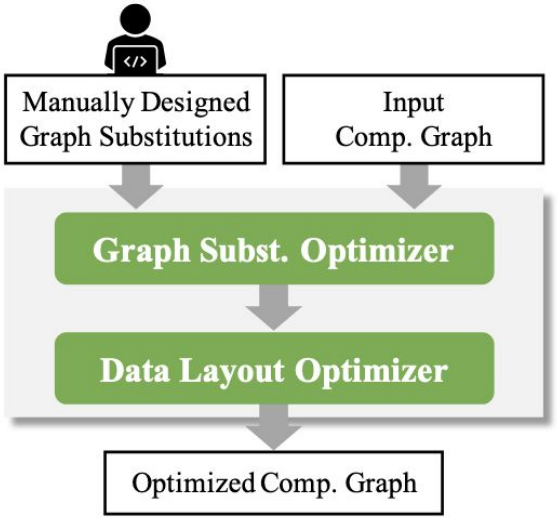
Tensor Algebra Superoptimizer (TASO)

Jia et al. (2019)

Timi Adeniran

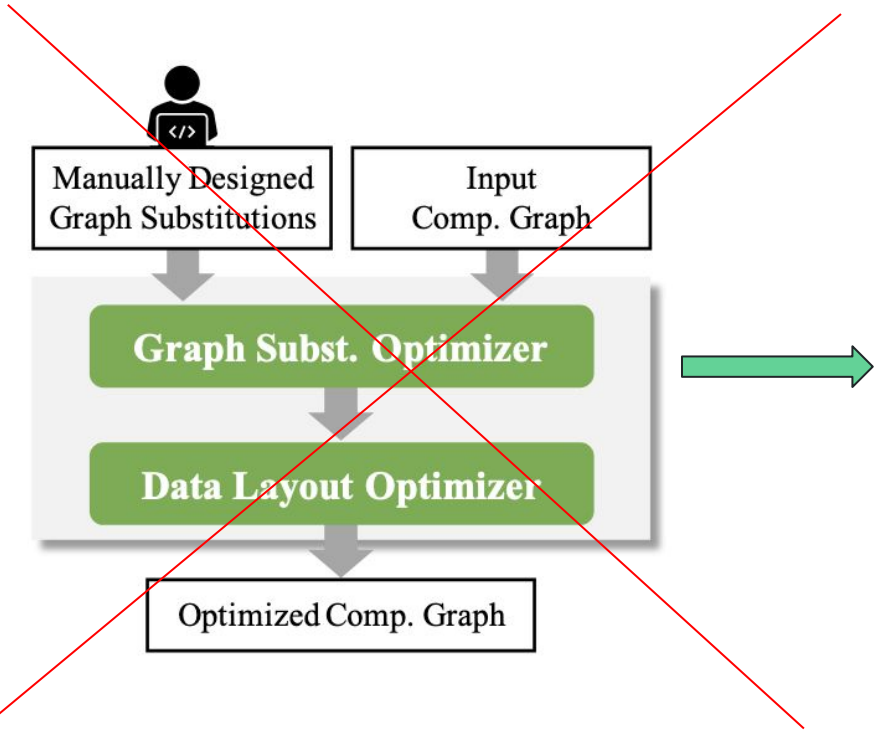
Research Questions



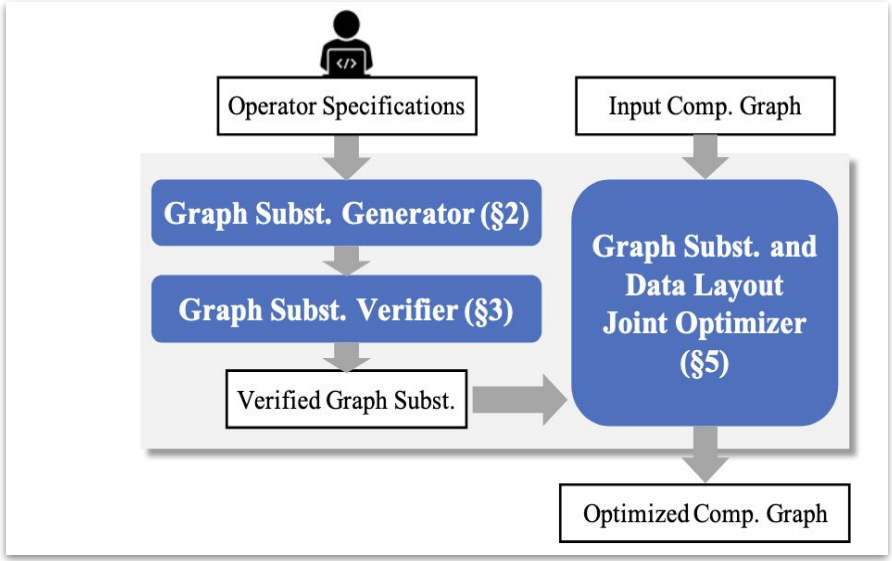
- Can graph substitutions be automated?
- Will jointly optimizing graph substitutions with the data layout yield better performance?

