Tensor Program Optimization with Probabilistic Programs

Junru Shao, Xiyou Zhou, Siyuan Feng, Bohan Hou, Ruihang Lai, Hongyi Jin, Wuwei Lin, Masahiro Masuda, Cody Hao Yu, Tianqi Chen

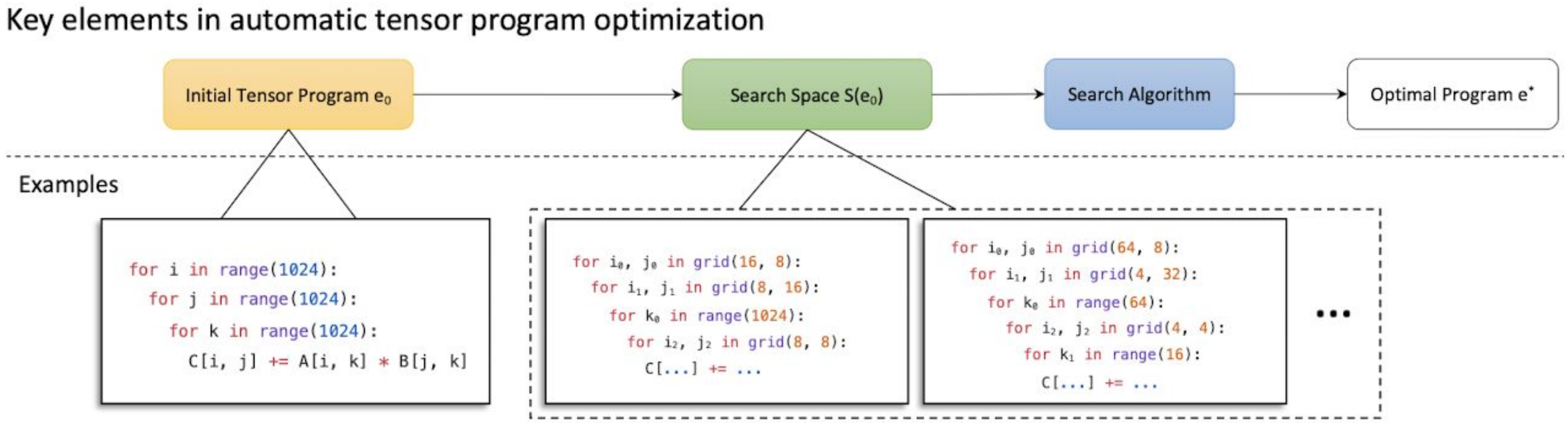
Presenter: Qianyi Liu

Context

- Current deep learning frameworks relies on **vendor-specific operator libraries** (e.g. CuDNN) to optimise deployment of neural networks on hardware
    - Choose from logically equivalent programs with significantly different performance
    - Huge engineering effort + domain knowledge

- **Automatic program optimisation** — machine learning
    - Two crucial components
    - A search space (loop transformation, vectorisation, threading patterns, and hardware acceleration)
    - Learning-based search algorithms

