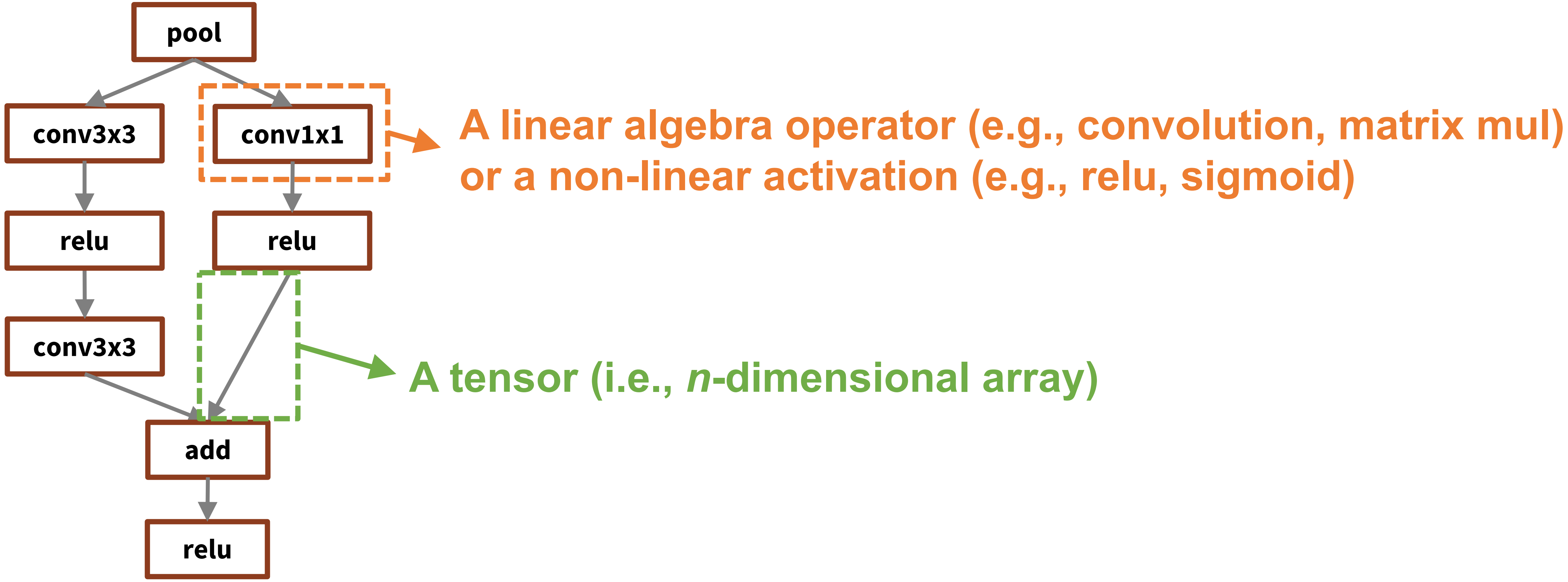
PET:
Optimizing Tensor Programs with Partially Equivalent Transformations and Automated Corrections

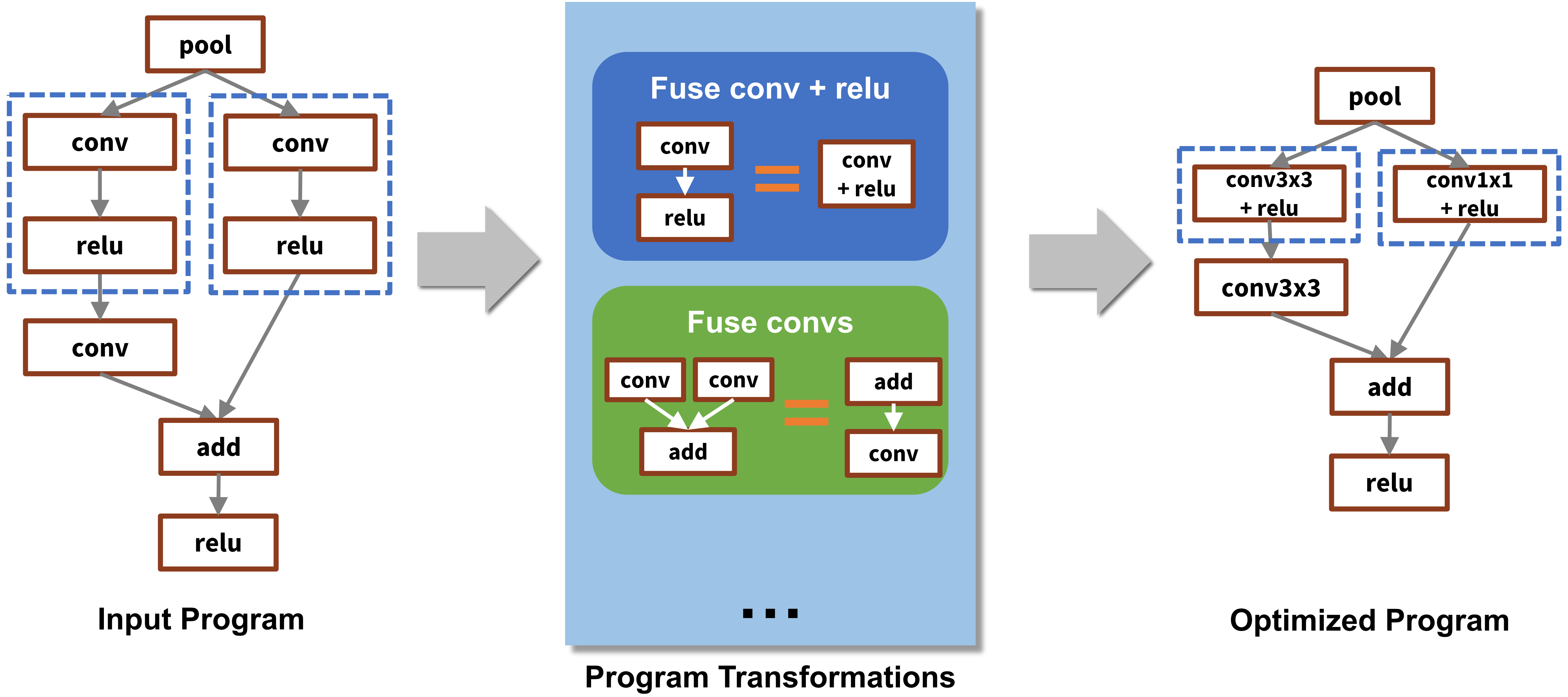
Haojie Wang, Jidong Zhai, Mingyu Gao, Zixuan Ma, Shizhi Tang,
Liyan Zheng, Yuanzhi Li, Kaiyuan Rong, Yuanyong Chen, **Zhihao Jia**

Tsinghua University Carnegie Mellon University Facebook

Tensor Program

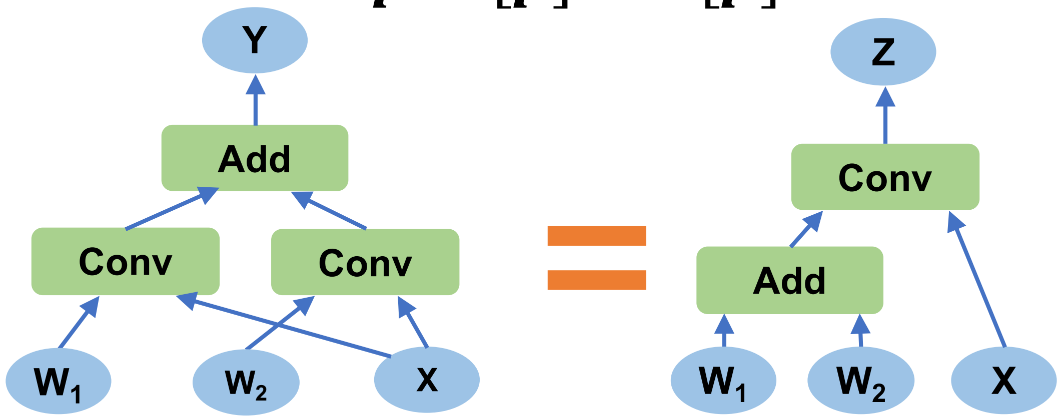


Tensor Program Transformations



Current Systems Consider only Fully Equivalent Transformations

$$\forall \boldsymbol{p}.\ \boldsymbol{Y}[\boldsymbol{p}] = \boldsymbol{Z}[\boldsymbol{p}]$$

