MDARTS: Multi-objective Differentiable Neural Architecture Search

Sunghoon Kim, Hyunjeong Kwon, Eunji Kwon, Youngchang Choi, Tae-Hyun Oh, and Seokhyeong Kang*
EE Department, POSTECH, Pohang, South Korea
*shkang@postech.ac.kr

***Abstract*—In this work, we present a differentiable neural architecture search (NAS) method that takes into account two competing objectives, quality of result (QoR) and quality of service (QoS) with hardware design constraints. NAS research has recently received a lot of attention due to its ability to automatically find architecture candidates that can outperform handcrafted ones. However, the NAS approach which complies with actual HW design constraints has been under-explored. A naive NAS approach for this would be to optimize a combination of two criteria of QoR and QoS, but the simple extension of the prior art often yields degenerated architectures, and suffers from a sensitive hyperparameter tuning. In this work, we propose a multi-objective differential neural architecture search, called MDARTS. MDARTS has an affordable search time and can find Pareto frontier of QoR versus QoS. We also identify the problematic gap between all the existing differentiable NAS results and those final post-processed architectures, where soft connections are binarized. This gap leads to performance degradation when the model is deployed. To mitigate this gap, we propose a separation loss that discourages indefinite connections of components by implicitly minimizing entropy.**

I. Introduction

In the face of the neural architecture deluge, automatic neural architecture search (NAS) [1]–[5] has gained considerable attention recently. This is because NAS has shown that an advanced neural architecture can be effortlessly designed and that outperforms hand-crafted designs (*e.g.*, [6], [7]) in many applications such as image classification [1], detection [8], and segmentation [9].

However, there are a few issues to leverage NAS for application-oriented hardware (HW) designers. First of all, most of early NAS methods entail an intractable search time. Early search strategy and search space lead to thousands of hours of search time [1]–[3]. Liu *et al.* [5] and many follow-up works [10]–[12] have dramatically reduced the search cost to a few days, but they focus only on reducing errors (*i.e.*, quality of result, QoR). Due to this biased development to QoR, the aforementioned NAS approaches typically sacrifice the resulting computational performance (*i.e.*, quality of service, QoS). This is not desirable to many QoS-critical applications, such as embedded AI applications, because a discovered architecture should be implemented on a restricted HW. In those applications, the failure to meet QoS constraints significantly degrades users' experience. Therefore, a NAS approach targeting to HW needs to find an architecture that satisfies both QoR and QoS constraints at the same time, called multi-objective NAS.

While there have been attempts to achieve the multi-objective NAS [13]–[19], they do not guarantee to hold hard constraints on QoS given HW constraints. Rather, they pose the problem into a single objective optimization by a linear combination of the QoR and QoS terms. These simple soft combination methods have been known to be often susceptible to yield biased solutions to either of the terms, *i.e.*, challenging to pursue to or control a favorable trade-off [20]. In addition, despite the advance on the heterogeneous execution capability of the state-of-the-art HWs [21]–[24], *e.g.*, pipelining, the previous studies have only focused on the evaluation of sequential and homogeneous executions, not heterogeneous executions.

This research was supported by National RD Program through the National Research Foundation of Korea(NRF) funded by Ministry of Science and ICT (NRF-2020M3F3A2A02082435).

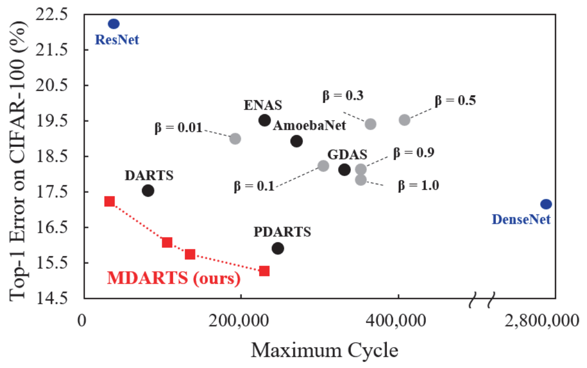


Fig. 1: Maximum cycle (QoS) *vs.* top-1 error (QoR) on CIFAR-100. Black dots: architectures obtained by [3], [5], [11], [12], [27]; Blue dots: hand-crafted architectures [6], [7]; Gray dots: architectures obtained by exploring combination losses according to a hyperparameter β. All the models are trained on CIFAR-100. MDARTS, proposed in this work, obtains notably better trade-offs than the others, and the red line can suggest a QoR and QoS trade-off guidance to users.

