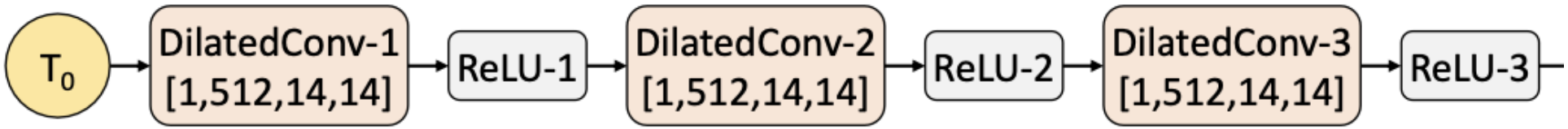
PET: Optimizing Tensor Programs with Partially Equivalent Transformations and Automated Corrections

KH. Wang, J. Zhai, M. Gao, Z. Ma, S. Tang, L. Zheng, Y. Li, K. Rong, Y. Chen, and Z. Jia

Reviewed by Andy (Wenyang) Zhou

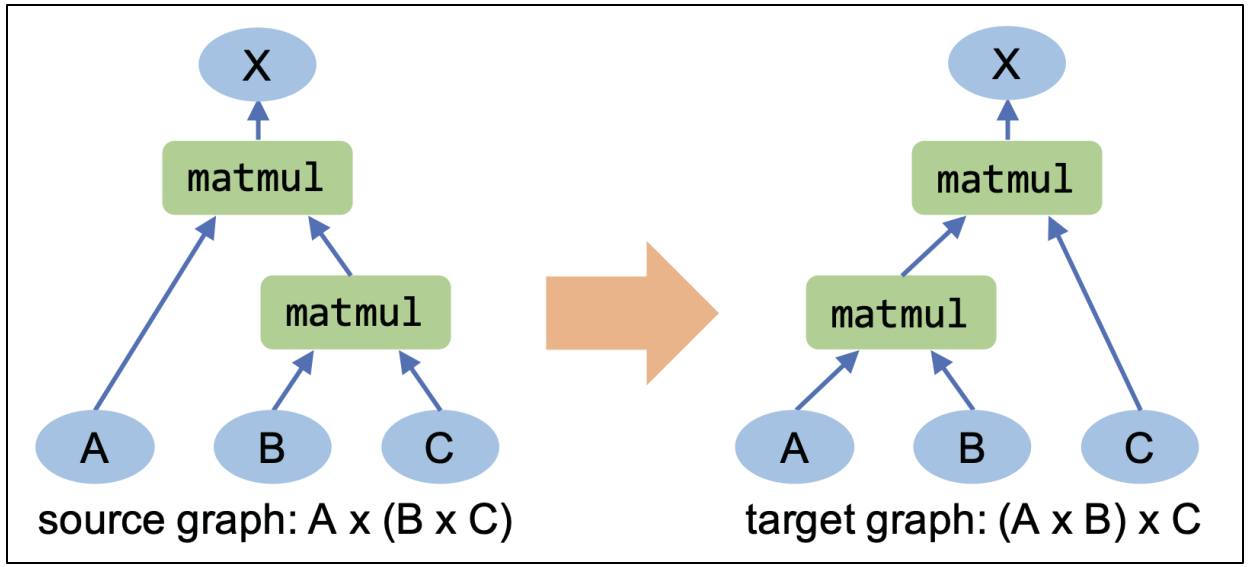
Tensor Programs

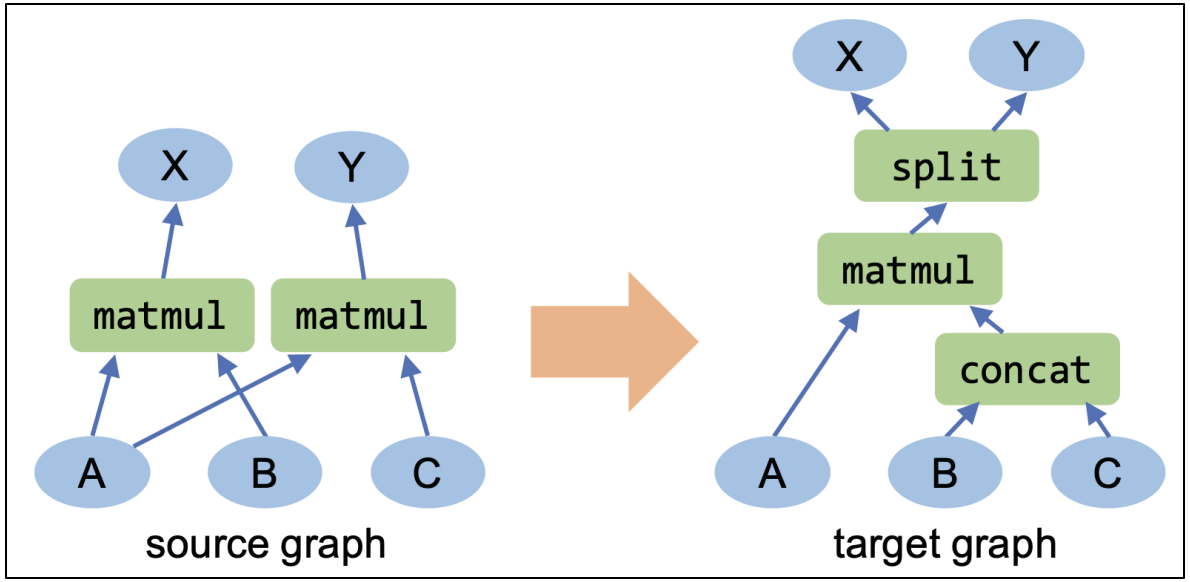
