Nokia Bell Labs Cambridge Interview

Zhihe (Kyrie) Zhao

Y3 Ph.D Candidate

Advisor: Prof. Guoliang Xing

AIoT Lab, Chinese University of Hong Kong

Dec, 2023

Zhihe (Kyrie) Zhao

Sys4AI | DNN Compiler | AIoT

- SenSys'22 Best Poster Award
- SenSys'22 Best Paper Finalist
- Huawei Spark Award, 2022
- BOSCH AIoT Fellowship, 2021
- Co-Founder of two high-tech start-ups, raised over 2M USD

UoL CST BS'19

- Conducted Sensing & Edge AI Research since 2017
- Published a poster AT SenSys'18; a demo AT IoTDI'18; a journal paper AT SENSORS
- Established a smart home company, raised 1M USD

Duke ECE MS'21

- Conducted research on Multi-user Mobile Augmented Reality
- Quit Ph.D with MS

CUHK IE PhD'24

Trying to Bridge the Gap Between HPC and AIoT

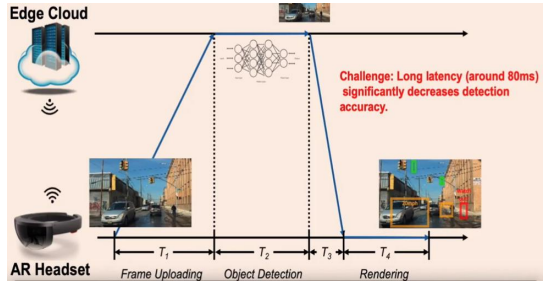
"Bringing DNN Compiler to AIoT"

[Edge–cloud CollabAI] EdgeML: An AutoML Framework for Real–Time Deep Learning on the Edge

The 6th ACM/IEEE International Conference on Internet–of–Things Design and Implementation (IoTDI 2021)

Zhihe Zhao[1,2], Kai Wang[1], Neiwen Ling[1], and Guoliang Xing[1]
[1]The Chinese University of Hong Kong
[2]Duke University

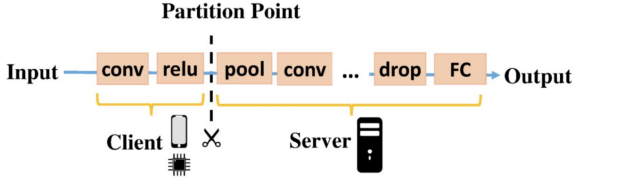
Background



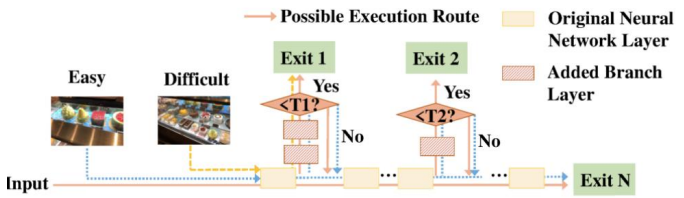
(1)

