Automatically Discovering Systems Optimizations for Deep Learning

Zhihao Jia

CMU and Facebook
The Rise of ML and Neural Networks

Adapted from Jeff Dean, HotChips 2017
The Rise of ML and Neural Networks

Adapted from Jeff Dean, HotChips 2017
Automated Approaches to Accelerate Machine Learning

Zhihao Jia | Stanford University
Deep Neural Networks for Machine Translation

1000x Productivity
Google shrinks language translation code from 500k imperative LoC to 500 lines of dataflow
What is a Deep Neural Network?

- Collection of simple trainable mathematical units that work together to solve complicated tasks

A tensor (i.e., \( n \)-dimensional array)

A tensor algebra operator (e.g., convolution, matrix mul)
ML Computation is Increasing Exponentially

Amount of computation in largest ML training doubles every 3.4 months

[OpenAI Blog, 2018]
ML Systems are a Key Ingredient in ML

- Heterogenous Processors
- Heterogenous Memories

High Performance Network

Distributed Heterogenous Hardware Architectures

~4,600 compute nodes

ML Model
Challenges of Building ML Systems

- **Massively Parallel**: Tensor algebra is parallelizable in many dimensions.
- **New ML Operators**: Continuously introduced into ML systems.
- **Heterogenous Hardware**: Different processor kinds and complex memory hierarchy.
**CMU Automated Learning Systems Lab**

Mission: Automate the design and optimization of ML systems by leveraging
1. Statistical and mathematical properties of ML algorithms
2. Domain knowledge of modern hardware platforms

[https://catalyst.cs.cmu.edu/](https://catalyst.cs.cmu.edu/)
CMU Automated Learning Systems Lab

1. Comp. Graph Optimization
   - ML Model
   - TASSO

2. Parallelization Optimization
   - Device 1
   - Device N
   - FlexFlow

3. Kernel Generation
   - Device 1
   - Device N
   - TVM

https://catalyst.cs.cmu.edu/
Current Rule-based Graph Optimizations

Input Graph

conv3x3 → relu → conv3x3

conv1x1 → relu

add → relu

Input Graph

Rule-based Optimizer

Fuse conv + relu

conv → relu → conv + relu

Optimized Graph

Input

conv3x3

add → relu

conv1x1

+ relu

relu
Current Rule-based Graph Optimizations

TensorFlow currently includes ~200 rules (~53,000 LOC)

Rule-based Optimizer

Fuse conv + relu
Fuse conv + batch normalization
Fuse multi. convs

Current Rule-based Graph Optimizations

TensorFlow currently includes ~200 rules (~53,000 LOC)

Rule-based Optimizer

Fuse conv + relu
Fuse conv + batch normalization
Fuse multi. convs

...
When I turned on XLA (TensorFlow’s graph optimizer), the training speed is **about 20% slower**

With XLA, my program is **almost 2x slower** than without XLA
Limitations of Rule-based Optimizations

Robustness
Experts’ heuristics do not apply to all models/hardware

Scalability
New operators and graph structures require more rules

TensorFlow currently uses ~4K LOC to optimize convolution
Limitations of Rule-based Optimizations

**Robustness**
Experts’ heuristics do not apply to all models/hardware

**Scalability**
New operators and graph structures require more rules

**Performance**
Miss subtle optimizations for specific models/hardware
Motivating Example (ResNet)

The final graph is **30% faster** on V100 but **10% slower** on K80.
Infeasible to manually design graph optimizations for all cases

Is it possible to generate them automatically?
TASO: Tensor Algebra SuperOptimizer

**Key idea**: replace manually-designed graph optimizations with *automated generation and verification* of graph substitutions for tensor algebra

- **Less engineering effort**: 53,000 LOC for manual graph optimizations in TensorFlow → 1,400 LOC in TASO
- **Better performance**: outperform existing optimizers by up to 3x
- **Stronger correctness**: formally verify all generated substitutions
Graph Substitution

\[
Y(n, c, h, w) = \left( \sum_{d,u,v} X(n, d, h + u, w + v) \ast W_1(c, d, u, v) \right) + \left( \sum_{d,u,v} X(n, d, h + u, w + v) \ast W_2(c, d, u, v) \right)
\]

\[\Leftrightarrow Y(n, c, h, w) = \sum_{d,u,v} X(n, d, h + u, w + v) \ast \left( (W_1(c, d, u, v) + W_2(c, d, u, v)) \right)\]
TASO Workflow

Operator Specifications

Graph Subst. Generator

Candidate Substitutions

Graph Subst. Verifier

Verified Substitutions

Graph Optimizer

Input Comp. Graph

Optimized Comp. Graph
Key Challenges

1. How to generate potential substitutions?
   - Graph fingerprints

2. How to verify their correctness?
   - Operator specifications + theorem prover
Graph Substitution Generator

Enumerate all possible graphs up to a fixed size using available operators.
Graph Substitution Generator

66M graphs with up to 4 operators

A substitution = a pair of equivalent graphs

Explicitly considering all pairs does not scale
Graph Substitution Generator

Compute output fingerprints with random input tensors
Graph Substitution Generator

Pairs of graphs with identical fingerprint are candidate substitutions

TASO generates **28,744** substitutions (pruned to **743**) by enumerating graphs with up to 4 operators

*Pruning details available in Z. Jia et al. SOSP’19*
Graph Substitution Verifier

Candidate Substitutions

Graph Subst. Verifier

Verified Substitutions

P1. conv is distributive over concatenation
P2. conv is bilinear
...
Pn.

