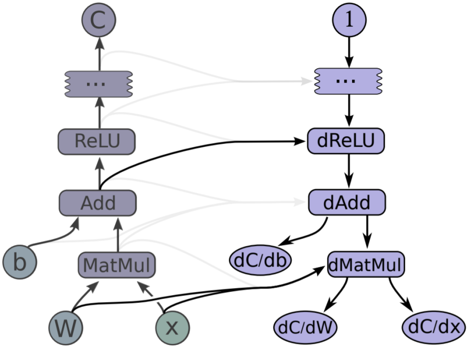
PET: Optimizing Tensor Programs with Partially Equivalent Transformations and Automated Corrections

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

Conor Perreault

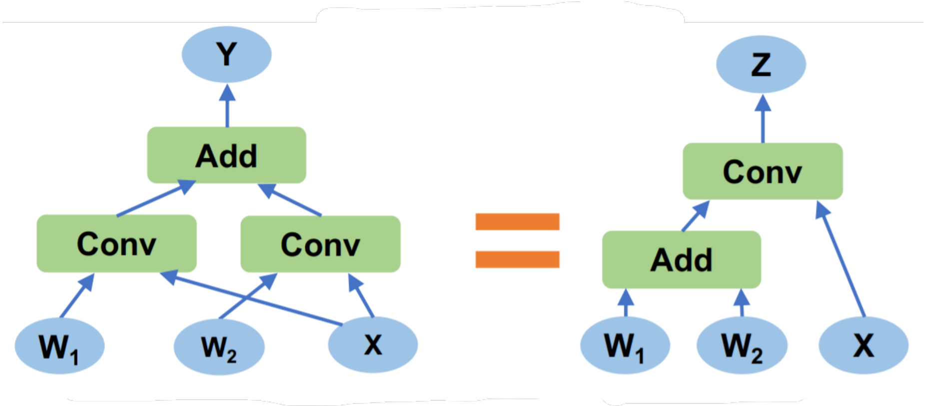
Review of Tensor Programs

- Deployment of neural networks uses optimized tensor programs (e.g. TensorFlow).
- Program structure is represented as a DAG of computation nodes, with tensors flowing across edges
- Graph structure allows for parallelization and distribution of computation.



Optimization of Tensor Programs

- Node assignment: some hardware is optimized for certain types of computations.
- Graph creation: there are often equivalent ways to solve a problem.

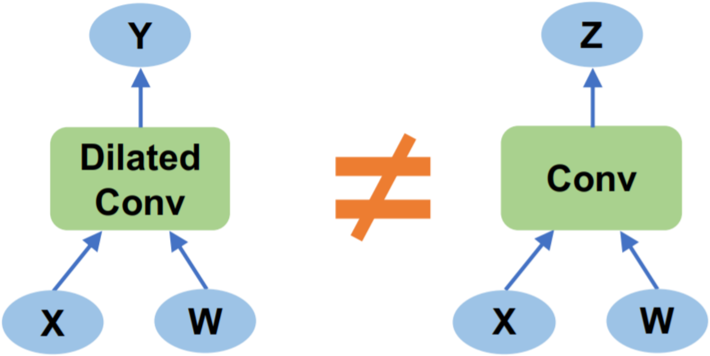


Related Work

- TensorFlow, Pytorch and several other systems use heuristic-based optimization to apply transformations.
- TASO: Automatically generates and verifies transformations using deep learning. Claims up to $2.8\times$ speedup compared to manual versions.
- All current systems rely on fully equivalent transformations.
- NeoCPU and other work explore optimizing CNNs by reducing layout transformations. PET builds on these ideas.
- Automatic statistician, TPOT use non-equivalent transformations to propose different neural network architectures, but require evaluations at the end to test the effects on accuracy.

Partially Equivalent Transformations

- Transformations that do not preserve outputs exactly.
- May be more efficient or allow for hardware specialization.
- Partially equivalent transformations require corrections to ensure model accuracy.



Overview

- PET uses partially equivalent transformations to further optimize tensor programs.
- Mutation Generator: the component that generates potential mutations to the graph layout that take same inputs and produce outputs with the same shape.
- Mutation Corrector: automatically produces correction kernels that adjust the outputs of a mutant subprogram to match the original.
    - Without optimizations, this is a combinatorially hard problem.
    - PET only tests a few representative input/output combinations to find these corrections.
- Program optimizer: makes the search for mutations efficient by splitting the program into subprograms and mutating individually, then applying optimizations across subprogram boundaries.

Mutation Generator

- Mutations are generated on linear portions of a neural network such as matrix multiplication and convolution, and not on non-linear portions such as activation functions.
- Using a defined set of operations, the generator uses DFS to explore all possible mutants (up to a certain depth), and prunes those that are invalid.
- Reshape and transpose: transform tensor layouts– well established for improving performance.
- Single operator mutants: take advantage of optimized kernels for convolution, matrix multiplication instead of their variants such as dilated convolutions.
- Multi-operator mutants: replace multiple operators at a time. For example, convolution on multiple inputs at the same time can be more efficient.

