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

R244 Presentation

Juntong Deng

jd2125@cam.ac.uk

Motivation — Challenges in Deploying Deep Learning

- Deep learning models are deployed on a wide variety of environments (cloud servers, mobile devices, CPUs, GPUs, etc.)
- Deep learning models are diverse (video understanding, natural language understanding, recommendation systems, etc.)
- Optimization becomes important (selecting among equivalent tensor programs with different characteristics to achieve maximum performance)
- Diverse models and hardwares constitute a huge search space
- Manual optimization has become a bottleneck; Automated optimization is needed
- Efficient search algorithm is needed to find the optimal tensor program in the search space

Motivation — Search Space Definition

- The search space definition contains a set of equivalent tensor programs
- Tensor programs in the search space have different characteristics, such as threading patterns, vectorization, memory access, hardware acceleration, etc
- Needs to find the optimal tensor program for the deployment environment
- Most previous work used pre-defined search spaces to encode domain knowledge once
- Most previous work focused on developing efficient search algorithms

Motivation — Search Space Definition

- The
- Ten

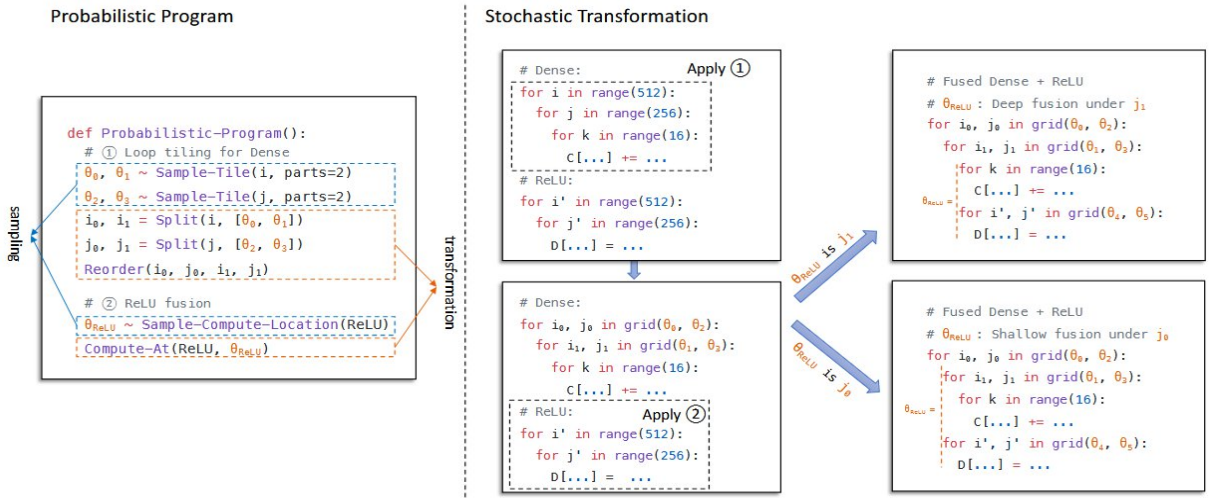
  patt
- Nee
- Mos
- Mos

What is the problem here?

- The pre-defined search spaces limit the best possible search results
- Expanding the search space for new tensor programs and new hardware primitives is difficult
- Changes to the search space construction require surgical modifications to the automated program optimization framework

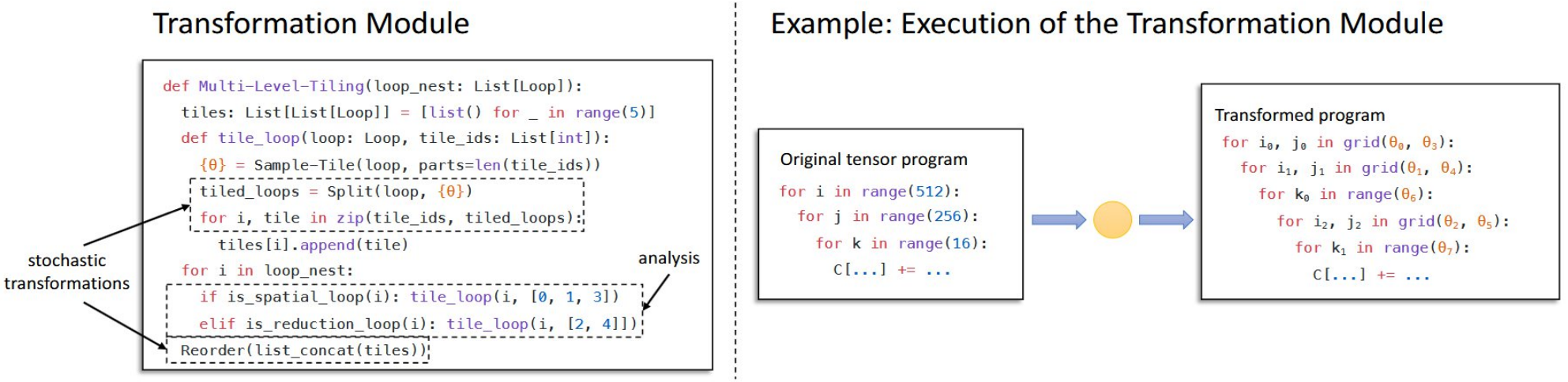
MetaSchedule — Probabilistic Programming