In this paper, we propose a multi-objective differentiable neural architecture search (MDARTS) that finds an optimal architecture in terms of QoS as well as QoR. In the optimization, QoS is evaluated by considering heterogeneous executions of recent HWs. In addition, we found two limitations when adopting QoS in differentiable neural architecture searches; 1) indefinite selection and 2) degenerated model. In this work, we propose two approaches to prevent the two problems in considering QoS, and provide the QoR and QoS guidance to users successfully (Figure 1). The primary contributions of our work are as follows:

- We propose a separation loss to avoid the indefinite selection. Differentiable NAS considering QoS inherently has the indefinite selection due to the mixed operation, which causes the uncorrelatedness between estimated QoS and actual QoS. The proposed separation loss helps to consider the final QoS in the process of seeking the architecture.
- MDARTS overcomes the inherent problem of skip-connection aggregation when the previous differential methods consider QoS in their search. MDARTS prevents the frequent selection of skip connections by accurately reflecting the lower bound QoS constraint in the search, so it avoids the selection of a degenerated architecture that have severe QoR degradation when solving other tasks.
- MDARTS conducted QoS evaluation according to the HW design trend of heterogeneous execution [21]–[24]. MDARTS considers the case of mapping the neural architecture to HW by considering the HW design trends during the architecture search.

The rest of this work is organized as follows. Section II presents the categories of NAS and our focal points. Section III presents MDARTS to overcome the weaknesses of previous works. Section IV presents the experimental setup and results, then Section V concludes this work.

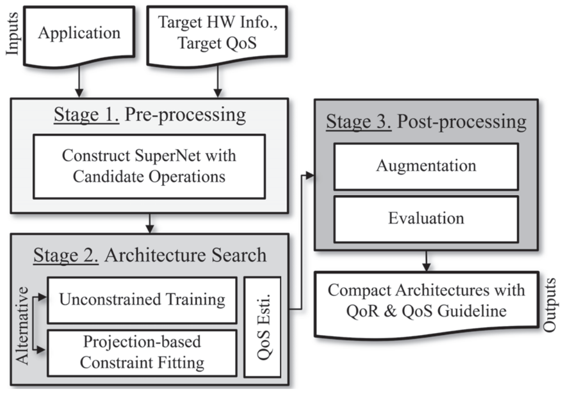
II. Related Work

NAS research consists of: (i) search strategy, (ii) search space, and (iii) evaluation metrics & HW platform .

Search Strategy in NAS can be categorized into three types: reinforcement learning (RL), evolutionary algorithm (EA), and differentiable method (DM). RL [1], [2], [27] improves the controller by optimizing reward, and returns a high-quality architecture corresponding to the highest reward. EA [3], [16], [28] explores the architecture design space by stochastic re-samplings. They consider architectures as a population, evaluates and compares the architectures, discards inferior ones, and generates next-generation architectures from the survivors by using crossovers and mutations. Both RL and EA require an enormous searching time to find the best operations due to ineffective search direction proposals. DM [5] dramatically reduces the search time by formulating a differentiable search space, where efficient and effective search directions are available to find a novel architecture. While our MDARTS is also a DM, we identify and tackle fundamental limitations shared across DMs when considering QoS: 1) degenerated model cases and 2) a gap between the output form of DM and the required form to be deployed.

Search Space can be divided into two types: macro-search and micro-search. Macro-search aims to directly discover the entire architecture by considering candidate operations, but the number of possible architectures increases exponentially with its number of layers. Therefore, macro-search is not scalable enough for a deep architecture search. Micro-search aims to discover cells and to design an entire architecture by stacking many copies of the discovered cells using two types: *normal* and *reduction* cells [2], [5]. Micro-search can significantly reduce search space and can build deeper architectures compared to that of macro-search. In this work, since our goal is to map a discovered neural architecture into heterogeneous HW architectures [21]–[24], we adopt a micro-search of which stacked cells are compatible to the heterogeneous ones.

Evaluation Metric & HW Platform can be categorized into three types: (i) QoR-centric [2], [3], [5], [10]–[12], (ii) QoR with the number of multiplication-and-accumulation operations (MAC) and parameters [16], [18], [29], and (iii) QoR with HW latency [13], [14], [17], [19], [30], [31]. The later two categories work in multi-objective regimes. One of difficulties of considering multi-objective, including QoS, stems from the fact that the number of MACs and the number of parameters do not directly reflect a latency of HW [32]. Template-based approaches [13], [30] search architectures with a predictable performance of FPGA accelerator, but their methods are limited to simple operations such as 3×3 convolution and 2×2 pooling. Lu *et al.* [31] introduced a joint architecture and quantization search based on FPGA implementation, but they exploited RL and a macro search space, so the number of layers was restricted to six to reduce the search space. Other prior arts [14], [17], [19] for general-purpose processors do not consider the parallel and heterogeneous executions. Our MDARTS considers both QoR and QoS reflecting the recent HW trends of heterogeneous executions; thereby, the discovered architecture is favorable to be implemented on the recent HWs.


Inputs
Application
Target HW Info., Target QoS
Stage 1. Pre-processing
Construct SuperNet with Candidate Operations
Stage 2. Architecture Search
Alternative
Unconstrained Training
Projection-based Constraint Fitting
QoS Esti.
Stage 3. Post-processing
Augmentation
Evaluation
Compact Architectures with QoR & QoS Guideline
Outputs
Fig. 2: Overview flow of MDARTS.