A typical workflow for automatic tensor program optimization

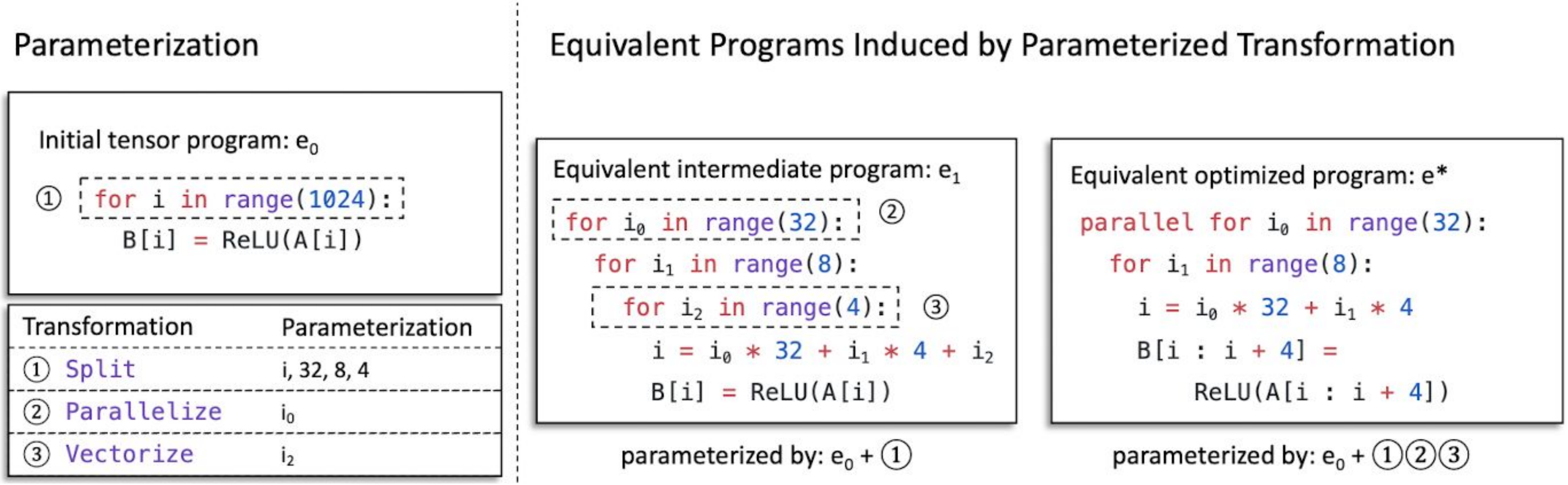


MetaSchedule

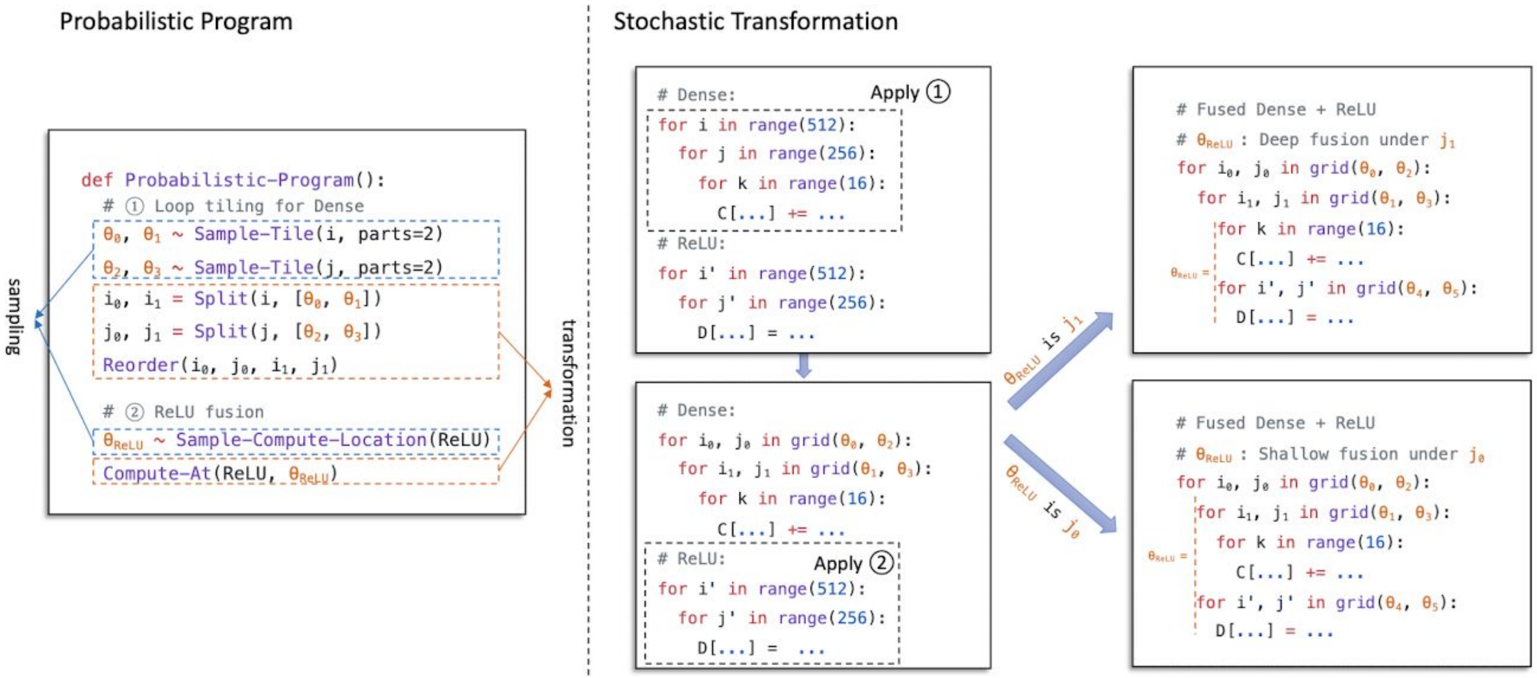
- The search space itself fundamentally limits the best possible performance search algorithms can get.
- Defining the search space for a wide range of tensor programs is challenging
  - S(e0) is highly dependent on e0
  - Differs in different hardware domains
  - Hardware and model settings evolve -> update S(e0)
- This paper aims to provide a programmable abstraction to construct S(e0) in a composable and modular way
- **MetaSchedule**: a domain-specific probabilistic programming language abstraction to construct a search space of tensor programs