- Deploying DNN on edge devices
- Edge offloading is a trend

Optimization Knobs

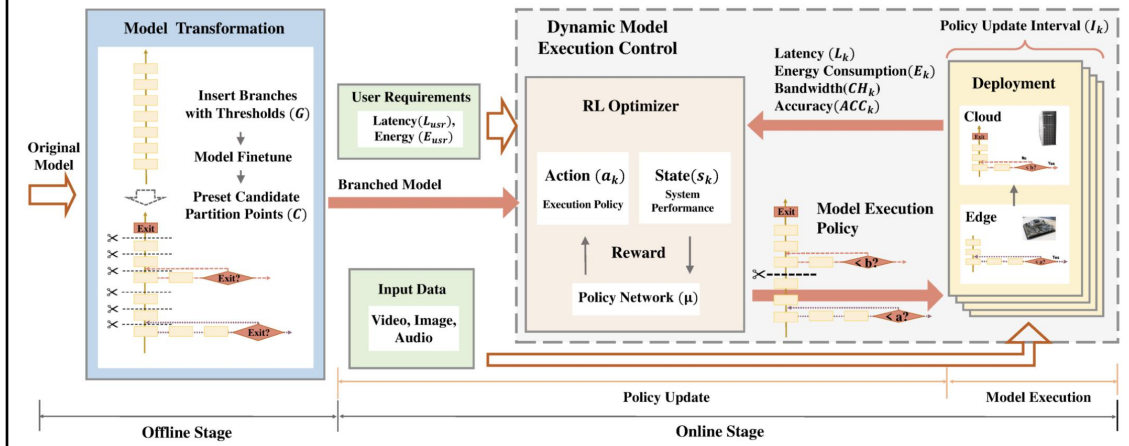


DNN Model Partition



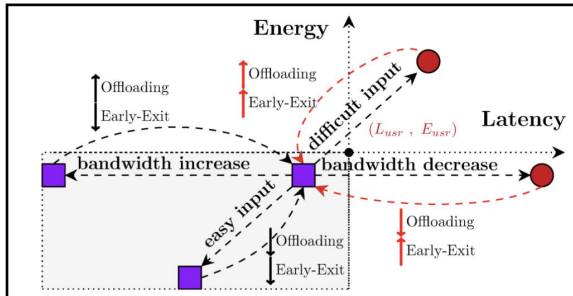
Progressive NN Architecture

System Architecture

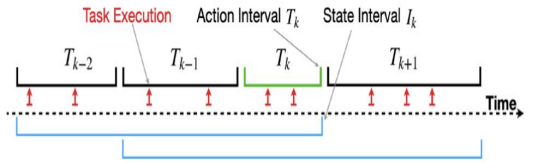


- Key challenge is to adapt the runtime dynamics (inputs, network bandwidth)
- EdgeML uses a RL-based automl framework to collaboratively achieve the optimial runtime settings model partition and the progressive nerual architecture

Design Details



- Illustration of how EdgeML adapts to environments
- Changes of inputs and bandwidth will cause system transitions



- Illustration of states of RL optimizer
- Preventing the optimizer from frequently changing the execution policy back and forth

Conclusion

- An AutoML framework that accelerates deep learning tasks on edge devices
- Adaptive environment-aware network structures

Reference

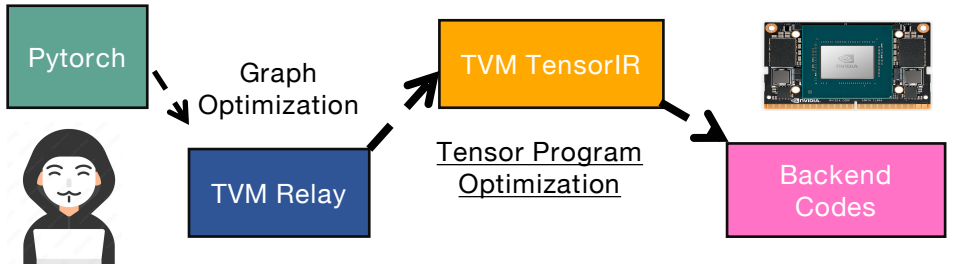
[1] Liu, Luyang, Hongyu Li, and Marco Gruteser. "Edge assisted real-time object detection for mobile augmented reality." The 25th annual international conference on mobile computing and networking. 2019.

[NNCompiler Opt] Moses: Exploiting Cross-device Transferable Features for On-device Tensor Program Optimization

The 24th International Workshop on Mobile Computing Systems and Applications (HotMobile 2023)

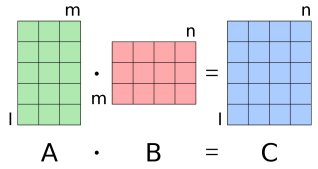
Zhihe Zhao[1], Xian Shuai[1], Neiwen Ling[1], Nan Guan[2], Zhenyu Yan[1], Guoliang Xing[1]
[1]The Chinese University of Hong Kong
[2]City University of Hong Kong

Background of DNN Compiler (tvm as example)