III. Proposed Method

Figure 2 shows the overview of our framework. Inputs include an application, target HW information, and target QoS. Our framework consists of three stages. In Stage 1, a *SuperNet* is created using candidate operations. In Stage 2, MDARTS searches architectures from a *SuperNet* considering QoR and QoS, simultaneously. Stage 3 performs augmentation and evaluation, and yields the architecture complying with guidelines for QoR and QoS to users. The following subsections describe the observed problems and our approaches in more detail.

A. Motivation

As discussed in Section II, we observed the limitations that should be carefully dealt with when considering deployment to recent HWs. When extending the prior arts to multi-objective optimization *w.r.t.* QoR and QoS as well as recent HW deployment, the following issues have been under-explored: (i) degenerated model and (ii) indefinite selection. In the following, we describe the issues in detail.

Two tasks of QoR and QoS in the multi-objective optimization are hard to jointly improve or balance because of their strong trade-offs. Thereby, the discovered architectures tend to abuse skip-connections to easily improve QoS [11], [33], [34]. The frequent skip-connection aggregation induces degenerated models that suffer from severe QoR degradation when applying to other applications. To overcome the skip-connection aggregation, existing approaches limited the number of skip-connections to two [11], [33] or tuned a balance hyperparameter, β, of a loss that is a simple combination of the tasks [34]. However, tuning β is computationally demanding, and there is no clear criterion for the optimal minimum number of skip-connections or for finding an optimal β, *c.f.* β's in Figure 1. Lower-bounding QoS can alleviate the skip-connection aggregation [29] by preventing the estimated QoS during the search from becoming smaller than the user-defined QoS lower bound. But unfortunately, there exists an inevitable uncorrelatedness between the estimated QoS and the actual QoS of HWs, due to the indefinite selection problem of DM methods.

Indefinite selection emerges in all the DM methods when the softmax based operation mechanism is used for differentibility. Given candidate primitive operations, the softmax probabilities provide soft selection according to the probabilities, *i.e.*, non-binary. Figure 3 empirically shows that, even in the converged point, there are multiple operations still survived (having mid-range probabilities). This introduces a gap between the output of DM based NAS and actual HW deployment, because the architecture output is post-processed

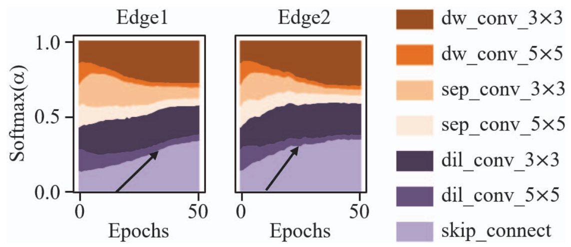


Fig. 3: The evolution of selection probabilities of all operations across the search epoch. It does not converge to a single operation (no clear zero or one), but the contributions remain distributed until the end, *i.e.*, indefinite selection, which causes the uncorrelatedness of the esticated QoS and the actual QoS.

so that the connections are binarized, *i.e.* connected or not. This yields a non-negligible discrepancy between the estimated QoS in the optimization process and the actual QoS of HWs, which is critical when considering HW deployment. In this regard, we postulate that encouraging definite selection can reduce the QoS gap, and with this better calibrated QoS, lower-bounding QoS can easily avoid degenerated model generation.

B. Separation Loss To Overcome Indefinite Problem And Skip-Connection Aggregation

MDARTS exploits two-phase iterative optimization [29], [40] to overcome the interference of the competing tasks of QoR and QoS. In the first phase, the QoR-centric optimization is performed using the following loss function.

$$\min_{\alpha} \mathcal{L}_{val}^{total}\left(w^*(\alpha),\alpha\right) \quad s.t.\ w^*(\alpha) = \underset{w}{\operatorname{argmin}}\ \mathcal{L}_{train}^{total}(w,\alpha), \quad (1)$$

where $\mathcal{L}_{\{train,val\}}^{total}$ denote the loss functions calculated using training and validation datasets, respectively.

Using the architecture parameter, α, in the first phase, QoS of the architecture is evaluated by: $\mathrm{L}_{co}(\alpha) = \sum_{i\leq M}\sigma(i)\cdot p_{o_i}(\alpha)\cdot co_{o_i}$, where $\sigma(\cdot)$ represents an indicator function, o_i is a candidate operation in *SuperNet*, $\mathcal{O} = \{o_1, o_2, \ldots, o_M\}$, that consists of M candidate operations, $p_{o_i}(\alpha) = \mathrm{softmax}(\alpha)$ is the probability that o_i is selected, and co_{o_i} is the cost function of o_i for evaluating QoS. Our QoS estimation for the heterogeneous HWs is described in detail in Section III-C. The second phase finds an optimal architecture that has a cost function value, Eq. III-B, between the user-defined lower-bound and upper-bound QoS constraints, C_L and C_H as follows: $\mathrm{C}_L \leq \mathcal{L}_{co}(\alpha) \leq C_H$. To project intermediate solutions to the feasible solution, we convert Eq. III-B to the following projection operation [5]: $\min_{\alpha_p}\phi(\alpha_p) = (\alpha_p - \alpha)^2 + \tau\cdot max(C_L - \mathcal{L}_{co}(\alpha_p), 0)$ $+\tau\cdot max(0, \mathcal{L}_{co}(\alpha_p) - C_H)$. Eq. III-B project α to its nearest point, α_p, so that $\mathcal{L}_{co}(\alpha_p)$ is in the bounds of Eq. III-B.