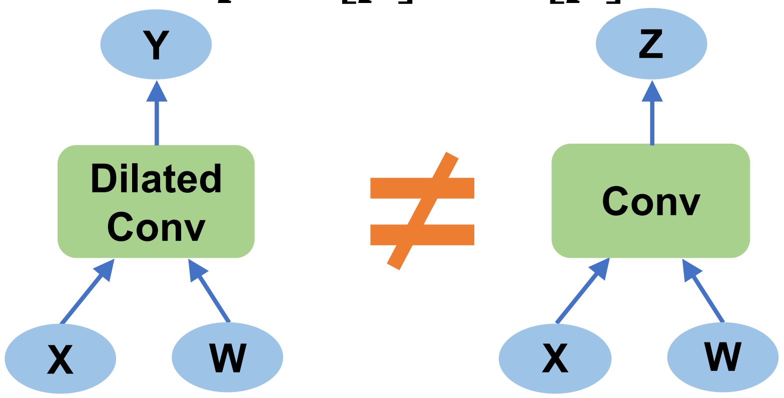


Fully Equivalent Transformations

Pro: preserve functionality

Con: miss optimization opportunities

$$\exists \boldsymbol{p}.\ \boldsymbol{Y}[\boldsymbol{p}] \neq \boldsymbol{Z}[\boldsymbol{p}]$$



Partially Equivalent Transformations

Pro: better performance

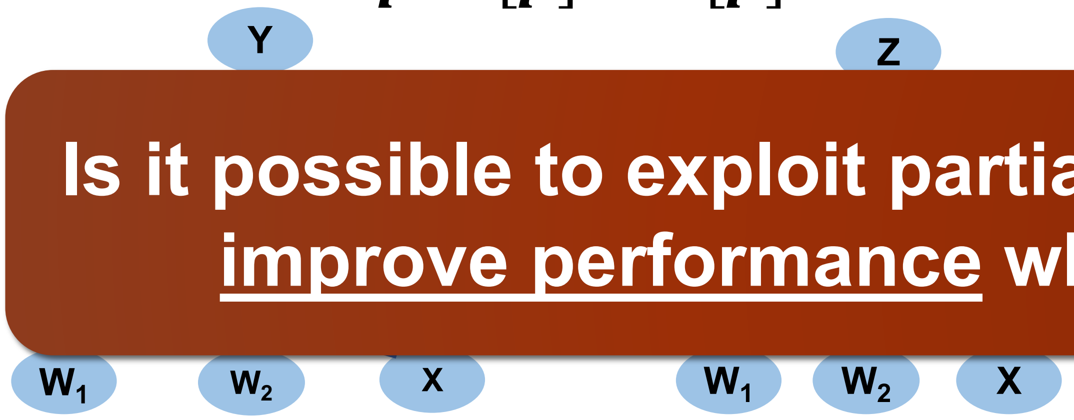
- Faster ML operators
- More efficient tensor layouts
- Hardware-specific optimizations

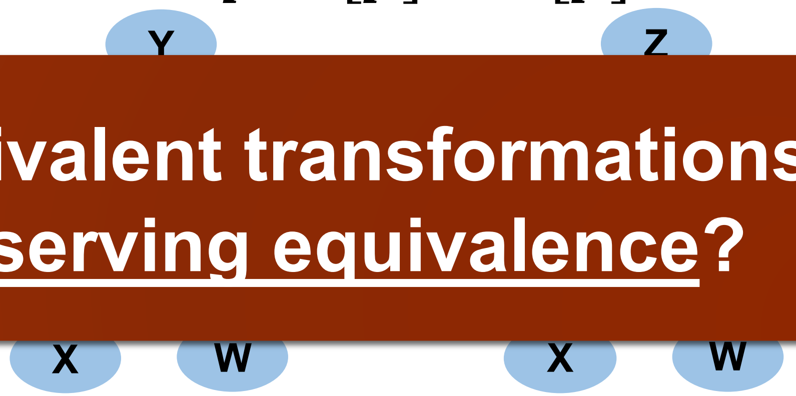
Con: potential accuracy loss

Current Systems Consider only Fully Equivalent Transformations

$\forall p.\ Y[p] = Z[p]$

$\exists p.\ Y[p] \neq Z[p]$





Is it possible to exploit partially equivalent transformations to <u>improve performance</u> while <u>preserving equivalence</u>?

