2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED) | 978-1-6654-3922-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISLPED52811.2021.9502478

DIAN: Differentiable Accelerator-Network Co-Search Towards Maximal DNN Efficiency

Yongan Zhang[1], Yonggan Fu[1], Weiwen Jiang[2], Chaojian Li[1], Haoran You[1], Meng Li[3], Vikas Chandra[3] and Yingyan Lin[1]
[1]Rice University, Houston, TX; [2]George Mason University, Fairfax, VA; [3]Facebook, Inc., Menlo Park, CA
Corresponding to: [1]yingyan.lin@rice.edu

***Abstract*—We present DIAN, a Differentiable Accelerator-Network Co-Search framework for automatically searching for matched networks and accelerators to maximize both the accuracy and efficiency. Specifically, DIAN integrates two enablers: (1) *a generic design space for DNN accelerators* that is applicable to both FPGA- and ASIC-based DNN accelerators; and (2) *a joint DNN network and accelerator co-search algorithm* that enables the simultaneous search for optimal DNN structures and their accelerators. Experiments and ablation studies based on FPGA measurements and ASIC synthesis show that the matched networks and accelerators generated by DIAN consistently outperform state-of-the-art (SOTA) DNNs and DNN accelerators (e.g., $3.04\times$ better FPS with a 5.46% higher accuracy on ImageNet), while requiring notably reduced search time (up to $1234.3\times$) over SOTA co-exploration methods, when evaluated over ten SOTA baselines on three datasets.**

***Index Terms*—Network Accelerator Co-design, DNN Accelerator, Neural Architecture Search**

I. Introduction

Powerful deep neural networks' (DNNs') prohibitive complexity has motivated intensive studies of efficient DNN solutions. Early works merely explore from either the algorithm or hardware level. For example, compression techniques attempt to trim down DNNs' complexity, while hardware accelerators [1], [2] develop customized *micro-architectures* (e.g., # of memory hierarchies and processing elements (PEs), PE array dimension and shape, size of different memories, and network-on-chip (NoC) design) and algorithm-to-hardware *mapping methods* (e.g., loop tiling strategy, loop size, and loop order) to boost DNN acceleration efficiency. Later, hardware-aware neural architecture search (HA-NAS) [3], [4] emerged to automate the design of efficient *network structures* (e.g., # of layers and channels, size of kernels, and layer operations). Recently, it has been recognized that maximizing DNN accelerators' efficiency requires joint exploration of both the networks and their accelerators (the latter refers to the accelerators' micro-architectures and mapping methods hereafter) [5]–[7].

Despite their promise, the great potential of jointly optimizing DNNs and their accelerators has not been unleashed. The key challenges include (1) the *prohibitively large* joint space consisting of the *coupled yet different* network and accelerator spaces with extremely sparse optima, (2) *non-differentiable* hardware costs, and (3) how to *algorithmically describe* a generic accelerator search space. To tackle the aforementioned challenges, this work makes the following contributions:

- We propose **DIAN**, a **DI**fferentiable **A**ccelerator-**N**etwork co-search framework (see Fig. 1) that jointly searches for DNNs' networks and their accelerators to boost the efficiency and expedite the development speed.
- We develop Differentiable Accelerator Search (DAS), a generic accelerator search engine for exploring the large and discrete design space of DNN accelerators. DAS distinguishes itself from existing methods which mostly adopt Reinforcement Learning (RL)-based methods and thus are limited in scalability.
- We construct a Generic DNN Accelerator Design Space (GADS) that is applicable to both FPGA- and ASIC-based accelerators and can enable algorithmically exploration of DNN accelerators' large and discrete design space, facilitating future development in DNN accelerator innovations.
- Through FPGA and ASIC synthesis, extensive experiments validate DIAN's effectiveness in outperforming SOTA DNNs/accelerators while requiring a notably reduced search time over SOTA methods.

II. Related Works

Hardware-Aware NAS (HA-NAS). HA-NAS has been developed to automate the design of efficient DNNs. Early works utilize RL-based methods, and thus suffer from substantially large search time [9]. Later, differentiable HA-NAS [4] emerged to greatly improve both the search and hardware efficiency. However, existing HA-NAS methods (1) mostly consider hardware costs (e.g., FLOPs) on one given device/accelerator and (2) have not yet fully explored the hardware space, motivating network-accelerator co-search techniques.

DNN accelerators. DNNs' powerful performance and prohibitive complexity have motivated extensive research in customized DNN accelerators. Given a DNN and its acceleration specification, SOTA accelerators [1], [10] explore different micro-architectures and algorithm-to-hardware mappings to maximize the acceleration efficiency. To reduce the design effort, recently, there has been a growing interest in design flow tools like [11] and DNN accelerator design automation such as [12], [13]. However, these works only explore the accelerator design space, leading to sub-optimal solutions.

