Co-Optimization of DNN and Hardware Configurations on Edge GPUs

Halima Bouzidi*, Hamza Ouarnoughi*, Smail Niar*, El-Ghazali Talbi† and Abdessamad Ait El Cadi*
*LAMIH/UMR CNRS, Université Polytechnique Hauts-de-France, Valenciennes, France
†Université de Lille, CNRS/CRISTAL INRIA Lille Nord Europe
Email: *{firstname.lastname}@uphf.fr
*El-ghazali.Talbi@univ-lille.fr

2022 25th Euromicro Conference on Digital System Design (DSD) | 978-1-6654-7404-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/DSD57027.2022.00060

***Abstract*—The ever-increasing complexity of both Deep Neural Networks (DNN) and hardware accelerators has made the co-optimization of these domains extremely complex. Previous works typically focus on optimizing DNNs given a fixed hardware configuration or optimizing a specific hardware architecture given a fixed DNN model. Recently, the importance of the joint exploration of the two spaces drew more and more attention. Our work targets the co-optimization of DNN and hardware configurations on edge GPU accelerators. We propose an evolutionary-based co-optimization strategy by considering three metrics: DNN accuracy, execution latency, and power consumption. By combining the two search spaces, a larger number of configurations can be explored in a short time interval. In addition, a better tradeoff between DNN accuracy and hardware efficiency can be obtained. Experimental results show that the co-optimization outperforms the optimization of DNN for fixed hardware configuration with up to 53% hardware efficiency gains with the same accuracy and inference time.**

***Index Terms*—DNN, Edge GPU, Hardware-aware Neural Architecture Search, Multi-objective optimization**

I. Introduction and related works

Deep Neural Networks (DNN) and hardware accelerators are both leading forces for the observed progress in Machine Learning (ML). However, DNN architectures are becoming more and more complex and resource-demanding. When these architecture have to be supported by edge systems, they need careful optimization to achieve the best tradeoff between accuracy and hardware efficiency. To meet this challenge, Hardware-aware Neural Architecture Search (HW-NAS) has been proposed [1] in which DNN hardware efficiency is considered during the exploration process.

Nevertheless, hardware efficiency depends not only on the DNN architecture but also on the hardware configuration [2], [3]. Most existing works on HW-NAS fall into the optimization of DNN without considering the reconfigurability of the hardware accelerator. As discussed in [4], this approach is sub-optimal because the HW-NAS search space is narrower when considering only a fixed hardware configuration. Thus, by considering the hardware design space, it is possible to find tailor-made DNNs for each hardware configuration and vice-versa. The joint exploration of both search spaces is referred to as *the co-optimization* in this paper.

Recent works [5]–[12] have tackled the co-optimization problem where DNN architectures are optimized jointly with hardware configurations. Multiple DNN-HW pairs are generated and evaluated during the exploration process to choose the best pair(s). However, this strategy incurs a huge search time given the complexity of the joint search space [13]. Therefore, another co-exploration strategy has been proposed in [14]–[16], where DNN and hardware optimizations are done in a separate way. The two optimization algorithms communicate and exchange information to adjust the exploration process at some points.

However, although this strategy solves the drawback of the first joint approach, the sub-optimality of the final results remains its critical issue. The works mentioned above can also be classified according to the following factors [4]: DNN search space, targeted hardware accelerator and the exploration algorithm implementation in terms of objective functions and fitness evaluation strategy.

Nevertheless, only few works have attempted to consider the co-optimization problem for GPU-based hardware platforms. Recent edge GPUs allow the reconfigurability of different components such as the number of the processing units or the operating frequencies. The impact of these parameters on DNN performance has been well discussed and analyzed in [17], [18].

For instance, the authors in [2] adjust both the configuration of the GPU operating frequencies and the batch size of the DNN to maximize the inference hardware efficiency. However, no prior work has studied the joint optimization of DNN and hardware configurations on edge GPUs to the best of our knowledge. Therefore, we tackle this problem and give a detailed study about the impact of the hardware configuration on the DNN efficiency. Our obtained results motivate considering the hardware reconfigurability in Hardware-aware Neural Architecture Search to support the DNN accuracy and latency requirements under minimal power consumption.

Our paper is structured as follows. In section II we present and explain the motivation of our work. In section III we first formulate our multi-objective co-optimization problem. Then, we describe and explain our optimization approach. Section IV presents our experimental setup and results. Then we discuss our obtained results and compare them to other approaches and state-of-the-art solutions. Finally, the conclusion will be given in section V.

2771-2508/22/$31.00 ©2022 IEEE
DOI 10.1109/DSD57027.2022.00060

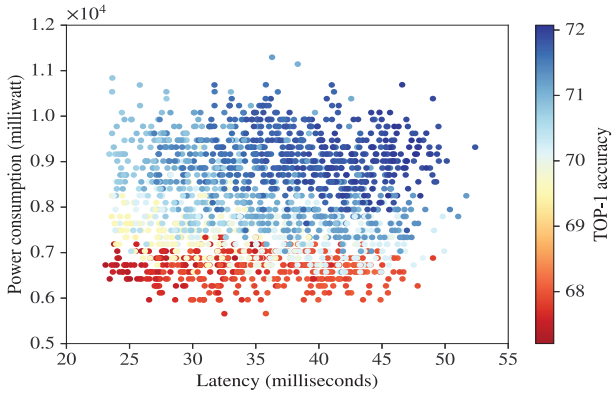

Fig. 1. The results of performing an HW-NAS under fixed edge GPU's hardware configuration.