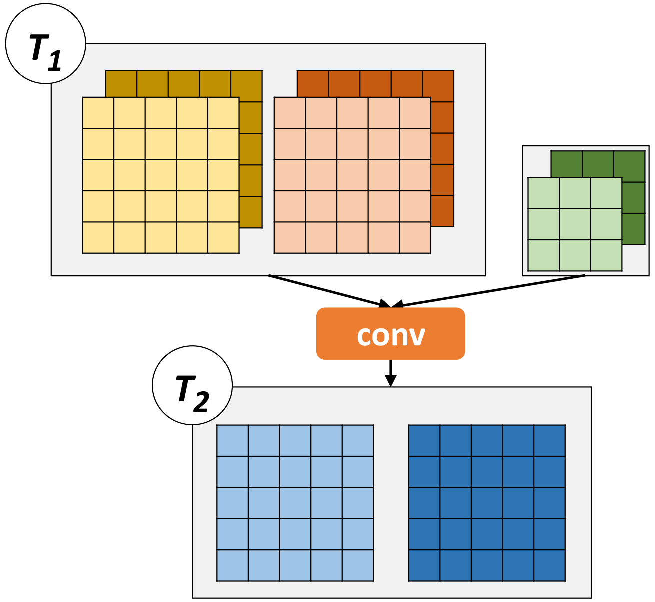
Fully Equivalent Transformations

- Pro: preserve functionality
- Con: miss optimization opportunities

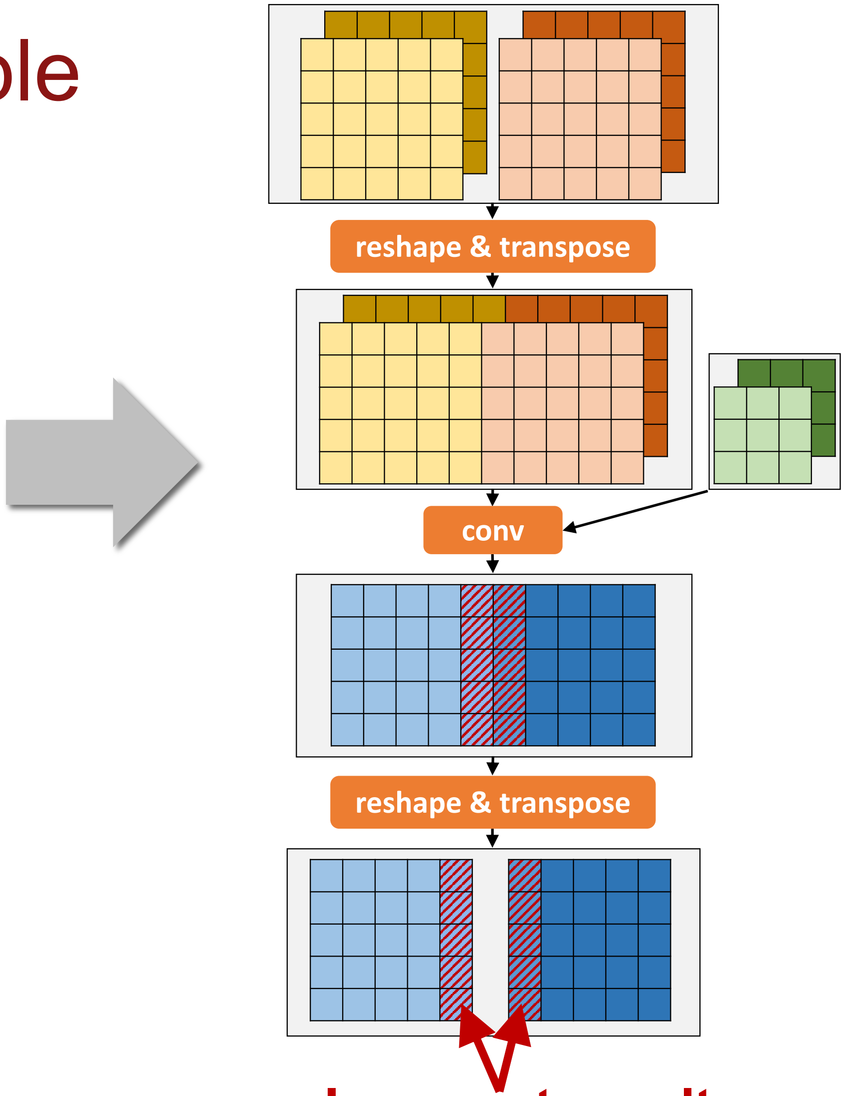
Partially Equivalent Transformations

- Pro: better performance
  - Faster ML operators
  - More efficient tensor layouts
  - Hardware-specific optimizations
- Con: potential accuracy loss

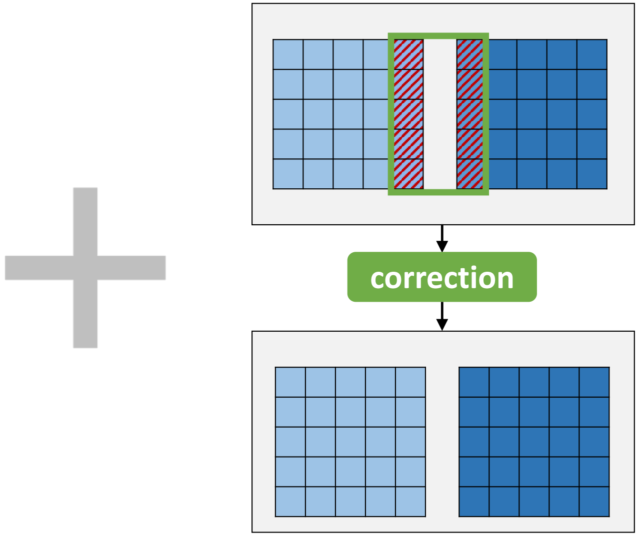
Motivating Example



Input Program

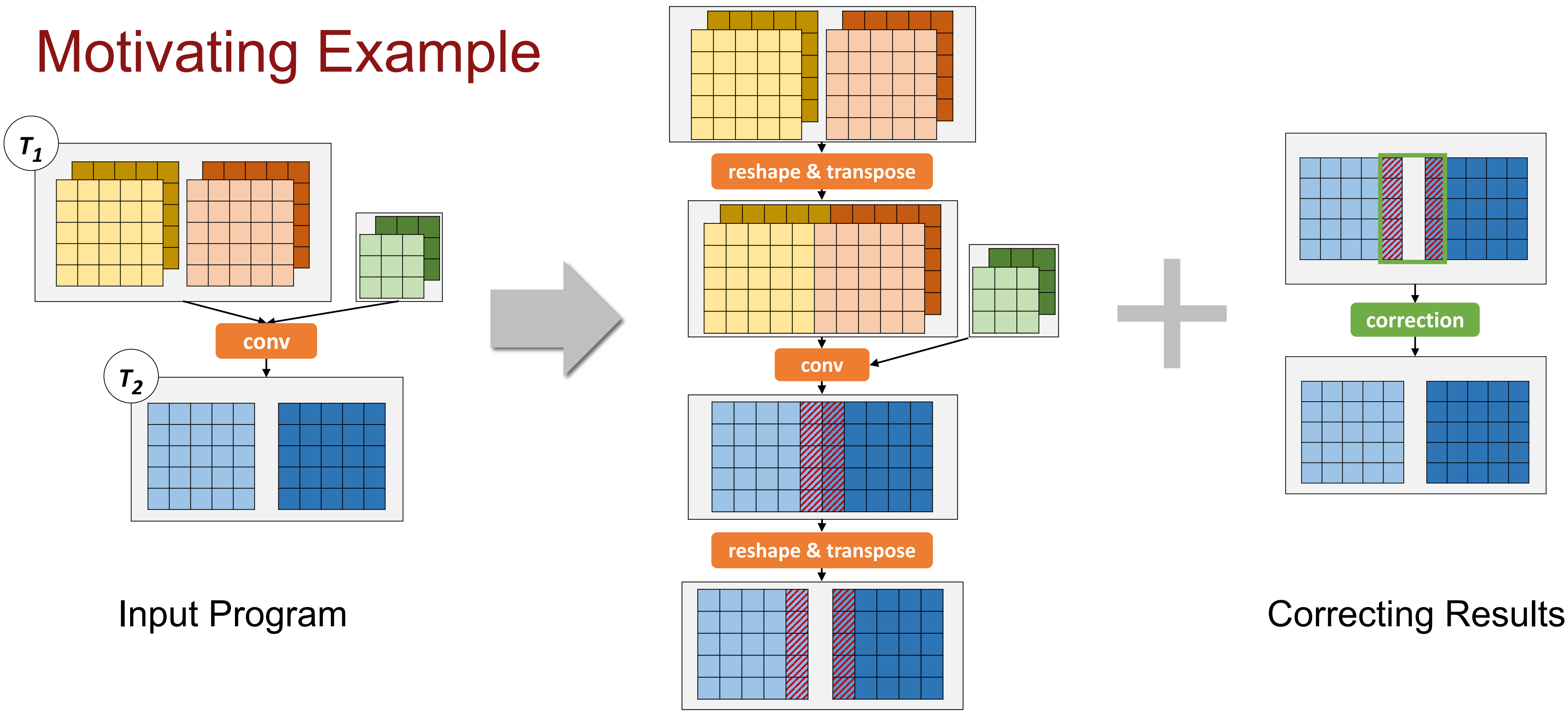


Partially Equivalent Transformation



Correcting Results

Motivating Example



Input Program

Correcting Results

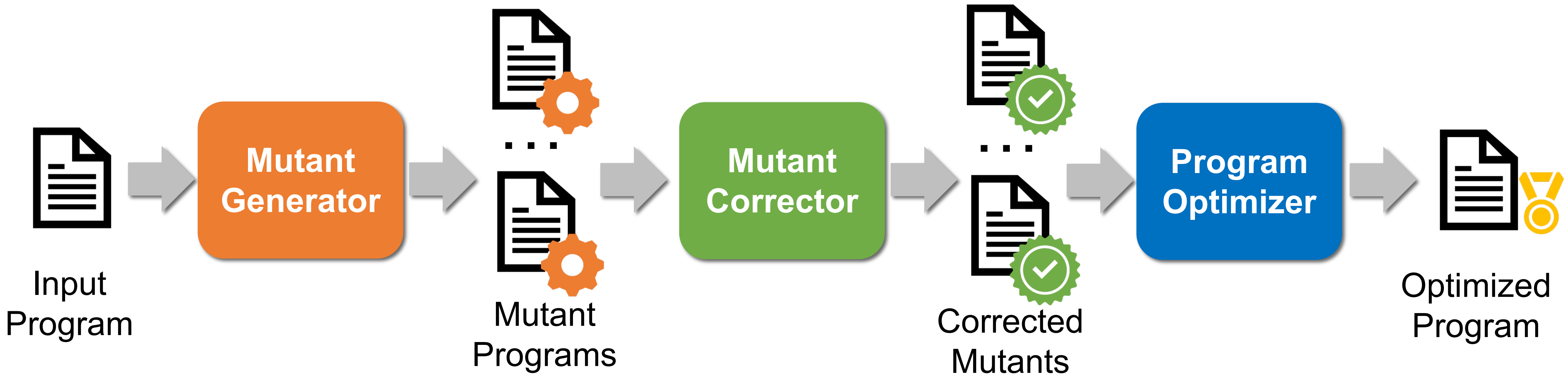
- Transformation and correction lead to **1.2x** speedup for ResNet-18
- Correction preserves end-to-end equivalence