∀\( x, w_1, w_2 \).
\[ \text{Conv}(x, \text{Concat}(w_1, w_2)) = \text{Concat}(\text{Conv}(x, w_1), \text{Conv}(x, w_2)) \]

Operator Specifications
∀x, w₁, w₂ .
(\text{Conv}(x, w₁), \text{Conv}(x, w₂))
= \text{Split(\text{Conv}(x, \text{Concat}(w₁, w₂)))}

P1. ∀x, w₁, w₂ .
\text{Conv}(x, \text{Concat}(w₁, w₂)) =
\text{Concat(\text{Conv}(x, w₁), \text{Conv}(x, w₂))}
P2. ...

**Operator Specifications**
TASO generates all 743 substitutions in 5 minutes, and verifies them against 43 operator properties in 10 minutes.

Supporting a new operator requires a few hours of human effort to specify its properties.

Operator specifications in TASO ≈ 1,400 LOC
Manual graph optimizations in TensorFlow ≈ 53,000 LOC
Search-Based Graph Optimizer

Cost Model
- Based on individual operators’ cost
- Measure the cost of each operator on hardware

Search-Based Graph Optimizer
- Cost-based backtracking search
- Optimizing an ML model takes less than 10 minutes

Input Comp. Graph

Verified Substitutions

Optimized Comp. Graph

End-to-end Inference Performance (Nvidia V100 GPU)

- **TensorFlow**: 1.0x
- **TensorFlow XLA**: 1.3x
- **TensorRT**: 3.1x
- **TASO**: 1.6x

**Competitive on standard models**

**Larger speedups on emerging models**

- ResNet-50
- NasNet-A
- ResNeXt-50
- NasRNN
- BERT-Large

- **Runtime (ms)**
  - 0
  - 3
  - 6
  - 9
  - 12
  - 15
First DNN graph optimizer that automatically generates substitutions

• Less engineering effort
• Better runtime performance
• Stronger correctness guarantee

1. TASO: Optimizing Deep Learning Computation with Automated Generation of Graph Substitutions. SOSP’19.
2. Optimizing DNN Computation with Relaxed Graph Substitutions. MLSys’19.
Can we improve TASO?

TASO only discovers fully equivalent subst. Can we use subst. that are partially equivalent and correct the results afterwards?
Motivating Example: Partially Equivalent Substs

(a) Input program.

(b) A partially equivalent transformation.

(c) Correcting results.
PET: Partially Equivalent Substs and Auto Corrections

Up to 2.5x faster than TASO

Generate both fully and partially equivalent substs

Auto-correct substs to maintain end-to-end equivalence

Optimize DNNs using both fully and partially equivalent substs
Can we improve TASO?

TASO uses backtracking search. Can we find an optimal solution in the search space?
Equality Saturation for TASO

- Key idea: a new representation that can express all possible computation graphs at once

Source: (matmul ?input_1 ?input_2), (matmul ?input_1 ?input_3)
Target: (split_0 (split 1 (matmul ?input_1 (concat_2 1 ?input_2 ?input_3))))
(split_1 (split 1 (matmul ?input_1 (concat_2 1 ?input_2 ?input_3))))

The optimal graph is **16%** better.
The search is **48x** faster.
Can we apply TASO to other problem domains?

- **Cross-optimizations between ML and DB operations**
  - Why: ML and DB operations are optimized separately in today’s DB systems
  - How: automatically generate co-optimizations of linear algebra and relational algebra operations

- **Optimizing Compilers for Quantum Computing**
  - Why: today’s quantum machines support different sets of instructions → impossible to manually design optimizations for all quantum architectures
  - How: automatically generate quantum program transformations given a set of instructions

- **Others?**
Automated Discovery of ML Optimizations

1. Comp. Graph Optimization

TASO

2. Parallelization

Device 1

Device N

FlexFlow
Stochastic Gradient Descent (SGD)

Train ML models through many iterations of 3 stages

1. **Forward propagation**: apply model to a batch of input samples and run calculation through operators to produce a prediction

2. **Backward propagation**: run the model in reverse to produce error for each trainable weight

3. **Weight update**: use the loss value to update model weights
Stochastic Gradient Descent (SGD)

Train ML models through many iterations of 3 stages

1. **Forward propagation**: apply model to a batch of input samples and run calculation through operators to produce a prediction

2. **Backward propagation**: run the model in reverse to produce error for each trainable weight

3. **Weight update**: use the loss value to update model weights
Stochastic Gradient Descent (SGD)

Train ML models through many iterations of 3 stages

1. **Forward propagation**: apply model to a batch of input samples and run calculation through operators to produce a prediction