Software/Hardware co-exploration. Network-accelerator co-exploration has been shown to be promising in boosting DNN acceleration efficiency: [5], [14] conduct RL-based search to jointly search for the networks and their accelerator parameters; [15] adopts RL-based controllers to search for the networks and select accelerators from two pre-configured templates; and [6], [16] extend differentiable NAS to network and accelerator co-search. Despite their promise, they have not

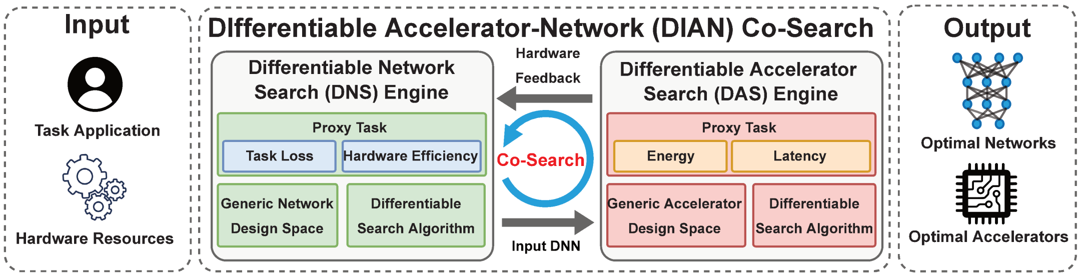

Input
Task Application
Hardware Resources
DIfferentiable Accelerator-Network (DIAN) Co-Search
Differentiable Network Search (DNS) Engine
Proxy Task
Task Loss
Hardware Efficiency
Generic Network Design Space
Differentiable Search Algorithm
Hardware Feedback
Co-Search
Input DNN
Differentiable Accelerator Search (DAS) Engine
Proxy Task
Energy
Latency
Generic Accelerator Design Space
Differentiable Search Algorithm
Output
Optimal Networks
Optimal Accelerators

Fig. 1: Overview of our DIAN co-search framework, which accepts target datasets and accelerator specifications and then automatically generates matched DNN and accelerator pairs to maximize both the accuracy and efficiency.

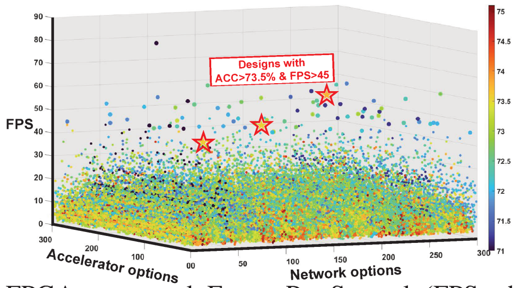


Fig. 2: FPGA measured Frame-Per-Second (FPS; the left axis) on a ZC706 FPGA [8] and CIFAR-100 based accuracy (the right colorbar) of 300 randomly sampled networks from the FBNet [4] search space, when each of the networks is accelerated by 300 randomly sampled accelerators from a generic accelerator space. Designs with $ACC > 73.5\%$ and $FPS > 45$ are marked as stars and extremely sparse.

yet tackled the challenges associated with generic algorithm-accelerator co-exploration. First, existing works have not yet considered a generic accelerator space, limiting their general applicability. Second, RL-based methods requiring large search costs suffer from poor scalability and achievable performance. For instance, [16] and [6] consider merely one or a few design factors in the accelerator design space to make the hardware cost differentiable, suffering from sub-optimal acceleration performance and thus calling for co-exploration innovations that adopt generic accelerator design space and more efficient co-search algorithms.

III. The Proposed DIAN Framework

This section describes our DIAN framework. We first provide an overview and the problem formulation, and then DIAN's co-search algorithm, followed by DIAN's two search engines and the proposed generic accelerator space.

A. DIAN: overview and formulation

As mentioned in Sec. I, the challenges of network-accelerator co-search include (1) the prohibitively large and irregular joint space versus very sparse optima (see Fig. 2), (2) non-differentiable hardware costs, and (3) the lack of a generic accelerator space description.