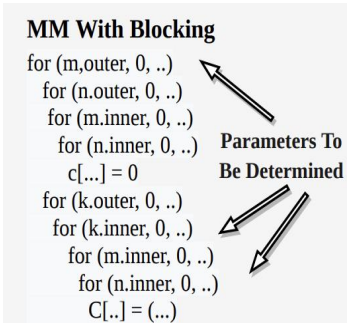


- Deploying diverse NN models on heterogenous devices
- Search-based tuning process for generating high performance kernels

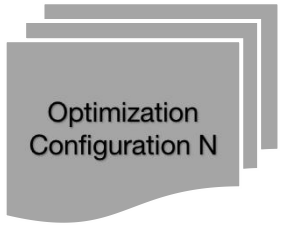
Motivation



Example: Efficient Matrix Multiplication



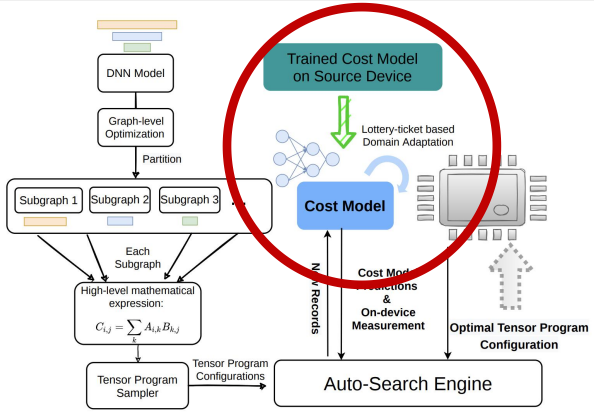
For-loop Level Tensor Program

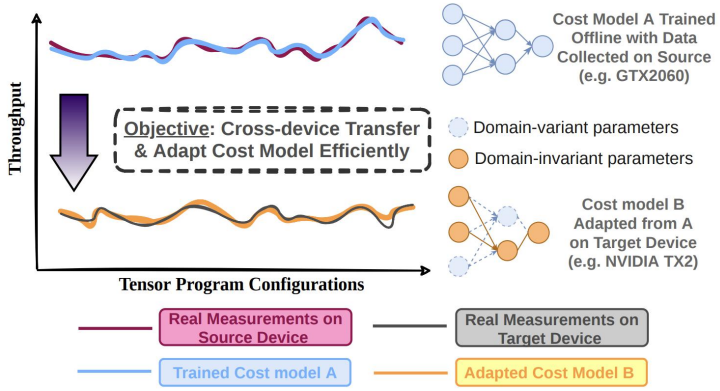


Huge search space

Time-consuming Compiling Optimization

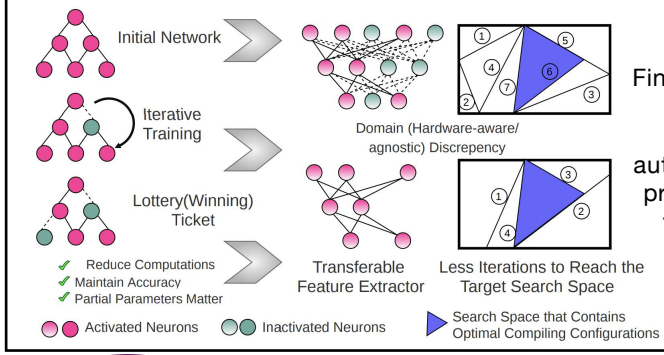
Key Insights





- Pipeline of automatic tensor program optimization
- Cost model can predict the performance of each tensor program configuration on specific device
- Cost model is usually trained online costly
- We aim to obtain the cost model on the new device highly efficiently and effectively.