The lower bound C_L on QoS can alleviate the skip-connection aggregation that easily minimizes Eq. III-B. However, as shown in Figure 3, despite the convergence, the selection probabilities of candidate operations have indefinite values. Since the expected QoS defined in Eq. III-B is calculated by the weighted sum of QoS for all operations with scattered selection probabilities, it has a non-negligible discrepancy with the actual QoS of HWs which requires the probabilities to be 0 or 1. As a result, the second phase could not reflect the user-defined C_L correctly, whereby the number of skip-connections cannot be effectively restricted.

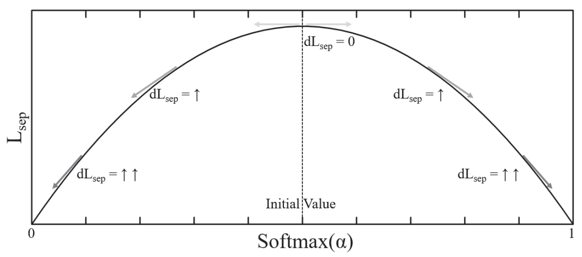


Fig. 4: Illustration of separation loss $\mathcal{L}_{sep}$

We propose to impose a separation loss into the total loss function, $\mathcal{L}^{total}$, to encourage definite selection. We define the proposed separation loss as follows:

$$\mathcal{L}_{sep}(\alpha) = -\sum\left(p_{o_i}(\alpha) - \mathit{initial\_value}\right)^2, \quad (2)$$

where $\mathit{initial\_value} = 1/M$, M denotes the number of operations in the set $\mathcal{O}$. Eq. 2 is illustrated in Figure 4. From the ambiguous initial value $1/M$, as long as $\mathrm{softmax}(\alpha)$ is started to be deviated from the initial value by optimization, $|d\mathcal{L}_{sep}|$ is increased. It makes $\mathrm{softmax}(\alpha)$ encourage to favor extreme values, which can be regarded as a type of entropy minimization, *i.e.,* disambiguate. By introducing the separation loss, $\mathcal{L}_{co}$ reflects the actual QoS better; thus, the lower bound C_L on QoS is more effectively reflected during the architecture search.

Our final MDARTS loss function used in the first phase, Eq. 1, can be formulated as:

$$\mathcal{L}^{total}(w,\alpha) = \mathcal{L}_{CE}(w,\alpha) + \mu_{sep}\cdot\mathcal{L}_{sep}(\alpha), \quad (3)$$

where $\mathcal{L}_{CE}$ denotes the cross entropy and μ_{sep} a balance parameter. By using Eq. 3 in the first phase, the optimization favors to explore directions to prevent the indefinite selection of the operations. This closes the gap between QoS estimation by Eq. III-B and the actual QoS. With the calibrated QoS, the second phase better reflects the user-defined C_L, whereby the skip-connection aggregation can be suppressed. The whole architecture search is done by alternatively solving Eqs. 1 and III-B.

C. QoS Estimation for Heterogeneous HWs

Recent HWs are enhanced to execute the operations of the neural architecture heterogeneously [21]–[24]. Therefore, QoS estimation in the architecture search needs to consider the heterogeneous executions. Because it is infeasible to measure the real run-time latency of the heterogeneous executions, we propose a simulation framework to consider the heterogeneous HWs in NAS. First, we assume that the execution of a layer is performed for two application-specific integrated circuits (ASICs), Eyeriss [26] and Diannao [38]. We then used the state-of-the-art simulator Timeloop [37] to figure out the cycles of the candidate operations (Table I). The cycle varies depending on the operation and the height, width, and channel of the feature map, and only the first cell is shown due to space constraints. Then the heterogeneous-aware QoS estimation of *the micro-search-based architecture* is closely related to the principle of the cell. c_{k-1} and c_{k-2} in Figure 5 denote the output feature map of previous two cells, and are used as an input feature maps in k^{th} cell to obtain the output feature map c_k. Each edge represents an operation, so intermediate feature maps are computed in parallel following [21]–[24] when they pass through edges. Intermediate feature maps

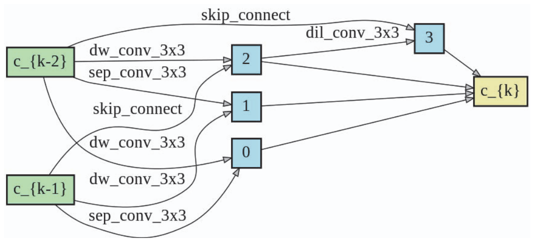


Fig. 5: A normal cell found on CIFAR-10 [25] with Eyeriss [26]

connected to each intermediate node (blue boxes in Figure 5) are summed. All operations of node 2 must be completed before the operations at node 3 are performed, so stalls inevitably occur during the dilated convolution 3×3 that connects node 2 to node 3 (Figure 5). Finally, results in intermediate nodes are concatenated to generate c_k. Thus the cycle of the node that is computed last is set as the maximum cycle to obtain c_k.