Fig. 1 shows an overview of our DIAN framework. To simultaneously tackle all the three challenges above, we propose an effective yet efficient joint co-search algorithm (see Sec. III-B) that leverages our developed DAS engine (see Sec. III-C) integrated with a generic accelerator design space (see Sec. III-D). Formally, DIAN's optimization can be formulated as:

$$\min_{\alpha} \; L_{val}(\omega^*, NET(\alpha)) + \lambda L_{hw}(NET(\alpha), HW(\gamma^*)) \quad (1)$$

$$s.t. \quad \omega^* = \arg\min_{\omega} \; L_{train}(\omega, NET(\alpha)), \quad (2)$$

$$s.t. \quad \gamma^* = \arg\min_{\gamma} \; L_{hw}(NET(\alpha), HW(\gamma)) \quad (3)$$

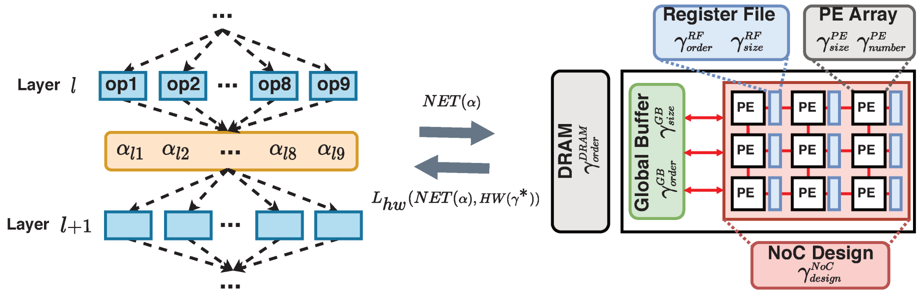
where L_{train} and L_{val} are the task loss of training and validation, respectively; ω, α, and γ are the supernet weights, DNNs' structure parameters [17], and the accelerator parameters, respectively; $NET(\alpha)$ and $HW(\gamma)$ denote the network and accelerator spaces parameterized by α and γ, respectively; L_{hw} is the hardware-cost loss determined by both the network and its accelerator; and λ is adjusted to balance the weighting of task and hardware-cost loss. Despite its simplicity in formulation, it is in general intractable to analytically solve Eq. 1~Eq. 3. DIAN integrates its Differentiable Network Search (DNS) and DAS engines to search for optimally matched networks and accelerators at much improved search efficiency, as in Fig. 3.

B. DIAN: the joint search algorithm

Here we describe DIAN's co-search algorithm. There are two practical issues to design differentiable co-search algorithms. First, ideally the hardware-cost penalty for each layer-wise network operator should be obtained when being executed on the final searched optimal accelerator, which is not yet available at each co-search epoch as the optimal network is still unknown, i.e., the chicken-and-egg problem. Second, the network is searched by regularizing α in a layer-wise manner (see Eq. 6), while the accelerator γ is determined by the whole DNN (see Eq. 7) due to the global accelerator parameters (e.g., the loop-order in Tab. I) shared among all the layers.

To address the above issues, DIAN obtains the hardware-cost loss L_{hw} at each co-search epoch by approximating the final optimal accelerator using the accelerator optimized for the network with the most probable operators. The hypothesis is that the network operators that have higher probabilities are also more likely to appear in the final optimal network, and thus, the corresponding optimal accelerators are likely to be the final optimal one. Specifically, at each epoch, DIAN samples M networks from the current network distribution $NET(\alpha)$ and obtains the optimal accelerator for each of them using its DAS engine (see Sec. III-C); the hardware-cost loss of each operator is then obtained using its average hardware cost on the M optimal accelerators generated from the previous step. For example, the hardware cost of the k-th operator in the l-th layer O_{lk} is formulated as:

$$E_{NET \sim P(NET|\alpha)}(L_{hw}(O_{lk}, HW(\gamma^*))) \approx \frac{1}{M}\sum_{m=1}^{M} L_{hw}(O_{lk}, HW_m^*) \quad (4)$$



Layer l
op1
op2
op8
op9
α_{l1} α_{l2} α_{l8} α_{l9}
Layer $l+1$
$NET(\alpha)$
$L_{hw}(NET(\alpha), HW(\gamma^*))$
DRAM γ^{DRAM}_{order}
Global Buffer γ^{GB}_{size} γ^{GB}_{order}
Register File γ^{RF}_{order} γ^{RF}_{size}
PE Array γ^{PE}_{size} γ^{PE}_{number}
PE
NoC Design γ^{NoC}_{design}

Fig. 3: DIAN's differentiable co-search of the network-accelerator joint space, where our DAS engine (right) optimizes the accelerator parameters based on Eqs. 3-7 based on the input networks $NET(\alpha)$ and then returns the corresponding hardware cost to the DNS engine (left) for which to penalize the costly operators. Here γ^{rf}_{order} denotes the accelerator parameter of *loop-order* in the register file (RF), and similar notations are adopted for other accelerator parameters in Tab. I.