- Program represented as DAG of computation nodes



Graph Transformation

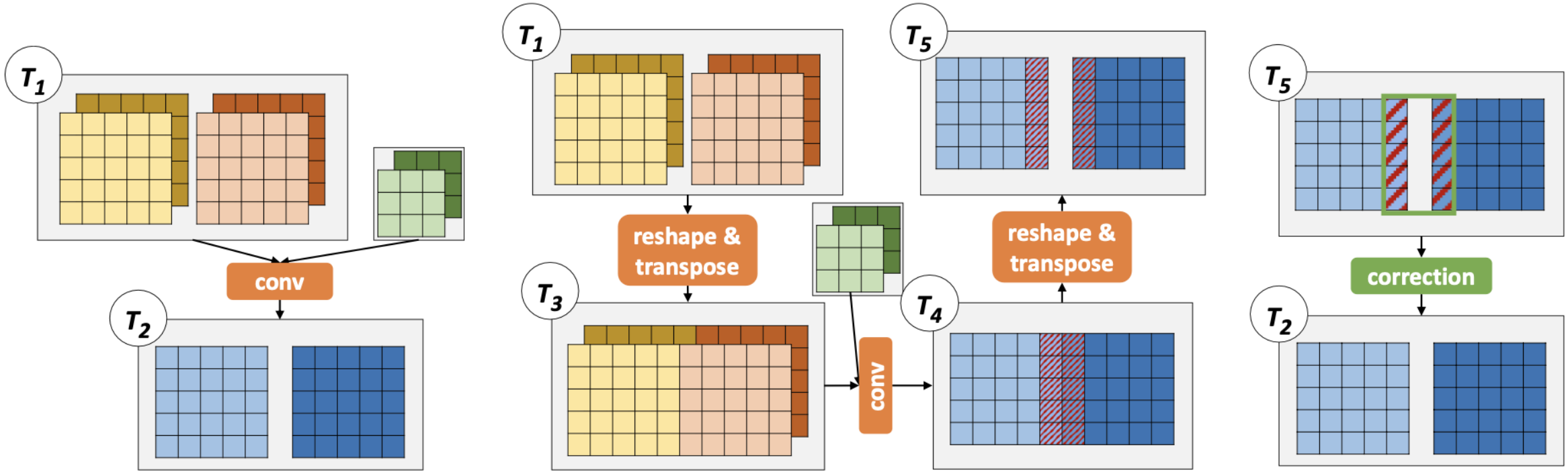
- To improve efficiency
- Most previous works: rule-based, fully equivalent transformations





Partially Equivalent Transformations

- Not equal on all elements of output tensor
- Faster, but requires correction lalter