Stochastic Search Space Construction

- Parameterize an optimisation search space by the initial program followed by a sequence of transformations on the program
- Allow further parameterization of each transformation step with random variables, drawn from sampling distributions

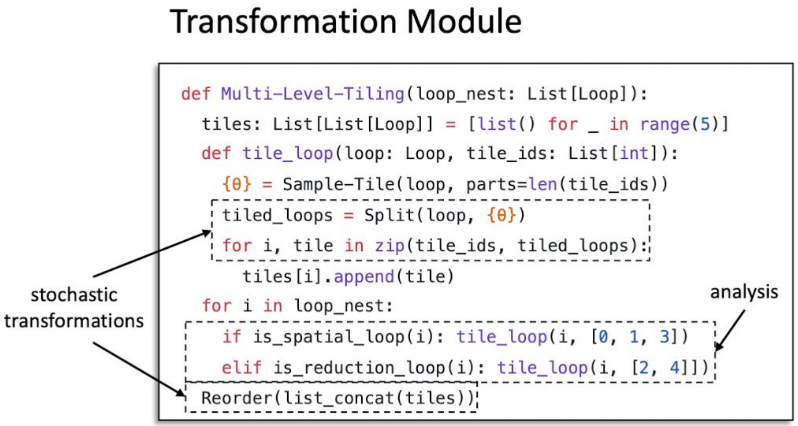


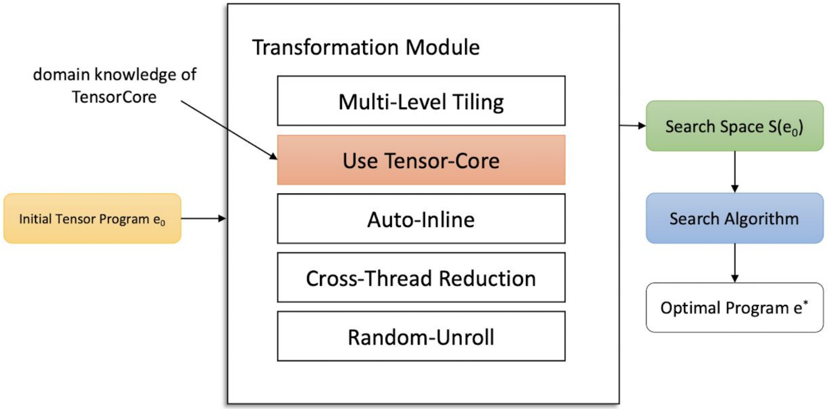
Defining stochastic transformation in MetaSchedule