Algorithm 1: DIAN's differentiable network-accelerator co-search algorithm.

```
Input: supernet weight $\omega$, the network space $NET(\alpha)$, the accelerator space $HW(\gamma)$, the total search epoch $max\_epoch$
Output: the optimal network $optimal\_net$ and the optimal accelerator $optimal\_accelerator$
while $epoch$ < $max\_epoch$ do
    Obtain optimized accelerators $HW^*_m(m = 1, ..., M)$ using Eq. 7 for $M$ DNNs sampled from the current network
      distribution $NET(\alpha)$ and calculate the average hardware cost for each operator based on Eq. 4
    for one training epoch do
        update $\omega$ based on Eq. 2
        update $\alpha$ based on Eq. 1 where $L_{hw}$ is calculated using Eq. 5
    end
end
Derive $optimal\_net$ by selecting the operator with the maximal weighted coefficients for each layer based on Eq. 6
Derive $optimal\_accelerator$ for the optimal $optimal\_net$ based on Eq. 7
return {$optimal\_net$, $optimal\_accelerator$}
```

where HW^*_m is the optimal accelerator generated using DIAN's DAS engine (see Eq. 7 in Sec. III-C) for the m-th sampled network, and $L_{hw}(O_{lk}, HW^*_m)$ is the hardware-cost loss of accelerating the operator O_{lk} using the accelerator HW^*_m. Therefore, the hardware-cost loss of the whole DNN in Eq. 1 can be formulated in a layer-wise manner:

$$\begin{aligned} & L_{hw}(NET(\alpha), HW(\gamma^*)) \\ & = \sum_{l=1}^{L}\sum_{k=1}^{K}\alpha_{lk}[E_{NET\sim P(NET|\alpha)}(L_{hw}(O_{lk}, HW(\gamma^*)))] \\ & \approx \frac{1}{M}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{m=1}^{M}\alpha_{lk}L_{hw}(O_{lk}, HW^*_m) \end{aligned} \tag{5}$$

Essentially, the hardware-cost loss L_{hw} at each co-search epoch is approximated using the weighted sum of the approximated hardware cost for all the layer-wise operators. DIAN's co-search algorithm is summarized in Alg. 1, the effectiveness of which is consistently validated in Sec. IV's experiments.

Without loss of generality, DIAN's network search adopts both SOTA hardware-friendly search space in [4], which searches the kernel size, channel expansion ratio, and group number for each building block, and network search algorithm. This is to better illustrate the benefits of co-search on top of SOTA HA-NAS techniques. In particular, the network search computes the output of the l-th layer A_l as a weighted sum of all candidate operators:

$$A_l = \sum_{k=1}^{K}\alpha_{lk}O_{lk}(A_{l-1}) \tag{6}$$

where K denotes the total number of layer-wise candidate operators, O_{lk} denotes the k-th operator for the l-th layer, and α_{lk} denotes the weighted coefficient of O_{lk}.

C. DIAN: the DAS engine

Our DAS engine realizes a generic differentiable accelerator search engine built on top of our GADS (see Sec. III-D). Specifically, we reformulate Eq. 3 and propose a differentiable method to solve it:

$$\gamma^* = \min_{\gamma} \sum_{s=1}^{S} GS(\gamma^s)\, L_{hw}(NET(\alpha^*), HW(GS(\gamma^1), ..., GS(\gamma^S))) \tag{7}$$

where the accelerator HW is characterized by its parameters γ^s $(s = 1, ..., S)$, which is a normalized vector representing the s-th accelerator parameter with each element of γ^s defining the probability of the corresponding choice of its represented accelerator parameter, and $GS(\gamma^s)$ denotes Gumbel-Softmax sampling [18] of the s-th accelerator parameter γ^s, so that the hardware cost is differentiable w.r.t. γ^s.

Unlike NAS, different options of one accelerator parameter are NOT additive, i.e., cannot be formulated as a sum weighted by their probability. As such, for each accelerator parameter, we apply Gumbel-Softmax sampling [18] to sample one choice $GS(\gamma^s)$ of the s-th accelerator parameter. Once all the accelerator parameters are sampled, the corresponding accelerator's hardware-cost are obtained using SOTA accelerator

TABLE I: The constructed generic accelerator search space, where TBS denotes "to be searched".

Memory Hierarchy	Loop-order	Loop-size
DRAM	TBS	-
Global Buffer	TBS	TBS
PE array	-	TBS
Register File (RF)	TBS	TBS

NoC design	Max # of PEs	Pipeline/Multi-cycle
TBS	TBS	TBS

performance estimators, where in this work we refer to [13] for FPGA-based accelerators and [19]–[21] for ASIC-based accelerators. We then multiply the resulting hardware-cost loss with the sampled $GS(\gamma^s)$ and relax it to Gumbel-Softmax during backpropagation for estimating the gradients.