- A domain-specific probabilistic programming language abstraction to construct a rich search space of tensor programs
- Parameterizing tensor programs with the initial program and sequence of transformations (with different characteristics or structures)
- Program transformation based on random variables (stochastic transformations)



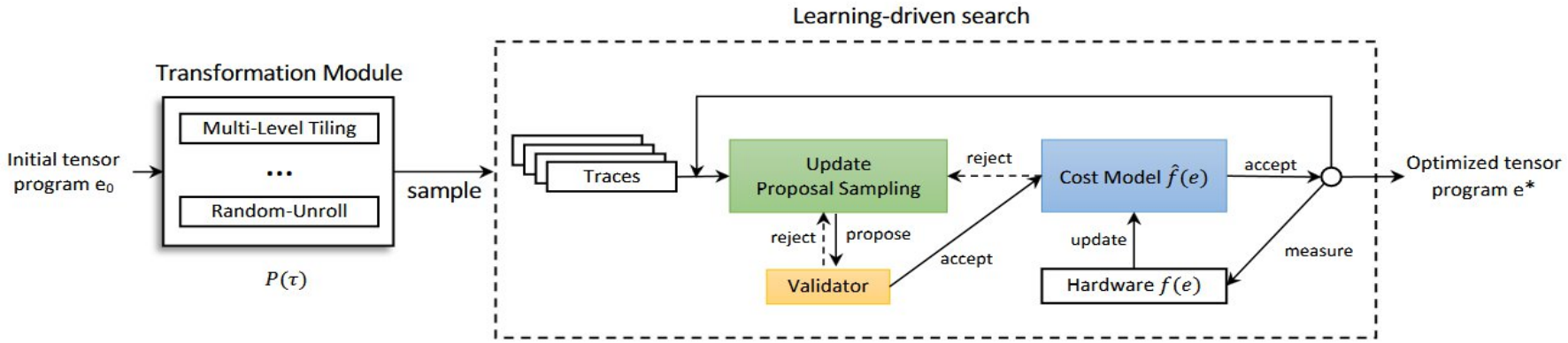
MetaSchedule — Modular Search Space Composition

- The constructed search space can be modularized into transformation modules (including stochastic transformations, analysis, etc) and reused by other workloads
- Transformation modules are implemented by practitioners of prior domain knowledge
- By combining hardware-specific modules and generic modules, the search space for any tensor program can be generated



MetaSchedule — Learning-driven Search

- **Objective formalization**: Assign a higher probability to the programs that perform well
- **Execution tracing**: Sample the program condition on the execution sequence
- **End-to-end search**: Using an evolutionary search algorithm based on the proxy cost model (continuously updated)
- **Trace validation**: Eliminate invalid traces chosen by random variables (e.g., beyond physical hardware limits)

