TASO: Optimizing Deep Learning Computation with Automatic Generation of Graph Substitutions

Zhihao Jia, Oded Padon, James Thomas, Todd Warszawski, Matei Zaharia, Alex Aiken

Presented by: Samuil Stoychev
Graph Substitutions

- DNNs are represented as computation graphs combining multiple operators.
- A substitution is a mapping from a source graph to a target graph.
- We want a target graph that is functionally equivalent and that has better performance.
Existing Solutions

• All use a greedy rule-based approach.
• Substitution heuristics manually designed by human experts.
Existing Approach

Manually designed graph substitutions

Graph Substitution Optimizer

Data Layout Optimizer

*Diagram adapted from Z. Jia et al., 2019
Existing Approach

Manually designed graph substitutions

Input graph

Graph Substitution Optimizer

Data Layout Optimizer

Output graph

*Diagram adapted from Z. Jia et al., 2019
Problems With Current Solutions

• **Maintainability**
  - Manually writing substitutions is time-consuming.
  - TensorFlow’s 155 substitutions implemented in ~53K lines of code.
  - Hard to add new operators.

• **Data layout**
  - Graph substitutions and data layout are interconnected.
  - However, current approaches treat them as isolated optimization problems.

• **Correctness**
  - Hard-coded substitution rules are error-prone.
  - No verification mechanism to validate substitutions.
TASO (Tensor Algebra SuperOptimizer)

• Automatic **generation of graph substitutions**.
• **Formal verification** of generated substitutions.
• **Joint optimization** over graph substitution and data layout.
TASO’s Approach

Operator specifications

Graph Substitution Generator

*Diagram adapted from Z. Jia et al., 2019*
TASO’s Approach

Operator specifications → Graph Substitution Generator → Graph Substitution Verifier → Verified Substitutions

*Diagram adapted from Z. Jia et al., 2019
TASO’s Approach

Operator specifications → Graph Substitution Generator → Graph Substitution Verifier → Verified Substitutions → Joint Optimizer (Graph Substitution + Data Layout)

*Diagram adapted from Z. Jia et al., 2019
TASO’s Approach

Operator specifications

Graph Substitution Generator

Graph Substitution Verifier

Verified Substitutions

Joint Optimizer (Graph Substitution + Data Layout)

*Diagram adapted from Z. Jia et al., 2019*
Graph Substitution Generator

Operator specifications

Graph Substitution Generator

Graph Substitution Verifier

Verified Substitutions

Joint Optimizer (Graph Substitution + Data Layout)

*Diagram adapted from Z. Jia et al., 2019
Graph Substitution Generator

1. Enumerate all possible graphs with up to \( N \) operators using depth-first-search.

2. Run the generated graphs on test inputs and obtain their fingerprints by hashing their outputs.

3. Maintain a dictionary \( D \) mapping \( \text{Fingerprint}(G) \mapsto G \).

4. Iterate over fingerprints in \( D \) and return all pairs \((G_1, G_2)\) with identical fingerprints.
Graph Substitution Verifier

*Diagram adapted from Z. Jia et al., 2019*
Graph Substitution Verifier

• Use a **theorem prover** to verify the generated substitutions satisfy **operator properties** defined in **first-order logic**.

• 43 operator properties defined in **TASO** including:
  
  • $\forall x. \text{transpose}(\text{transpose}(x)) = x$
  • $\forall x. \text{matmul}(x, I_{\text{matmul}}) = x$
  • $\forall x, y, z. \text{matmul}(x, \text{matmul}(y, x)) = \text{matmul}(\text{matmul}(x, y), z)$

• **Additional validation steps**
  
  • Testing operator properties on small tensors.
  • Checking if operator properties are consistent.
Pruning Redundant Substitutions

- A substitution is **redundant** if it can be inferred from another valid substitution.
- Pruning strategies in TASO:
  - Input tensor renaming
  - Common subgraph
- Pruning reduces the number of verified substitutions in TASO by **39 times**.
Joint Optimizer

Operator specifications → Graph Substitution Generator

Verified Substitutions

Joint Optimizer (Graph Substitution + Data Layout)

*Diagram adapted from Z. Jia et al., 2019
Joint Optimizer

- Layouts of input tensors (A, B) and output tensor (X) are fixed.
- But we can change the layout of intermediate tensors.
Joint Optimizer

• Using cost-based backtracking search defined by MetaFlow.

• Extended by TASO to account to account for different possible data layouts.

• The $\alpha$ parameter controls the search space. ($\alpha = 1.05$ in TASO)

**Algorithm 2 Cost-Based Backtracking Search**

1: **Input**: an input graph $G_{in}$, verified substitutions $S$, a cost model $\text{Cost}(\cdot)$, and a hyper parameter $\alpha$.
2: **Output**: an optimized graph.
3: 
4: $P = \{G_{in}\}$ // $P$ is a priority queue sorted by $\text{Cost}$.
5: **while** $P \neq \{\}$ **do**
6:   $G = P$.dequeue()
7:   **for** substitution $s \in S$ **do**
8:     // $\text{LAYOUT}(G, s)$ returns possible layouts applying $s$ on $G$.
9:     **for** layout $l \in \text{LAYOUT}(G, s)$ **do**
10:    // $\text{APPLY}(G, s, l)$ applies $s$ on $G$ with layout $l$.
11:    $G' = \text{APPLY}(G, s, l)$
12:    **if** $G'$ is valid **then**
13:     **if** $\text{Cost}(G') < \text{Cost}(G_{opt})$ **then**
14:       $G_{opt} = G'$
15:     **if** $\text{Cost}(G') < \alpha \times \text{Cost}(G_{opt})$ **then**
16:       $P$.enqueue($G'$)
17: **return** $G_{opt}$
Implementation

• TASO implementation includes **12 operators** and **43 operator properties**.

• Built on top of MetaFlow
  • **8000** lines of code in total.
  • **1400** lines for operator reference implementations + operator properties.

• Framework-agnostic.
Evaluation Setup

• Evaluated on optimizing 5 deep neural networks:
  • ResNet-50
  • ResNeXt-50
  • NasNet-A
  • NasRNN
  • BERT

• Substitution generation:
  • Enumerated graphs with up to 4 operators.
  • Generated 743 verified substitutions.
Performance Comparison

- Compared against TensorFlow, TensorRT, MetaFlow, TVM both on the cuDNN and TVM backends.
Why Does TASO Perform Better?

- Different architectures require different substitutions.
- Old heuristics do not necessarily apply to new models.
Why Does TASO Perform Better?

- Joint optimization compared with:
  - Only graph substitution optimizations.
  - Only data layout optimizations.
  - Sequential optimization.

- Joint optimization reduces execution time by 12% compared to sequential optimization.
Review

**Strengths**
- Novel approach to optimizing DNNs via graph substitutions.
- Evaluation demonstrates clear improvement in performance.
- Good presentation of motivations and findings.

**Potential Improvements**
- Approach does not scale beyond graphs of 4 operators.
- No evaluation of system performance (e.g. memory consumption).
- Evaluation done only on a single machine.
Impact

- TASO builds on the findings of the MetaFlow paper:
- TASO is publicly available on GitHub.
  - https://github.com/jiazhihao/TASO
Thank you for the attention!