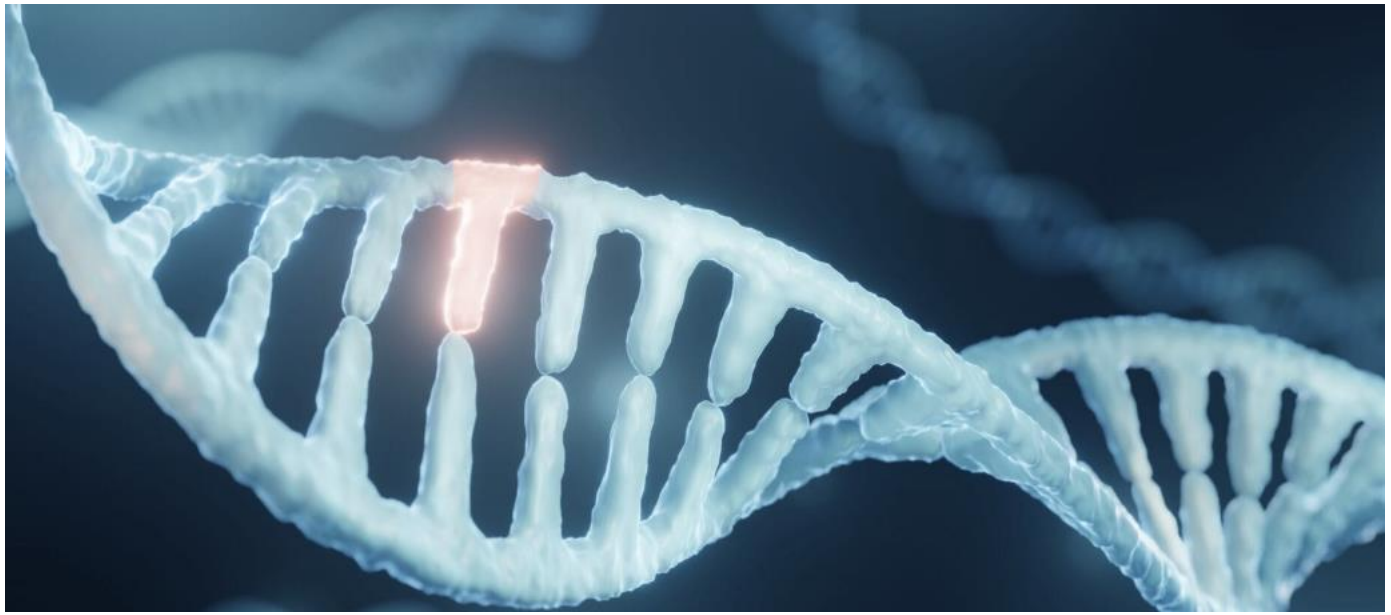


(a) Input program. (b) A partially equivalent transformation. (c) Correcting results.

Mutant

P1 is a **mutant** of P0 if:

same #input and #output, each input/output has same shape



PET in a systematic way

- Split a tensor program into sub-programs
- Mutate, correct, and evaluate each part iteratively, keeping the top K programs