Lottery-ticket hypothesis guided transfer learning



Finding the domain discrepancy between two auto-tuning search processes on the two hardware platforms

Conclusion

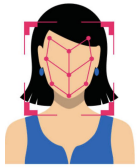
- A new framework to optimize the auto-tuning process in the DNN compiler
- A simple yet efficient design based on the lottery ticket hypothesis
- Achieves cross-device adaptation of a trained cost model by updating the domain invariant parameters during online learning

[Multi-DNN Inference] Aaron: Compile-time Kernel Adaptation for Multi-DNN Inference Acceleration on Edge GPU

The 20th ACM Conference on Embedded Networked Sensor Systems (Poster, SenSys 2022), **best poster award**

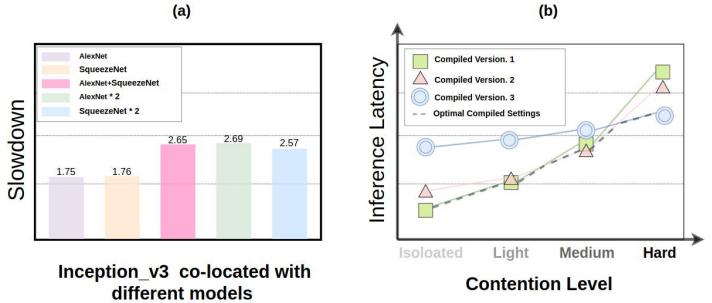
Zhihe Zhao[1], Neiwen Ling[1], Nan Guan[2] and Guoliang Xing[1]
[1]The Chinese University of Hong Kong
[2]City University of Hong Kong

Background



- Execution of multiple DNNs on a single device has gained significant interests.
- Multiple deep learning tasks must share limited on-board resources e.g., memory, caches and processing elements.
- Leverages DNN compiler techniques that incorporate both inter-kernel and intra-kernel optimizations.

Motivation



(a) Slowdown of Inception_v3 when co-running against isolation inference.
(b) Performance of three compiled versions under different contention levels.

- Our work focuses on edge GPU (NVIDIA GTX 2060 as testbed).
- Co-execution kernels with the same contention channel can easily lead to resource contention.
- The optimal compiled graph structures are not static when co-locating with different DNN tasks.

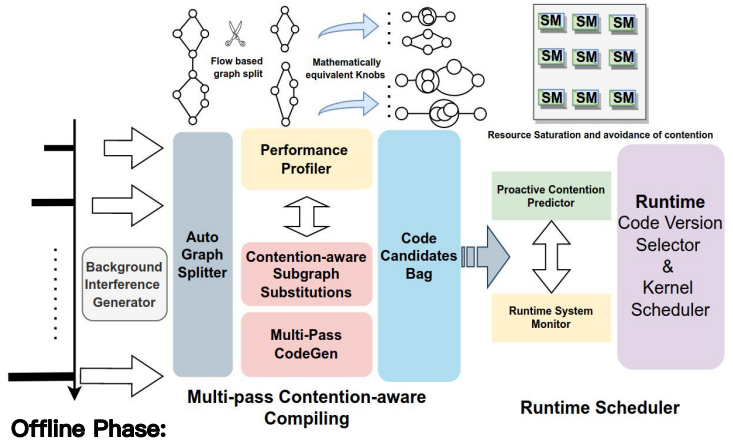
Problem Formulation

- **The objective:** To minimize the run-time contention while maximizing the on-device parallelism to fulfill resource utilization. The objective function as:

$$max[P(s_1) + P(s_2) + ...P(s_n)]$$
$$min[C(s_1) + C(s_1) + ...C(s_n)]$$

- P refers to parallelism on GPU; s represents the run-time stages. A stage contains operators from different concurrent DNN sequences; C is the measured contention metric.

Aaron System Architecture



Offline Phase:
- A profiling-based background contention generator.
- An adaptive contention-aware kernel generator.

Online Phase:
- A lightweight predictor for the upcoming contention.
- An online adaptive kernel scheduler.

Preliminary Results

- Evaluate performance of the contention-aware compiling module.
- Aaron exhibits much higher performance: 1.35x and 1.8x over solo compiling and 1.34x and 1.5x over original.
- Aaron can capture on-device contention behaviors including memory bandwidth and computation, and provide adaptive kernels at run-time.

Conclusion & Future Work

- We present Aaron, a multi-DNN inference accelerator by leveraging adaptive compiling and online scheduling.
- Aaron mainly attempts to mitigate the contention incurred by concurrent DNN inference on edge GPU.
- Extending this work by analyzing memory and compute resource contention and verifying the online scheduler.