Modular Search Space Composition

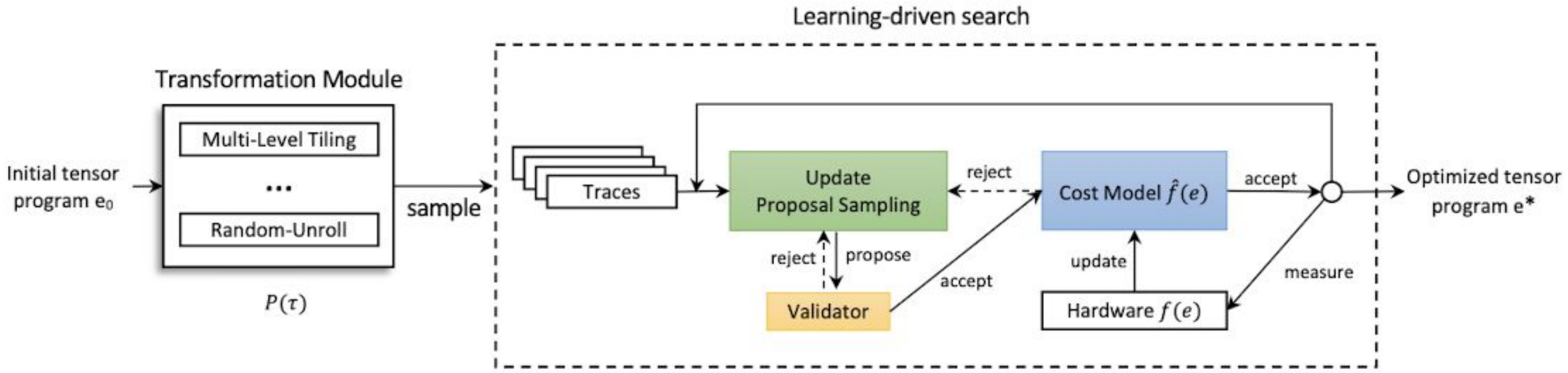
- Aim: make transformation reusable, make MetaSchedule more easy to use
- Introduce **transformation module**
  - Atomic stochastic transformation
  - Composition of program analysis, sampling as well as smaller transformations





A generic learning-driven framework to find an optimized program

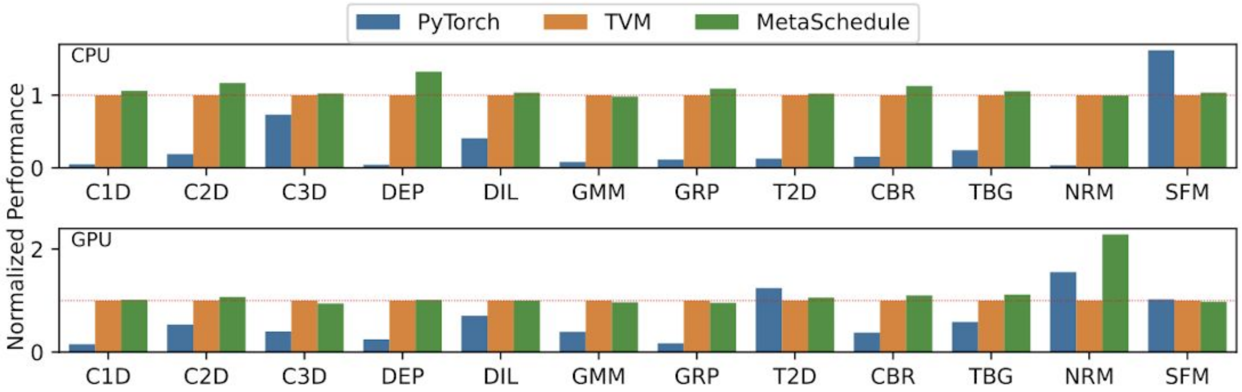
1. Search algorithm samples the MetaSchedule program to obtain a collection of traces
2. An evolutionary search algorithm that proposes a new variant of the trace by mutating the RV -> validator + cost model -> accept
3. Proxy cost model: a tree-boosting-based cost model – updated throughout the process