$P: S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow \cdots$

Heap $H = \{P\}$

Round 1: mutate S_1. Mutants $M_1^1, M_1^2, M_1^3 \cdots$

Evaluate each $P.replace(S_1, M_1^i)$

Leave top K in H: $H = \{P.replace(S_1, M_1^1), \cdots\}$

Round 2: mutate S_2. Mutants $M_2^1, M_2^2 \cdots$

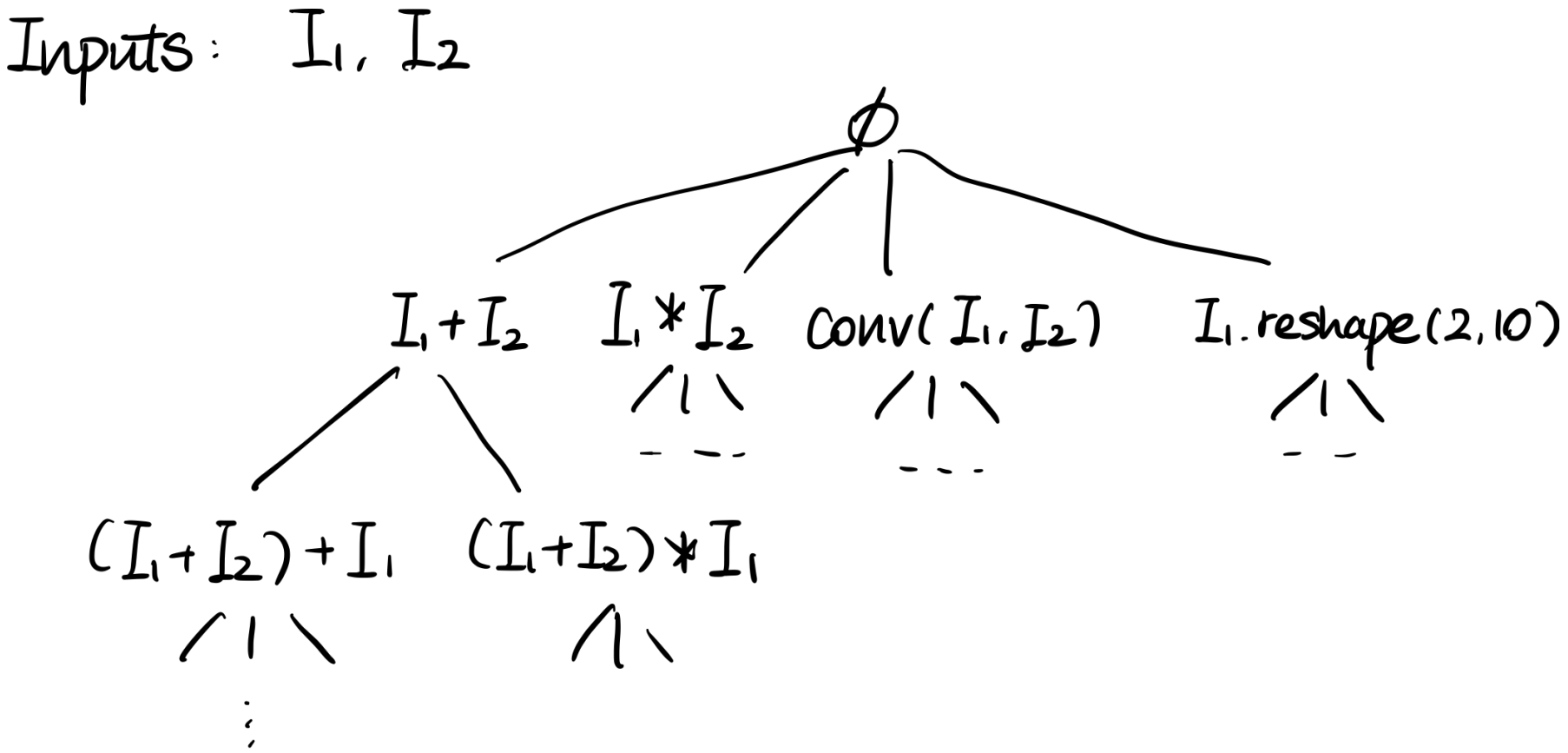
for each P' in H, replace S_2 with each M_2^j and evaluate

Leave top K in H. $H = \{P.replace(S_1, M_1^i).replace(S_2, M_2^j)$

.....