Can we do better?

Manual Substitutions

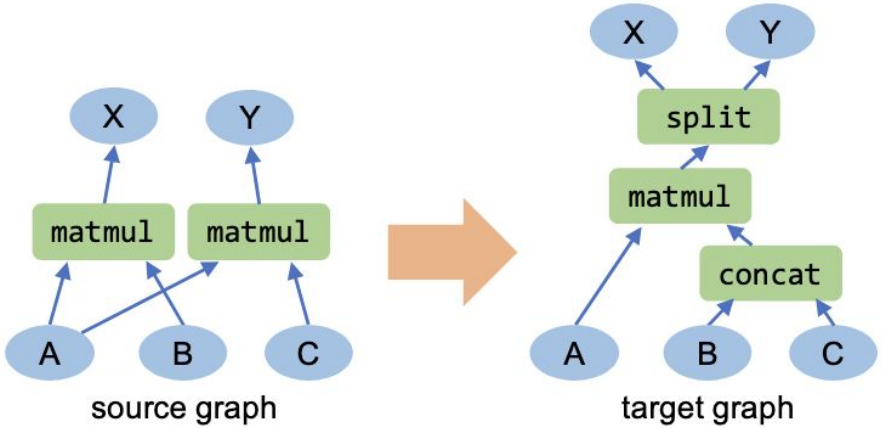


TASO



TASO

- Automatically generates substitutions.
- Verifies their correctness.
- Jointly optimizes substitutions with data layout on an input graph.



Generating Substitutions

Algorithm 1 Graph substitution generation algorithm.

```
1:  Input: A set of operators 𝒫, and a set of input tensors ℐ.
2:  Output: Candidate graph substitutions 𝒮.
3:
4:  // Step 1: enumerating potential graphs.
5:  𝒟 = {} // 𝒟 is a graph hash table indexed by their fingerprints.
6:  BUILD(1, ∅, ℐ)
7:  function BUILD(n, 𝒢, ℐ)
8:      if 𝒢 contains duplicated computation then
9:          return
10:     𝒟 = 𝒟 + (FINGERPRINT(𝒢), 𝒢)
11:     if n < threshold then
12:         for op ∈ 𝒫 do
13:             for i ∈ ℐ and i is a valid input to op do
14:                 Add operator op into graph 𝒢.
15:                 Add the output tensors of op into ℐ.
16:                 BUILD(n + 1, 𝒢, ℐ)
17:                 Remove operator op from 𝒢.
18:                 Remove the output tensors of op from ℐ.
19:
20: // Step 2: testing graphs with identical fingerprint.
21: 𝒮 = {}
22: for 𝒢1, 𝒢2 ∈ 𝒟 with the same FINGERPRINT(·) do
23:     if 𝒢1 and 𝒢2 are equivalent for all test cases then
24:         𝒮 = 𝒮 + (𝒢1, 𝒢2)
25: return 𝒮
```

- DFS algorithm to generate all possible graphs for a given set of operators and input tensors.
- Evaluates graphs on random inputs and hashes their result (fingerprint).
- Graphs with the same fingerprint are good candidates.

Verifying Substitutions

- Tests candidate substitutions against user-provided operator properties.
- Users provide a symbolic representation of properties in Python.
- Uses the Z3 theorem prover to verify properties holds for substitutions.

Pruning Substitutions

- Eliminates input tensor renaming redundancies.

- Prunes substitutions with a common subgraph between source and target.