IV. Experiment Setup and Results

We conducted experiments on the standard benchmark classification datasets to evaluate our method. All experiments were conducted on a server with Intel ® Xeon Gold 6134 CPU @ 3.2 GHz and one NVIDIA Titan XP 12 GB GPU. Our code was implemented based on PyTorch 1.4.0 and Python 3.6.10.

A. Experiment Setup

Dataset. We used two datasets: CIFAR-10 and 100 [25], which consist of 50K training images and 10K test images with the resolution of 32×32, and have 10 and 100 classes, respectively.

Training Procedure. During the first phase of the training, we divided each 50K training set into two subsets of 25K; then we set the batch size to 96, and search epoch to 50, following [5], [10]. To optimize architectural weights w, one subset and stochastic gradient decent (SGD) were used (initial learning rate = 0.025 with cosine scheduling, momentum = 0.9, and weight decay = 3×10^4). To optimize architecture parameters α, the other subset and Adam [35] were used (initial learning rate = 3×10^{-4}, momentum = (0.5, 0.999), and weight decay = 10^{-3}). During the second phase, we also used Adam with the same setting, but without training subsets. The number of constraint fitting iteration (u_{max}) for Eq. III-B were set to 100. The lower and upper bounds for target QoS (number of cycles) were selected in the range of 0.5×10^6 to 1.7×10^6 in increments of 0.2×10^6 partition.

The compact architecture is composed of 20 cells (18 *normal cells* and 2 *reduction cells*). Following [2], [5], [10]–[12], [18], we trained the compact architecture from scratch using data augmentation techniques, so this is so-called post-processing (Stage 3 in Figure 2). For CIFAR-10/-100 [25], we trained from scratch for 600 epochs with a batch size of 128. We used the SGD with an initial learning rate of 0.025, it decayed to zero with cosine scheduling. We used Cutout [39] for data augmentation unless mentioned.

Search Space. Following [5], [10]–[12], [18], [29], [33], *reduction cells* were placed at 1/3 and 2/3 of the total depth of the architecture between two *normal cells*. Similar to [12], we reduced candidate operations in *reduction cell* with three operations: 3×3 max-pooling, 3×3 average-pooling and skip-connection. Similar to [36], we substituted 3×3 max-pooling, 3×3 average-pooling to 3×3 and 5×5 depth-wise convolution for *normal cell*, respectively. Other candidate operations are identical to [5], [10]–[12], [18], [29],

TABLE I: Comparison of the first cell's cycle with two different ASICs. (Obtained by the state-of-the-art simulator, Timeloop [37])

Operations	Eyeriss [26] on CIFAR [25]	Diannao [38] on CIFAR [25]
dw_conv_3x3	12,288	9,216
dw_conv_5x5	40,960	25,600
sep_conv_3x3	28,672	20,480
sep_conv_5x5	86,016	53,248
dil_conv_3x3	15,920	11,428
dil_conv_5x5	53,888	33,424

TABLE II: Four architectures (A to D) found by MDARTS according to different QoS constraints.

Architectures	QoS Constraints $\mathcal{L}_{co}$ $(\times 10^6)$	
	Lower bound (C_L)	Upper bound (C_H)
A	0.5	0.7
B	0.9	1.1
C	1.1	1.3
D	1.5	1.7

[33]. This approach diminishes the search space from $8^{14} \times 8^{14}$ = 1.93×10^{25} [5] to $7^{14} \times 3^{14}$ = 3.24×10^{18}, which brings out reducing the usage of GPU memory and search cost effectively.

B. Experimental Results

In our experiments, MDARTS has found four architectures according to QoS constraints (Table II). We first verified the proposed separation loss to prevent the production of degenerated architectures. Figure 6a shows the heat map of $p = \mathrm{softmax}(\alpha)$ for edges in a *SuperNet* at the last epoch of the search stage. MDARTS can determine exactly which operation to choose for each edge with guidance of the $\mathcal{L}_{sep}$ and constraints. In addition, MDARTS successfully prevents the continual increase in the number of skip-connections by accurately reflecting the user-defined lower bound for QoS that has a high correlation with the actual QoS.

We compared the resulting optimal architectures and the search costs of MDARTS and the benchmark methods on CIFAR-10 and CIFAR-100 (Table III). Based on the Eyeriss [26], MDARTS has searched the advanced architecture for CIFAR-10 and CIFAR-100, and it has taken 0.27 GPU-days and 0.3 GPU-days, respectively. We reported the classification error as "mean±std" for *all* six random seeds which means that we do *not* pick the best. Cutout [39] was used except [7], [28].