D. Generic Accelerator Design Space (GADS)

Similar to NAS, a generic accelerator search space is a prerequisite for algorithmic accelerator exploration. However, it is challenging to develop such a space for DNN accelerators due to their *large* and *discrete* design space. First, there are numerous choices for the algorithm-to-hardware mapping methods (i.e., how to *temporally* and *spatially* schedule all the DNN's operations to be executed in the target accelerators). Second, there are many ways to design the accelerators' micro-architectures, which are characterized by the number of memory hierarchies and PEs, the size of each memory hierarchy, the shape and size of the PE array, and the NoC design [1].

We construct a generic accelerator search space as shown in Tab. I by leveraging the commonly used nested *for-loop* accelerator description [21], [22] which naturally bridges the accelerator's micro-architectures and mapping methods with DNNs' network parameters. Next, we introduce each accelerator parameter in Tab. I:

loop-order: the orders of the loops within each memory hierarchy, each of which has to consider all the data dimensions;

loop-size: the size of each loop in the *for-loop* description;

NoC design: the parallel execution pattern of MACs (multiply–accumulate operations) when accelerating DNNs on an accelerator. In this work, we consider three NoC options:

- parallelizing the computation over the output partial sums, where the dimensions of output channels, output rows, and output columns are executed in parallel.
- parallelizing the computation over the kernels, where the dimensions of output channels, input channels, kernel rows, and kernel columns are executed in parallel.
- parallelizing the computation over both the kernel and output dimensions, where the dimensions of output channels, kernel rows, and output columns are executed in parallel.

max number of PEs: the maximal number of PEs in the design which are determined by the area constraint.

pipeline/multi-cycle: a binary choice between a chunk-based pipeline or multi-cycle micro-architecture shared by all layers.

Note that these accelerator parameters are platform agnostic and thus compatible with both FPGA- and ASIC-based DNN accelerators, for which only the implementation flows and cost models are different as elaborated in Sec. IV-A.

IV. Experiment results

In this section, we introduce the experiment setup, evaluate DIAN over both SOTA (1) HW/SW co-exploration works and (2) HA-NAS methods, and present ablation studies to evaluate DIAN's co-search algorithm and DAS engine.

A. Experiment setup

Evaluation baselines and datasets. For evaluating DIAN over SOTA co-exploration works, we consider (1) three co-exploration FPGA baselines: HS-Co-Opt [14], BSW [5], EDD [6] and (2) two co-exploration ASIC baselines: NASAIC [15] and DANCE [16]. For benchmarking over SOTA HA-NAS methods, we consider four baselines: EfficientNet-B0 [9], FBNet [4], FBNet-V2 [23], and ProxylessNAS [3]. For evaluating DIAN's DAS engine, we consider both expert-designed and tool-generated SOTA accelerators in [2], [24]. Our experiments consider three datasets: CIFAR-10, CIFAR-100, and ImageNet.

Search and evaluation on CIFAR-10/100. We adopt the same search space as [4], except the stride settings for each group to adapt to the resolution of the input images in CIFAR-10/100, and the same search and evaluate settings as [4].

Search and evaluation on ImageNet. Search space: we adopt the same search space as [4] which is a SOTA search space for generating efficient DNNs via NAS. Search and evaluation settings: we follow the same hyper-parameter settings as [4] for searching and evaluating on ImageNet, while additionally updating the hardware accelerator parameters at each epoch using an SGD optimizer with a momentum of 0.9 and a fixed learning rate of 1E-9.

Hardware cost and hyperparameters. We adopt Frame-Per-Second (FPS), latency, or Energy-Delay-Product (EDP) as the hardware-cost loss (i.e., L_{hw} in Eq. 1) to compare with various baselines and set M = 10 for Eq. 4 in all the experiments. The performance of our DIAN is not sensitive to M for M > 5, and we thus choose M = 10.

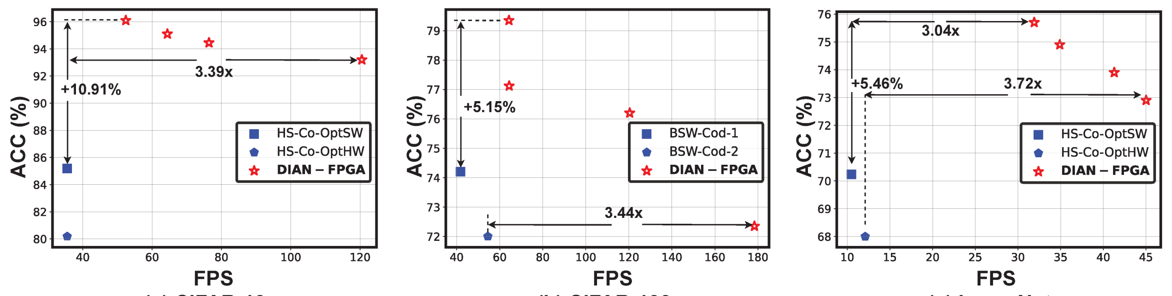
Accelerator evaluation methodology. To evaluate DIAN's generated accelerators, we adopt standard FPGA and ASIC evaluation flows. For FPGA-based accelerators, we employ the Vivado HLS flow [25] and use a SOTA accelerator performance predictor [13] to obtain fast and reliable estimation during search. For ASIC-based accelerators, we use SOTA accelerator performance estimators Timeloop [20] and Accelergy [19] during and after the search, with CACTI7 [26] and Aladdin [27] based on a commercial 32nm or 45nm CMOS technology for the unit energy/latency.