Mutation from DFS

target: $conv(I_1, I_2)$



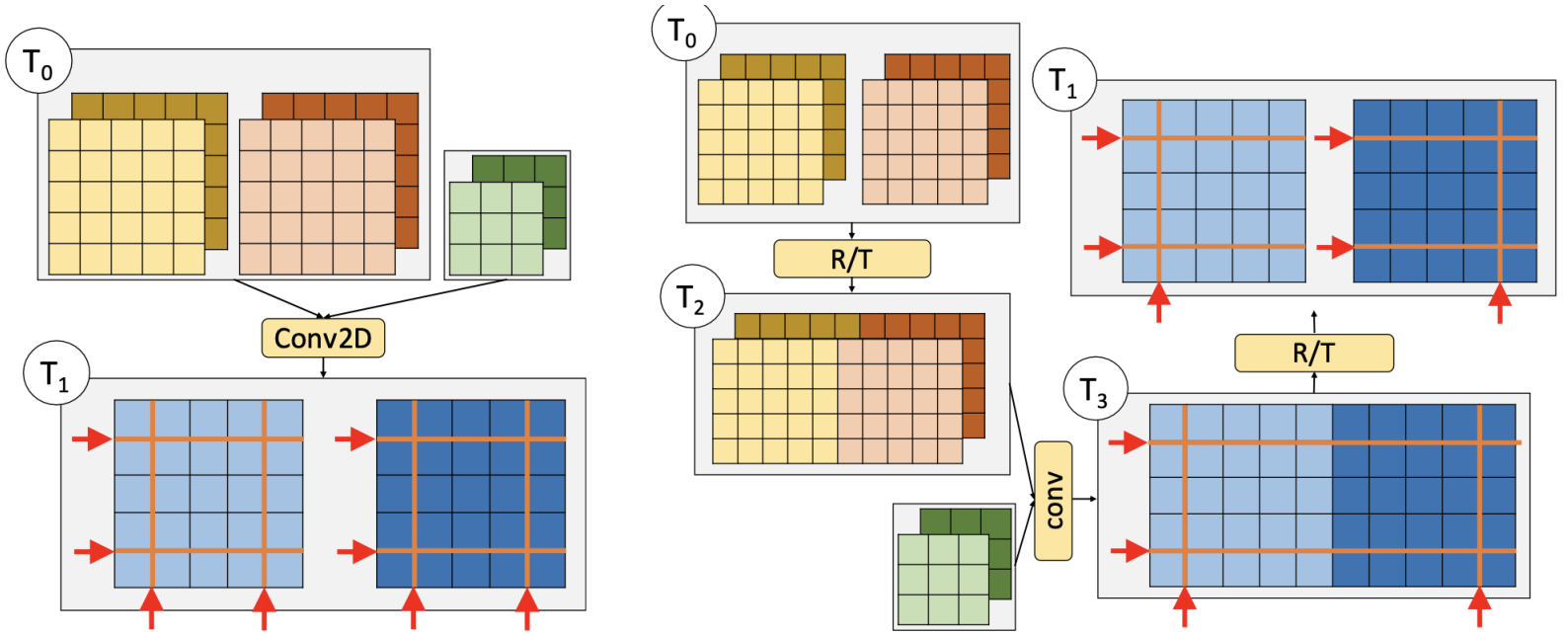
At depth 4, accept all programs with agreeing shape of IO

Table 1: Multi-linear tensor operators used in PET.

Operator	Description
`add`	Element-wise addition
`mul`	Element-wise multiplication
`conv`	Convolution
`groupconv`	Grouped convolution
`dilatedconv`	Dilated convolution
`batchnorm`	Batch normalization
`avgpool`	Average pooling
`matmul`	Matrix multiplication
`batchmatmul`	Batch matrix multiplication
`concat`	Concatenate multiple tensors
`split`	Split a tensor into multiple tensors
`transpose`	Transpose a tensor's dimensions
`reshape`	Decouple/combine a tensor's dimensions

Program correction

- For original and mutated program, divide output region into *boxes*, according to *summation interval*
- Verify equivalence of each intersection
- Only re-compute disagreeing intersections



(a) Input program $\mathcal{P}_0$.