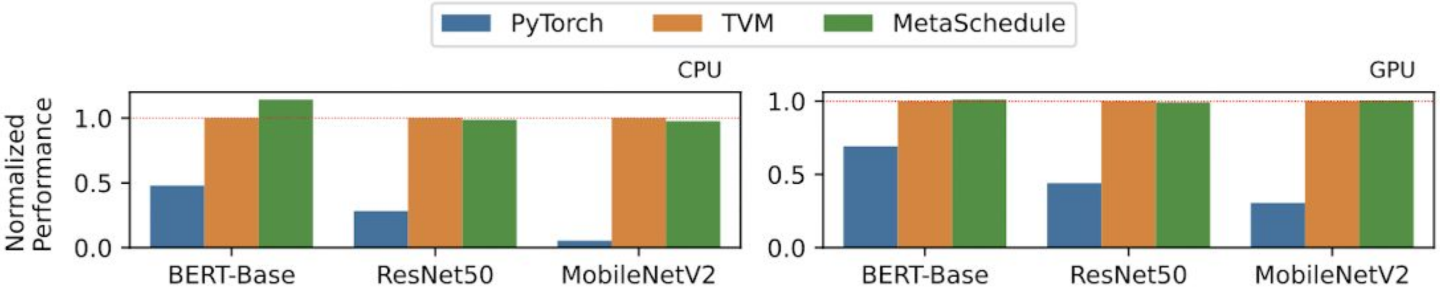
Experiment 1: Expressiveness to cover common optimisation techniques

Target: a diverse set of operators and subgraphs

- MetaSchedule: our approach
- TVM (AutoTVM and Ansor) — SOTA tensor program optimisation system
- PyTorch — optimised with vendor libraries



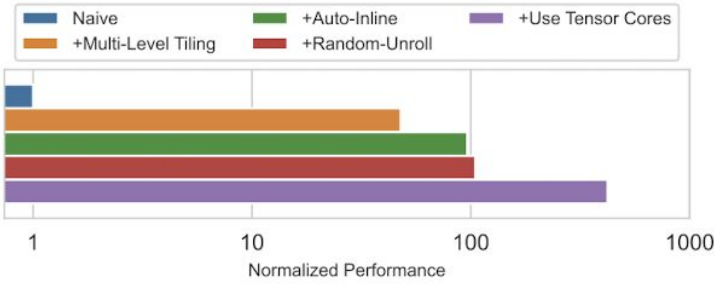
Experiment 2: optimising End-to-End deep learning models



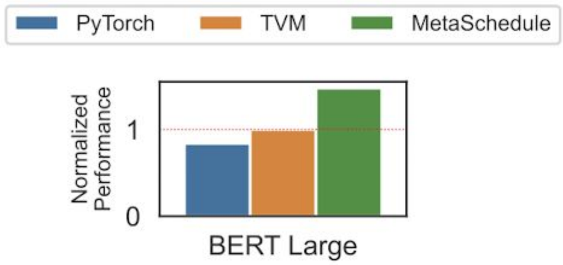
Conclusion: MetaSchedule performance is on parity with TVM, while surpassing PyTorch in all cases -> the MetaSchedule framework delivers end-to-end performance

Experiment 3: Search space composition and hardware-specific modules

- By progressively enriching the search space, the performance of optimized tensor programs consistently increases -> translate to end-to-end model performance
- Convenience of customization and composition



(a) Performance with different search spaces.



(b) BERT-Large Performance.

Takeaways

Pros:

- A novel piece of work — MetaSchedule, probabilistic programmable abstraction
- Decouples the search space construction from the search — enabling further customisation without surgical changes to the system
- A simple yet powerful generalisation of existing tensor program optimisation methods

Cons:

- Lack of further evaluation on the search space construction process and program optimisation process
- Didn’t explain in detail the advantage over previous deterministic approaches using other DSLs

Discussion