MDART finds the architecture with the best QoR for the necessary QoS constraints. GDAS [12] fixed the *reduction cell* with one configuration, and thereby reduced the search space extremely, so the search cost is better than MDARTS, but the error rate is 0.27% and 1.59% higher than MDART-C10-A on CIFAR-10 and CIFAR-100, respectively. Moreover, the maximum cycle of GDAS is 7.44× greater than MDARTS-C10-A. PC-DARTS [10] introduced a partial connection concept to reduce memory usage and thus dramatically reduce the search cost. However, QoS is not considered, so the configuration of the cell is complicated. Furthermore, the test error on CIFAR-10 is 0.22% higher for PC-DARTS than for MDART-C10-C with 2.64× greater cycles.

For CIFAR-100, this trend and has already been described in Section I (Figure 1). Kindly note that when the MDARTS-C10 architectures were post-processed on CIFAR-100, the MDARTS-C10-C had the lowest error rate (14.99%). A slightly incorrect guideline stems from MDARTS-C10 searching CIFAR-10 rather than CIFAR-100. We also conducted searches for suitable architectures with Diannao [38]

TABLE III: Comparison of architectures on CIFAR-10 and CIFAR-100. Top-1 test error of MDARTS denotes "mean ± standard deviation" across six runs with different random seeds. The maximum cycle is computed using the configurations of micro cell provided by the authors with Eyeriss [26][5] by assuming that the heterogeneous execution is available following [21]–[24]. For architectures designed by human experts [6], [7], it is difficult to directly compare maximum cycle, so these are estimated values. ‡ indicate results reported in [12]. ◇ indicate results reported in [10], [11]. ∗ averaged across ten runs from the pretrained architecture provided by the author.

Architecture		Search Method	Search Cost (GPU-days)	Test Err. (%) on		Maximum Cycle	# of Params (millions)
				CIFAR-10	CIFAR-100		
ResNet [6]‡		Human Expert	-	4.61	22.1	49,152	1.7
DenseNet-BC [7]‡		Human Expert	-	3.46	17.18	2,764,800	25.6
NASNet-A [2]		RL (micro)	1,800	2.65	-	561,152	3.8
ENAS [27]		RL (micro)	0.45	2.89	19.53	229,376	4.0
AmoebaNet-A [3]		EA (micro)	3,150	3.12	18.93	270,336	3.2
LEMONADE [28]		EA (macro)	80	3.05	-	-	4.7
NSGANet [16]		EA (micro)	8	2.75	-	139,904	3.3
DARTS_V2 [5]		DM (micro)	1	2.76	17.54	81,456	3.4
GDAS [12]		DM (micro)	0.21	2.93	18.13	331,776	3.4
PDARTS [11]		DM (micro)	0.30	2.62	15.92	246,576	3.6
PC-DARTS [10]		DM (micro)	**0.10**	2.57	-	323,584	3.4
RAPDARTS [18]		DM (micro)	12	2.83	-	68,704	2.6
RC-DARTS [29]		DM (micro)	1	2.83	-	94,208	3.3
proxylessNAS [14]		DM (macro)	4◇	2.72 ± 0.04*	-	-	5.7
MDARTS-C10 (ours)	A	DM (micro)	0.27	2.56 ± 0.04	16.54 ± 0.17	44,592	2.1
	B	DM (micro)	0.27	2.51 ± 0.17	16.10 ± 0.21	49,152	2.4
	C	DM (micro)	0.27	**2.35** ± 0.13	**14.99** ± 0.23	122,416	3.4
	D	DM (micro)	0.27	**2.35** ± 0.16	15.69 ± 0.21	172,032	3.7
MDARTS-C100 (ours)	A	DM (micro)	0.30	-	17.22 ± 0.16	32,768	1.9
	B	DM (micro)	0.30	-	16.08 ± 0.19	106,496	3.0
	C	DM (micro)	0.30	-	15.74 ± 0.19	135,168	2.7
	D	DM (micro)	0.30	-	**15.26** ± 0.20	230,016	3.8

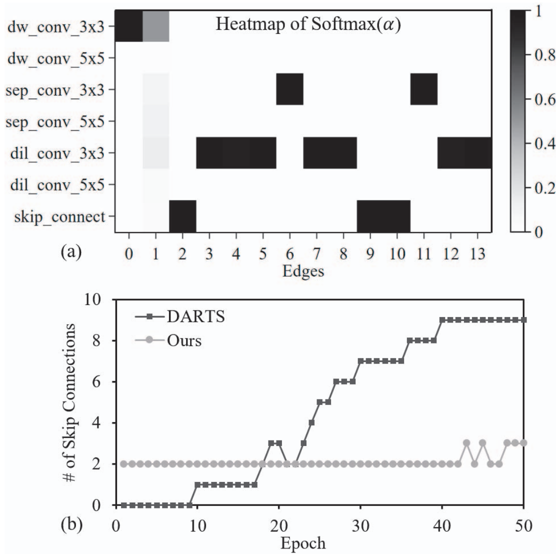


Fig. 6: (a) Heatmap of $p_{o_i} = \mathrm{softmax}(\alpha)$ in each edge of *SuperNet* at the last epoch of search stage. (b) The number of skip-connections when searching with DARTS [5] and MDARTS on CIFAR-10 [25].