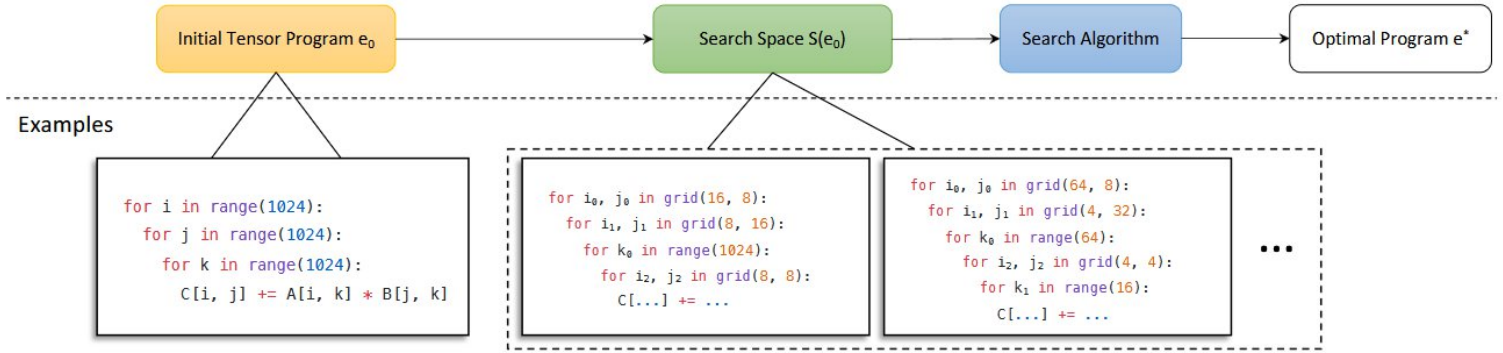


Evaluation

- Expressiveness (a diverse set of operators and subgraphs)
  - The performance of MetaSchedule is comparable to or even better than TVM, and in most cases, it outperforms PyTorch by a significant margin
- End-to-End Deep Learning Models Optimization (BERT-Base, ResNet-50, MobileNet-v2)
  - The performance of MetaSchedule is comparable to TVM, and in all cases, it outperforms PyTorch
- Ablation Study for Transformation Modules Composition
  - Hardware-specific module Use-Tensor-Core
  - MetaSchedule with Use-Tensor-Core delivers 48% speedup compared with TVM

Conclusion & Key Contributions

- Proposed MetaSchedule, abstracting the program transformation search space using probabilistic programming language
- Proposed a learning-driven algorithm to explore search space
- Decoupled the search space definition and the search algorithm
- Made the search space expandable and customizable, and modular
- Evaluated MetaSchedule and compared it with advanced optimization frameworks
- Achieved performance improvements for workloads



Agree & Strength

- Optimization in deep learning model deployment is an important topic and this paper is a timely research
- Decoupling the search space construction and the search algorithm is a great idea. It makes expanding the search space and modularization easier
- Abstracting the tensor program search space into a probabilistic programming language is novel idea
- The idea of abstracting, programmable, and modularizing the search space influenced subsequent research
- The performance improvements in the evaluation are encouraging

Disagree & Weaknesses

- Search space definition still depends on the expertise of domain experts (not fully automatic)
- No evaluation for learning costs of probabilistic programming languages. Is the learning curve steep? How much time would an inexperienced engineer need?
- Only evaluated the performance improvement of the workload, but did not evaluate the cost of the search. Is learning-driven search time-consuming?
- No diverse hardware was evaluated (all CPU experiments on Intel Xeon Platinum 8124M, and all GPU experiments on GeForce RTX 3070). Can performance be improved on other hardwares?
- No comparison of cost-effectiveness with other optimization methods. Would other methods cost less? Is using MetaSchedule worthwhile?

Related Research

- This paper was influenced by previous works:
  - Tensor Program Transformations: Halide [7], TVM [4], etc. (no search space abstraction)
  - Automatic Tensor Program Optimization: Metatune [8], Tenset [12], etc. (orthogonal contributions)
  - Probabilistic Programming Language: Church [5], Stan [2], etc. (new scenario)
  - ...
- This paper influenced subsequent works (programmable and composable search space):
  - SparseTIR [11] (composable abstractions for sparse compilation)
  - Allo [3] (programming model for composable accelerator design)
  - Relax [6] (composable abstractions for end-to-end dynamic machine learning)
  - ...

Impact & Possible Impact

- Impact:
  - This paper was accepted by NeurIPS 2022 [9]
  - MetaSchedule was discussed in the TVM community [1]
  - MetaSchedule integrated into TVM as an alternative to AutoTVM and AutoScheduler [10]
- Possible Impact:
  - Other fields may adopt probabilistic programming abstractions (such as traditional compilers)
  - Search space modules may be packaged as third-party libraries (module sharing community)
  - Hardware primitives from different manufacturers may be standardized (in order to be integrated into probabilistic programming abstractions)

Possible Discussion Questions

- Besides equivalent tensor program selection, is probabilistic programming suitable for other scenarios? (e.g., traditional compilers, OS configurations, etc.)
- When using learning-driven search, how do we balance search time, generality, and validation costs?
- What are the risks of using learning-driven search? (reproducibility? debuggability?)
- What other aspects do we need to optimize when deploying deep learning models?
- How to balance development costs, human resource costs, and hardware costs? (Would buying more hardware be less costly than training employees or adopting new optimization methods?)