Related Publications

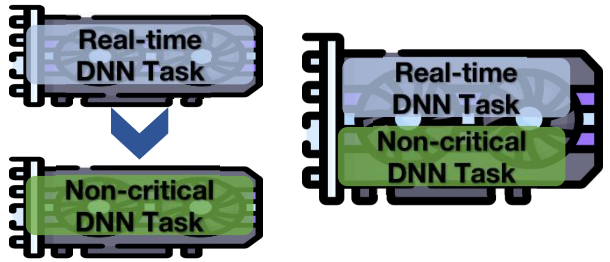
[1] Yaoyao et al. Ding. 2021. ios: Inter-operator scheduler for cnn acceleration. Proceedings of Machine Learning and Systems (2021)
[2] Tianqi Chen et al. 2018. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. In 13th USENIX Symposium on Operating Systems Design and Implementation (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)

[Multi-DNN Inference] Miriam: Exploiting Elastic Kernels for Real-time Multi-DNN Inference on Edge GPU

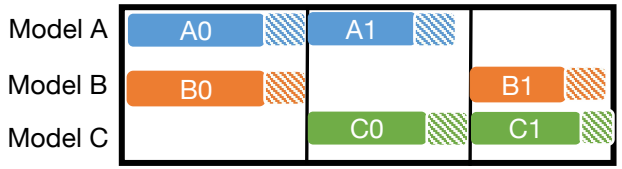
The 21th ACM Conference on Embedded Networked Sensor Systems (SenSys 2023)

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

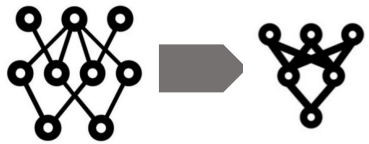
Background



Prior Work



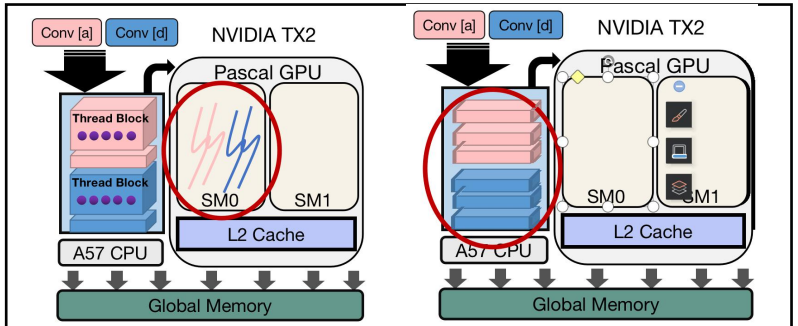
Operator Grouping [1]
Pipeline Automation



Model Compression [2]
Multi-level Trade-off

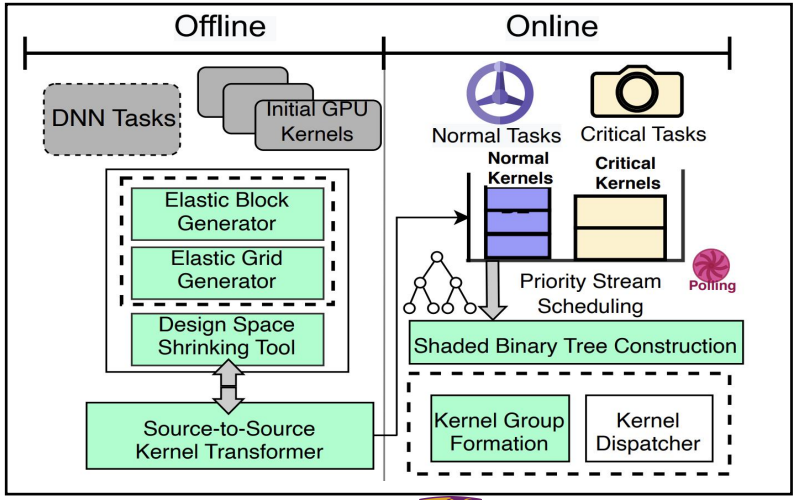
Miriam: A new DNN Compiler-Runtime Approach [3]