(Table IV). After searching on CIFAR-10, the post-processing was performed on both CIFAR-10 and CIFAR-100. MDARTS-Dian-D obtained test errors of 2.26% on CIFAR-10 and 15.25% on CIFAR-100, respectively. Moreover, compared to MDARTS-C10-D and MDARTS-C100-D, MDARTS-Dian-D achieved 1.62× and 2.16× better QoS, respectively.

TABLE IV: Architectures searched on CIFAR-10 [25] with Dianao [38] and post-processed on CIFAR-10/100.

Architecture	Test Err. (%)		Maximum Cycle	# of Params (millions)
	C-10	C-100		
MDARTS-Dian-A	2.62	17.51	24,576	1.7
MDARTS-Dian-B	2.35	15.84	47,268	2.4
MDARTS-Dian-C	2.30	15.65	76,800	3.7
MDARTS-Dian-D	2.26	15.25	106,496	3.8

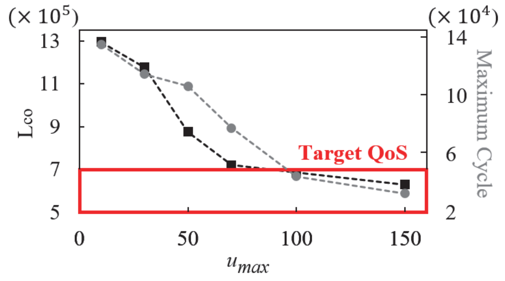


Fig. 7: Effects of the constraint fitting parameter u_{max} (x-axis) in the architecture search stage to $\mathcal{L}_{co}$ (left y-axis) and Maximum cycle (right y-axis).

Finally, we figured out the effect of the hyperparameter, u_{max}, of the QoS-centric second phase search. Figure 7 shows $\mathcal{L}_{co}$ for different u_{max}. Target QoS was not reached when u_{max} was small because the number of times to solve Eq. III-B is overly reduced, but the relationship between $\mathcal{L}_{co}$ and final QoS (maximum cycle) can be clearly observed. Moreover, the computation overhead is also smaller than other metrics. We have conducted the experiments by setting the u_{max} that could reach the smallest target QoS.

V. Conclusion

In this paper, we introduce the MDARTS framework, a multi-objective differentiable neural architecture search for heterogeneous HWs. We first identified the over-looked issues when considering the combination of NAS and HW deployment. Then, we proposed a separation loss to prevent indefinite selection of the operation, whereby we achieved a high correlation between the estimated QoS and the actual QoS, and also the prevention of the degenerated model. With the guidance of the trade-off between QoR and QoS, MDARTS can find an architecture just within 0.3 GPU-days while it has 2.35% and 14.99% test errors on CIFAR-10/-100, respectively, which is 2.64× better QoS compared with [10].

Our contributions have verified on the the standard benchmark datasets, CIFAR-10/-100, through various experiments. While the CIFAR dataset has typically shown consistent results with that of ImageNet in many machine learning literatures, it is still an interesting research question whether the algorithmic behavior would be consistent for a larger scale dataset. Due to the limited resource for ImageNet experiments, we leave it as future work.

References

[1] B. Zoph, and Q. V. Le, "Neural Architecture Search with Reinforcement Learning", *Proc. ICLR*, 2017.

[2] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition", *Proc. CVPR*, 2018.

[3] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized Evolution for Image Classifier Architecture Search", *Proc. AAAI*, 2019.

[4] A-C. Chengy, J-D. Dongy, C-H. Hsuy, S-H. Changy, M. Suny, S-C. Changy, J-Y. Panz, Y-T. Chenz, W. Weiz, and D-C. Juan, "Searching Toward Pareto-Optimal Device-Aware Neural Architectures", *Proc. ICCAD*, 2018.

[5] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable Architecture Search", *Proc. ICLR*, 2019.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for ImageRecognition", *Proc. CVPR*, 2016.

[7] G. Huang, Z. Liu, L. Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks". *Proc. CVPR*, 2017.

[8] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, "DetNAS: Backbone Search for Object Detection", *Proc. NeurIPS*, 2019.

[9] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. Yuille, and L. Fei-Fei, "Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation", *Proc. CVPR*, 2019.

[10] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian and H. Xiong, "PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search", *Proc. ICLR*, 2020.

[11] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive Differentiable Architecture Search: Bridging the Depth Gap between Search and Evaluation", *Proc. ICCV*, 2019.

[12] X. Dong and Y. Yang "Searching for A Robust Neural Architecture in Four GPU Hours", *Proc. CVPR*, 2019.

