Proposal: Broadening IREE Compatibility for Enhanced Performance

CSC4050 Computing Capstone Project

Alae-eddine Nasr
School of Data Science
Chinese University of Hong Kong, Shenzhen
alae-eddinenasr@link.cuhk.edu.cn

Vincentius Janssen
School of Data Science
Chinese University of Hong Kong, Shenzhen
vincentiusjanssen@link.cuhk.edu.cn

Yohandi
School of Data Science
Chinese University of Hong Kong, Shenzhen
yohandi@link.cuhk.edu.cn

Prof. Hsu Wei-Chung
School of Data Science
Chinese University of Hong Kong, Shenzhen
hsuweichung@cuhk.edu.cn

Background

The increased adoption of machine-learning (ML) for complex problem-solving and decision-making demands ever-increasing computational resources. The development of the recent models, such as OpenAI's GPT-4 and Google's Gemini, is indicative of this trend: model performance scales with model size, suggesting that demand for compute will increase even further as we move towards more advanced ML models.

Training state-of-the-art (SOTA) models requires both large quantities of both data and computational power, which is straining the limits of current hardware. Meanwhile, the development of recent large language models (LLM) such as OpenAI's GPT-4 and Google's Gemini, indicates a clear trend: model performance scales with model size, suggesting that demand for compute resources will escalate as the industry moves towards more advanced ML models.

This trend is in opposition with the projected decline of Moore's Law, a historical predictor of growth in the scaling of computing power. This underscores the crux of the issue: next-gen AI/ML applications demand increasingly larger amounts of compute, while cost and performance scaling of silicon processors is slowing down.

In response to these challenges, compiling ML models is emerging as a viable solution to enhance compute efficiency on existing hardware platforms. By optimizing model code for specific hardware architectures, compilers can significantly improve execution speed and reduce resource consumption, thereby mitigating the compute limitations that currently constrain the development and deployment of SOTA models. This approach not only maximizes the potential of existing hardware but also paves the way for the continued evolution and application of ML models, ensuring that advancements can be sustainably scaled and integrated into a wide array of domains.

An ML compiler is a sophisticated tool designed to optimize machine learning models for various hardware architectures. It translates the high-level representations of ML models into efficient, low-level code that can be executed directly on target hardware, such as CPUs, GPUs, and specialized application specific processors such as TPUs. This process involves several stages, including analyzing the model's computational graph, optimizing operations for efficiency, and applying hardware-specific optimizations in code-gen. The goal is to enhance the performance and efficiency of machine learning models, enabling them to run faster and consume less power without sacrificing model accuracy.

IREE (Intermediate Representation Execution Environment) is an end-to-end compiler framework that uses Multi-Level Intermediate Representation (MLIR) to enable the deployment of ML models on a diverse range of hardware platforms. IREE is designed to lower machine learning models written in high-level frameworks such as PyTorch or TensorFlow into MLIR dialects than can be optimized and run on various hardware targets. It supports a variety of ML frameworks, offering a flexible and powerful tool for developers looking to deploy AI/ML applications. By providing an efficient pathway from model development to hardware execution, IREE addresses the computational challenges of running and developing advanced ML models, facilitating efficient deployment and scaling of AI/ML applications across various domains.

MLIR is an IR based off LLVM that can be used to represent high-level machine learning models and facilitate their transformation and optimization for target hardware platforms. IREE utilizes MLIR as its representation for ML models, allowing it to optimize models described in various high-level frameworks by manipulating their common (MLIR) transformation.

Related Work

[1] Jieyang Chen et al. "TSM2: optimizing tall-and-skinny matrix-matrix multiplication on GPUs". In: *Proceedings of the ACM International Conference on Supercomputing*. ICS '19. Phoenix, Arizona: Association for Computing Machinery, 2019, pp. 106–116. ISBN: 9781450360791. DOI: 10.1145/3330345.3330355. URL: https://doi.org/10.1145/3330345.3330355.

[2] Tianqi Chen et al. *TVM: An Automated End-to-End Optimizing Compiler for Deep Learning*. 2018. arXiv: 1802.04799 [cs.LG].

[3] Chris Lattner et al. *MLIR: A Compiler Infrastructure for the End of Moore's Law*. 2020. arXiv: 2002.11054 [cs.PL].

[4] Mingzhen Li et al. "The Deep Learning Compiler: A Comprehensive Survey". In: *IEEE Transactions on Parallel and Distributed Systems* 32.3 (Mar. 2021), pp. 708–727. ISSN: 2161-9883. DOI: 10.1109/tpds.2020.3030548.

[5] Hsin-I Cindy Liu et al. *TinyIREE: An ML Execution Environment for Embedded Systems from Compilation to Deployment*. 2022. arXiv: 2205.14479 [cs.PL].

[6] Hongbin Zhang et al. "Compiler Technologies in Deep Learning Co-Design: A Survey". In: *Intelligent Computing* 2 (2023), p. 0040. DOI: 10.34133/icomputing.0040.

Approach Proposal

Our approach is focused on improving IREE by addressing several critical areas that have been addressed in a recent OpenXLA community meeting with engineers from Google and Nvidia: improving model correctness, optimizing performance across different hardware architectures, and expanding support for GPU libraries or interfaces. These areas represent a significant opportunity to advance the capabilities of IREE.

A paramount concern in the deployment of machine learning models is ensuring their correctness across various benchmarks. Currently not all models compiled with IREE, particularly those benchmarked against Torchbench for PyTorch, are passing or yielding valid results, with pass rates varying between 52% to 94%. This variability indicates an opportunity to enhance the correctness of IREE to ensure reliable model compilation. Our approach will involve root-cause analysis to identify and rectify the problems causing these discrepancies, aiming to achieve a higher pass rate across all models.

We have also found through our research that there are performance issues in some models when deployed on the RISC-V architecture, suggesting an area ripe for potential optimization, particularly with improved code generation for the RISC-V Vector Extension. Furthermore, while the OpenXLA project as a whole has historically been focused on TPU performance, with some support for CPU as well, GPU performance has not been optimal. To address this, we propose to extend and improve support for GPU libraries, such as AMD ROCm and Intel OneAPI. By doing so, we aim to make IREE more performant on a wider range of GPU platforms.

Code generation is another critical area for enhancement. For CPU, we have thought about optimizing narrow M/N matrix kernels and enhanced RVV support. Whereas for GPU, our approach concentrates on pipelining optimizations and matrix optimization.

By using existing insights from the papers listed above, we aim to contribute to the development of IREE as a valuable tool for efficiently deploying and scaling AI/ML applications across diverse hardware platforms.