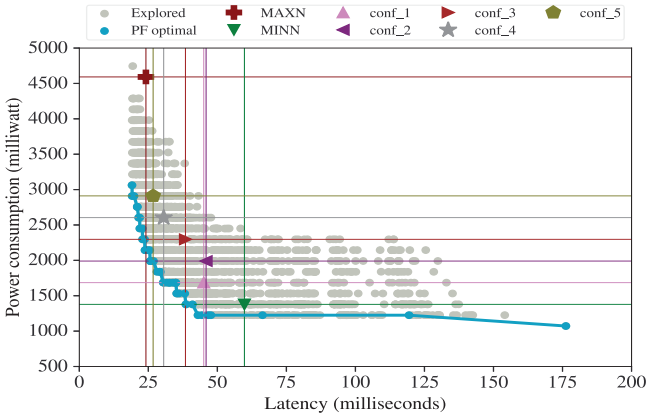

Fig. 2. The results of optimizing edge GPU's hardware configuration for a fixed DNN model.

II. Motivation

Figure 1 shows that different DNN models give different tradeoffs between accuracy and hardware efficiency under a fixed hardware configuration. In this paper the term *hardware efficiency* refers to the trade-off between latency and power consumption in the inference. This figure gives the results of a Hardware-aware Neural Architectural Search (HW-NAS) that we performed on the NVIDIA Jetson AGX edge GPU. The DNN models are sampled from the trained supernet proposed by AttentiveNAS [19]. Each point represents a DNN model in the search space. The x-axis and y-axis represent the measured latency and power consumption, respectively, on the edge GPU. The color of the points indicates the TOP-1 accuracy of the DNN. Figure 1 shows that different tradeoffs are obtained between accuracy and hardware efficiency for each explored DNN model. Figure 2 illustrates that the hardware efficiency of a single DNN varies when varying the hardware configuration. This figure gives the results of an exhaustive exploration of hardware configurations for a fixed DNN model, EfficientNet-B0 [20] in this case. To showcase the impact of the hardware configuration on the overall hardware efficiency of the DNN, we compare the obtained results with the predefined default hardware configurations proposed by NVIDIA. In this figure MAXN (resp. MINN) is the NVIDIA Jetson AGX configuration with the highest (resp. the smallest) allowed clock frequency. MAXN (resp. MINN) in general maximizes (resp. minimises) the processing speed at the cost of a high (resp. low) power consumption. The other configurations (i.e., from conf_1 to conf_5) are proposed to achieve a tradeoff between MAXN and MINN[1]. Remarkably, the optimal Pareto front, marked in blue in Figure 2) and obtained by an exhaustive exploration identifies hardware configurations that completely dominate all NVIDIA's predefined default configurations. It's important to mention that the Pareto front does not contain any of the NVIDIA's predefined configurations (MAXN, MINN and conf_1 to conf_5). Furthermore, the dominant solutions in the Pareto front improve upon the default configurations of NVIDIA (i.e., MAXN and MINN) by 57% and 40% for power consumption and latency, respectively This result shows the necessity to explore the space of the hardware configurations, to enhance the hardware efficiency of the DNN.

[1]Jetson developer kits and modules: https://docs.nvidia.com/jetson/l4t/

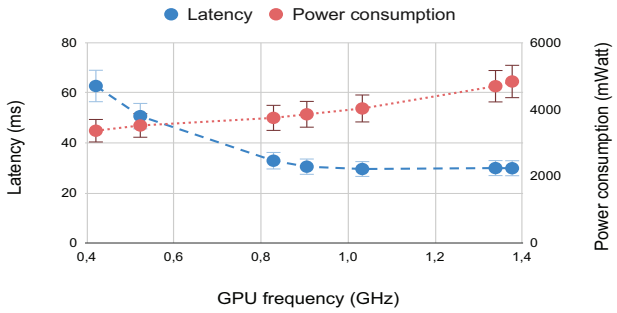

Fig. 3. Impact of GPU frequency on latency and power consumption

To better understand the impact of the hardware parameters, in figure 3 we report the impact of the GPU frequency values on the DNN hardware efficiency (i.e., latency and average power consumption). We vary one frequency at once while fixing the other frequencies (i.e., CPU and EMC frequencies) to their maximum value. We note that similar trend has been also observed when varying the CPU and EMC frequency.