PET

- **First tensor program optimizer** with partially equivalent transformations

- **Larger optimization space** by combining fully and partially equivalent transformations
- **Better performance**: outperform existing optimizers by up to **2.5x**
- **Correctness**: automated corrections to preserve end-to-end equivalence

PET Overview



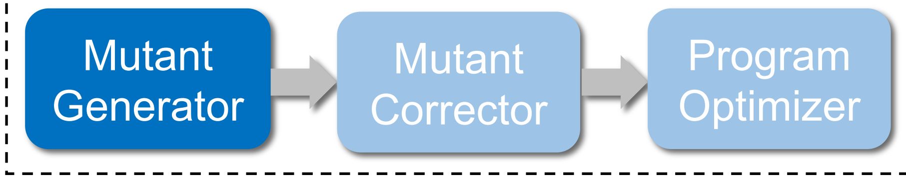
Key Challenges

1. How to generate partially equivalent transformations?

Superoptimization

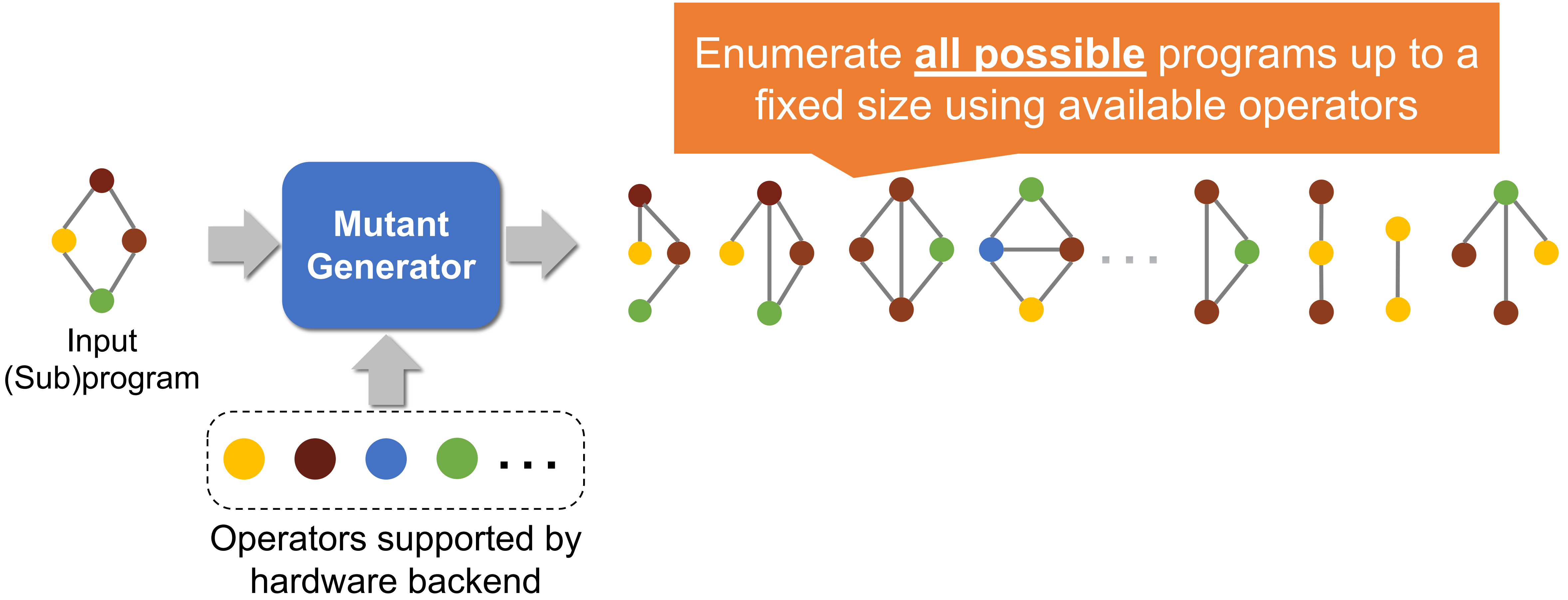
2. How to correct them?

Multi-linearity of DNN computations



Mutant Generator

Superoptimization adapted from TASO[1]

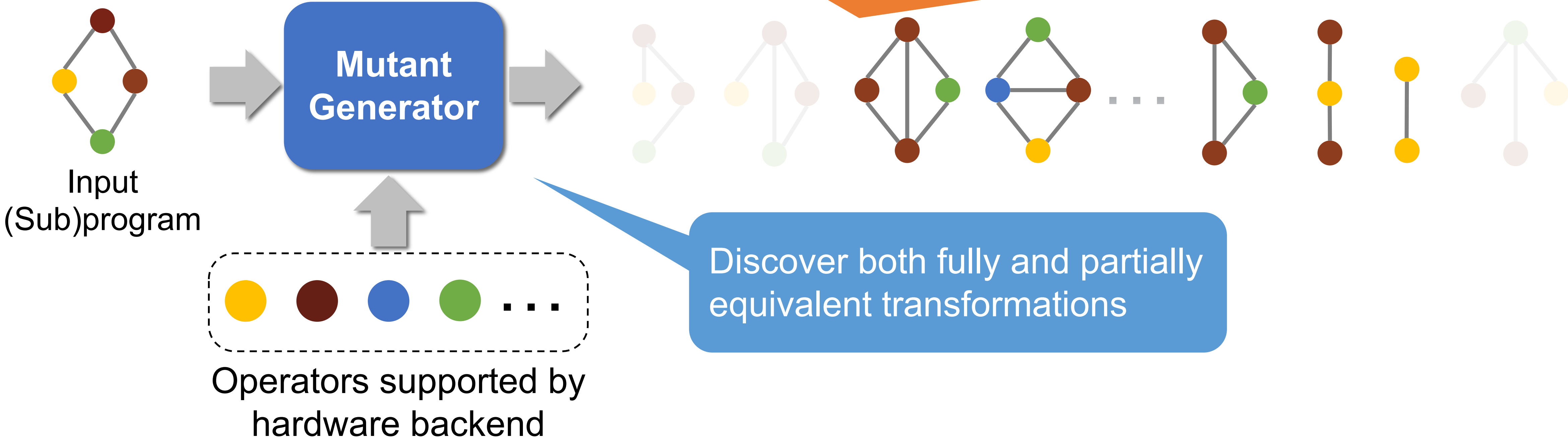


1. TASO: Optimizing Deep Learning Computation with Automated Generation of Graph Substitutions. SOSP'19.

Mutant Generator

Superoptimization adapted from TASO[1]