Thanks

Thank you for listening

References

[1] Michael Canesche. 2024. [RFC][MetaSchedule] Adding an Exploitation Phase to MetaSchedule to Improve Scheduling. https://discuss.tvm.apache.org/t/rfc-metaschedule-adding-an-exploitation-phase-to-metaschedule-to-improve-scheduling/17365. Accessed on Nov 11, 2025.

[2] Bob Carpenter, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. Stan: A Probabilistic Programming Language. Journal of Statistical Software 76, 1 (2017), 1–32. doi:10.18637/jss.v076.i01

[3] Hongzheng Chen, Niansong Zhang, Shaojie Xiang, Zhichen Zeng, Mengjia Dai, and Zhiru Zhang. 2024. Allo: A Programming Model for Composable Accelerator Design. Proc. ACM Program. Lang. 8, PLDI, Article 171 (June 2024), 28 pages. doi:10.1145/3656401

[4] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Meghan Cowan, Haichen Shen, Leyuan Wang, Yuwei Hu, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. 2018. TVM: an automated end-to-end optimizing compiler for deep learning. In Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation (Carlsbad, CA, USA) (OSDI'18). USENIX Association, USA, 579–594.

[5] Noah D. Goodman, Vikash K. Mansinghka, Daniel Roy, Keith Bonawitz, and Joshua B. Tenenbaum. 2008. Church: a language for generative models. In Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence (Helsinki, Finland) (UAI'08). AUAI Press, Arlington, Virginia, USA, 220–229.

References

[6] Ruihang Lai, Junru Shao, Siyuan Feng, Steven Lyubomirsky, Bohan Hou, Wuwei Lin, Zihao Ye, Hongyi Jin, Yuchen Jin, Jiawei Liu, Lesheng Jin, Yaxing Cai, Ziheng Jiang, Yong Wu, Sunghyun Park, Prakalp Srivastava, Jared Roesch, Todd C. Mowry, and Tianqi Chen. 2025. Relax: Composable Abstractions for End-to-End Dynamic Machine Learning. Association for Computing Machinery, New York, NY, USA, 998–1013. https://doi.org/10.1145/3676641.3716249

[7] Jonathan Ragan-Kelley, Connelly Barnes, Andrew Adams, Sylvain Paris, Frédo Durand, and Saman Amarasinghe. 2013. Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. In Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation (Seattle, Washington, USA) (PLDI '13). Association for Computing Machinery, New York, NY, USA, 519–530. doi:10.1145/2491956.2462176

[8] Jaehun Ryu and Hyojin Sung. 2021. MetaTune: Meta-Learning Based Cost Model for Fast and Efficient Auto-tuning Frameworks. arXiv:2102.04199 [cs.LG] https://arxiv.org/abs/2102.04199

[9] Junru Shao, Xiyou Zhou, Siyuan Feng, Bohan Hou, Ruihang Lai, Hongyi Jin, Wuwei Lin, Masahiro Masuda, Cody Hao Yu, and Tianqi Chen. 2022. Tensor program optimization with probabilistic programs. In Proceedings of the 36th International Conference on Neural Information Processing Systems (New Orleans, LA, USA) (NIPS '22). Curran Associates Inc., Red Hook, NY, USA, Article 2593, 14 pages.

[10] TVM. 2024. tvm.meta_schedule. https://tvm.apache.org/docs/reference/api/python/meta_schedule.html. Accessed on Nov 11, 2025.

References

[11] Zihao Ye, Ruihang Lai, Junru Shao, Tianqi Chen, and Luis Ceze. 2023. SparseTIR: Composable Abstractions for Sparse Compilation in Deep Learning. In Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3 (Vancouver, BC, Canada) (ASPLOS 2023). Association for Computing Machinery, New York, NY, USA, 660–678. doi:10.1145/3582016.3582047

[12] Lianmin Zheng, Ruochen Liu, Junru Shao, Tianqi Chen, Joseph E. Gonzalez, Ion Stoica, and Ameer Haj Ali. 2021. TenSet: A Large-scale Program Performance Dataset for Learned Tensor Compilers. In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1). https://openreview.net/ forum?id=aIfp8kLuvc9