From figure 3, we observe that the latency decreases by increasing the frequency from 400 MHz to 900MHz. However, after 900MHz, the latency remains the same, but the power consumption keeps increasing drastically. Hence, it is necessary to determine which configurations give the minimal latency and the lowest power consumption. From figure 3, it is evident that a frequency of 900 MHz gives the best tradeoff between latency and power consumption. However, this value can be further optimized by choosing adequate CPU and EMC frequency values. Furthermore, the optimal values of frequen-

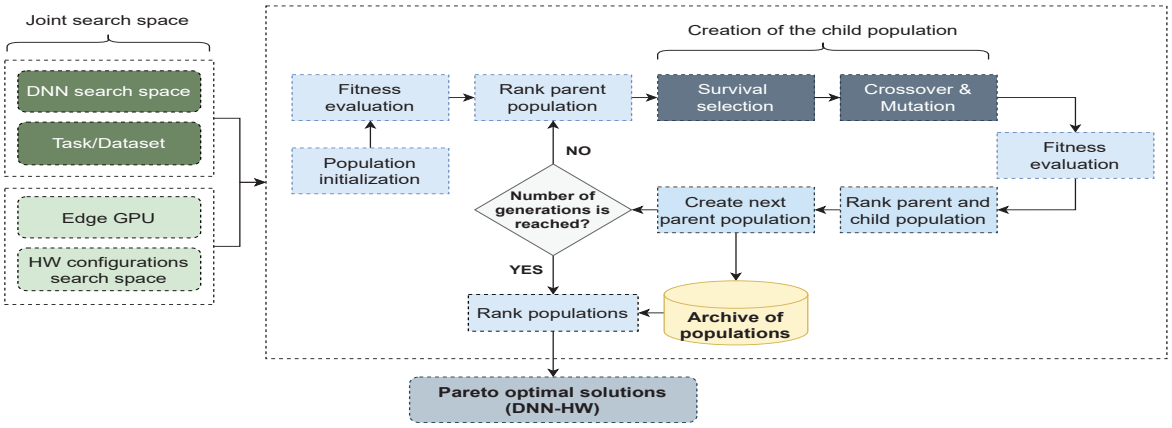

Fig. 4. Overview on the proposed co-optimization approach based on NSGA-II

cies change according to the DNN workload. Thus, there is no optimal hardware configuration for all the DNNs. From the above discussion, we can conclude that the performances of a DNN model are determined by the DNN architecture and the HW configuration. However, understanding the impact of these factors is not straightforward, which motivates the co-optimization of DNN and hardware configurations.

III. Proposed Approach

A. Problem formulation

Our DNN-HW co-optimization problem can be formulated as a multi-objective problem. As exploring the whole HW space and the whole DNN space is time-consuming and costly in terms of development efforts, we need a tool for a rapid DNN/HW co-design space exploration. Furthermore, while accelerating the co-design space exploration, the tool must adequately provide good approximation of the Pareto front. In this paper, we focus on solutions depicting the highest DNN accuracy and hardware efficiency. The mathematical formulation of our problem is as follows:

$$\min F(dnn, hc) = [(E(dnn), L(dnn, hc), P(dnn, hc)]^T$$
$$\text{s.t.} \quad (dnn, hc) \in (DNN \times HWConf) \quad (1)$$

Where dnn represents a DNN model defined by the DNN decision variables detailed in table I. hc represents a hardware configuration defined by the hardware decision variables listed in table I. DNN and $HWConf$ are the decision spaces of DNNs and hardware configurations, respectively, detailed in table I.

Each hardware configuration hc consists of 3 components: a Central Processing Unit (CPU), a Graphical Processing Unit (GPU) and an External Memory Controller (EMC). Each component has an independent and programmable clock frequency. Finally, F is the objective vector to optimize by minimizing the DNN error E (i.e., maximizing accuracy) on a user-defined $dataset$, DNN latency L and power consumption P on the hardware configuration hc.

We note that the $E(aka., Error)$ is measured by calculating the TOP-1 error rate, which is the percentage of images from $dataset$ for which the correct label is not the class label predicted by the DNN. $L(aka., Latency)$ is the execution time (in milliseconds) of dnn on the hardware accelerator HW_{ACC}. Finally, $P(aka., Powerconsumption)$ is the average power consumption (in milliwatt) observed when executing dnn on the hardware accelerator hc.

B. Optimization Methodology

To solve the above problem, we propose an evolutionary-based co-optimization strategy, where we search for both the optimal DNN architecture and hardware configuration. The search is done by exploring DNN and HW search spaces. Figure 4 details the proposed co-optimization approach. Our methodology includes three main components:

- **Joint search space:** We extend the search space of the HW-NAS by including the hardware configurations. Furthermore, by definition, the joint search space can be generalized to any DNN, task, dataset, and hardware accelerator. Thus, these four factors are considered as inputs in our co-optimization framework. In this paper, we use the joint search space detailed in table I. We note that all the considered decision variables are discrete.
- **Optimization algorithm:** We choose NSGA-II [21] as an evolutionary algorithm to explore the joint search space. NSGA-II is a widely used algorithm for NAS problems in general, and HW-NAS in particular [22]. Moreover, it typically provides a fast and efficient convergence by searching a wide range of solutions. These two abilities are due to its selection strategy based on non-dominated sorting and crowding distance, which allow for both convergence and diversity of solutions. In this paper, the parameters used for NSGA-II are detailed in table II. We first initialize the population using the Latin HyperCube Sampling (LHS) method. Then, next populations are generated from: 1) Selecting the best solutions using the non-dominated sorting algorithm of NSGA-II and 2) Applying