Nature of Resource Contention on Edge GPU

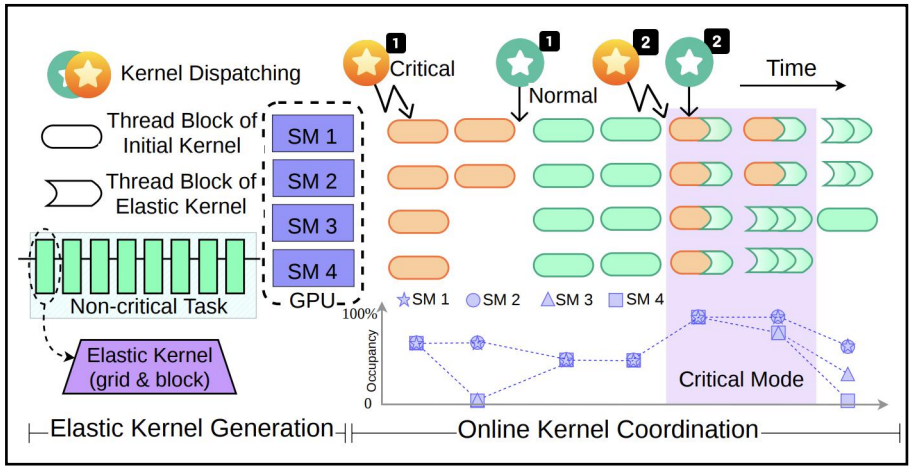


Intra-SM Contention **Inter-SM Contention**

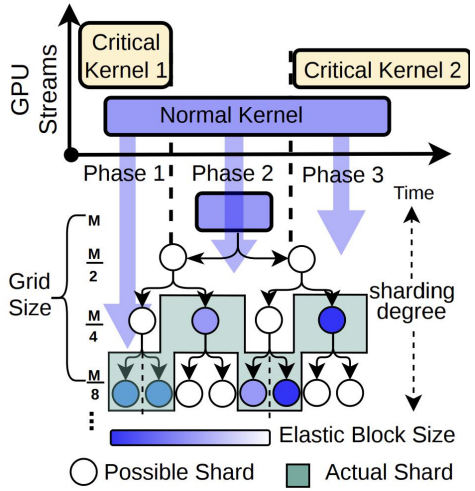
Miriam System Design



Miriam Workflow



Runtime Kernel Management



Conclusion

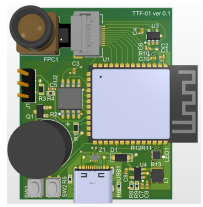
- A novel compiler-runtime synergistic framework that addresses latency and throughput problems of co-running multiple DNN inference tasks on edge GPUs
- Facilitate fine-grained GPU resource re-mapping
- Real-time multi-DNN inference on edge GPUs

[TinyML | LLM/MLOps] Entrepreneurship Project – ThingX (2023.3–now)

THiNGX

Introduction/Achievements:

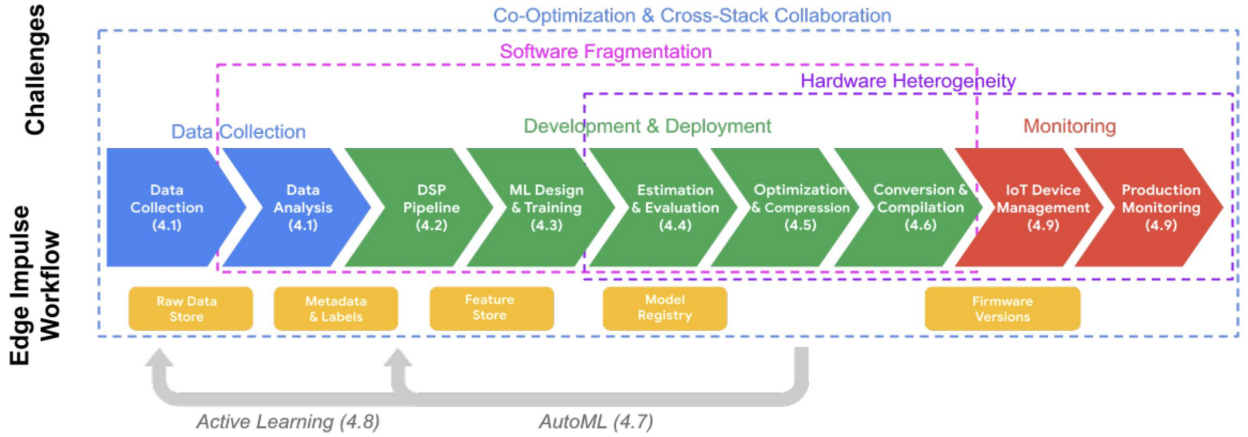
- Developed two smart healthcare hardware products to fit market need.100+ order in total in HK, UK, Japan, Singapore and Spain.
- The first ToF/Thermal Sensing based fall-detection product in HK
- TinyML (e.g. ESP32/STM32F4) | NPU (RK3568)
- Large Foundation Model in the loop
- Attended/ing exhibitions/conferences such as CES'24 and TinyML Asia'23, Korea.



