358, 2020.

Chen, D., Cong, J., Fan, Y., Han, G., Jiang, W., and Zhang, Z. xpilot: A platform-based behavioral synthesis system. *SRC TechCon*, 5, 2005.

Chen, D., Cong, J., Fan, Y., and Wan, L. Lopass: A low-power architectural synthesis system for FPGAs with interconnect estimation and optimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18 (4):564–577, 2009.

Chen, Y., Krishna, T., Emer, J., and Sze, V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *JSSC 2017*, 52(1):127–138, 2017.

Chen, Y., Meng, G., Zhang, Q., Zhang, X., Song, L., Xiang, S., and Pan, C. Joint neural architecture search and quantization. *arXiv preprint arXiv:1811.09426*, 2018.

Chen, Y.-H., Emer, J., and Sze, V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. *ACM SIGARCH Computer Architecture News*, 44(3):367–379, 2016.

Dong, X. and Yang, Y. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pp. 1761–1770, 2019.

Du, Z., Fasthuber, R., Chen, T., Ienne, P., Li, L., Luo, T., Feng, X., Chen, Y., and Temam, O. Shidiannao: Shifting vision processing closer to the sensor. In *ACM SIGARCH Computer Architecture News*, volume 43, pp. 92–104. ACM, 2015.

Elthakeb, A. T., Pilligundla, P., Mireshghallah, F., Yazdanbakhsh, A., and Esmaeilzadeh, H. Releq: A reinforcement learning approach for automatic deep quantization of neural networks. *IEEE Micro*, 2020.

Fu, Y. et al. Autogan-Distiller: Searching to compress generative adversarial networks. In *ICML'20*.

Gao, M., Pu, J., Yang, X., Horowitz, M., and Kozyrakis, C. Tetris: Scalable and efficient neural network acceleration with 3d memory. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 751–764, 2017.

Gong, C., Jiang, Z., Wang, D., Lin, Y., Liu, Q., and Pan, D. Z. Mixed precision neural architecture search for energy efficient deep learning. In *ICCAD*, pp. 1–7, 2019.

Guan, Y., Liang, H., Xu, N., Wang, W., Shi, S., Chen, X., Sun, G., Zhang, W., and Cong, J. FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates. In *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 152–159. IEEE, 2017.

Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., and Sun, J. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pp. 544–560. Springer, 2020.

Hong, W., Li, G., Zhang, W., Tang, R., Wang, Y., Li, Z., and Yu, Y. Dropnas: Grouped operation dropout for differentiable architecture search. In *International Joint Conference on Artificial Intelligence*, 2020.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1314–1324, 2019.

Hu, S., Xie, S., Zheng, H., Liu, C., Shi, J., Liu, X., and Lin, D. Dsnas: Direct neural architecture search without parameter retraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12084–12092, 2020a.

Hu, Y., Wu, X., and He, R. Tf-nas: Rethinking three search freedoms of latency-constrained differentiable neural architecture search. *arXiv preprint arXiv:2008.05314*, 2020b.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jiang, W., Lou, Q., Yan, Z., Yang, L., Hu, J., Hu, X. S., and Shi, Y. Device-circuit-architecture co-exploration for computing-in-memory neural accelerators. *IEEE Transactions on Computers*, 2020a.

Jiang, W., Yang, L., Dasgupta, S., Hu, J., and Shi, Y. Standing on the shoulders of giants: Hardware and neural architecture co-search with hot start. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):4154–4165, 2020b.

Jiang, W., Yang, L., Sha, E. H.-M., Zhuge, Q., Gu, S., Dasgupta, S., Shi, Y., and Hu, J. Hardware/software co-exploration of neural architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020c.

Jin, Q., Yang, L., and Liao, Z. Adabits: Neural network quantization with adaptive bit-widths. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2146–2156, 2020.

Jouppi, N. P. et al. In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pp. 1–12. IEEE, 2017.

Li, G., Zhang, X., Wang, Z., Li, Z., and Zhang, T. Stacnas: Towards stable and consistent differentiable neural architecture search. *arXiv*, pp. arXiv–1909, 2019.

Li, W., Xu, P., Zhao, Y., Li, H., Xie, Y., and Lin, Y. Timely: Pushing data movements and interfaces in pim accelerators towards local and in time domain. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pp. 832–845, 2020a. doi: 10.1109/ISCA45697.2020.00073.

Li, Y., Hao, C., Zhang, X., Liu, X., Chen, Y., Xiong, J., Hwu, W.-m., and Chen, D. Edd: Efficient differentiable dnn architecture and implementation co-search for embedded ai solutions. *arXiv preprint arXiv:2005.02563*, 2020b.

Lin, Y., Hafdi, D., Wang, K., Liu, Z., and Han, S. Neural-hardware architecture search. 2020.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018a.

Liu, S., Lin, Y., Zhou, Z., Nan, K., Liu, H., and Du, J. On-demand deep model compression for mobile devices: A usage-driven model selection framework. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '18, pp. 389–400, New York, NY, USA, 2018b. Association for Computing Machinery. ISBN 9781450357203. doi: 10.1145/3210240.3210337. URL https://doi.org/10.1145/3210240.3210337.

Parashar, A., Raina, P., Shao, Y. S., Chen, Y., Ying, V. A., Mukkara, A., Venkatesan, R., Khailany, B., Keckler, S. W., and Emer, J. Timeloop: A systematic approach to dnn accelerator evaluation. In *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 304–315, 2019.

Qiu, Y., Leng, J., Guo, C., Chen, Q., Li, C., Guo, M., and Zhu, Y. Adversarial defense through network profiling based path extraction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4777–4786, 2019.