TABLE I
THE JOINT SEARCH SPACE OF DNN AND HARDWARE PARAMETERS

Decision variables	DNN search space					Hardware search space		
	Input resolution	Width	Depth	Kernel size	Expand ratio	CPU frequency	GPU frequency	Memory frequency
Values	[192, 288]	[16, 1984]	[1, 8]	[3, 5]	[1, 6]	[0.1, 2.3]	[0.1, 1.4]	[0.2, 2.1]
Cardinality	4	16	8	2	4	29	14	9

TABLE II
NSGA-II PARAMETERS

Parameter	Value
Number of generations	50
Population size	100
Population initialization	LHS
Mutation, probability	Uniform, 30%
Crossover, probability	Uniform, 80%

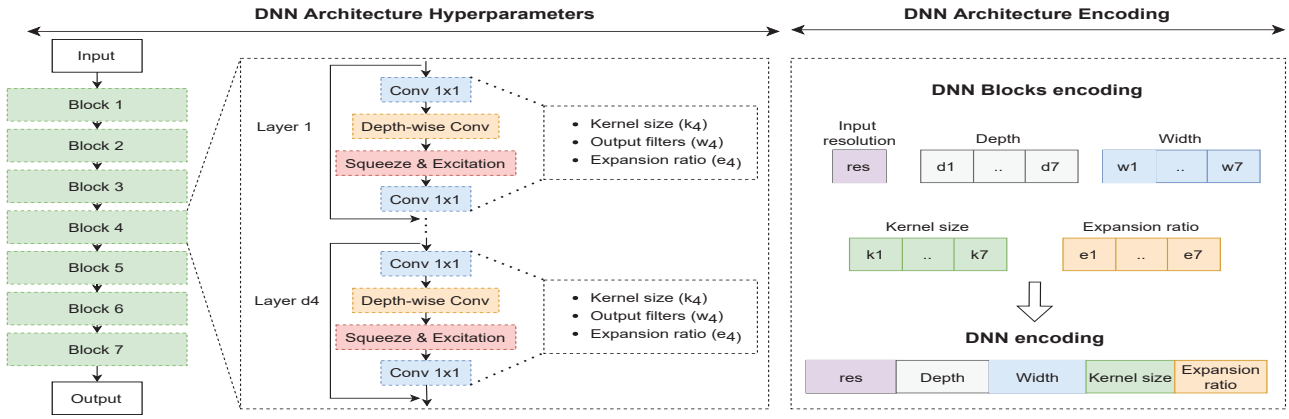

Fig. 5. **DNN search space encoding:** A candidate DNN architecture is real-encoded using a single vector that comprises five sub-vectors depicting: input resolution, depth, width, kernel size, and expansion ratio of each block

mutation and crossover on these best solutions to create the offspring population. We choose a high crossover probability of 80% to increase the reproducibility of good candidate solutions. However, we decrease the probability of mutation to 30% to prevent the risk of losing traces of good candidate solutions. Crossover and mutation are chosen uniformly.

- **Evaluation strategy:** The explored pairs are evaluated regarding the DNN accuracy and hardware efficiency. A) The DNN accuracy is evaluated in two stages: 1) We use a fast evaluation technique using an accuracy prediction model proposed in [19] to quickly determine the DNN accuracy during the exploration, then after the exploration, 2) We perform a more complete evaluation of the DNN accuracy for the elite solutions. We note that the results of the two evaluation techniques are highly correlated. B) The DNN hardware efficiency (latency and power) is directly measured by executing the DNN on the hardware accelerator under the specified configuration. As a reduced number of configurations is explored in the search algorithm, this step is not time consuming. Using a surrogate model for the hardware efficiency evaluation is considered as one possible extension of the work.

IV. EVALUATION

A. Experimental Setup

Our co-optimization problem has the following inputs:

- *DNN search space:* We use the same search space provided by [19], [23]. The search space contains 10^{11} DNN architectures, as detailed in table I and figure 5. The authors in [19] provide a prediction model for accuracy assessment, as a fast evaluation tool, and a pretrained supernet, as a complete evaluation tool. However, the second strategy is time-consuming as the sampled DNN needs to be calibrated on the entire training dataset (ImageNet). Thus, we used a fast evaluation strategy during the exploration then performed a complete evaluation using the pretrained supernet only for the elite solutions.
- *Dataset:* We choose to explore the joint search space for a state-of-art dataset such as ImageNet. Thus, the DNN accuracy are calculated on the ImageNet [24] validation dataset. All images are preprocessed using data augmentation techniques such as whitening, upsampling, random cropping, and random horizontal flipping, before feeding them to the DNN.
- *Hardware search space:* We choose the NVIDIA Jetson AGX Xavier GPU as as a hardware accelerator[2]. NVIDIA Jetson GPU accelerators allow the reconfigurability of different hardware parameters such as the number of operating cores. It also allows to have different operating clock frequency in the cores, GPU, and memory units. The chosen values of these parameters depend on the application requirements. For our case, we only vary the operating frequencies as detailed in table I

[2]Jetson AGX Xavier: https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit

B. Experimental Results

1) ***Co-optimization Results****:* In this section we will discuss the obtained results from two main perspectives:

- *Efficiency of the co-exploration:* To underline the co-exploration's importance, we compare the results obtained when co-exploring the joint search space and when performing a typical HW-NAS under fixed hardware configurations. We choose two default configurations proposed by the hardware manufacturer, NVIDIA in our case: MAXN and MINN. Figure 6 gives the results of the co-exploration, where figure 7 depicts a comparison between the results of the three approaches (i.e., joint, MAXN, and MINN), marked with different points shapes. Each point in the figures corresponds to a (dnn, hc) configuration.