B. DIAN over SOTA co-exploration works

Search efficiency. Here we benchmark our DIAN over SOTA co-exploration works [5], [6], [14], [15] in terms of search space size and search time. As shown in Tab. II, DIAN can handle a remarkably larger (e.g., 2.21E+48 vs. 3.63E+09) joint search space while requiring the shortest search time (e.g., 4.2 vs. 5184 GPU hours), compared with all the SOTA baselines. Specifically, on ImageNet, DIAN achieves a 5.46% and 0.7% higher accuracy with a 3.04× and 1.94× higher FPS under a 450 and 900 Digital-Signal-Processor

TABLE II: DIAN vs. SOTA co-exploration works for generating both FPGA- and ASIC-based accelerators. Note that we quantize the DIAN generated networks to 8-bit when comparing with EDD [6] for a fair comparison.

Method target on FPGA	Dataset	Network Space	Accelerator Space	Joint Space	Search Time (GPU Hours)	DSP Limit	Accuracy (%)	FPS
HS-Co-Opt [14] DIAN (Proposed)	CIFAR-10	1.15E+18 9.85E+20	1 2.24E+27	1.15E+18 2.21E+48	103.9 **4.2** (↓24.7×)	450	85.19 **96.10** (↑10.91)	35.5 **52.4** (↑1.48×)
BSW [5] DIAN (Proposed)	CIFAR-100	4.20E+05 9.85E+20	8.64E+03 2.24E+27	3.63E+09 2.21E+48	5184 **4.2** (↓1234.3×)	512	72.00 **79.35** (↑7.35)	54.5 **64.3** (↑1.18×)
HS-Co-Opt [14] DIAN (Proposed)	ImageNet	2.22E+18 9.85E+20	1 2.24E+27	2.22E+18 2.21E+48	266.8 **144** (↓1.9×)	450	70.24 **75.70** (↑5.46)	10.5 **31.9** (↑3.04×)
EDD [6] DIAN 8-bit (Proposed)	ImageNet	3.65E+19 9.85E+20	- 2.24E+27	- 2.21E+48	- **144**	900	74.40 **75.10** (↑0.7)	40.2 **78.1** (↑1.94×)
Method target on ASIC	**Dataset**	**Network Space**	**Accelerator Space**	**Joint Space**	**Search Time (GPU Hours)**	**Area (mm^2)**	**Accuracy (%)**	**EDP (J * clock cycle)**
NASAIC [15] DIAN (Proposed)	CIFAR-10	1.70E+03 9.85E+20	9.84E+05 2.24E+27	1.67E+09 2.21E+48	4.6 **4.2** (↓1.1×)	3.34E+03 **5.92E-01**	92.62 **96.50** (↑3.88)	1.62E+06 **4.99E+03** (↓324.0×)



(a) CIFAR-10 (b) CIFAR-100 (c) ImageNet

Fig. 4: DIAN generated FPGA-based accelerators over those of SOTA co-exploration works HS-CO-Opt [14] and BSW [5], where we adopt the same DSP limits as the baselines, i.e., 450/512/450 on CIFAR-10/100/ImageNet, respectively.

(DSP) limit over HS-Co-Opt [14] and EDD [6], respectively; on CIFAR-100, DIAN achieves a 1234.3× speed-up in search time when handling a 6.1E+38× larger search space, resulting in a 7.35% better accuracy and 1.18× higher FPS. This set of experiments validate that DIAN's differentiable co-search can indeed handle a notably larger joint space with much improved search efficiency. Note that HS-Co-Opt [14] considers only one choice in its accelerator search space, because it uses an algorithm to automatically infer the optimal accelerator given the specific network structure. However, its inefficient RL search strategy still leads to much longer search time compared with DIAN.

Achieved FPS on FPGA. Fig. 4 compares DIAN's generated accelerators with SOTA co-exploration methods, HS-Co-Opt [14] and BSW [5] under the same FPGA resource budgets and datasets. We can see that DIAN generated accelerators consistently achieve the highest FPS with the same or even a higher accuracy: on CIFAR-10 with a 450 DSP limit, DIAN boosts the FPS by 1.48× ~ 3.39×, while offering a 8.01% ~ 10.91% higher accuracy over HS-Co-Opt; on CIFAR-100 with a 512 DSP limit, DIAN achieves a 1.18× ~ 3.44× higher FPS while boosting the accuracy by 0.35% ~ 5.15% over BSW; and on ImageNet with a 450 DSP limit, DIAN achieves a 3.04× higher FPS with a 5.46% higher accuracy over HS-Co-Opt.

