DACO: Pursuing Ultra-low Power Consumption via DNN-Adaptive CPU-GPU CO-optimization on Mobile Devices

Yushu Wu[1], Chao Wu[1], Geng Yuan[2], Yanyu Li[1], Weichao Guo[3], Jing Rao[4], Xipeng Shen[5], Bin Ren[6], Yanzhi Wang[1]
Northeastern Univerisity[1], *University of Georgia*[2], *Oppo Mobile Communications Co, Ltd.*[3],
University of New South Wales[4], *North Carolina State University*[5], *William & Mary*[6]
{wu.yushu,cha.wu,li.yanyu,yanz.wang}@northeastern.edu

***Abstract*—As Deep Neural Networks (DNNs) become popular in mobile systems, their high computational and memory demands make them major power consumers, especially in limited-budget scenarios. In this paper, we propose DACO, a DNN-Adaptive CPU-GPU CO-optimization technique, to reduce the power consumption of DNNs. First, a resource-oriented classifier is proposed to quantify the computation/memory intensity of DNN models and classify them accordingly. Second, a set of rule-based policies is deduced for achieving the best-suited CPU-GPU system configuration in a coarse-grained manner. Combined with all the rules, a coarse-to-fine CPU-GPU auto-tuning approach is proposed to reach the Pareto-optimal speed and power consumption in DNN inference. Experimental results demonstrate that, compared with the existing approach, DACO could reduce power consumption by up to 71.9% while keeping an excellent DNN inference speed.**

***Index Terms*—DVFS, Heterogeneous, DNN, Power management**

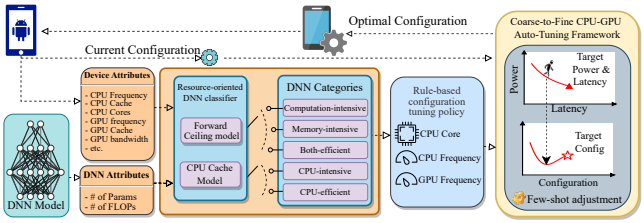
I. Introduction

The ubiquitous Deep Neural Networks (DNNs) are posing huge challenges on mobile devices with constrained computation/power budgets [5]. Different DNN applications might have particular demands. For example, some require real-time system responsiveness, while others prefer a lower power consumption to preserve a longer device standby time. DNN applications may also have different patterns representing each stage of their inference (e.g., input pre-processing, feed-forward computation, and output post-processing). For example, ResNet [2] is computation-intensive due to its fewer weights/intermediate results, and DNN-based super-resolution [7] is memory-intensive because the amount of data in the pre/post-processing stage required high I/O performance.

To cater to the diverse demands of DNN applications, prior methods have been proposed based on the conventional dynamic voltage and frequency scaling (DVFS) technique. However, one observation from our study shows that DVFS is not the only contributor to system optimization in terms of the power budget. The number of activated CPU cores could also determine the device's power consumption besides the DVFS scheduler, yielding a huge amount of configuration combinations for heterogeneous CPU-GPU systems. Thus, manually searching for the best-suited DNN-adaptive configuration for modern CPU-GPU systems is expensive.

For the diverse DNN patterns, prior wisdom classifies the DNN applications into computation- and memory-bound with a roofline model.However, the current roofline model or its extension [3] does not consider the role of CPU in DNN inference, the frequent data reuse of DNN weights, and the patterns of each DNN stage, thus failing to get the best-suited system configuration in terms of power and speed.

Fig. 1: The architecture of DACO.



Therefore, we propose DACO, a DNN-Adaptive CPU-GPU CO-optimization framework, to address this issue. Based on the study, we make the following contributions. 1) We propose a resource-oriented DNN classifier to reveal the relationship between mobile hardware capacity and the patterns of various stages in DNN inference. 2) We propose a set of rule-based policies to guide the designers in approaching the DNN-adaptive CPU-GPU configuration. 3) We propose a Coarse-to-Fine CPU-GPU Auto-tuning framework, which automatically selects configurations based on DNN architecture and mobile hardware resources. 4) Experimental results demonstrate that our proposed approach can reduce the power consumption of DNNs on mobile devices by up to 71.9% compared with existing works while keeping the DNN inference speed.