Mutation Corrector

- To maintain predictability, PET corrects partial equivalencies so that outcomes are identical.
- PET first identifies all the elements of the output tensor that may not be identical to the original program.
- Then, PET generates a kernel to correct those outputs.
- Both of these operations are infeasible if the program just tests every possibility, but the authors introduce 2 theorems that help generate these kernels in $O(1)$ time.

Mutation Corrector: Theorem 1

- Any single output of a multi linear tensor program can be described as

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

  where $\mathcal{R}(\vec{v})$ describes the summation region for an output.
- **Theorem 1**: If a program has an m-dimensional output tensor, then only $m + 1$ positions have to be evaluated for equivalence in each summation region.
- Example: in a convolution, the edges of a matrix have a different summation region than the center.

Mutation Corrector: Theorem 2

- **Theorem 2**: If two programs with n dimensional inputs are not equivalent and inputs are drawn from p values, then for a random input vector $\vec{v}$, the probability that the outputs are the same is at most $\frac{n}{p}$.
- If tensors are allowed to take values from a large set of integers, this means only a few inputs need to be sampled to test equivalence.

Mutation Corrector Algorithm

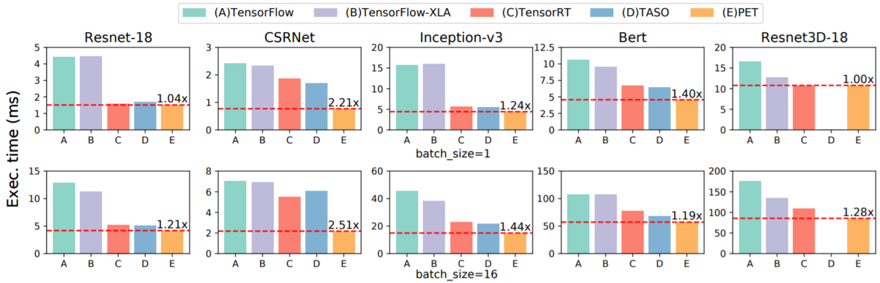
- Box propagation: find a set of split points that define the boundaries of summation regions and propagate through a program.
- Random testing: For each box, randomly test $m + 1$ positions at t random points.
- Correction kernel generation: PET runs the original program on the boxes that are not equivalent and uses existing libraries to generate correction kernels.

Program Optimizer

- Break program into subprograms, and generate mutant versions.
- Estimate runtime of each version using a similar algorithm to TASO based on summing the execution times of each component, keep top k.
- Apply post optimizations: remove inverse transformations, fuse operators when possible
- Choose program with fastest expected runtime.

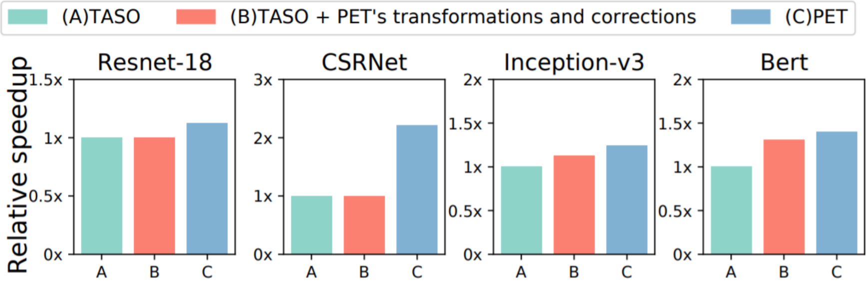
Experiments and Results

The authors test 5 models optimized by PET and compare it to contemporary systems:



Experiments and Results

The authors perform an ablation study by using pieces of TASO to test which components of PET improve performance:



Major Contributions

- PET is the first use of partially equivalent transformations for optimizing tensor programs. This provides a much larger search space for optimizations than other frameworks.
- The authors provide 2 theorems that prove automatic correction generation can be efficient.
- Efficient search is used to explore a large program space.
- PET achieves up to $2.5\times$ speed up compared to TASO.

Criticism

- Well written paper:
  - Contributions in theory and applications
  - Good experiments and case studies
  - Ablation study
- Extending framework to optimize training as well would be even more useful
- Authors mention that some operations are optimized for different hardware types. It would be useful to explicitly include this computation in the execution chosen.
- In most cases, very low improvement over TASO. This is hardly discussed.
- Search time is up to 25 minutes. This could be unreasonable in some cases.

References

Haojie Wang, Jidong Zhai, Mingyu Gao, Zixuan Ma, Shizhi Tang, Liyan Zheng, Yuanzhi Li, Kaiyuan Rong, Yuanyong Chen, Zhihao Jia. *PET: Optimizing Tensor Programs with Partially Equivalent Transformations and Automated Corrections*. OSDI '21. 2021.