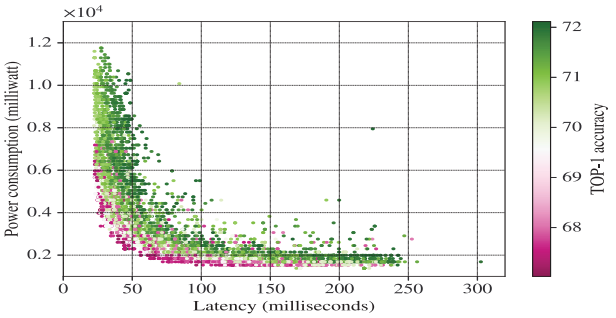

Fig. 6. Co-exploration results on the joint search space

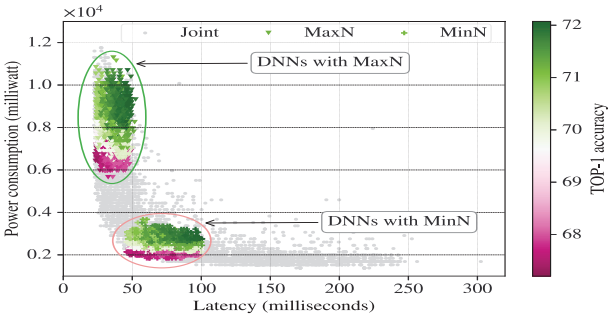

Fig. 7. Results of the three exploration approaches

After analyzing the two figures, we can clearly see that the region explored in the joint space is much larger than the regions explored when fixing the hardware configuration to MAXN or MINN. Furthermore, the explored regions when fixing the hardware configuration are included in the co-exploration. Indeed, the joint search space allows for exploring much larger solutions and hence different tradeoffs between DNN accuracy and hardware efficiency. The obtained hypervolume results presented in figure 8 confirm this observation. The obtained hypervolume from the co-exploration is larger than

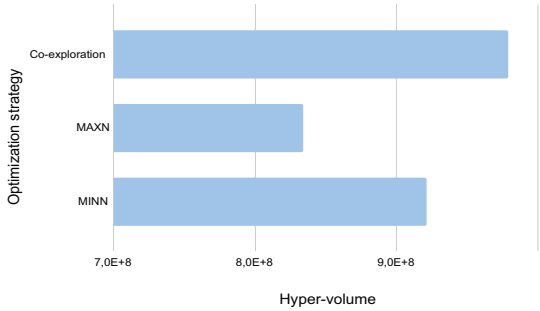

Fig. 8. Hypervolume results of the three optimization approaches

the hypervolumes of the HW-NAS under MAXN and MINN.

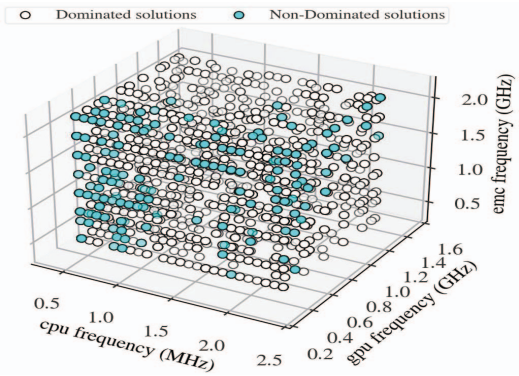

Fig. 9. Explored hardware configurations

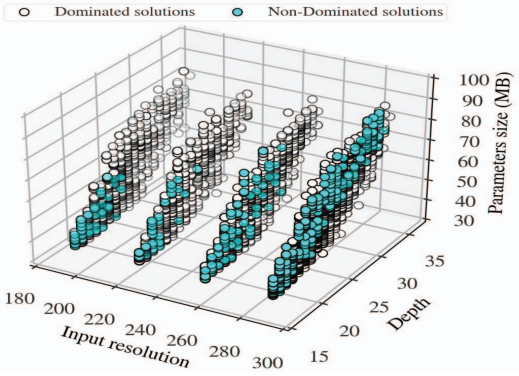

Fig. 10. Explored DNN models

Furthermore, we give figures 10 and 9 to show the diversity of the explored solutions. The white points correspond to all the explored solutions, whereas the solutions of the Pareto set are marked in blue. In figure 9, we show the explored hardware decision space. From this figure, we can observe that the Pareto optimal solutions are diverse and well distributed. This confirms our earlier

TABLE III
PERFORMANCE OF THE BASELINE MODELS PROPOSED BY ATTENTIVENAS [19] COMPARED TO OUR TOP SOLUTIONS FROM THE PARETO FRONT

DNN, hw_conf	TOP-1 Acc (%)	Latency (ms)	Power consumption (mw)
AttentiveNAS-A2, MAXN	78.8%	29.91	6575
AttentiveNAS-A3, MAXN	79.1%	33.51	6575
AttentiveNAS-A4, MAXN	79.8%	32.67	7033
AttentiveNAS-A5, MAXN	80.1%	35.66	6881
Ours-B0, hc0	**79.0%**	**28.85**	**4744**
Ours-B1, hc1	**79.6%**	**30.82**	**4591**
Ours-B2, hc2	**79.9%**	**33.03**	**4591**
Ours-B3, hc3	**80.2%**	**34.10**	**6118**

observation that no a priori knowledge can be used to choose the best-suited hardware configuration without actual exploration and evaluation. Figure 10 gives the characteristics of the explored DNN models in terms of input resolution, depth (i.e., number of layers), and size of trainable parameters (in Mega-Bytes). Similarly, the Pareto optimal solutions are well distributed and diverse. This also supports the importance of the exploration as we can assume a priori which DNN model will be Pareto optimal without actual evaluation of its performance.