[13] C. Hao, X. Zhang, Y. Li, S. Huang, J. Xiong, K. Rupnow, W. Hwu, and D. Chen, "FPGA/DNN Co-Design: An Efficient Design Methodology for IoT Intelligence on the Edge", *Proc. DAC*, 2019.

[14] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware", *Proc. ICLR*, 2019.

[15] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q.V. Le, "MnasNet: Platform-Aware Neural Architecture Search for Mobile", *Proc. CVPR*, 2019.

[16] Z. Lu, L. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-Net: Neural Architecture Search using Multi-Objective Genetic Algorithm", *Proc. GECCO*, 2019.

[17] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "FBNet: Hardware-aware Efficient ConvNet Design via Differentiable Neural Architecture Search", *Proc. CVPR*, 2019.

[18] S. Green, C. M. Vineyard, R. Helinski, and C. Koc, "RAPDARTS: Resource-Aware Progressive Differentiable Architecture Search", *arXiv preprint arXiv:1911.05704v1*, 2019.

[19] S. Zeng, H. Sun, Y. Xing, X. Ning, Y. Shan, X. Chen ,Y. Wang, and H. Yang, "Black Box Search Space Profiling for Accelerator-Aware Neural Architecture Search", *Proc. ASP-DAC*, 2020.

[20] O. Sener, and V. Koltun "Multi-Task Learning as Multi-Objective Optimization", *Proc. NeuralPS*, 2018.

[21] L. Song, F. Chen, Y. Zhuo, X. Qian, H. Li, and Y. Chen, "AccPar: Tensor Partitioning for Heterogeneous Deep Learning Accelerators", *Proc. HPCA*, 2020.

[22] H. Kwon, L. Lai, T. Krishna, and V. Chandra, "HERALD: Optimizing Heterogeneous DNN Accelerators for Edge Devices", *arXiv preprint arXiv:1909.07437*, 2019.

[23] Cerebrase System, "Wafer-Scale Deep Learning", *Proc. Hotchips*, 2019.

[24] Y. Shao, J. Clemons, R. Venkatesan, B. Zimmer, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S.G. Tell, Y. Zhang, W.J. Dally, J. Emer, C.T. Gray, B. Khailany, and S.W. Keckler "Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture", *Proc. MICRO*, 2019.

[25] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 and CIFAR-100 datasets". 2009, available at https://www.cs.toronto.edu/kriz/cifar.html6.

[26] Y. Chen, T. Krishna, J. Emer and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks", *Proc. ISSCC*, 2016.

[27] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient Neural Architecture Search via Parameter Sharing", *Proc. ICML*, 2018.

[28] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient Multi-objective Neural Architecture Search via Lamarckian Evolution", *Proc. ICLR*, 2019.

[29] X. Jin, J. Wang, J. Slocum, M.-H. Yang, S. Dai, S. Yan, and J. Feng, "RC-DARTS: Resource Constrained Differentiable Architecture Search", *arXiv preprint arXiv:1912.12814*, 2019.

[30] C. Hao, Y. Chen, X. Liu, A. Sarwari, D. Sew, A. Dhar, B. Wu, D. Fu, J. Xiong, W. Hwu, J. Gu, and D. Chen "NAIS: Neural Architecture and Implementation Search and its Applications in Autonomous Driving", *Proc. ICCAD*, 2019.

[31] Q. Lu, W. Jiang, X. Xu, Y. Shi, and J. Hu, " On Neural Architecture Search for Resource-Constrained Hardware Platforms", *Proc. ICCAD*. 2019.

[32] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design", *Proc. EECV*, 2018.

[33] H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, and Z. Li, "DARTS+: Improved Differentiable Architecture Search with Early Stopping", *arXiv preprint arXiv:1909.06035*, 2019.

[34] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox and F. Hutter, "Understanding and Robustifying Differentiable Architecture Search", *Proc. ICLR* 2020.

[35] D.P. Kingma, and J. Ba "Adam: A Method for Stochastic Optimization", *Proc. ICLR*, 2015.

[36] Y. Weng, T. Zhou, L. Liu and C. Xia, "Automatic Convolutional Neural Architecture Search for Image Classification Under Different Scenes", *in IEEE Access*, 2019, pp. 38495-38506.

[37] A. Parashar, P. Raina, Y. S. Shao, Y-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler and J. Emer, "Timeloop: A Systematic Approach to DNN Accelerator Evaluation", *IEEE Proc. ISPASS*, 2019.

[38] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "DianNao: a small-footprint high-throughput accelerator for ubiquitous machine-learning" *Proc. ASPLOS*, 2014.

[39] T. DeVries, and G. W. Taylor "Improved Regularization of Convolutional Neural Networks with Cutout", *arXiv preprint arXiv:1708.04552*, 2017.

[40] J. Zhang and M. Styblinski, *Yield and Variability Optimization of Integrated Circuits*, Boston, MA: Kluwer, 1995.