Rupnow, K., Liang, Y., Li, Y., Min, D., Do, M., and Chen, D. High level synthesis of stereo matching: Productivity, performance, and software constraints. In *2011 International Conference on Field-Programmable Technology*, pp. 1–8. IEEE, 2011.

Shao, Y. S., Reagen, B., Wei, G., and Brooks, D. Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pp. 97–108, 2014.

Shen, Y., Ferdman, M., and Milder, P. Maximizing cnn accelerator efficiency through resource partitioning. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, pp. 535–547, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450348928. doi: 10.1145/3079856.3080221. URL https://doi.org/10.1145/3079856.3080221.

Stamoulis, D., Ding, R., Wang, D., Lymberopoulos, D., Priyantha, B., Liu, J., and Marculescu, D. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 481–497. Springer, 2019.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

Tian, Y., Liu, C., Xie, L., Jiao, J., and Ye, Q. Discretization-aware architecture search. *arXiv preprint arXiv:2007.03154*, 2020.

Venkatesan, R., Shao, Y. S., Wang, M., Clemons, J., Dai, S., Fojtik, M., Keller, B., Klinefelter, A., Pinckney, N., Raina, P., et al. MAGNet: A Modular Accelerator Generator for Neural Networks. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, 2019.

Wan, A., Dai, X., Zhang, P., He, Z., Tian, Y., Xie, S., Wu, B., Yu, M., Xu, T., Chen, K., et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12965–12974, 2020.

Wang, J., Lou, Q., Zhang, X., Zhu, C., Lin, Y., and Chen, D. Design flow of accelerating hybrid extremely low bit-width neural network in embedded FPGA. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, 2018a.

Wang, K., Liu, Z., Lin, Y., Lin, J., and Han, S. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8612–8620, 2019.

Wang, T., Wang, K., Cai, H., Lin, J., Liu, Z., Wang, H., Lin, Y., and Han, S. Apq: Joint search for network architecture, pruning and quantization policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2078–2087, 2020.

Wang, Y., Xu, J., Han, Y., Li, H., and Li, X. Deepburning: Automatic generation of fpga-based learning accelerators for the neural network family. DAC '16, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342360. doi: 10.1145/2897937.2898003. URL https://doi.org/10.1145/2897937.2898003.

Wang, Y., Su, H., Zhang, B., and Hu, X. Interpret neural networks by identifying critical data routing paths. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8906–8914, 2018b.

Wu, B., Wang, Y., Zhang, P., Tian, Y., Vajda, P., and Keutzer, K. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018a.

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10734–10742, 2019.

Wu, J., Wang, Y., Wu, Z., Wang, Z., Veeraraghavan, A., and Lin, Y. Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5363–5372. PMLR, 10–15 Jul 2018b. URL http://proceedings.mlr.press/v80/wu18h.html.

Wu, Y. N., Emer, J. S., and Sze, V. Accelergy: An architecture-level energy estimation methodology for accelerator designs. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2019.

Xilinx Inc. Vivado High-Level Synthesis, a. https://https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html, accessed 2019-09-16.

Xilinx Inc. Xilinx zynq-7000 soc zc706 evaluation kit. https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html, b. (Accessed on 09/30/2020).

Xu, P., Zhang, X., Hao, C., Zhao, Y., Zhang, Y., Wang, Y., Li, C., Guan, Z., Chen, D., and Lin, Y. AutoDNNchip: An automated dnn chip predictor and builder for both FPGAs and ASICs. *The 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Feb 2020. doi: 10.1145/3373087.3375306. URL http://dx.doi.org/10.1145/3373087.3375306.

Yang, L., Yan, Z., Li, M., Kwon, H., Lai, L., Krishna, T., Chandra, V., Jiang, W., and Shi, Y. Co-exploration of neural architectures and heterogeneous asic accelerator designs targeting multiple tasks. *arXiv preprint arXiv:2002.04116*, 2020.

Yang, X., Pu, J., Rister, B. B., Bhagdikar, N., Richardson, S., Kvatinsky, S., Ragan-Kelley, J., Pedram, A., and Horowitz, M. A systematic approach to blocking convolutional neural networks, 2016.

You, H., Chen, X., Zhang, Y., Li, C., Li, S., Liu, Z., Wang, Z., and Lin, Y. Shiftaddnet: A hardware-inspired deep network. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 2771–2783. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/

1cf44d7975e6c86cffa70cae95b5fbb2-Paper.pdf.

Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P.-J., Tan, M., Huang, T., Song, X., Pang, R., and Le, Q. Bignas: Scaling up neural architecture search with big single-stage models. *arXiv preprint arXiv:2003.11142*, 2020.

Zhang, C., Li, P., Sun, G., Guan, Y., Xiao, B., and Cong, J. Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '15, pp. 161–170, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450333153. doi: 10.1145/2684746.2689060. URL https://doi.org/10.1145/2684746.2689060.

Zhang, C., Sun, G., Fang, Z., Zhou, P., Pan, P., and Cong, J. Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018a.

Zhang, X., Wang, J., Zhu, C., Lin, Y., Xiong, J., Hwu, W.-m., and Chen, D. Dnnbuilder: An automated tool for building high-performance dnn hardware accelerators for fpgas. In *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '18, New York, NY, USA, 2018b. Association for Computing Machinery. ISBN 9781450359504. doi: 10.1145/3240765.3240801. URL https://doi.org/10.1145/3240765.3240801.

Zhao, Y., Chen, X., Wang, Y., Li, C., You, H., Fu, Y., Xie, Y., Wang, Z., and Lin, Y. SmartExchange: Trading higher-cost memory storage/access for lower-cost computation. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pp. 954–967, 2020. doi: 10.1109/ISCA45697.2020.00082.

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