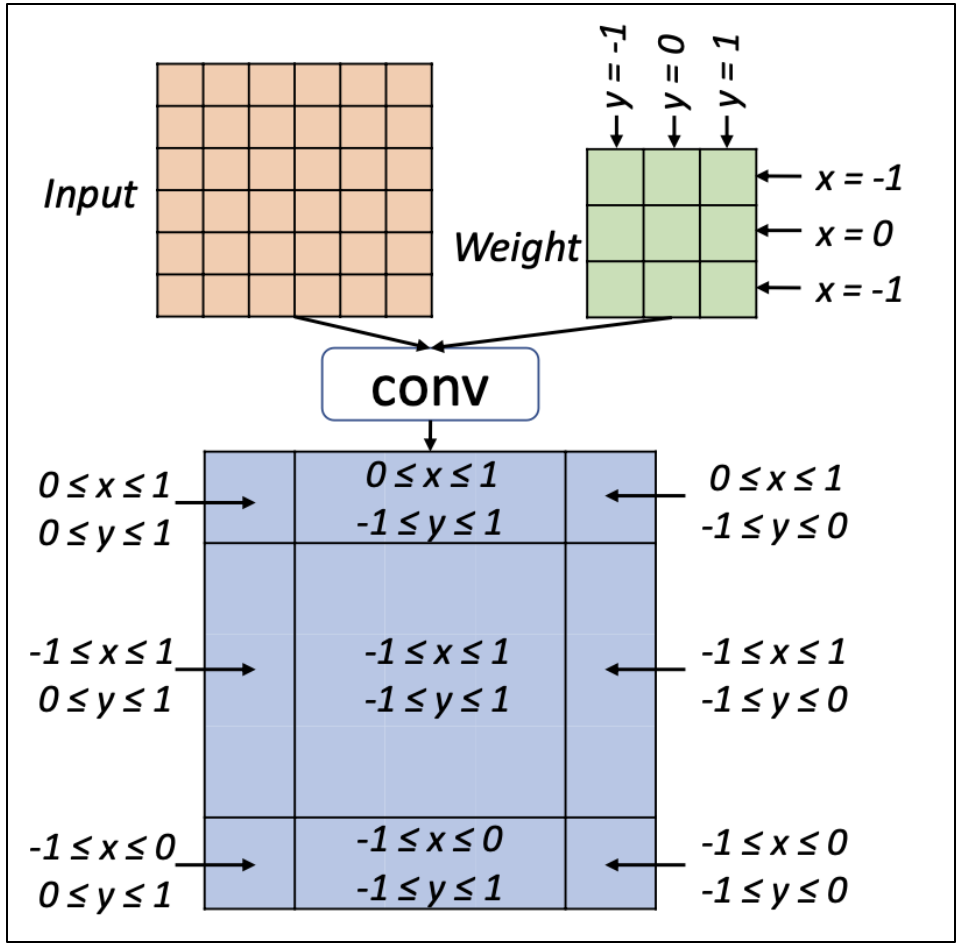
(b) A potential mutant $\mathcal{P}_1$.

Summation Interval

Multi-linear tensor programs (MLTPs). We first define multi-linear tensor operators. An operator *op* with n input tensors $I_1, \ldots, I_n$ is *multi-linear* if *op* is linear to all inputs I_k:

$$op(I_1, \ldots, I_{k-1}, X, \ldots, I_n) + op(I_1, \ldots, I_{k-1}, Y, \ldots, I_n)$$
$$= op(I_1, \ldots, I_{k-1}, X + Y, \ldots, I_n)$$
$$\alpha \cdot op(I_1, \ldots, I_{k-1}, X, \ldots, I_n) = op(I_1, \ldots, I_{k-1}, \alpha \cdot X, \ldots, I_n)$$

$$\mathcal{P}(I_1, \ldots, I_n)[\vec{v}] = \sum_{\vec{r} \in \mathcal{R}(\vec{v})} \prod_{j=1}^{n} I_j[\mathbf{L}_j(\vec{v}, \vec{r})]$$



Correctness of an intersection?

Do we need to verify each output cell?

No. For each box, if output is m-dimensional, only need to verify m+1 cells.

For each output cell, how to verify equivalence?

To verify a cell, just test on a few cases. The error is small and controlled.

Then…

- If a box intersection is good, done
- Otherwise, re-compute the box
  - (some optimization to reduce overhead…)