Achieved EDP/latency on ASIC. For benchmarking on ASIC-based accelerators, we consider two SOTA co-search works, NASAIC [15] and DANCE [16], based on their reported results. Fig. 5 shows that DIAN generated accelerators achieve a 324.0× lower EDP and a 3.88% higher accuracy over NASAIC.

TABLE III: Evaluating DIAN generated ASIC-based accelerators over DANCE [16] under the same setting.

Optimization Methods	Accuracy (%)	Latency (ms)	Area (mm^2)	Precision (bit)
DANCE [16]	68.70	8.13	2.73	16
DIAN (Proposed)	**72.20** (↑3.50)	**2.85** (↓64.94%)	**2.12** (↓22.34%)	16

Second, like the cases in FPGA, DIAN generated ASIC-based accelerators can flexibly trade-off between the accuracy and efficiency (i.e., EDP here). Note that the surprisingly higher EDP/area for NASAIC, as noted in Tab. II, is caused by the much reduced hardware utilization due to their target of heterogeneous tasks. Tab. III shows that DIAN generated accelerators outperform DANCE in all aspects: a 3.50% higher accuracy on ImageNet with a 64.94% and 22.34% reduction in the required latency and area, respectively, under the same dataset (ImageNet), precision (16-bit), and accelerator metric (latency) as DANCE.

C. DIAN ablation: DIAN over SOTA HA-NAS works

Here we evaluate DIAN over SOTA HA-NAS works by comparing the acceleration efficiency of DIAN generated accelerators and SOTA HA-NAS generated networks when they are accelerated by their optimal accelerators, which are generated by the DAS engine under a DSP constraint of 900 (the maximum one in a ZC706 board [8]) to maximize the achieved FPS, as shown in Fig. 6. Note that we consider two kinds of precision, i.e., 16-bit and 8-bit, and adopt 8-bit for a fair comparison with EDD which reports performance under 8-bit and 16-bit when comparing with all other HA-NAS works to maintain their reported accuracy. We can see that compared

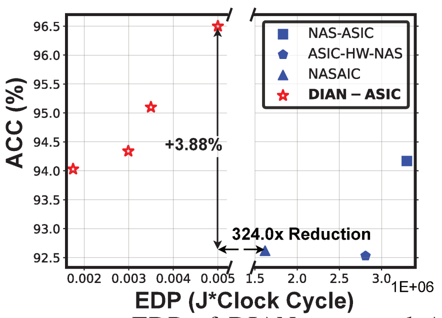


Fig. 5: Accuracy vs. EDP of DIAN generated ASIC-based accelerators over three SOTA co-exploration designs [15].

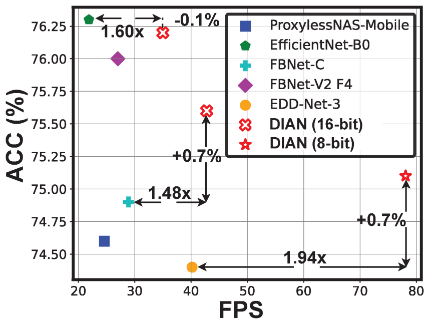


Fig. 6: Accuracy vs. FPS of DIAN and SOTA HA-NAS generated networks [3], [4], [9], [23] and EDD [6].

with (1) EDD, DIAN (8-bit) achieves a 0.7% higher accuracy and 1.94x higher FPS; and (2) SOTA HA-NAS methods, DIAN (16-bit) achieves a 1.48× higher FPS while having a +0.7% higher accuracy over FBNet and a 1.60× higher FPS with a comparable accuracy (-0.1%) over EfficientNet-B0.

D. DIAN ablation: DAS over SOTA DNN accelerators

The proposed DAS is a key enabler of DIAN. To evaluate its efficacy, we compare the hardware efficiency of DAS generated accelerators with SOTA accelerators under the same conditions. We consider two representative FPGA-based accelerators including [2], [24], when accelerating VGG16 on ImageNet. The results in Tab. IV show that our DIAN generated accelerators outperform both expert-designed and tool-generated SOTA accelerators under the same dataset, DNNs, and FPGA resources. For example, DAS generated accelerators achieve up to 2.12× improvement in throughput on VGG16 under the same setting. The consistent better performance of our DAS generated accelerators validates the effectiveness of our DAS engine in navigating over the large and discrete design space of DNN accelerators to search for optimal DNN accelerators. Note that when using DAS to generate optimal accelerators, we adopt the same precision and FPGA resources as the baselines for a fair comparison.

