Understanding ML Compilers: A Benchmark on Heterogeneous Hardware
MSc Thesis

Benjamin Ramhorst, Gustavo Alonso

Autumn Semester 2024

1 Introduction

Machine Learning (ML) compilation is an active area of research and industrial development. By representing a neural network as a computation graph, compilers can optimise the graph for execution on the target hardware (e.g. CPU, GPU or FPGA). Compilers apply various optimisations such as quantisation, operator folding, constant propagation etc. Over the years, many compilers, both from academic and industrial efforts, have been proposed, including: TVM, OpenVINO, TensorRT, MiGraphX etc., with some compilers also being capable of targeting different devices. The objective of this project is to gain an in-depth understanding of the inner-workings of ML compilers and develop extensions to a tool for automated benchmarking of compilers on heterogeneous hardware. While the main deliverables are fixed, there is plenty of room for extensions, depending on the student's area of interest: FPGA compilers, power measurements, model-specific evaluations, costs of data movement etc. Successful completion of the project will contribute to a paper on a related topic.

2 Work plan

The work consists of the following units:

1. Literature review on ML compilers: the goal is to gain an in-depth understanding of ML compilers, their inner-works and typical applications. The student should understand the various types of IR and their advantages/disadvantages, various optimisations and hardware-specific optimisations, such as unrolling and tiling.
2. Introduction and hands-on experimentation: Experiment with an internal tool for benchmarking compilers and evaluate already supported compilers (MiGraphX, OpenVINO, XLA) with some of the following models: ResNet50, VGG19, GPT-2, BERT. Report latency, throughput, power and compare to baselines. Obtain the full accuracy and compare to online sources (papers, HuggingFace etc.). Any inconsistencies or issues (e.g. memory leaks) should be documents so that they can be reproduced.

3. Backend extension: Add support for benchmarking other compilers, including TVM, PlaidML etc. This will involve parsing high-level models, creating classes for compiled models and supporting synchronous / asynchronous inference. Detailed analysis of the compilers properties.
4. Analysis and comparison: Compare the newly added compilers with the previously supported ones and perform an in-depth analysis of accuracy, throughout and latency based on the choice of run-time parameters.
5. Open-ended work: Add extensions to support FPGA compilers, improve the power measurement flow, extend support to more irregular architectures (e.g. graph neural networks or recommender models) etc.

3 Deliverables

The project should result in the following deliverables:

1. Project report / MSc Thesis. This should contain at least the following:
   - Background: an in-depth summary of ML compilers, their application and typical features. A list and short comparison (supported hardware, supported frontends, implementation etc.) of the common compilers (e.g. TVM, OpenVINO etc.)
   - Description of the implementation: Extensions to the backends, functional verification, limitations.
   - Results: Thorough analysis of supported compilers, in terms of latency, throughput and power. Benchmarks performed both on both CPU and GPU. Results for any newly added compilers (e.g. TVM).
2. Complete code, integrated with the existing infrastructure. The source code should also be accompanied with enough documentation to allow complete reproduction of the experimental results in the thesis.
3. Presentation of the results and demonstration of functionality.

4 Grading

The minimum requirements for a grade of 5.0 are as follows:

- Successful completion of work items 1–4. Attempted work item 5 with some conclusions drawn.
- Production of deliverables 1–3 to a satisfactory standard.

The grade will be reduced if these goals are not achieved, except in the case of extreme extenuating circumstances (such as an unforeseeable and unsolvable technical barrier to completing the work, accompanied by an acceptable alternative work item).

A grade of 5.50 or higher will be awarded for the completion of the work to an unusually high quality, the addition of extra work (e.g., support for unit and integration tests), or particularly detailed insights to the results (e.g. understanding memory management issues, breaking points, data movement overheads etc.)

5 Prerequisites

Knowledge of Python and machine learning. Good software development and version control (git) skills. Knowledge of compilers and hardware acceleration (GPU, FPGA) beneficial but not required.

6 Contact

If interested, please contact Benjamin Ramhorst (benjamin.ramhorst@inf.ethz.ch) to arrange a meeting.