Evaluation - Speedup

- Up to 2.5x improvement
- Improvement even on heavily-optimized models

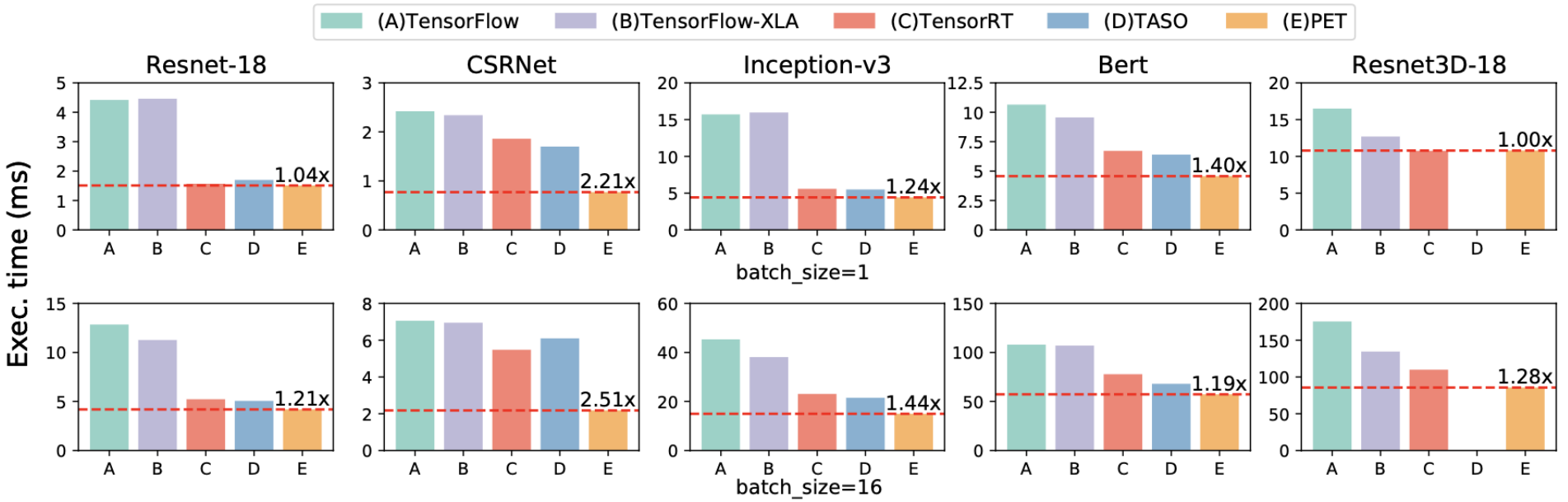


Figure 8: End-to-end performance comparison between PET and existing frameworks. For each DNN, the numbers above the PET bars show the speedups over the best baseline.

Evaluation – Search Time

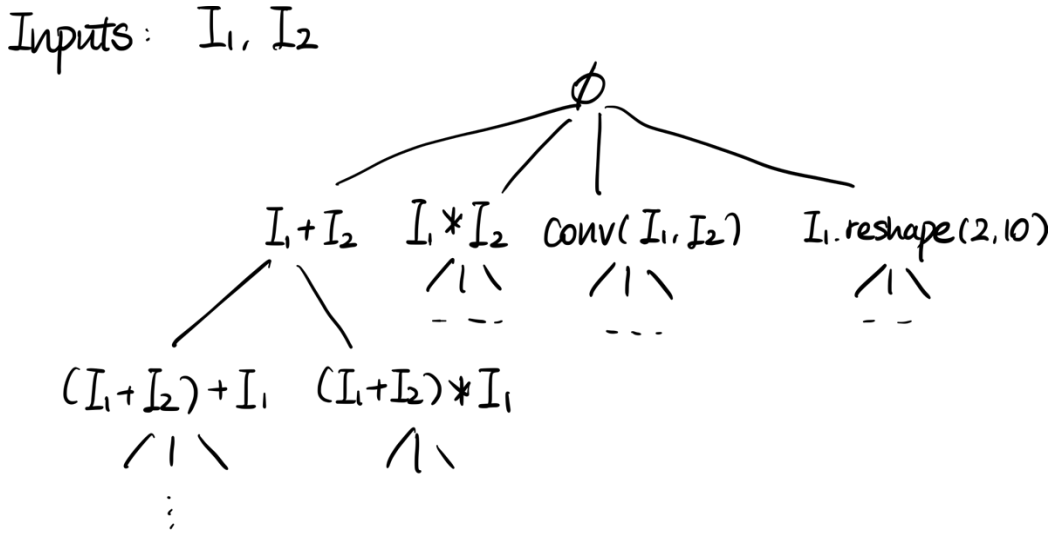
- Under 3 minutes usually
  - *“89 s, 88 s, 91 s, and 165 s on Resnet-18, CSRNet, BERT, and Resnet3D-18, respectively”*
- 25 min for Inception-v3
  - “*due to the multiple branches in the Inception modules*”

Impact

- Considering partially equivalent transformations opens up optimization space
- Overhead well controlled
- Solid theoretical foundations

Criticism

- Complex system with unclear details
  - e.g. how is the reshape params determined?
- Estimates performance based on cost model
  - “Measures execution time of each tensor operator once for each configuration” – costly, not counted in the 3 min I assume
- Only supports predefined operators
- Analysis on #boxes missing



Discussion

- Currently inference-only. Can be used for training as well?

References

- https://www.yourgenome.org/theme/what-is-a-mutation/