V. Conclusion

We propose DIAN, a generic network-accelerator co-search framework, to enable automated search for matched DNNs and their accelerators that maximize both task accuracy and hardware efficiency. Extensive experiments validate that DIAN generated networks and accelerators consistently outperform SOTA baselines in terms of both task accuracy and hardware efficiency, while notably boosting the search efficiency.

TABLE IV: DAS generated vs. SOTA FPGA accelerators on a Zynq XC70Z45 FPGA, with VGG16 on ImageNet at 200MHz.

	[2]	[24]	**DAS generated**
Resource Utilization	680/900 DSP	824/900 DSP	723/900 DSP
Performance (GOP/s)	262	230	**291** (↑1.11× - ↑2.12×)

Acknowledgment

This work was supported in part by the National Institutes of Health under Award R01HL144683, National Science Foundation under Awards CCF-1838873 and CAREER-2048183.

References

[1] Y. Chen *et al.*, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *JSSC 2017*, 2017.
[2] X. Zhang *et al.*, "Dnnbuilder: An automated tool for building high-performance dnn hardware accelerators for fpgas," in *ICCAD*, 2018.
[3] H. Cai *et al.*, "Proxylessnas: Direct neural architecture search on target task and hardware," *arXiv preprint arXiv:1812.00332*, 2018.
[4] B. Wu *et al.*, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *CVPR*, 2019.
[5] M. S. Abdelfattah *et al.*, "Best of both worlds: Automl codesign of a cnn and its hardware accelerator," in *DAC*, 2020.
[6] Y. Li *et al.*, "Edd: Efficient differentiable dnn architecture and implementation co-search for embedded ai solutions," in *DAC*, 2020.
[7] Y. Lin *et al.*, "Neural-hardware architecture search," 2019.
[8] Xilinx Inc., "Xilinx zc706 evaluation kit," https://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html, (Accessed on 09/30/2020).
[9] M. Tan *et al.*, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019.
[10] Y. Zhao, X. Chen, Y. Wang, C. Li, H. You, Y. Fu, Y. Xie, Z. Wang, and Y. Lin, "Smartexchange: Trading higher-cost memory storage/access for lower-cost computation," *arXiv preprint arXiv:2005.03403*, 2020.
[11] D. Chen *et al.*, "xpilot: A platform-based behavioral synthesis system," *SRC TechCon*, vol. 5, 2005.
[12] R. Venkatesan *et al.*, "MAGNet: A Modular Accelerator Generator for Neural Networks," in *ICCAD*, 2019.
[13] P. Xu *et al.*, "Autodnnchip: An automated dnn chip predictor and builder for both fpgas and asics," *arXiv preprint arXiv:2001.03535*, 2020.
[14] W. Jiang *et al.*, "Hardware/software co-exploration of neural architectures," *TCAD*, 2020.
[15] L. Yang, *et al.*, "Co-exploration of neural architectures and heterogeneous asic accelerator designs targeting multiple tasks," in *DAC*, 2020.
[16] K. Choi *et al.*, "Dance: Differentiable accelerator/network co-exploration," *arXiv preprint arXiv:2009.06237*, 2020.
[17] H. Liu *et al.*, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.
[18] E. Jang *et al.*, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.
[19] Y. Wu *et al.*, "Accelergy: An architecture-level energy estimation methodology for accelerator designs," in *ICCAD*, 2019.
[20] A. Parashar *et al.*, "Timeloop: A Systematic Approach to DNN Accelerator Evaluation," in *ISPASS*, 2019.
[21] Y. Zhao *et al.*, "Dnn-chip predictor: An analytical performance predictor for dnn accelerators with various dataflows and hardware architectures," in *ICASSP*, 2020.
[22] X. Yang *et al.*, "A systematic approach to blocking convolutional neural networks," *CoRR*, vol. abs/1606.04209, 2016. [Online]. Available: http://arxiv.org/abs/1606.04209
[23] A. Wan *et al.*, "Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions," *arXiv preprint arXiv:2004.05565*, 2020.
[24] Q. Xiao *et al.*, "Exploring heterogeneous algorithms for accelerating deep convolutional neural networks on fpgas," in *DAC*, 2017.
[25] Xilinx Inc., "Vivado High-Level Synthesis," https://https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html, accessed 2019-09-16.
[26] R. Balasubramonian *et al.*, "Cacti 7: New tools for interconnect exploration in innovative off-chip memories," *ACM Trans. Archit. Code Optim.*, vol. 14, no. 2, Jun. 2017. [Online]. Available: https://doi.org/10.1145/3085572
[27] Y. S. Shao *et al.*, "Aladdin: A pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures," in *ISCA*, 2014.