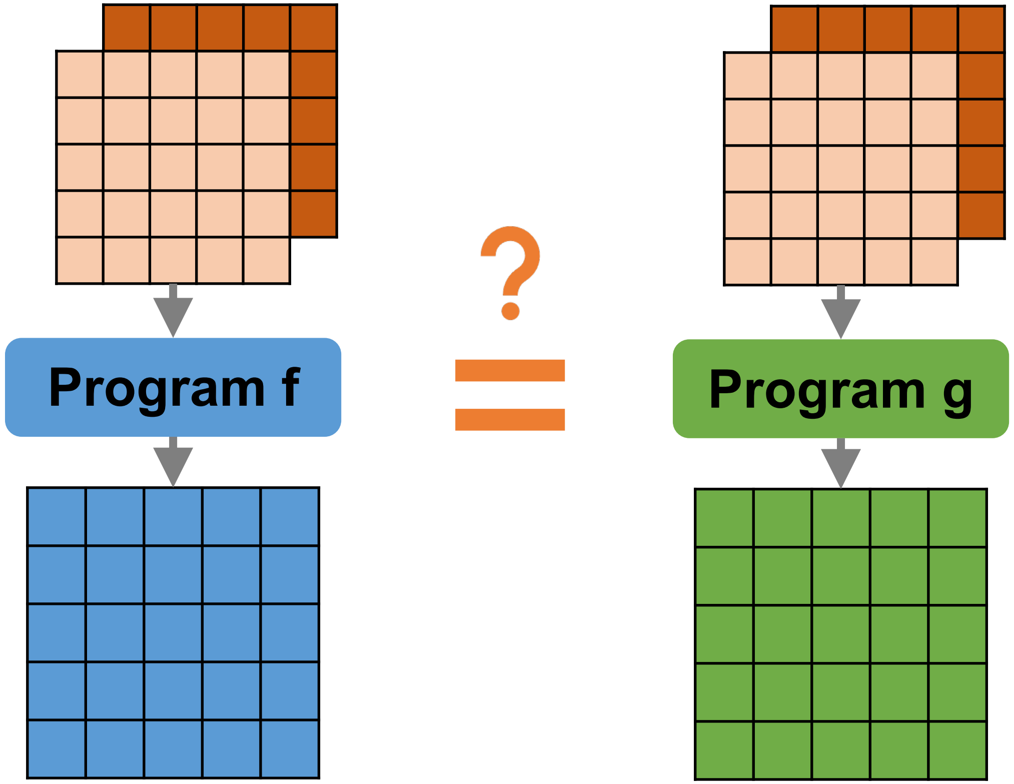
Programs with the same input/output shapes are potential mutants



Discover both fully and partially equivalent transformations

1. TASO: Optimizing Deep Learning Computation with Automated Generation of Graph Substitutions. SOSP'19.

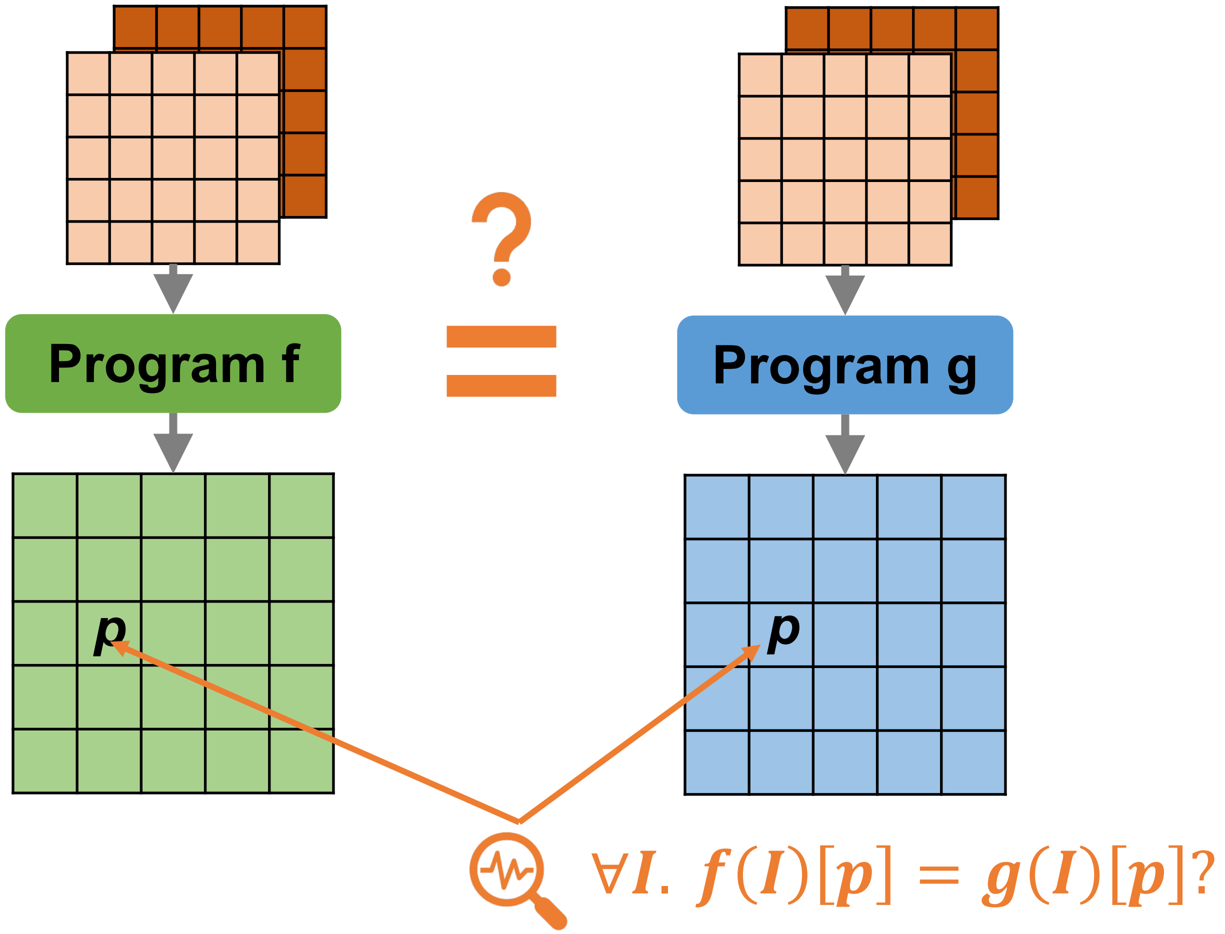
Challenges: Examine Transformations



1. Which part of the computation is not equivalent?
2. How to correct the results?

A Strawman Approach

- **Step 1**: Explicitly consider all output positions (m positions)
- **Step 2**: For each position ***p***, examine all possible inputs (n inputs)



Require O(m * n) examinations, but both m and n are too large to explicitly enumerate

Multi-Linear Tensor Program (MLTP)

- A program $\boldsymbol{f}$ is multi-linear if the output is linear to all inputs
  - $\boldsymbol{f}(I_1, \ldots, X, \ldots, I_n) + \boldsymbol{f}(I_1, \ldots, Y, \ldots, I_n) = \boldsymbol{f}(I_1, \ldots, X + Y, \ldots, I_n)$
  - $\alpha \cdot \boldsymbol{f}(I_1, \ldots, X, \ldots, I_n) = \boldsymbol{f}(I_1, \ldots, \alpha \cdot X, \ldots, I_n)$

- DNN computation = MLTP + non-linear activations

Majority of the computation

O(m * n) examinations in strawman approach

MLTP

O(1) examinations in PET's approach

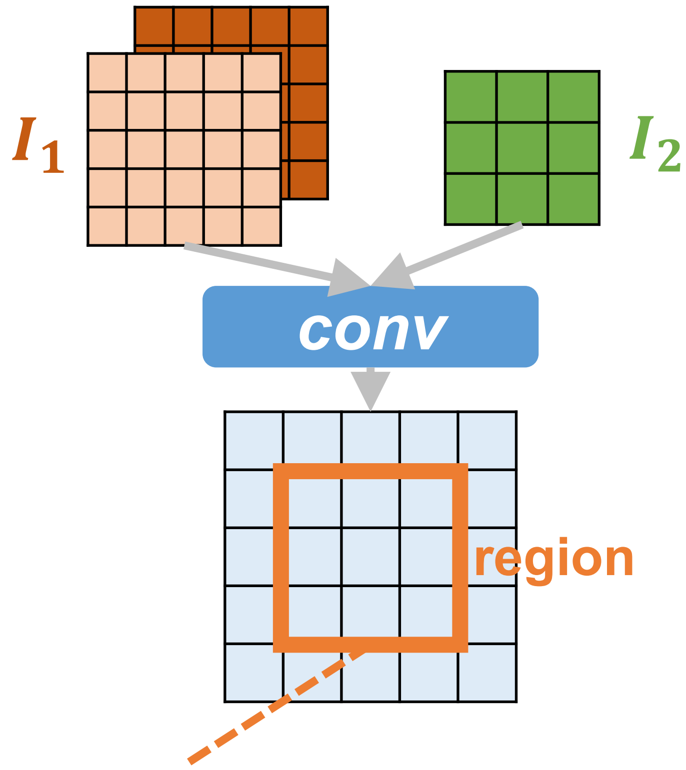
No Need to Enumerate All Output Positions

Group all output positions with an identical summation interval into a region

***Theorem 1**: For two MLTPs **f** and **g**, if **f=g** for O(1) positions in a region, then **f=g** for all positions in the region

Only need to examine O(1) positions for each region.