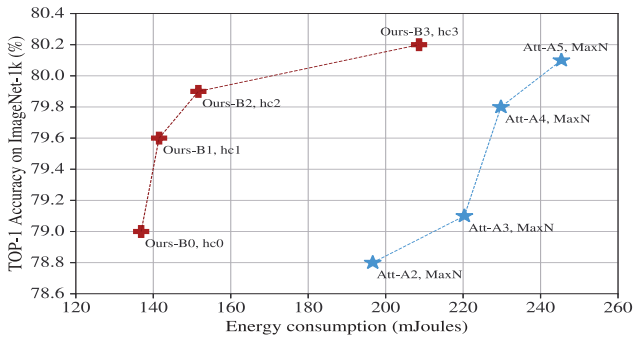

Fig. 11. ImageNet-1k accuracy v.s. average energy consumption of the top (DNN, hw-conf) pairs from the co-optimization Pareto front and the baseline configurations from AttentiveNas [19]

- *Optimality of the obtained results:* To further investigate the efficiency of the co-exploration, we select four pairs of (DNN, hardware configuration) from the Pareto front and compare them to state of the art DNN models under the widely used default configuration proposed by NVIDIA, MAXN. The four pairs of (DNN, hardware configuration) give different trade-offs between (accuracy, inference time and power consumption). Table III and figure 11 detail the obtained results. We also give a comparison with the top four (04) models from AttentiveNAS (A2-A5) [19]. We refer to these models to as *AttentiveNAS-A(2-5)*. Our obtained solutions (i.e., combinations of DNN models and hardware configuration) are referred to as *Ours-B(0-3), hc(0-3)*. Our co-optimization approach was able to identify better solutions in terms of accuracy and hardware efficiency. We can notice a power gains of up to 53% under the same latency constraints. Furthermore, we observe an accuracy improvement of up to 0.5% on the ImageNet dataset with a better hardware efficiency.

*2) **Fine-tuning on CIFAR10 and CIFAR100**:* To further investigate the performance of the obtained top models, we apply a fine-tuning to CIFAR10 and CIFAR100 datasets. Since the obtained models are sampled from the pretrained SuperNet of AttentiveNAS [19], we do not need to train them from scratch. Instead, we apply a transfer learning from ImageNet to the new datasets. We apply the same fine-tuning strategy proposed in [20] by taking the ImageNet pretrained models checkpoints and fine-tuning on the new datasets. Thus, the fine-tuning is applied to the overall model's parameters (not only the classifier). For training, we apply the same settings used to train the SuperNet in [19].

TABLE IV
FINE-TUNING RESULTS TO CIFAR-10 AND CIFAR-100

Models	ImageNet-1k	Cifar-100	Cifar-10
AttentiveNAS-A2[3]	78.8%	86.13%	97.45%
AttentiveNAS-A3[3]	79.1%	86.21%	97.61%
AttentiveNAS-A4[3]	79.8%	86.72%	97.71%
AttentiveNAS-A5[3]	80.1%	86.92%	97.83%
Ours-B0	**79.0%**	**86.12%**	**97.47%**
Ours-B1	**79.6%**	**86.69%**	**97.75%**
Ours-B2	**79.9%**	**86.95%**	**97.62%**
Ours-B3	**80.2%**	**87.03%**	**98.05%**

The models are trained using the SGD as an optimizer, with a weight decay of 1e-5 and momentum of 0.9. The learning rate is initialized at 0.1 and decays by 97% every three epochs. The training is done on eight distributed GPUs with a budget fixed to 100 epochs and a batch size of 128. Table IV summarizes the obtained results. We compare the obtained results with the baseline models of AttentiveNAS.

*3) **Neural Graph Post-Optimization with TensorRT**:* We further boost the hardware efficiency of the obtained models by post-optimizing the neural graph using TensorRT[4]. TensorRT is an SDK developed by NVIDIA to accelerate the deep learning inference. Figure 12 depicts the workflow to create an optimized inference engine using the TensorRT Builder module.