Pruning Techniques	Remaining Substitutions	Reduction v.s. Initial
Initial	28744	1×
Input tensor renaming	17346	1.7×
Common subgraph	743	39×

Joint Optimization

- New: TASO jointly optimizes graph substitution and data layout.
- TASO uses a cost-based backtracking algorithm and picks the best* layout for each substitution.
- Cost model sums execution time of each operator in a graph.

*best according to the cost model

Joint Optimization - Backtracking Algorithm

Algorithm 2 Cost-Based Backtracking Search

Input: an input graph $\mathcal{G}_{in}$, verified substitutions $\mathcal{S}$, a cost model $Cost(\cdot)$, and a hyper parameter α.
Output: an optimized graph.

$\mathcal{P} = \{\mathcal{G}_{in}\}$ // $\mathcal{P}$ *is a priority queue sorted by Cost.*
while $\mathcal{P} \neq \{\}$ **do**
    $\mathcal{G} = \mathcal{P}$.dequeue()
    for substitution $s \in \mathcal{S}$ **do**
        // LAYOUT$(\mathcal{G}, s)$ *returns possible layouts applying* s *on* $\mathcal{G}$.
        for layout $l \in$ LAYOUT$(\mathcal{G}, s)$ **do**
            // APPLY$(\mathcal{G}, s, l)$ *applies* s *on* $\mathcal{G}$ *with layout* l.
            $\mathcal{G}' =$ APPLY$(\mathcal{G}, s, l)$
            if $\mathcal{G}'$ is valid **then**
                if $Cost(\mathcal{G}') < Cost(\mathcal{G}_{opt})$ **then**
                    $\mathcal{G}_{opt} = \mathcal{G}'$
                if $Cost(\mathcal{G}') < \alpha \times Cost(\mathcal{G}_{opt})$ **then**
                    $\mathcal{P}$.enqueue($\mathcal{G}'$)
return $\mathcal{G}_{opt}$

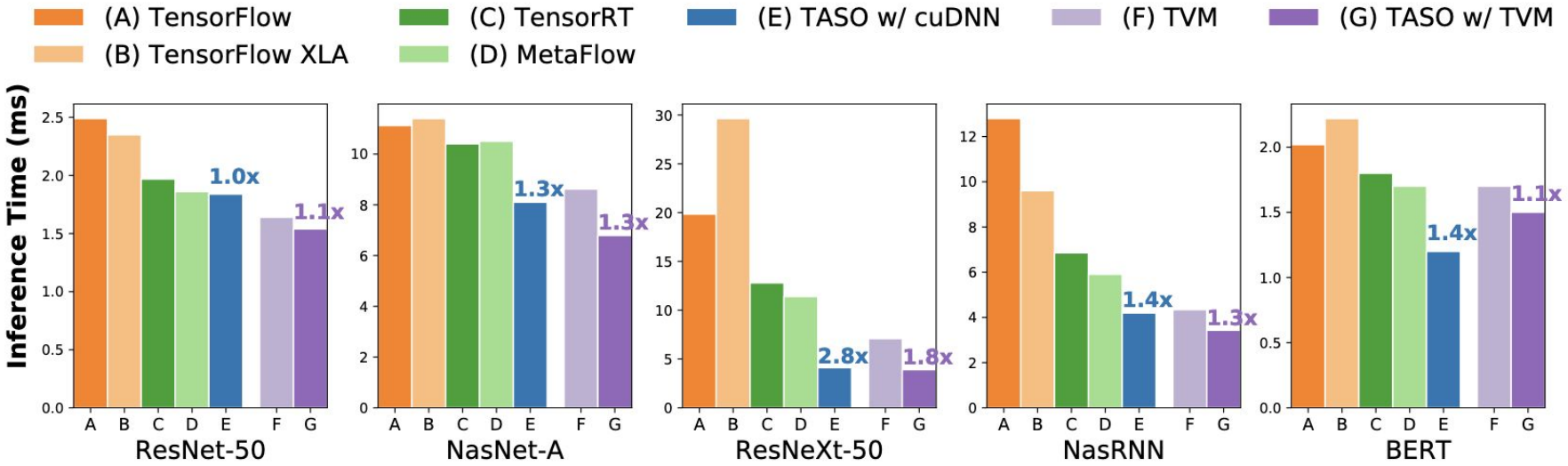
Evaluation

- Applied 743 substitutions generated from up to four operators on ResNet-50, ResNeXt-50, NasNet-A, NasRNN and BERT.