Complexity: **O(m * n) → O(n)**



$$conv(c,h,w) = \boxed{\sum_{d=0}^{D-1} \sum_{x=-1}^{1} \sum_{y=-1}^{1}} I_1(d, h+x, w+y) \times I_2(d, c, x, y)$$

Summation interval

*Proof details available in the paper

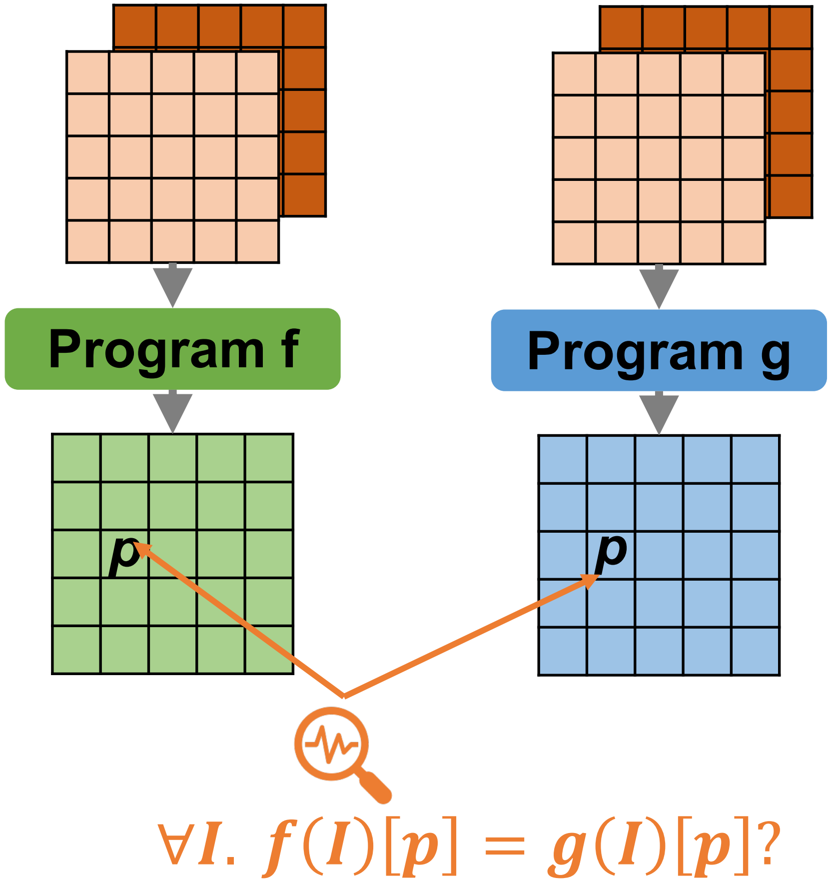
No Need to Consider All Possible Inputs

Examining equivalence for a single position is still challenging

***Theorem 2**: If $\exists I.\ f(I)[p] \neq g(I)[p]$, then the probability that **f** and **g** give identical results on t random integer inputs is $(\frac{1}{2^{31}})^t$