[3]In the original paper of AttentiveNas [19], the authors didn't report the results on CIFAR-10 and CIFAR-100. Therefore, we fine-tune the baselines models on the new datasets using our implementation

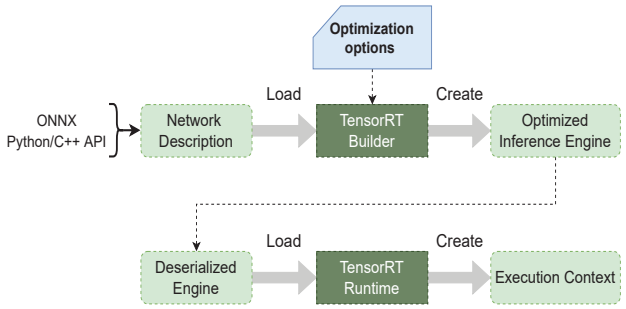

Fig. 12. Workflow of generating an optimized TensorRT inference engine

First, the network description and other optimization options such as kernel/data quantization and computation mapping strategy should be provided as input. Then, TensorRT creates an optimized engine by applying the specified optimization options and other internal optimization strategies such as layer fusion. The optimized engine creates an execution context to perform the inference. Finally, an execution context is mapped into a single (or multiple) CUDA stream(s) for execution. We optimize the obtained top models (i.e., Ours-B(0-3), hc(0-3)) by exploring the aforementioned TensorRT optimization parameters. We report the results of this post-optimization in figures 13 and 14.

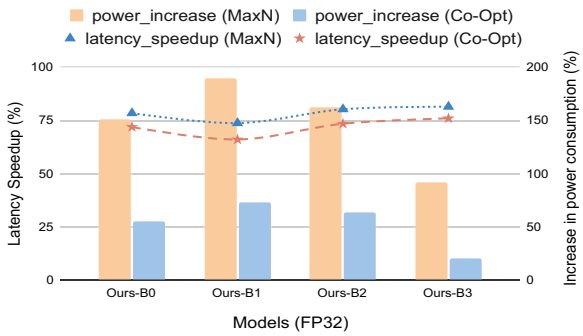

Fig. 13. Graph Neural Post-optimization results using TensorRT without post-quantization

These two figures depict latency and power consumption during inference of the obtained optimized models. We report the speedup in latency by calculating the ratio of the optimized models with TensorRT to the non-optimized models before applying TensorRt. For a better interpretation, the results are reported in percentage (%). Regarding power consumption, TensorRT incurs higher power consumption compared to the non-optimized implementation. We compare the performance when choosing MaxN as hardware configuration (referred to as *MaxN* in the figures) and when choosing the optimal hardware configuration found by our co-optimization approach (referred to as *Co-Opt* in the figures). In figure 13, we only apply

[4]NVIDIA TensorRT: https://developer.nvidia.com/tensorrt/

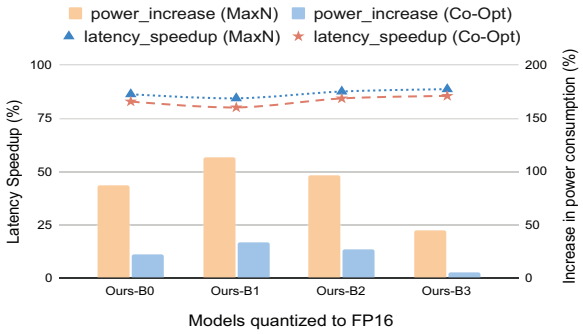

Fig. 14. Graph Neural Post-optimization results using TensorRT with FP16-bits post-quantization

the internal optimization of TensorRT, whereas in figure 14, we apply a post-quantization to FP16. First, when executing our models with the MaxN configuration, TensorRT boosts up the hardware efficiency of the models with a latency speedup up to 81% and 88% for non-quantized and quantized models, respectively. However, this comes at the expense of an increased power consumption up to 189% and 112% for non-quantized and quantized models, respectively. This is explained by the ability of TensorRT to take advantage of the GPU computation units during the inference. The optimized engine is highly parallel, which maximizes the GPU utilization ratio and thus incurs an increase in power consumption.

Second, by optimizing the hardware configuration (Co-Opt)), we were able to reduce the increase in power consumption from 189% and 112% to 73% and 33% for non-quantized and quantized models, respectively. Regarding latency, we notice a small average gap of 7% in the latency speedups of the MaxN and the optimal hardware configurations found by our algorithm. Nevertheless, since the two hardware configurations (i.e., MaxN and optimal configurations (hc-(0-3)) speed up the computations latency by a ratio of 85% to 88%, this small gap becomes negligible when considering the power saving from of up to 61% achieved by the optimal hardware configuration found by our co-optimization approach. Therefore, to balance latency speedup and power saving, co-optimization remains important even when using high-performance SDK such as TensorRT.

*4) **On the importance of the co-optimization**:* Our results demonstrated that the co-optimization of both DNN and hardware configurations on Edge GPUs is extremely important to balance both DNN performance and hardware efficiency. On the one hand, edge GPUs are heterogeneous embedded systems and can execute multiple applications concurrently. In this case, the availability of hardware resources (e.g., computation units and operating frequencies) varies due to other concurrent applications' execution. Thus, the performance of an application changes under different resource availability scenarios. Therefore, knowing beforehand which DNN model

is adapted to each scenario can prevent the violation of the application's performance requirements and meet the power and resources budget constraints. On the other hand, the optimal hardware configurations for a single application can be used at runtime to dynamically scale the hardware platform for the performance requirements and the power budget constraints. Therefore, our proposed co-optimization framework can be integrated into a real-time scheduling algorithm to dynamically adjust DNN and hardware configurations according to resource availability. It can also be used as a dynamic frequency scaling governor at runtime instead of the currently used heuristics that have been proved to be weak in capturing the correlation between the hardware configuration, the application workload and the availability of hardware resources.