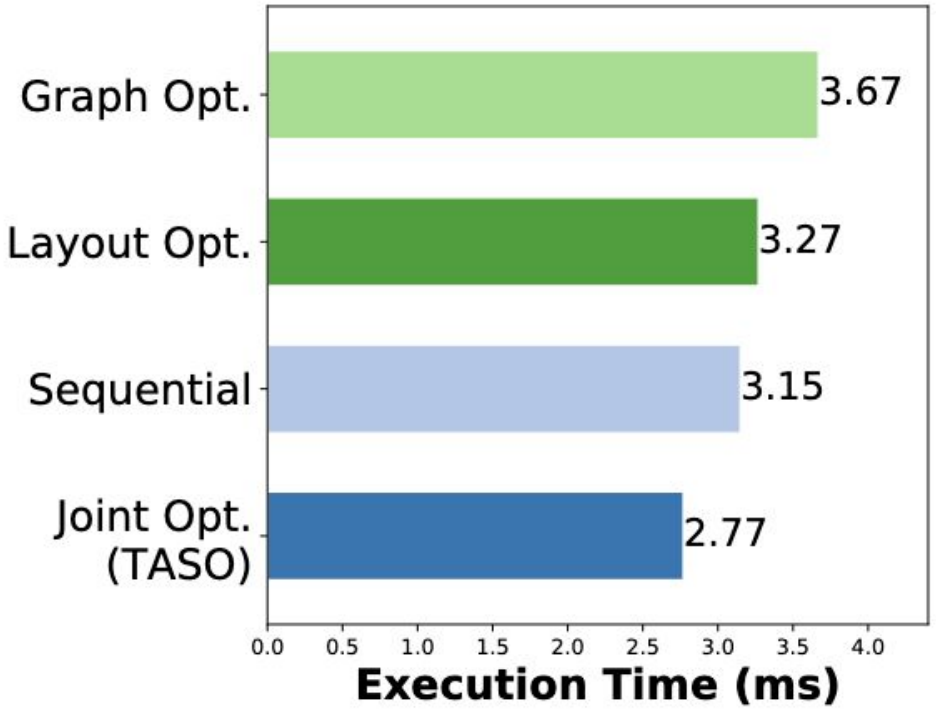
- Compared inference performance with TensorFlow, TensorFlow XLA, TensorRT, TVM, and MetaFlow on a V100 GPU.

- Compared inference performance on BERT of joint optimization with independent optimizations.

Results - End-to-end Inference Performance Comparison



Results - Joint Optimization



Discussion - Strengths

- Novel approach to tensor graph optimization:
  - Automatically generating graph substitutions.
  - Use of formal methods to verify them.
  - Jointly optimizing graph substitutions with data layout.
- Produces good results.

Discussion - Limitations (1)

- Only supports a maximum of four operators.
- Manual approach for adding operators.
- Too many redundant substitutions.
- Evaluation does not consider time taken to optimize.

Discussion - Limitations (2)

Oversimplified cost model

Table 1. Discrepancy between TASO's cost model estimates and TASO's end-to-end inference latency on some unoptimised DNNs. E2E stands for end-to-end inference latency. Time is measured in milliseconds.

DNNs	Cost model	E2E	Diff (%)
DALL-E	1.8269	1.7324	5.2%
InceptionV3	8.3650	9.2098	10.1%
BERT	1.0453	1.1264	7.8%
Squeezenet	1.3082	1.4006	7.1%
ResNext-50	6.1545	7.6498	24%
T-T	2.4828	2.7281	9.9%

From X-RLflow by He et al.

Impact

- Inspired follow-up work into automatic graph substitutions:
  - PET (2021): Wang et al. “PET: Optimizing Tensor Programs with Partially Equivalent Transformations and Automated Corrections,” OSDI 2021.
  - TENSAT (2021): Yang et al. “Equality Saturation for Tensor Graph Superoptimization,” MLSys 2021.
  - Unity (2022): Unger et al. “Accelerating DNN Training Through Joint Optimization of Algebraic Transformations and Parallelization”. OSDI 2022.
  - X-RLFlow (2023): He at al. “X-RLflow: Graph Reinforcement Learning for Neural Network Subgraphs Transformation,” MLSys 2023.

Questions?