II. Methodology

Figure 1 describes the architecture of DACO, which mainly consists of three parts. 1) Resources-aware DNN Classifier; 2) Rule-based Configuration Tuning Policy; 3) Coarse-to-fine CPU-GPU Auto-Tuning Framework.

A. Resource-oriented DNN Classifier

Accurately classifying DNN models helps reveal the relationship between hardware resources and DNNs. However, the current roofline model is less effective due to the neglect of the CPU, the patterns of DNN stages, and the data reusability in on-chip caches. For this reason, **forward ceiling model** and **CPU-cache model** are introduced.

979-8-3503-4859-0/DATE24/© 2024 EDAA

The forward ceiling model is deduced by scaling the original roofline model with the cache usage during the feed-forward calculation stage. Hence, the forward ceiling model reflects the relationship of the cache re-usability and computation-memory bound of a DNN model. This model classifies a DNN model as **computation-intensive**, **memory-intensive**, or **both-efficient**.

The CPU-cache model classifies DNN models in the pre/post-processing stage. The model compares the waiting time of CPU cores and the memory system to identify if a DNN model is CPU-bound. This model can classify the DNN model as **CPU-intensive** or **CPU-efficient**.

B. Rule-based Configuration Tuning Policy

The classification results of the resource-oriented DNN classifier provide many hints. The optimization objective could be either speed- or power-oriented. The speed-oriented tuning rule is shown as an example. The power-oriented rule can be obtained by flipping the speed-oriented rule. The reason is the contribution of each component (e.g., CPU cores, CPU frequency, GPU frequency) to power and speed is naturally consistent under different optimization goals, and the order of tuning is also reversed for the two opposite goals. **Rule 1:** Computation-intensive and both-efficient models follow the order of GPU frequency, CPU frequency, CPU core. **Rule 2:** Memory-intensive models follow the order of GPU frequency, CPU core, CPU frequency. **Rule 3:** CPU-intensive models should follow the order of CPU core, CPU frequency. **Rule 4:** CPU–efficient models should follow the order of CPU frequency, CPU core. **Rule 5:** A DNN model could both fit Rule 1-2 and 3-4. For speed-oriented tuning, we obey Rule 1-2 first. For power-oriented tuning, we obey Rule 3-4 first.

C. Coarse-to-Fine CPU-GPU Auto-Tuning

Rule-based policies can only provide hints for coarse-grained optimization. A coarse-to-fine CPU-GPU auto-tuning method is proposed using the DNN-based performance estimator for the latency and power consumption of the corresponding configuration. According to Sec. II-B, different DNN categories have different characteristics. Thus, `each DNN category will have its own performance estimator.` Each pre-trained estimator can initially provide a coarse estimation for the performance. With a few shots of fine-tuning, the pre-trained estimator can adapt to new devices.

III. Evaluation

TABLE I: Comparison with SOTA methods. T is DNN inference latency and ΔP is the dynamic power consumption.

DNN	Method	core_{CPU}	f_{CPU}(GHz)	f_{GPU}(MHz)	T(ms)	ΔP(W)
ResNet50	DACO-p	$4\times CPU_{little}$	1.80	587	37.91	1.59
	DACO-s	$4\times CPU_{little}$	1.25	305	64.24	0.22
	Schedutil	default	default	default	37.88	4.32
	zTT	default	default	default	43.17	2.41
	CDC	default	default	default	48.31	1.15
WDSR	DACO-p	$3\times CPU_{mid}$	2.05	587	36.04	2.83
	DACO-s	$4\times CPU_{little}$	0.79	305	71.50	0.22
	Schedutil	default	default	default	35.89	4.81
	zTT	default	default	default	44.12	1.70
	CDC	default	default	default	48.82	1.53
EfficientNet-b0	DACO-p	$4\times CPU_{little}$	1.80	587	14.37	1.02
	DACO-s	$4\times CPU_{little}$	0.78	305	28.97	0.05
	Schedutil	default	default	default	14.34	3.63
	zTT	default	default	default	16.09	0.95
	CDC	default	default	default	15.95	0.76

default: following their default configuration for each method.