V. Conclusion

In this paper, we investigated the importance of the joint exploration of DNN and hardware configurations for edge GPU accelerators. We propose a co-optimization approach based on an evolutionary algorithm (NSGA-II) to explore these two search search spaces. The aim was to minimize three objective functions: DNN TOP-1 error, latency and power consumption. Experimental results on the Jetson AGX Xavier NVIDIA GPU demonstrated the efficiency of the co-optimization compared to typical HW-NAS under fixed hardware configurations. Moreover, the top pairs found by our co-optimization are more energy-efficient with up to 53% gains than solutions found by state-of-the-art models under the same accuracy and latency constraints. We have also demonstrated the importance of the co-optimization when using a high performance SDK such as TensorRT. The results depict a latency speedup up to 81% and power saving of 61% compared to the MaxN configuration. As future work, we plan to enhance our co-optimization strategy by proposing more efficient selection and recombination operators for the optimization algorithm. We also aim to investigate more hardware configurations and DNN benchmarks to showcase the importance of co-optimization.

VI. Acknowledgement

This work is supported by Institut Carnot ARTS

References

[1] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "A comprehensive survey on hardware-aware neural architecture search," *CoRR*, vol. abs/2101.09336, 2021. [Online]. Available: https://arxiv.org/abs/2101.09336

[2] S. M. Nabavinejad, S. Reda, and M. Ebrahimi, "Coordinated batching and DVFS for DNN inference on gpu accelerators," *IEEE Transactions on Parallel and Distributed Systems*, 2022.

[3] L. Yang, Z. Yan, M. Li, H. Kwon, L. Lai, T. Krishna, V. Chandra, W. Jiang, and Y. Shi, "Co-exploration of neural architectures and heterogeneous ASIC accelerator designs targeting multiple tasks," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[4] W. Jiang, L. Yang, E. H.-M. Sha, Q. Zhuge, S. Gu, S. Dasgupta, Y. Shi, and J. Hu, "Hardware/software co-exploration of neural architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4805–4815, 2020.

[5] Y. Lin, D. Hafdi, K. Wang, Z. Liu, and S. Han, "Neural-hardware architecture search," *NeurIPS WS*, 2019.

[6] Y. Li, C. Hao, X. Zhang, X. Liu, Y. Chen, J. Xiong, W.-m. Hwu, and D. Chen, "Edd: Efficient differentiable dnn architecture and implementation co-search for embedded ai solutions," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[7] W. Jiang, L. Yang, S. Dasgupta, J. Hu, and Y. Shi, "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*, vol. 39, no. 11, pp. 4154–4165, 2020.

[8] W. Chen, Y. Wang, S. Yang, C. Liu, and L. Zhang, "You only search once: A fast automation framework for single-stage DNN/accelerator co-design," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 1283–1286.

[9] K. Choi, D. Hong, H. Yoon, J. Yu, Y. Kim, and J. Lee, "Dance: Differentiable accelerator/network co-exploration," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 337–342.

[10] Y. Liang, L. Lu, Y. Jin, J. Xie, R. Huang, J. Zhang, and W. Lin, "An efficient hardware design for accelerating sparse CNNs with NAS-based models," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[11] Y. Zhou, X. Dong, B. Akin, M. Tan, D. Peng, T. Meng, A. Yazdanbakhsh, D. Huang, R. Narayanaswami, and J. Laudon, "Rethinking co-design of neural architectures and hardware accelerators," *arXiv preprint arXiv:2102.08619*, 2021.

[12] M. Pinos, V. Mrazek, and L. Sekanina, "Evolutionary neural architecture search supporting approximate multipliers," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2021, pp. 82–97.

[13] L. Sekanina, "Neural architecture search and hardware accelerator co-search: A survey," *IEEE Access*, vol. 9, pp. 151 337–151 362, 2021.

[14] Q. Lu, W. Jiang, X. Xu, Y. Shi, and J. Hu, "On neural architecture search for resource-constrained hardware platforms," *arXiv preprint arXiv:1911.00105*, 2019.

[15] M. S. Abdelfattah, Ł. Dudziak, T. Chau, R. Lee, H. Kim, and N. D. Lane, "Best of both worlds: Automl codesign of a cnn and its hardware accelerator," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

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

[17] O. Spantidi, I. Galanis, and I. Anagnostopoulos, "Frequency-based power efficiency improvement of cnns on heterogeneous iot computing systems," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. IEEE, 2020, pp. 1–6.

[18] S. Liu and A. Karanth, "Dynamic voltage and frequency scaling to improve energy-efficiency of hardware accelerators," in *2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC)*. IEEE, 2021, pp. 232–241.

[19] D. Wang, M. Li, C. Gong, and V. Chandra, "Attentivenas: Improving neural architecture search via attentive sampling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[20] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

[21] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *International conference on parallel problem solving from nature*. Springer, 2000.

[22] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE transactions on neural networks and learning systems*, 2021.

[23] D. Wang, C. Gong, M. Li, Q. Liu, and V. Chandra, "Alphanet: Improved training of supernets with alpha-divergence," in *International Conference on Machine Learning*. PMLR, 2021.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.