Poster Abstract: Aaron: Compile-time Kernel Adaptation for Multi-DNN Inference Acceleration on Edge GPU

Zhihe Zhao[1], Neiwen Ling[1], Nan Guan[2] and Guoliang Xing[1,*]
[1]The Chinese University of Hong Kong, Hong Kong SAR, China
[2]City University of Hong Kong, Hong Kong SAR, China

ABSTRACT

AI applications powered by deep learning are increasingly running on edge devices. Meanwhile, many real-world IoT applications demand multiple real-time tasks to run on the same device, for example, to achieve both object tracking and image segmentation simultaneously on an augmented reality glass. However, the current solutions can not yet support such multi-tenant real-time DNN inference on edge devices. Techniques such as on-device model compression trade inference accuracy for speed, while traditional DNN compilers mainly focus on single-tenant DNN model optimization. To fill this gap, we propose Aaron, which leverages DNN compiling techniques to accelerate multi-DNN inference on edge GPU based on compile-time kernel adaptation with no accuracy loss. Aaron integrates both DNN graph and kernel optimization to maximize on-device parallelism and minimize contention brought by concurrent inference.

CCS CONCEPTS

• **Computer systems organization** → *Real-time System*;

KEYWORDS

Efficient DNN Processing, DNN Compiler, Real-time System

1 INTRODUCTION

Recently, efficient execution of multiple DNNs on a single device has gained significant interests, which requires the device to process several DNN based tasks concurrently at run-time. These tasks run in a concurrent manner which poses several challenges for **efficient on-device multi-DNN inference**. Multiple deep learning tasks have to share limited on-board resources including memory bandwidth, caches and processing elements.

To address resource contention among multiple DNN execution, recent works such as [3] exploit *model compressibility* to perform latency-accuracy trade-offs among multiple DNN models. As a result, the model with less importance may sacrifice its own accuracy due to the contention from a more important model execution. Although it is a promising solution to resource contention, the model compression ratio is highly model-dependent. One needs to search for the optimal model compression methods for emerging DNN models. What's more, this kind of approach will also lead to substantial accuracy loss. Another class of emerging solutions to support high-performance on-device multi-DNN inference is *runtime scheduling*. Multi-DNN with multiple parallel Directed Acyclic Graphs(DAG)' co-locations leads to extensive inter-operator contention, which requires careful operator scheduling among DNN tasks. Works such as [4] provide either learning-based or heuristic-based search algorithms to generate pipelines for multi-DNN inference. The key idea behind them is to re-schedule the DNN kernels execution order to avoid the potential resource contention.

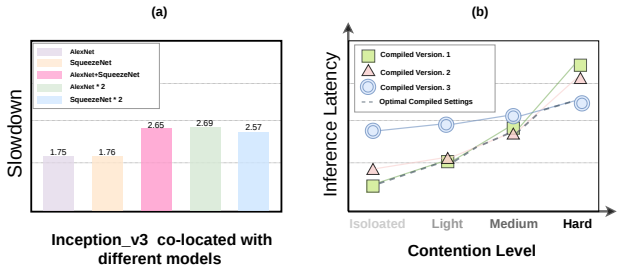

Figure 1: (a) Slowdown of Inception_v3 during co-running against isolation inference. (b) Performance of three compiled versions under different contention levels. The grey dotted line shows the optimal performance traces.

Our work leverages DNN compiling techniques that incorporate both inter-kernel and intra-kernel optimizations to mitigate the contention brought by concurrent kernel executions in a fine-grained manner. Traditional deep learning compilers such as [2, 5] usually accelerate DNN models by generating high-performance DNN kernels and transformed DNN graphs. However, they mainly target optimizing one model under static environments. **In this work, we propose Aaron, a compile-time DNN kernel adaptation framework to provide elastic inference schemes during concurrent inference on edge GPU with no accuracy loss.**

2 MOTIVATION STUDY

Our work focuses on edge GPU. We use NVIDIA GTX 2060, which consists of 34 streaming multiprocessors (SMs)Fig. 1a shows the slowdown for Inception_v3 when co-running with different background models. The slowdown is defined by the ratio of inference latency with or without co-runners.

We observe that variations occur among co-running settings, which is caused by different "contention channels"—shared resources such as L1 memory, DRAM, and compute units within the GPU. In another word, co-located "compute-bound" or "memory-bound" DNN kernels affect each other's inference speed. Without

*Corresponding email: glxing@cuhk.edu.hk.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SenSys '22, November 6–9, 2022, Boston, MA, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9886-2/22/11…$15.00
https://doi.org/10.1145/3560905.3568050

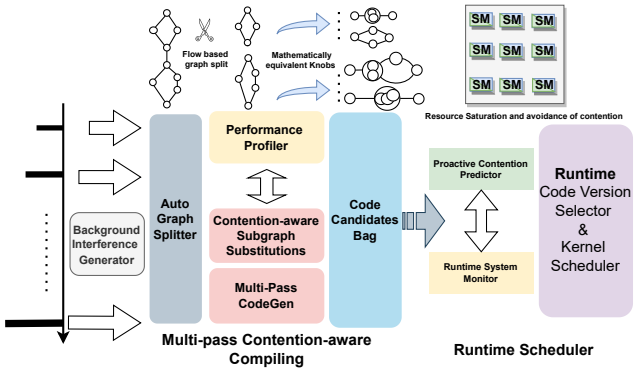

Figure 2: Aaron System Design.

carefully scheduling, co-execution kernels with the same contention channel can easily lead to resource contention. This motivates us to carefully design an online kernel scheduler to wisely "bin-pack" different concurrent kernel groups to avoid executing overlapped kernels with the same contention channel.