2. **Backward propagation**: run the model in reverse to produce error for each trainable weight

3. **Weight update**: use the loss value to update model weights

\[ w_i := w_i - \gamma \nabla L(w_i) = w_i - \frac{\gamma}{n} \sum_{j=1}^{n} \nabla L_j(w_i) \]
Current Strategies to Parallelize ML Training: Data and Model Parallelism

\[ w_i := w_i - \gamma \nabla L(w_i) = w_i - \frac{\gamma}{n} \sum_{j=1}^{n} \nabla L_j(w_i) \]
Current Strategies to Parallelize ML Training: Data and Model Parallelism

Model Parallelism

\[ w_i := w_i - \gamma \nabla L(w_i) = w_i - \frac{\gamma}{n} \sum_{j=1}^{n} \nabla L_j(w_i) \]
Are there strategies beyond data/model parallelism? Can we discover fast ones automatically?
FlexFlow: Automated Search for Fast Strategies

Define a search space of possible parallelization strategies

+ A cost model and a search algorithm

= Optimized Parallelization strategies
The SOAP Search Space

- Samples
- Operators
- Attributes
- Parameters
The SOAP Search Space

- Samples: partitioning training samples (Data Parallelism)
- Operators
- Attributes
- Parameters

Parallelizing a 1D convolution in Sample
The SOAP Search Space

- **Samples**: partitioning training samples (Data Parallelism)
- **Operators**: partitioning ML operators (Model Parallelism)
- **Attributes**
- **Parameters**

Parallelizing multiple convolutions in **Operator**
The SOAP Search Space

- **Samples**: partitioning training samples (Data Parallelism)
- **Operators**: partitioning ML operators (Model Parallelism)
- **Attributes**: partitioning attributes in a sample (e.g., pixels)
- **Parameters**

Parallelizing a 1D convolution in **Attribute**
The SOAP Search Space

- **Samples**: partitioning training samples (Data Parallelism)
- **Operators**: partitioning ML operators (Model Parallelism)
- **Attributes**: partitioning attributes in a sample (e.g., pixels)
- **Parameters**: partitioning parameters in an operator

Parallelizing a 1D convolution in **Parameter**
**Hybrid Parallelism in SOAP**

Different strategies perform the same computation.
Automated Approaches to Accelerate Machine Learning

Zhihao Jia | Stanford University

Data parallelism

Parameter

Sample

Data parallelism

A parallelization strategy in SOAP *(1.2x faster)*
Automated Approaches to Accelerate Machine Learning

Zhihao Jia | Stanford University

Parameter

Sample

GPU 1
GPU 2
GPU 3
GPU 4

Data parallelism

A parallelization strategy in SOAP (1.2x faster)
Challenges of Discovering Fast Strategies in SOAP

1. SOAP contains billions or more possible strategies

MCMC search algorithm

2. Evaluating a strategy on hardware is too slow

Execution simulator
FlexFlow Overview

- **Computation Graph**
  - MatMul
  - Concat
  - Conv
  - Pool

- **Hardware Topology**
  - CPU
  - GPU

- **Execution Simulator**
  - (Cost Model)

- **MCMC Search Alg.**

- **Distributed Runtime**

- **FlexFlow**
Deep Learning Recommendation Model (DLRM)
A deep learning model for ads recommendation

![Graph showing training throughput vs number of GPUs with Data Parallelism, Model Parallelism, and FlexFlow compared. Data Parallelism and Model Parallelism show gradual improvement, while FlexFlow shows a significant 10x speedup as the number of GPUs increases.](image)
FlexFlow: Automatically Discover Fast Parallelization for DNNs

https://flexflow.ai

Performance Autotuning
FlexFlow accelerates DNN training by automatically discovering fast parallelization strategies for a specific parallel machine.

Keras Support
FlexFlow provides a drop-in replacement for TensorFlow Keras and requires only a few lines of changes to existing Keras programs.

Large-Scale GNNs
FlexFlow enables fast graph neural network training and inference on large-scale graphs by exploring attribute parallelism.

Hardware Topology

Fast Parallel Strategy

[Image] Hardware Topology

[Image] Fast Parallel Strategy

[Image] TensorFlow

[Image] PyTorch

[Image] ONNX

[Image] Roc [MLSys’20]

[Image] https://flexflow.ai
Can we improve FlexFlow?

- FlexFlow takes network topology as an input.
- Can we co-optimize network topology and parallelization strategies?
Testbed with Reconfigurable Network Topology
Joint Optimization of Parallelization Strategy and Network Topology

Up to 5.7x faster than FlexFlow w/ Fat-tree interconnect

Optimizing the Network Topology for Distributed DNN Training. W. Wang et al.
Automated Approaches to Accelerate Machine Learning

Zhihao Jia | Stanford University

Automated Machine Learning Systems

1. Comp. Graph Optimization
   - ML Model
   - TASO

2. Parallelization Optimization
   - Device 1
   - Device N
   - FlexFlow

3. Kernel Generation
   - Device 1
   - Device N
   - TVM

https://catalyst.cs.cmu.edu/