A. Experimental Setup

We apply DACO on the OnePlus 8T to verify the performance of DNN inference and its power consumption. The $\Delta P = P_{avg} - P_{idle}$ is taken as the power consumption for the DNN inference, where the P_{avg} is the average power during DNN inference and P_{idle} is the power of the idle state. ResNet50 [2], EfficientNet-b0 [6], and WDSR [7] are chosen to represent the scalability of DACO in evaluation. We compare DACO with **Schedutil**, **zTT**, and **CDC**. **Schedutil** is a classic DVFS governor and the default governor in our experimental device. **zTT** [4] and **CDC** [1] adopt deep reinforcement-learning (DRL) to optimize power consumption via DVFS configurations.

B. Experiment Results and Analysis

Table I shows the experimental results of various approaches. For better comparison, we define three searching policies in the searched Pareto-optimal curve of DACO. `DACO-p` means speed-oriented searching and `DACO-s` means power-oriented searching. DACO shows a superior Pareto-optimal speed and power consumption compared with SOTA works. Compared with DNN-oriented approaches, e.g., `CDC`, DACO delivers similar or even faster inference speed (-0.2%, 5.2%, 1.3% faster in ResNet50, WDSR, EfficientNet-b0) with less power consumption (14.5%, 11.8%, 1.0% lower in ResNet50, WDSR, EfficientNet-b0), due to pre/post-processing stages also play important roles during DNN inference, which is ignored in `CDC`. Compared with approaches proposed for other applications, DACO represents an even better power efficiency for DNNs. For example, DACO outperforms `zTT` in both faster inference speed (4.3%, 2.3%, 1.9% faster in ResNet50, WDSR, EfficientNet-b0) and lower power consumption (51.5%, 14.8%, 21.1% lower in ResNet50, WDSR, EfficientNet-b0). Compared with `Schedutil`, DACO-p could reduce the power consumption by 63.2% for ResNet50, 41.2% for WDSR, 71.9% for EfficientNet-b0 with inference speed difference less than 1%. Since DNN inference has different workload preferences compared to ordinary tasks, these ordinary tasks-oriented approaches fail to adapt to various DNN patterns.

In summary, DACO could automatically search for Pareto-frontiers for each DNN category and configure the CPU-GPU system accordingly, thus delivering better power efficiency with excellent speed over existing works.

References

[1] M. Han et al., "HERTI: A Reinforcement Learning-Augmented System for Efficient Real-Time Inference on Heterogeneous Embedded Systems," in PACT. IEEE, 2021, pp. 90–102.

[2] K. He et al., "Deep residual learning for image recognition," in CVPR, 2016, pp. 770–778.

[3] M. Hill and V. Janapa Reddi, "Gables: A Roofline Model for Mobile SoCs," in HPCA, 2019, pp. 317–330.

[4] S. Kim et al., "zTT: Learning-Based DVFS with Zero Thermal Throttling for Mobile Devices," in MobiSys, 2021, p. 41–53.

[5] A. Sampson et al., "EnerJ: Approximate Data Types for Safe and General Low-Power Computation," in Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation. Association for Computing Machinery, 2011, p. 164–174.

[6] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in ICML, 2019, pp. 6105–6114.

[7] J. Yu et al., "Wide activation for efficient and accurate image super-resolution," arXiv preprint arXiv:1808.08718, 2018.