To mitigate inevitable resource contention among co-running models, we extend IOS [1], a DNN graph compiler that generates different concurrent strategies for different stages in a DNN graph with branches (e.g. Inception_v3). We generate three compiled versions of Inception_v3 based on IOS under four levels of contention pressure: isolation; light; medium and heavy kernels. Fig. 1b compares the performance across these three implementations under four contention levels and produces an optimal performance trace (the grey dotted line). The key observation here is that the optimal compiled graph structures are not static when the background co-locating DNN tasks are different.

These findings motivate our design of Aaron to profile and deal with the hardware resources contention when DNN tasks co-executing on edge GPU.

3 PROBLEM FORMULATION AND SYSTEM DESIGN

The objective of Aaron is to minimize the run-time contention caused by overlapped kernels while maximizing the on-device parallelism to fulfill resource utilization and maximize the overall DNN tasks throughput. The objective function can be formulated as:

$$\begin{aligned} &max[P(s_1) + P(s_2) + ...P(s_n)] \\ &min[C(s_1) + C(s_1) + ...C(s_n)] \end{aligned} \tag{1}$$

where P refers to parallelism on GPU; s_n represents the run-time stages. A stage contains operators from different concurrent DNN sequences streams; C is the measured contention metric. Fig. 2 shows our system design, the main components inside Aaron can be divided into offline and online phases.

Offline Phase: We design a background contention generator based on our profiling results of resources contention on edge GPU to guide the adaptive compiling process. We also propose an adaptive contention-aware kernel generator, which is to generate the adaptive compiled kernel bags with different versions, corresponding to different levels of contention.

Online Phase: We propose a lightweight predictor to predict the upcoming contention for different adaptive kernels combinations from different DNN sequences. We also design an online adaptive kernels scheduler, aimed to coordinate both the kernel version selections and the concurrency work groups from different DNN sequences. The scheduler is intermittently activated.

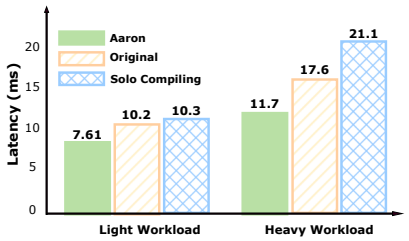

Figure 3: Preliminary Results of Aaron.

4 PRELIMINARY RESULTS

We evaluate the performance of the contention-aware compiling module in Aaron by comparing it against the baseline *solo compiling* and *original*. *Solo compiling* compiles an Inception_v3 model with no other co-located DNN models. *Original* refers to the inference of the original DNN model with no acceleration from the compiling aspect. We use the latency of each model to quantify how well the performance is improved. We select *MobileNet_v3_small* as the light background workload and *MobileNet_v3_small* co-located with *ShuffleNet* as the heavy background workload. As shown in Fig. 3, Aaron exhibits much higher performance: 1.35x and 1.8x over *solo compiling* and 1.34x and 1.5x over *original* under two background workloads. This is because our system can capture on-device contention behaviors including memory bandwidth and computation unit sharing, and provide adaptive kernels at run-time.

5 CONCLUSION AND FUTURE WORK

We present Aaron, a multi-DNN inference accelerator. Our work mainly attempts to mitigate the contention incurred by concurrent DNN inference on edge GPU by leveraging adaptive compiling and online scheduling, thus accelerating DNN tasks and increasing the overall throughput. In the future, we will extend this work by analyzing memory and compute resource contention on edge GPU and providing detailed verification on the online scheduler.

ACKNOWLEDGEMENT

The work described in this article was partially supported by the Research Grants Council of Hong Kong under Grant No. 14203420, and by the Centre for Perceptual and Interactive Intelligence (CPII) Ltd under the Innovation and Technology Fund.

REFERENCES

[1] Yaoyao et al. Ding. 2021. Ios: Inter-operator scheduler for cnn acceleration. *Proceedings of Machine Learning and Systems* 3 (2021), 167–180.
[2] Tianqi Chen et al. 2018. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. USENIX Association, Carlsbad, CA, 578–594.
[3] Neiwen Ling, Kai Wang, Yuze He, Guoliang Xing, and Daqi Xie. 2021. RT-mDL: Supporting Real-Time Mixed Deep Learning Tasks on Edge Platforms. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. 1–14.
[4] Fuxun et al. Yu. 2021. Automated Runtime-Aware Scheduling for Multi-Tenant DNN Inference on GPU. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–9.
[5] Zhihe Zhao, Xian Shuai, Yang Bai, Neiwen Ling, Nan Guan, Zhenyu Yan, and Guoliang Xing. 2022. Moses: Efficient Exploitation of Cross-device Transferable Features for Tensor Program Optimization. *arXiv preprint arXiv:2201.05752* (2022).