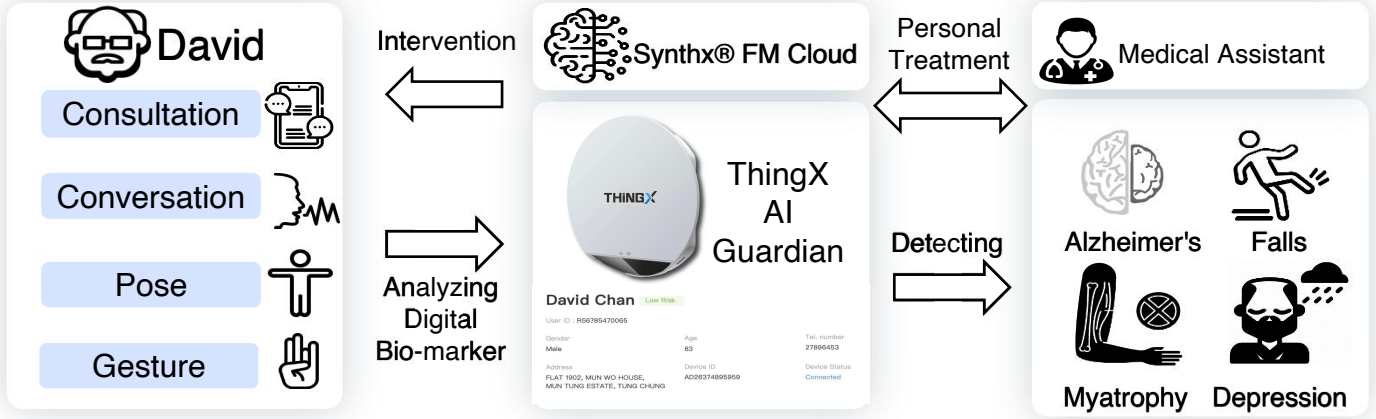
My Responsibilities:

- Team formation (R&D).
- Bridged the gap between the research team at CUHK AIoT Lab and the R&D team; Facilitating the translation of research into practical applications.
- Defined company product functions/specifications;
- Supply chain management (sensors, NPU, OEM)
- Fund raising
- Tech Lead

Edge Impulse MLOps Handbook, MLSys'23



Our Game-changing Product: [AI-Guardian]



Challenges of MLOps/ML in the Wild We Met

- Data drift & Too many corner cases
- Too many labor efforts on labeling
- Algorithm/Deployment Conflict
- Continual learning, personalization

[LLM for X | X for LLM] Tried–but–Gaveup/Interested Projects/On–going – (2023.4–now)

Democratizing LLM on Edge

LLM Compilation Landscape

- Deployment on fragmented edge devices
  - Virtual assistent on low-power smart-home devices (Alexa)
  - Picture editing on phones
- Performance
  - Efficient use of CPU/GPU/ARM ISA and on-device resources

Based on TVM

NVIDIA GPUs

Hardware/GPU	OS	Tokens/sec
GTX 1650 ti (4GB)	Fedora	15.6
GTX 1060 (6GB)	Windows 10	16.7
RTX 3080	Windows 11	26.0
RTX 3060	Debian bookworm	21.3
RTX 2080Ti	Windows 10	24.5
RTX 3090	N/A	25.7
GTX 1660ti	N/A	23.9
RTX 3070	N/A	23.3

What's the problem?