Run t random tests for each position p

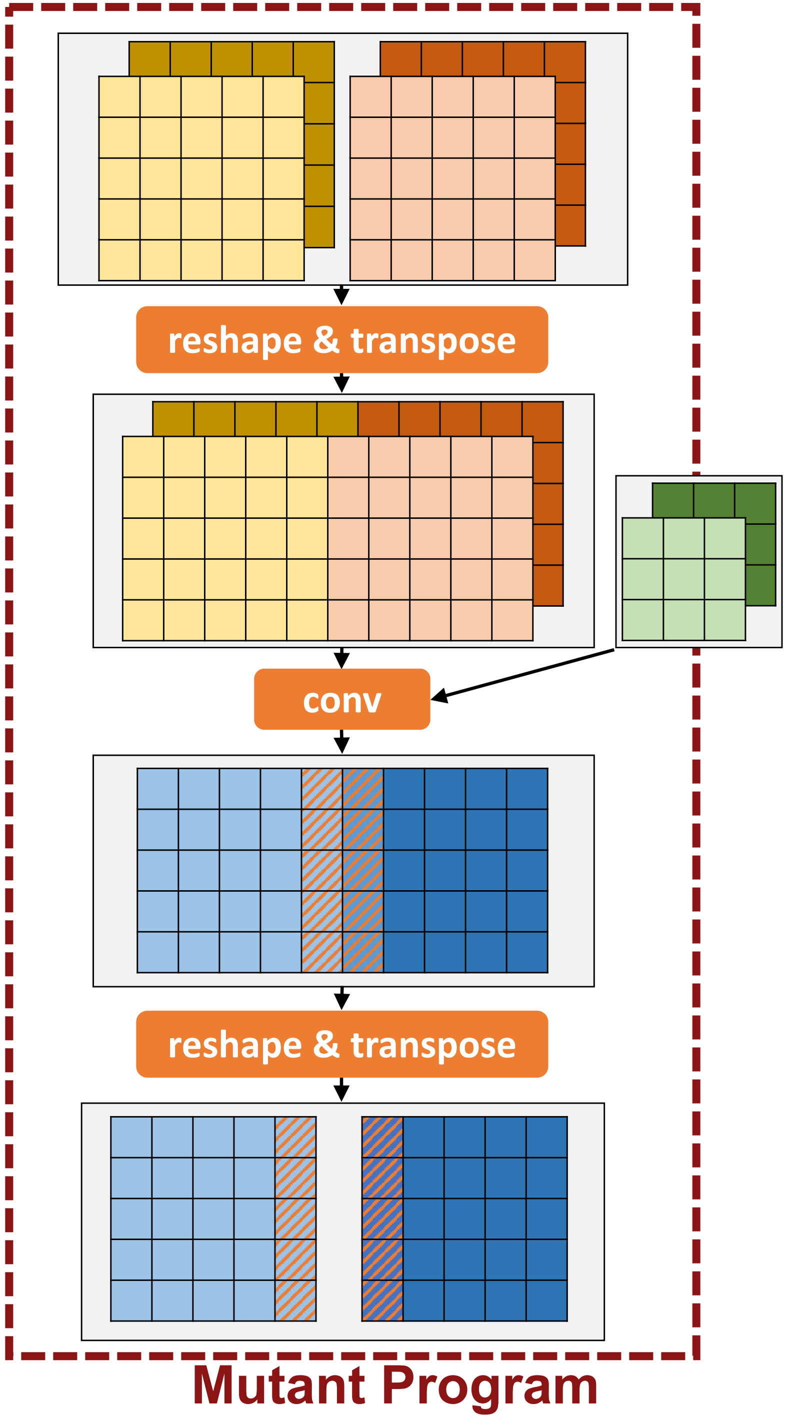
Complexity: **O(n) → O(t) = O(1)**



*Proof details available in the paper

Mutant Corrector

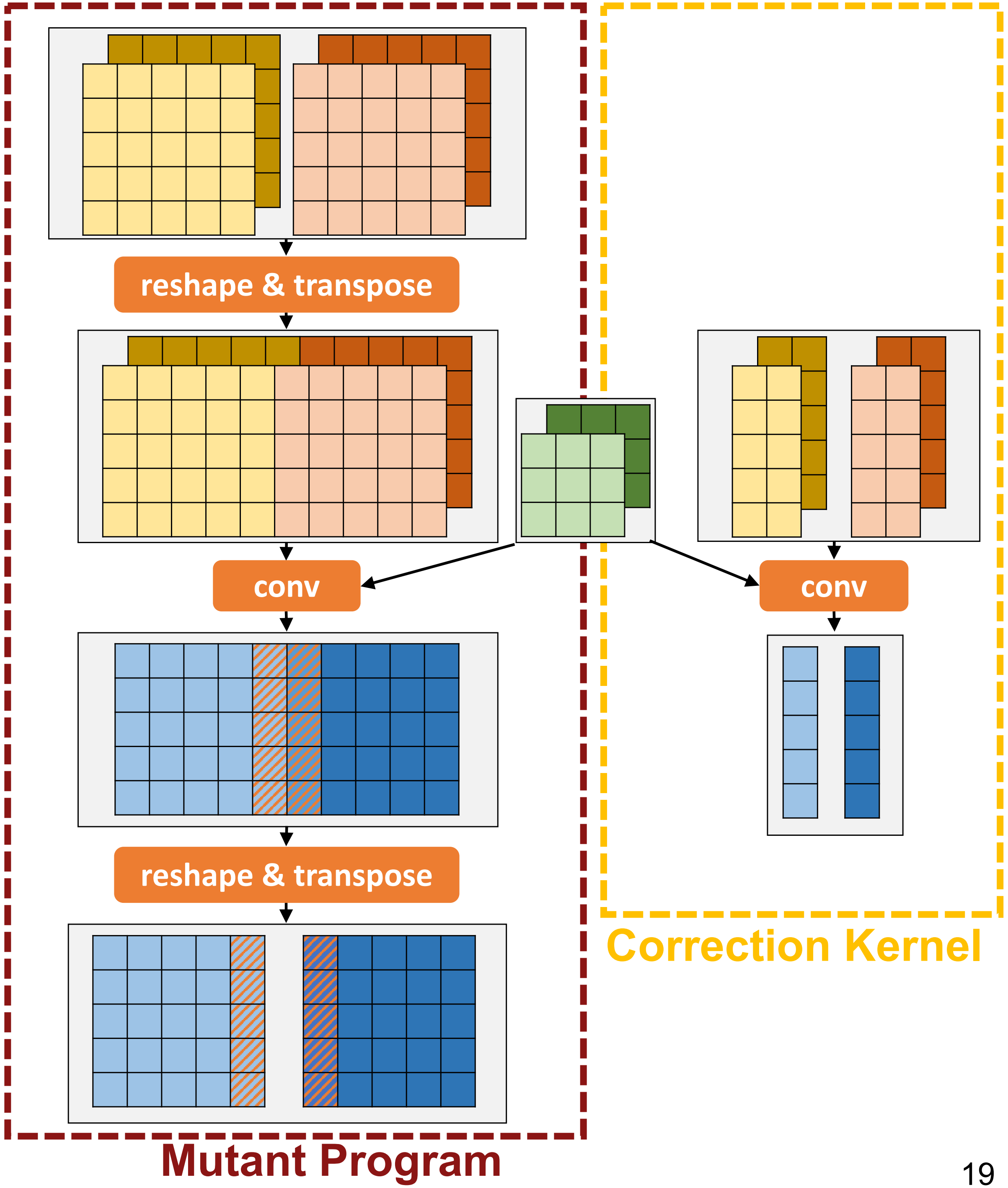
Goal: **quickly** and **efficiently** correcting the outputs of a mutant program



Mutant Corrector

Goal: **quickly** and **efficiently** correcting the outputs of a mutant program

Step 1: recompute the incorrect outputs using the original program



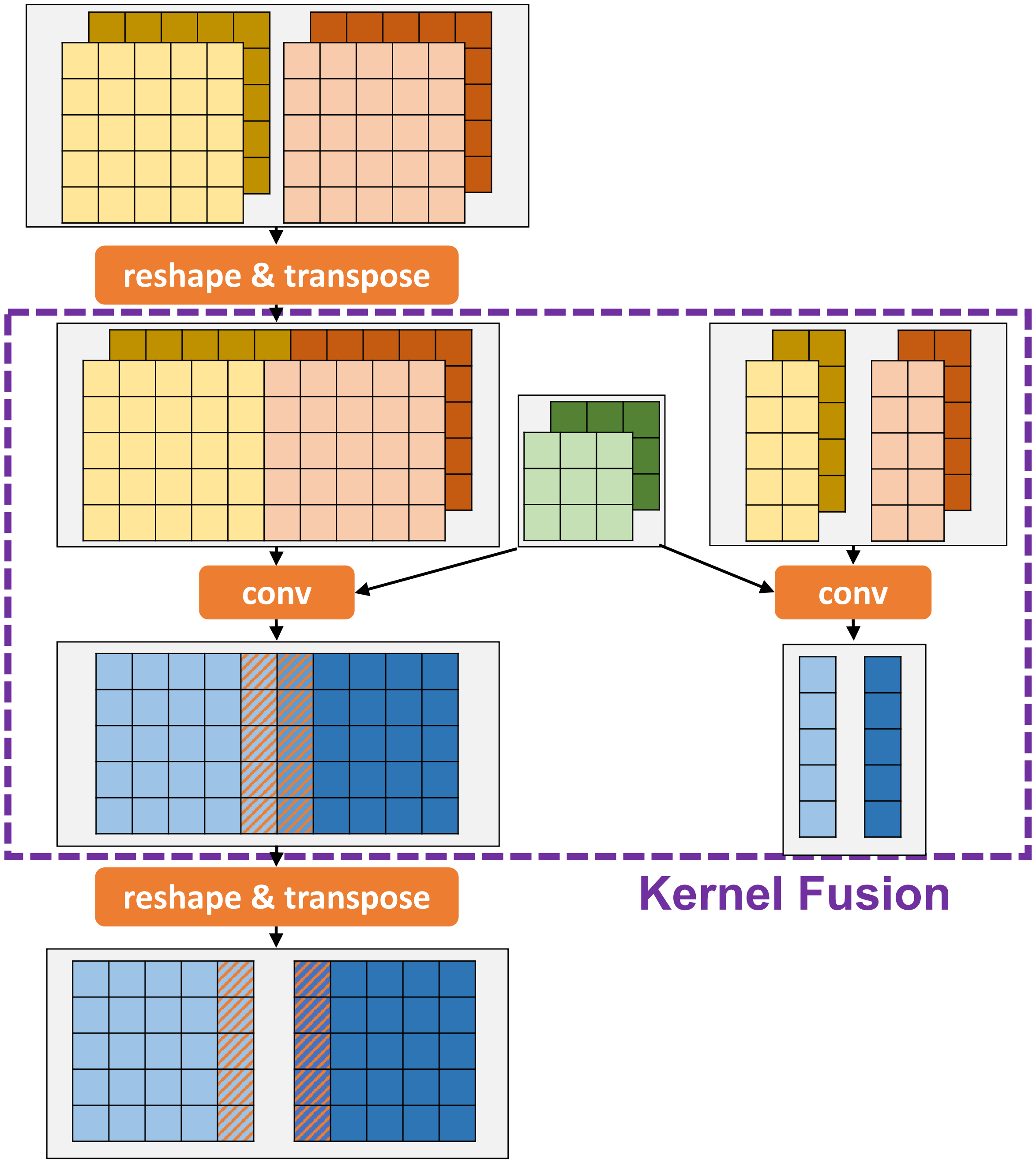
Mutant Corrector

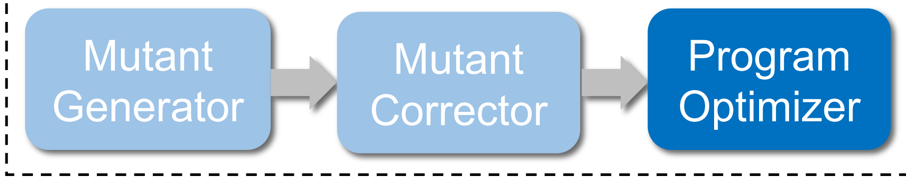
Goal: **quickly** and **efficiently** correcting the outputs of a mutant program