- ➢ Tuning is mandatory for code generation
- ➢ Tuning space is huge => time-consuming
- ➢ Few support for sparse model & mixed-precision operators
- ➢ No guarantee for significantly better performance

Goal

Provide high-performance for LLM workloads on edge devices with limited or no tuning

Design Directions: Analyze-then-Schedule (Thoughts Inspired from TVM community)

Observations:
Many ops share similar characteristics, e.g. softmax, layernorm, rmsnorm (reduction-dominant)

Clustering Schedule Rules:
Composing a set of rules to cover common ARM/CPU/GPU ops

Reason of not digging deep

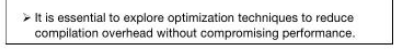
- ➢ Basic blocks of popular llm archtecture are not diverse (transformer, RNN)
- ➢ Too engineering

Utilizing FM for ISA Unification

TensorBind: Unifying On-device Tensor Program Optimization through Foundation Model

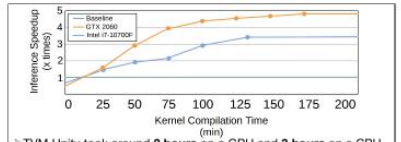
Zhihe Zhao[1], Neiwen Ling[1], Kaiwei Liu[1], Nan Guan[2] and Guoliang Xing[1]
[1]The Chinese University of Hong Kong
[2]City University of Hong Kong

Background



- ➢ Deployment of high-performance machine learning models on heterogeneous devices is challenging.
- ➢ DNN compilers reduce the model deployment cycle and enhance performance. However, the compilation overhead increases.
- ➢ It is essential to explore optimization techniques to reduce compilation overhead without compromising performance.

Motivation



- ➢ TVM Unity took around **2 hours** on a GPU and **3 hours** on a CPU to identify the best-performing kernel for a transformer encoder.
- ➢ DNN models involve multiple kernels, resulting in significant compilation overhead, especially for IoT devices.
- ➢ The long optimization time is due to the large optimization space and complex tensor compilation process.

TensorBind is a general-purpose solution for **UNIFYING** cross-device tensor program optimizations.

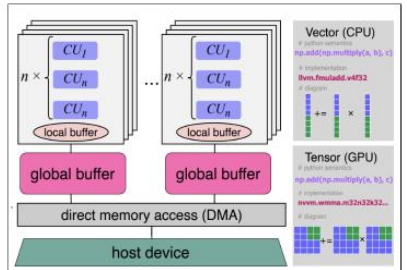
Problem Formulation

- ➢ The **objective** is to create a unified mapping function F that transforms optimized code representations for DNN computational graphs across different hardware platforms.

$$C_n = F(G, P_n; T_m)$$

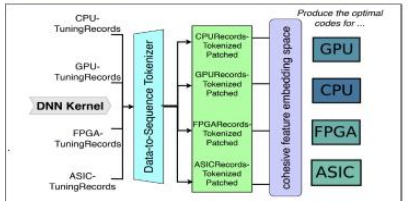
- ➢ The function F uses the optimal compilation record T_m on a specific hardware platform P_m to generate an optimized code representation C_n for the target hardware platform P_n.

Key Observations



- ➢ Previous research has attempted to unify the compilation process from two perspectives: the **"top-down"** approach (e.g., TensorIR) and the **"bottom-up"** approach (hardware architecture).
- ➢ However, they have been **limited in jointly optimizing the compilation process from both perspectives** due to the lack of representation capabilities across heterogeneous devices.

Methodology



- ➢ TensorBind utilizes the **transformer** architecture to create embeddings.
- ➢ The design of TensorBind includes **sequence-based data tokenization** and cohesive feature embedding space.

Acknowledgement

This work is supported in part by the Research Grants Council (RGC) of Hong Kong under Collaborative Research Fund (CRF) grants C4072-21G and C4034-21G

Learning-based DNN Decompilation