Step 1: recompute the incorrect outputs using the original program

Step 2: opportunistically fuse correction kernels with other operators

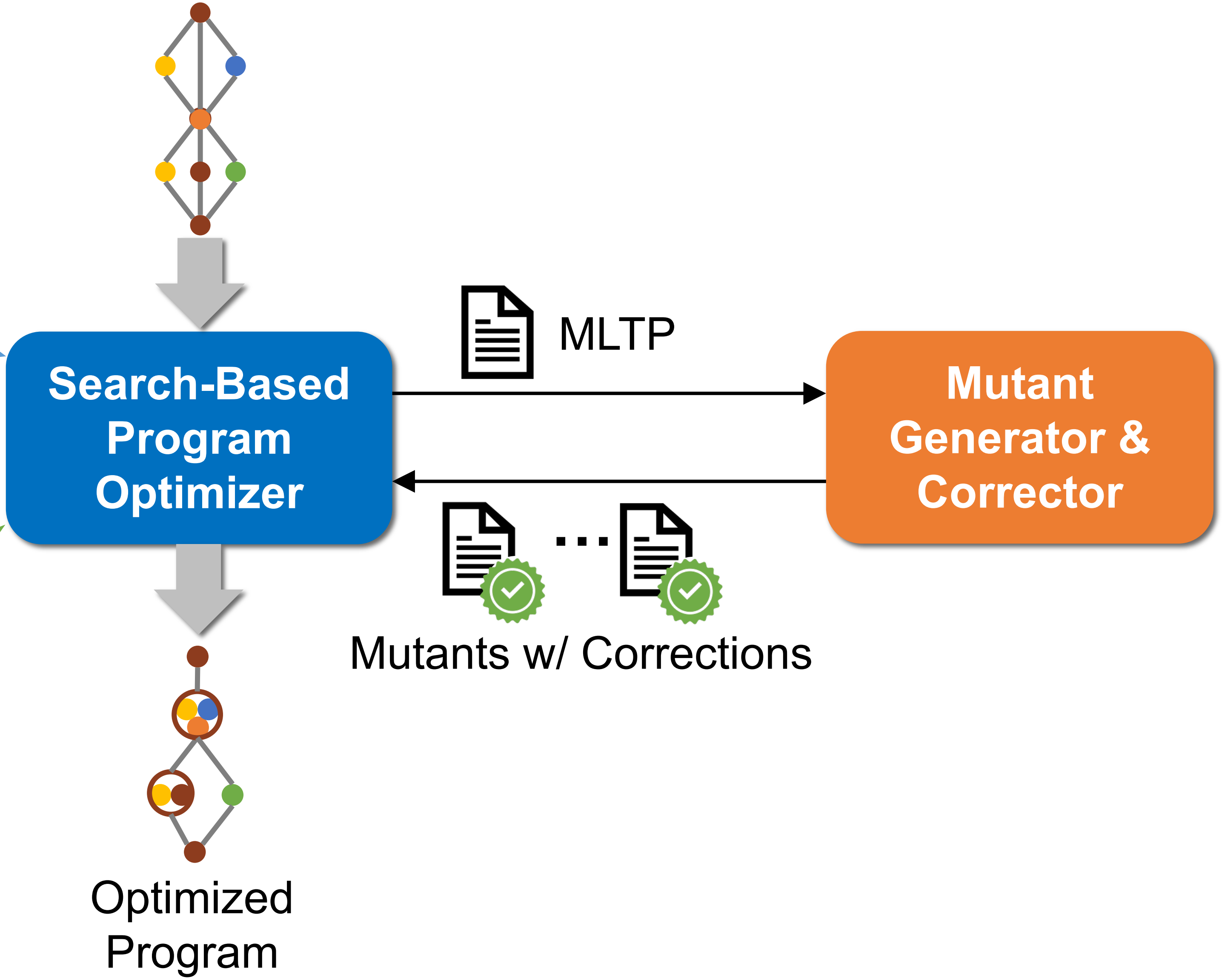
Correction introduces less than <u>1%</u> overhead





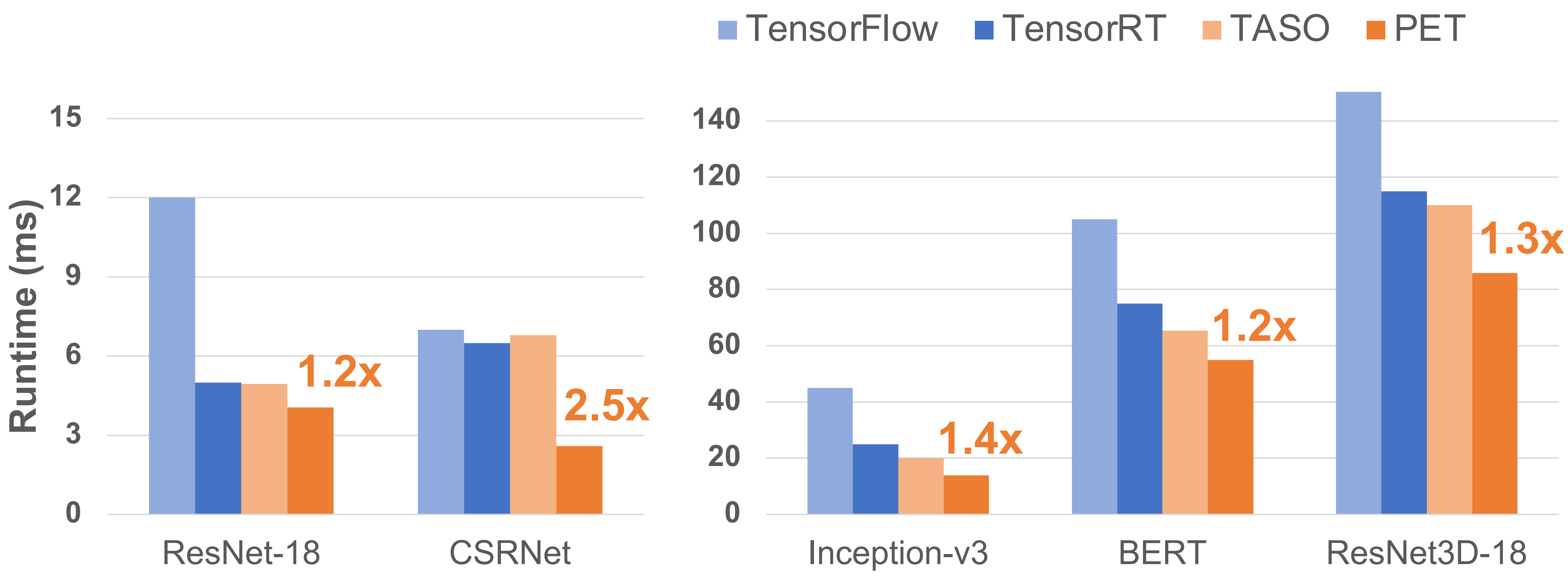
Program Optimizer

- **Beam search**
- Optimizing a DNN architecture takes less than <u>30</u> minutes



Other optimizations:
- Operator fusion
- Constant folding
- Redundancy elimination

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



PET outperforms existing optimizers by 1.2-2.5x
by combining fully and partially equivalent transformations

More Evaluation in Paper

1. A case study on tensor-, operator-, and graph-level optimizations discovered by PET

2. Both fully and partially equivalent transformations are critical to performance

3. PET consistently outperforms existing optimizers on various backends (cuDNN/cuBLAS, TVM, Ansor)

4. Partially equivalent transformations w/ corrections can directly benefit existing optimizers

PET

- A tensor program optimizer with partially equivalent transformations and automated corrections

- **Larger optimization space** by combining fully and partially equivalent transformations
- **Better performance**: outperform existing optimizers by up to **2.5x**
- **Correctness**: automated corrections to preserve end-to-end equivalence

Available at: https://github.com/thu-pacman/PET

zhihao@cmu.edu

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED