ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

# EINNET: Optimizing Tensor Programs with Derivation-Based Transformations

OSDI23 @ Boston, July 12, 2023

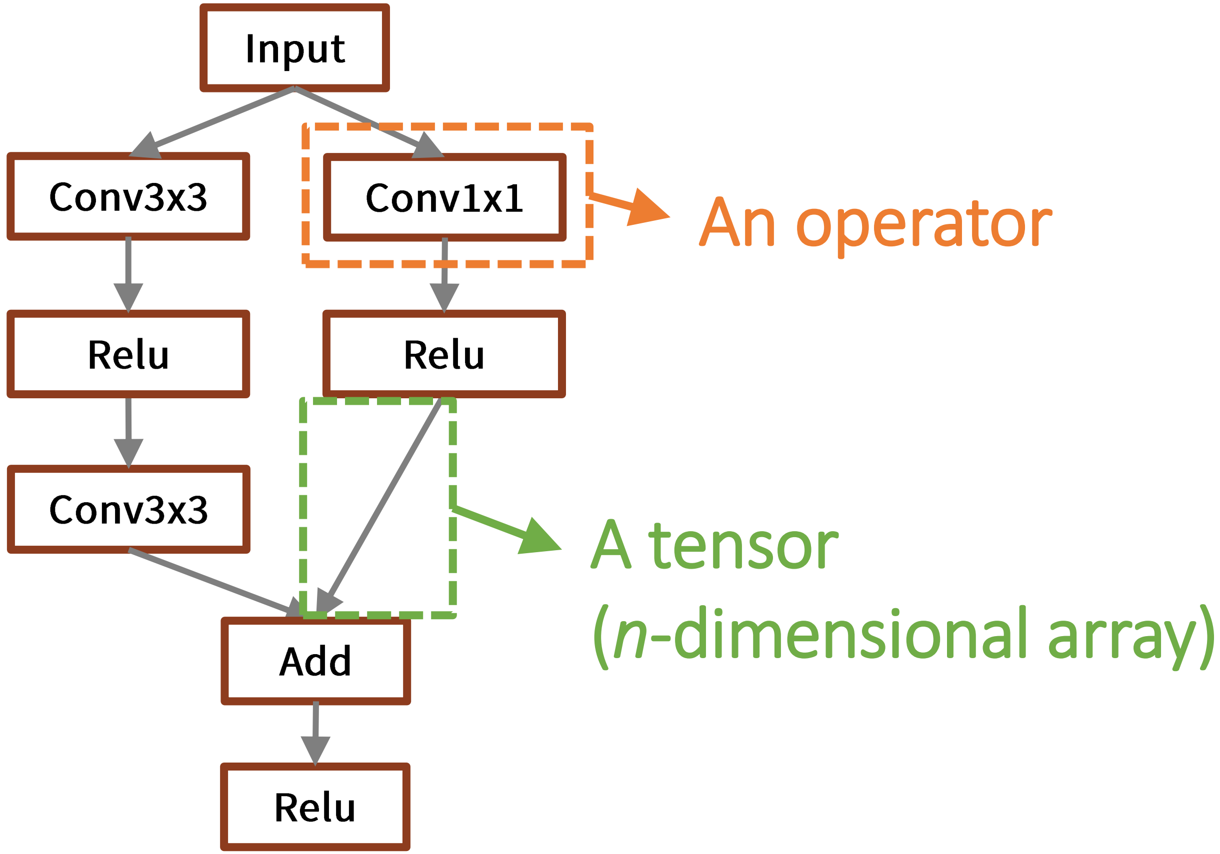
**Liyan Zheng**, Haojie Wang, Jidong Zhai,
Muyan Hu, Zixuan Ma, Tuowei Wang, Shuhong Huang,
Xupeng Miao, Shizhi Tang, Kezhao Huang, Zhihao Jia

Tsinghua University

Carnegie Mellon University

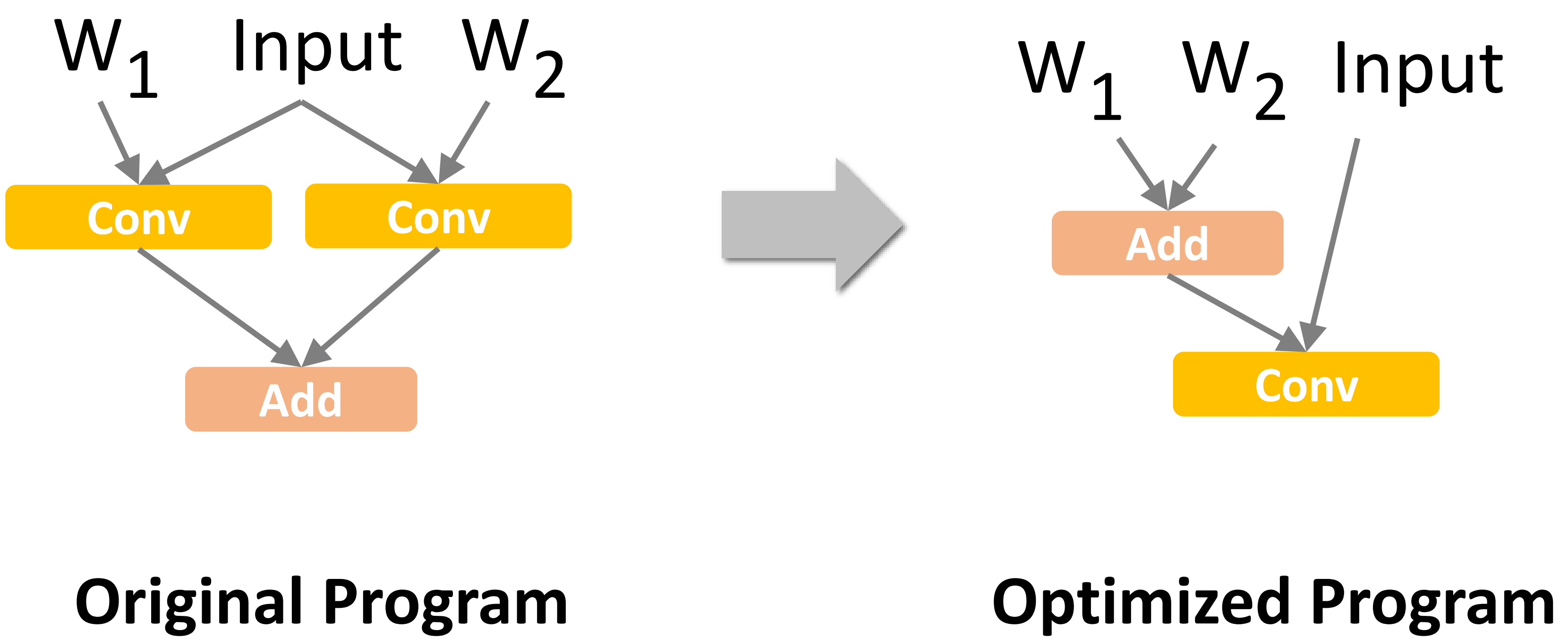
# Tensor Programs

- Widely used in deep learning
- Represented as computation graphs



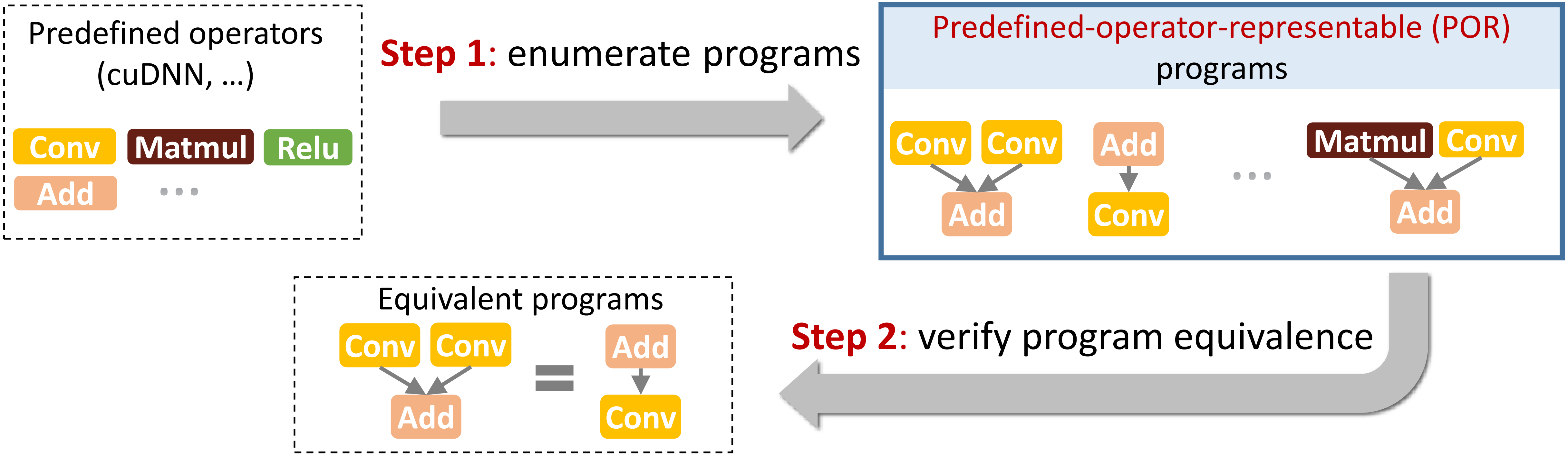
# Tensor Program Transformations

- Goal: optimizing program performance



# Automatic Transformations

- Superoptimization-based approaches (TASO[1] and PET[2])



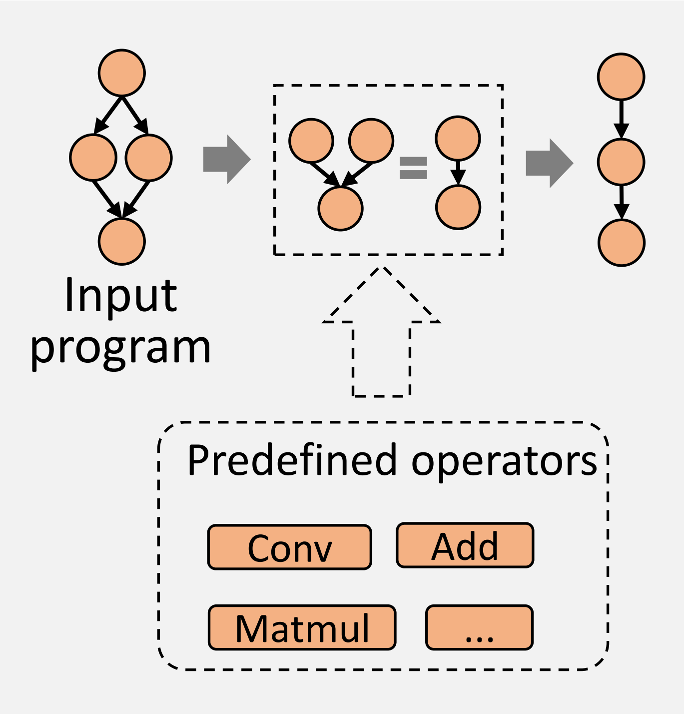

Automatic exploration · Only POR transformations · Time-consuming verifications

1. TASO: Optimizing Deep Learning Computation with Automated Generation of Graph Substitutions. SOSP'19
2. PET: Optimizing Tensor Programs with Partially Equivalent Transformations and Automated Corrections. OSDI'21

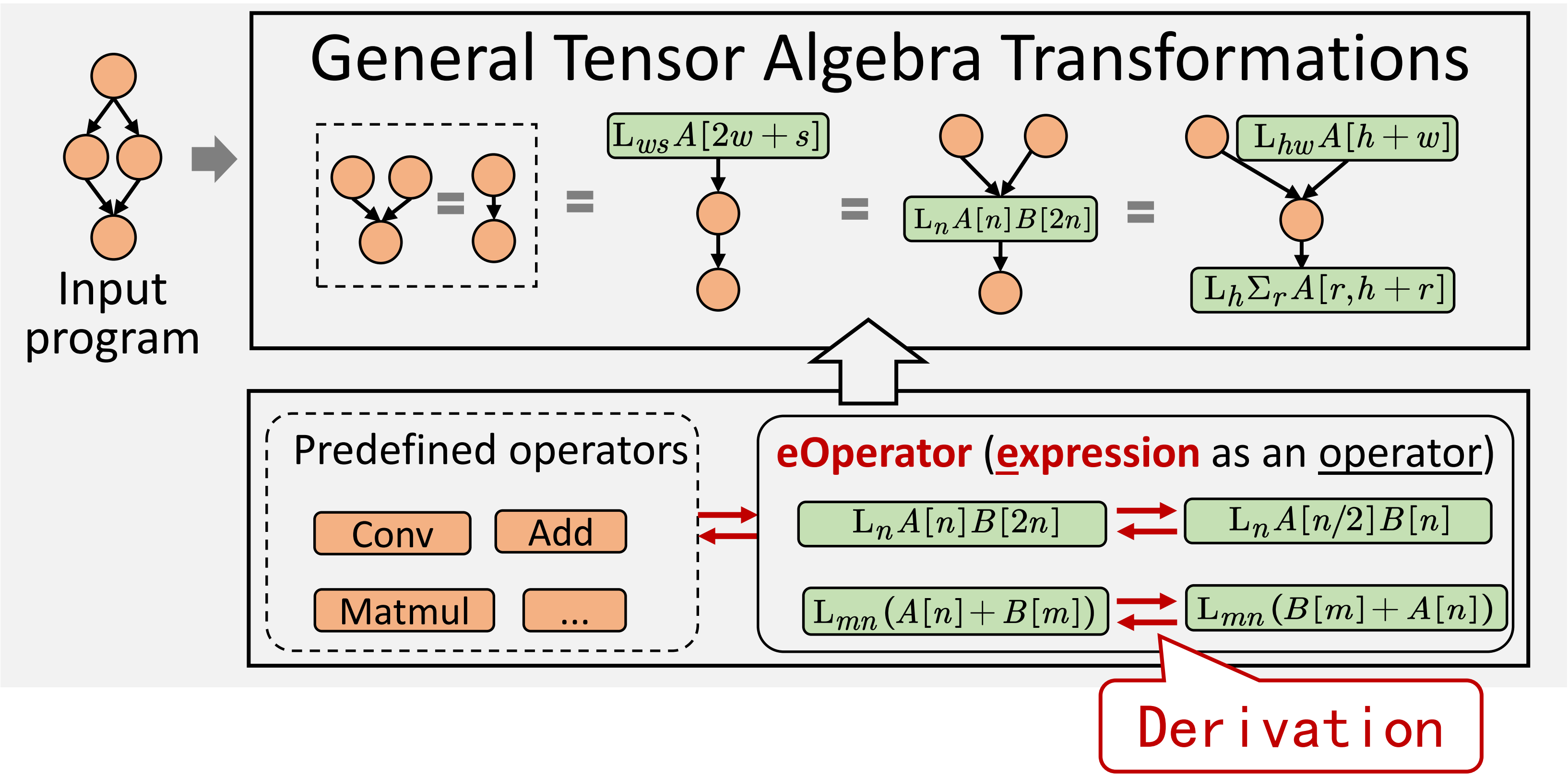
# EINNET

- A tensor program optimizer based on tensor expression derivation

## Prior work

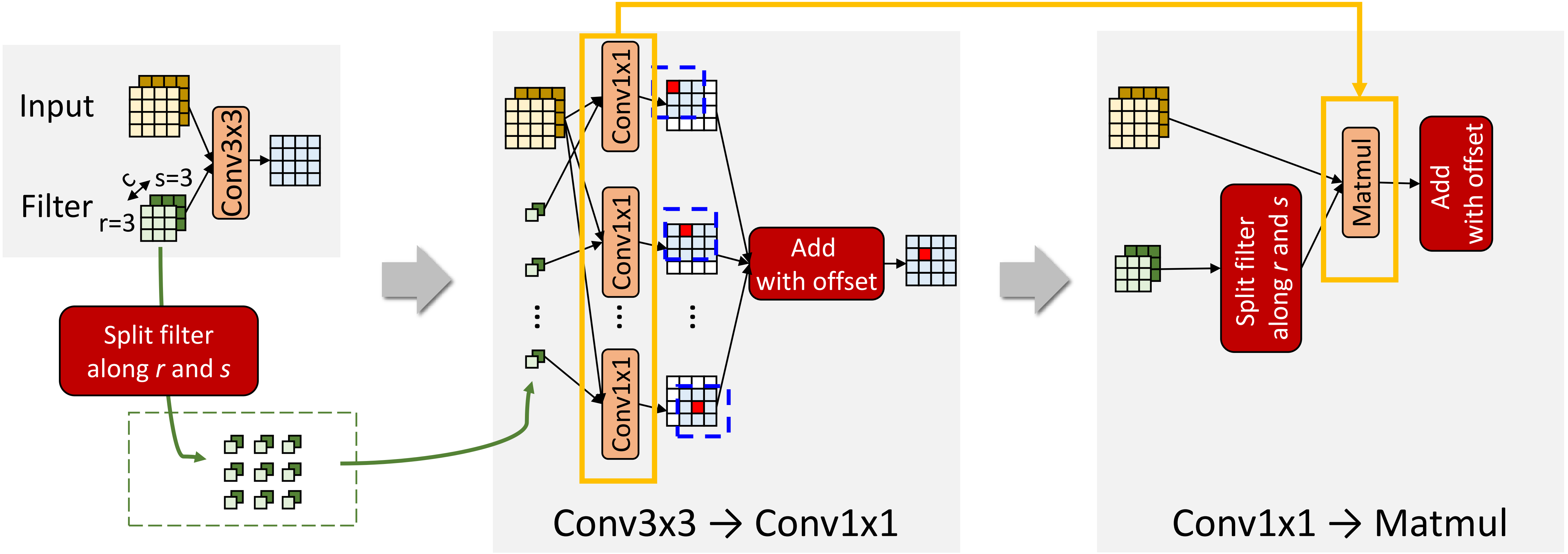



## EINNET



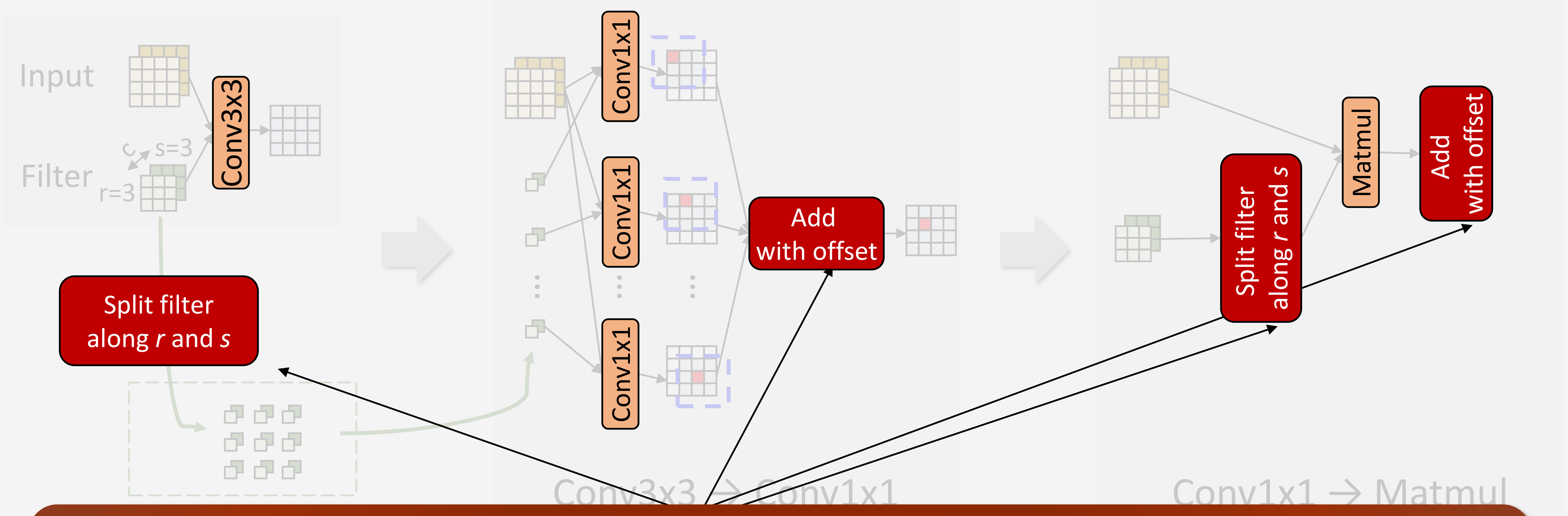

**Larger optimization space** and up to 2.7x speedup

# Motivating Example: Convolution to Matmul




**2x speedup on Nvidia A100 GPUs**

# Motivating Example: Convolution to Matmul



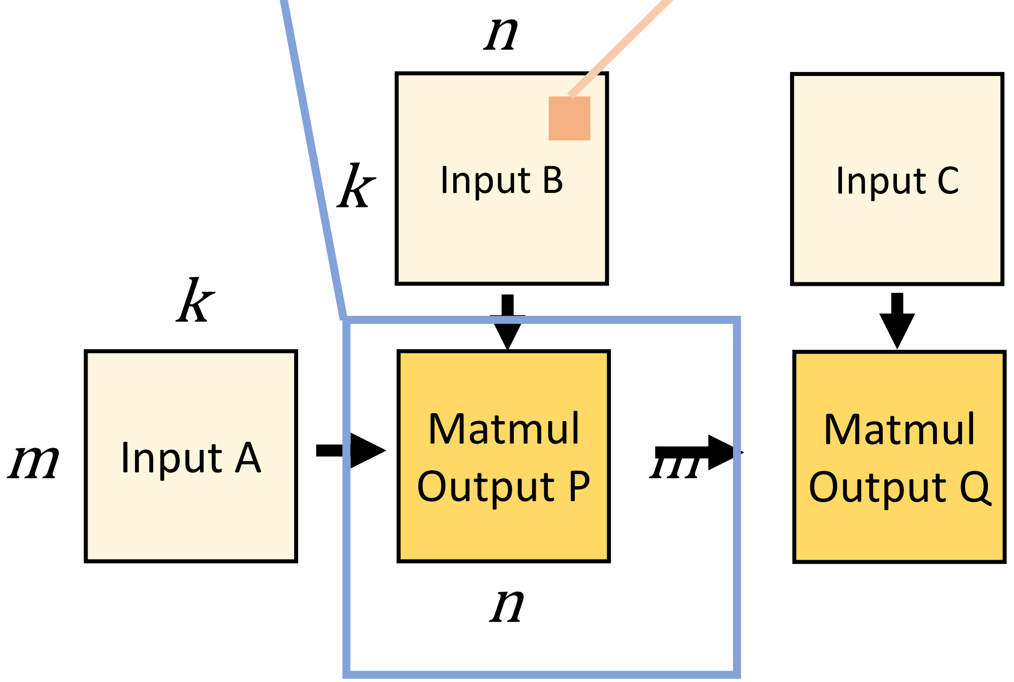

**Non-predefined operators** enable more optimizations

# Tensor Expressions

- Specify the computation semantics

$$P[m,n] = \mathrm{Matmul}(A,B)$$

$$= \underset{mn}{L} \sum_{k} A[m,k]B[k,n]$$



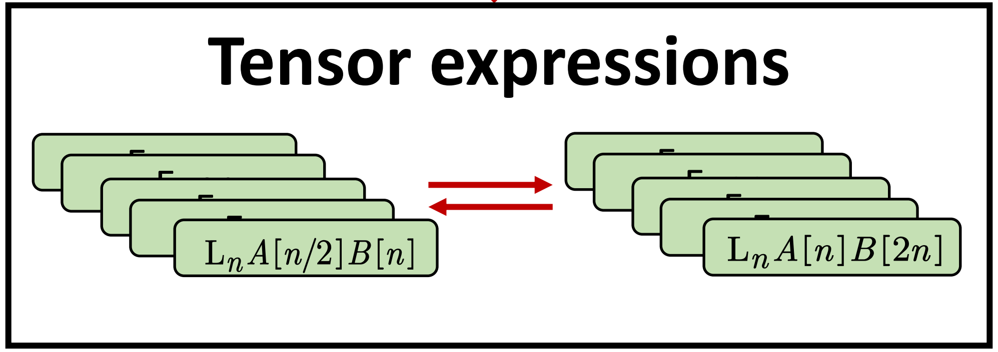

$$= \underset{mn}{L} \sum_{k} \left\{ \underset{mn}{L} \sum_{k} A[m,k]B[k,n] \right\} [m,k]C[k,n]$$

Nested expressions for multiple operators

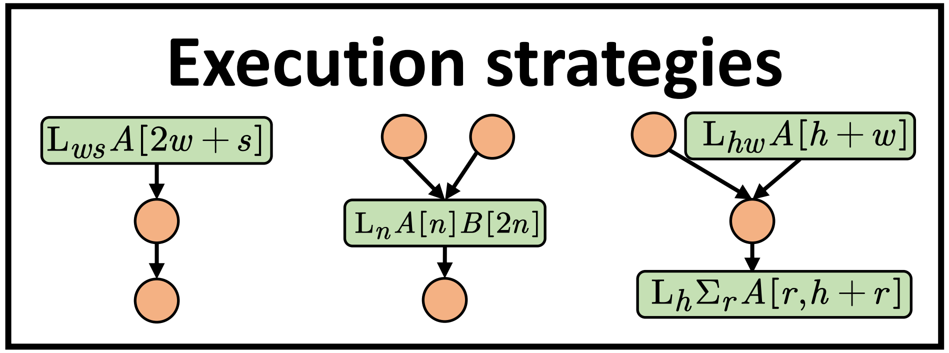
# Key Challenges

**1. discover transformations**

**2. execute expressions on HW**




×




**3. explore the large search space**

# Challenge#1: Expression Transformation

A strawman approach: superoptimization for expressions

Input expression

$2A[i] + B[j]$

Enumerate $\longrightarrow$

Candidates

$2A[i] \times B[j]$

$A[i] + B[j]$

$B[j] + 2A[i]$ Verified

…

- Limitation
  - Infinitely many expressions
  - Hard to verify equivalence

EinNet discovers equivalent expressions by **derivations**

# Derivation Rules

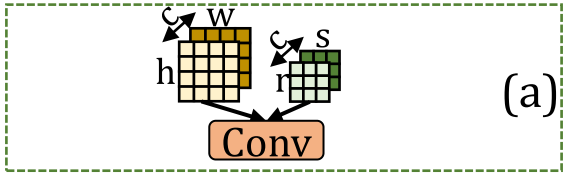
- Mathematically equivalent rewrites

| Intra-expression derivations | |
|---|---|
| Summation splitting | $\underset{\vec{x}}{L}\underset{\vec{s}_1,\vec{s}_2}{\Sigma} f\left(\vec{T},\vec{x},\vec{s}_1,\vec{s}_2\right) \Longrightarrow \underset{\vec{x}}{L}\underset{\vec{s}_1}{\Sigma}\left\{\underset{\vec{x},\vec{s}_1}{L}\underset{\vec{s}_2}{\Sigma} f\left(\vec{T},\vec{x},\vec{s}_1,\vec{s}_2\right)\right\}[\vec{x},\vec{s}_1]$ |
| Variable substitution | $\overset{\mathbb{X}}{\underset{\vec{x}}{L}} f\left(\vec{T},\vec{x}\right) \Longrightarrow \overset{\mathbb{X}}{\underset{\vec{x}}{L}}\left\{\overset{\Phi(\mathbb{X})}{\underset{\vec{y}}{L}} f\left(\vec{T},\Phi^{-1}(\vec{y})\right)\right\}[\Phi(\vec{x})]$ |
| Traversal merging | $\overset{\mathbb{X}}{\underset{\vec{x}}{L}}\left\{\overset{\mathbb{Y}}{\underset{\vec{y}}{L}} f\left(\vec{T},\vec{y}\right)\right\}[\Phi^{-1}(\vec{x})] \Longrightarrow \overset{\mathbb{X}}{\underset{\vec{x}}{L}} f\left(\vec{T},\vec{x}\right)$ |
| Boundary relaxing | $\overset{\mathbb{X}}{\underset{\vec{x}}{L}} f\left(\vec{T},\vec{x}\right) \Longrightarrow \overset{\mathbb{X}\cup\mathbb{Y}}{\underset{\vec{x}}{L}} f\left(\vec{T},\vec{x}\right)$ |
| Boundary tightening | $\overset{\mathbb{X}\cup\mathbb{Y}}{\underset{\vec{x}}{L}} f\left(\vec{T},\vec{x}\right) \Longrightarrow \overset{\mathbb{X}}{\underset{\vec{x}}{L}} f\left(\vec{T},\vec{x}\right)$ |

| Inter-expression derivations | |
|---|---|
| Expression splitting | $\overset{X}{\underset{\vec{x}}{L}} f\left(\vec{T},\vec{x}\right) \Longrightarrow \overset{X_0}{\underset{\vec{x}}{L}} f\left(\vec{T},\vec{x}\right) \sim \overset{X_1}{\underset{\vec{x}}{L}} f\left(\vec{T},\vec{x}\right)$ |
| Expression merging | $\overset{X_0}{\underset{\vec{x}}{L}} f\left(\vec{T},\vec{x}\right) \sim \overset{X_1}{\underset{\vec{x}}{L}} f\left(\vec{T},\vec{x}\right) \Longrightarrow \overset{X}{\underset{\vec{x}}{L}} f\left(\vec{T},\vec{x}\right)$ |
| Expression fusion | $\overset{X}{\underset{\vec{x}}{L}} f\left(\vec{T},\vec{x}\right) \rightarrow \overset{X'}{\underset{\vec{x}}{L}} g\left(\vec{T}',\vec{x}\right) \Longrightarrow \overset{X}{\underset{\vec{x}}{L}} g\left(f\left(\vec{T},\vec{x}\right)\right)$ |

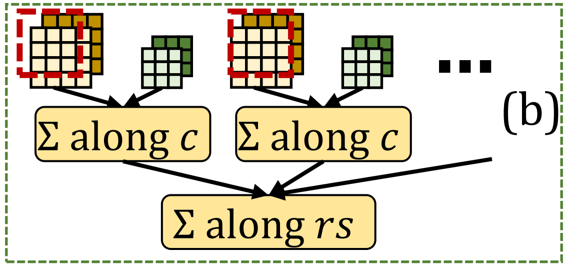
- Support custom derivation rules

# Derivations in the Motivating Example




$$L_{hwf} \sum_{crs} A[h+r, w+s, c] K[r, s, f, c]$$

**Summation splitting**: divides summation into two steps



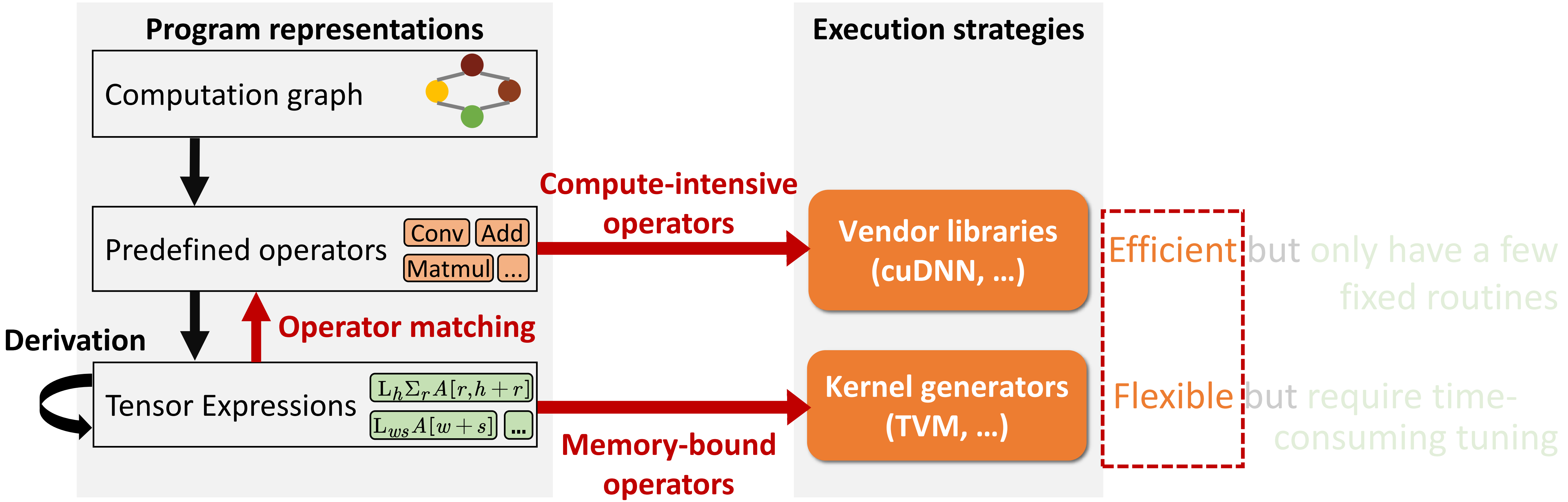

$$= L_{hwf} \sum_{rs} \left\{ L_{rshwf} \sum_{c} A[h+r, w+s, c] K[r, s, f, c] \right\} \cdots$$

**Variable substitution**: replace $h, w$ with $t_1, t_2$ $\begin{cases} t_1 = h + r \\ t_2 = w + s \end{cases}$

$$= \cdots \left\{ L_{rshwf} \left\{ L_{rsf} \; L_{t_1 = r}^{H+r} \; L_{t_2 = s}^{W+s} \; \sum_{c} A[t_1, t_2, c] K[r, s, f, c] \right\} [r, s, f, h+r, w+s] \right\} \cdots$$

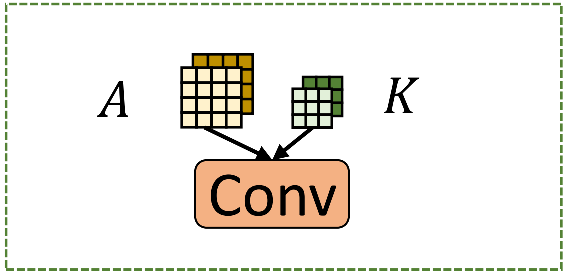
**Derivation creates new equivalent expressions**

# Challenge#2: Executing Expressions




**Combine the benefits of vendor libraries and kernel generators**

# Operator Matching & Kernel Generation




$$\sum_{hwf}\sum_{crs} A[h+r,w+s,c]K[r,s,f,c]$$

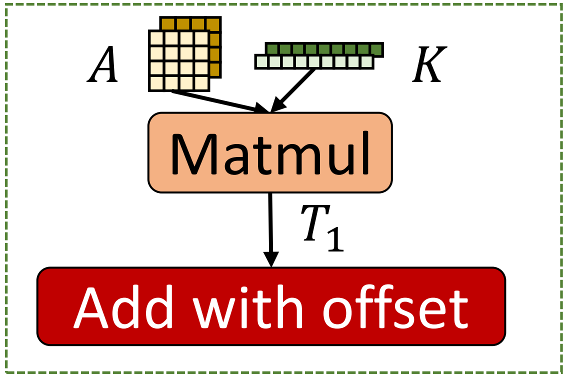
Several steps of derivations

$$=\sum_{hwf}\sum_{rs}\left\{\sum_{rsf}\sum_{t_1=0}^{H}\sum_{t_2=0}^{W}\sum_{c} A[t_1,t_2,c]K[r,s,f,c]\right\}[r,s,f,h+r,w+s]$$

**Operator matching**: replace compute-intensive expressions with $Matmul(K,\ A)$

$$=\sum_{hwf}\sum_{rs} T_1[r,s,f,h+r,w+s]$$

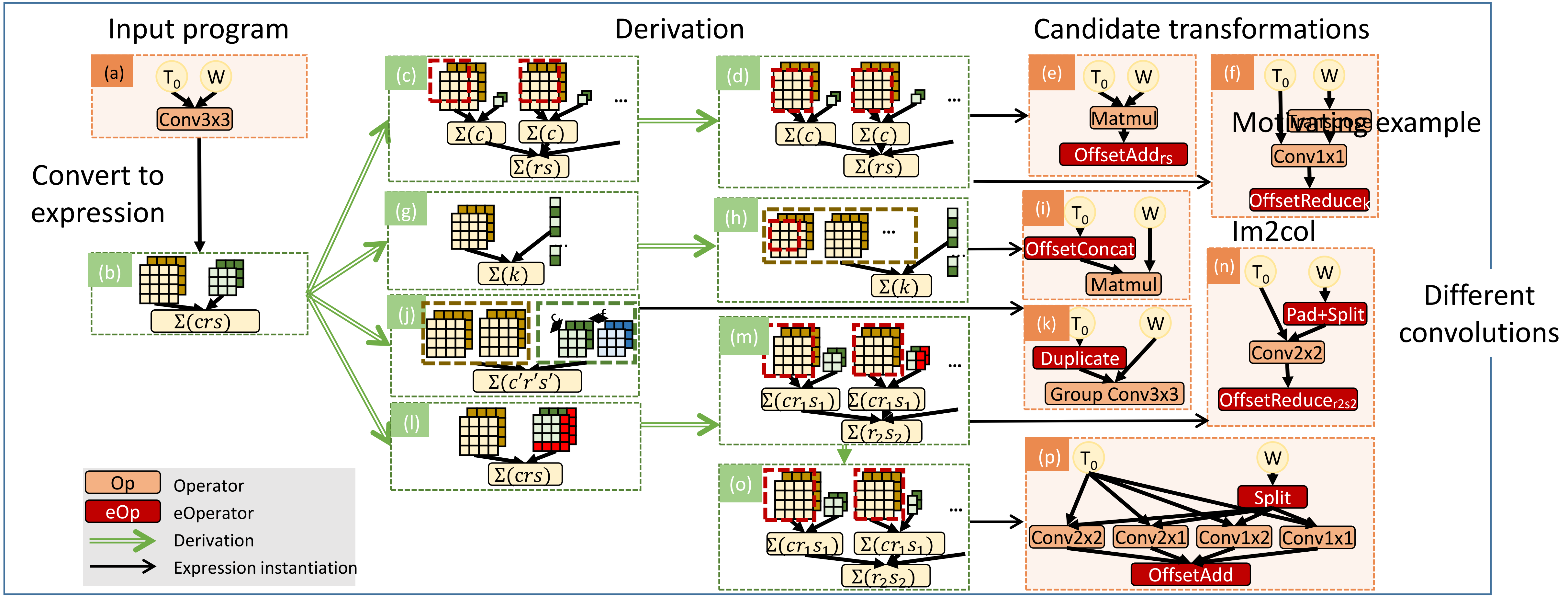
$$=T_2$$




**Kernel generation**: tune kernels for memory-bound expressions

* Details available in the paper

# Challenge#3: Large search space




**More optimization opportunities**

# Challenge#3: Large search space

- A computation graph of a single convolution
  - $\sim 10$ steps of derivation
  - $\sim 10^8$ candidates
  - $\sim 10$ hours

Solution: expression distance to guide search

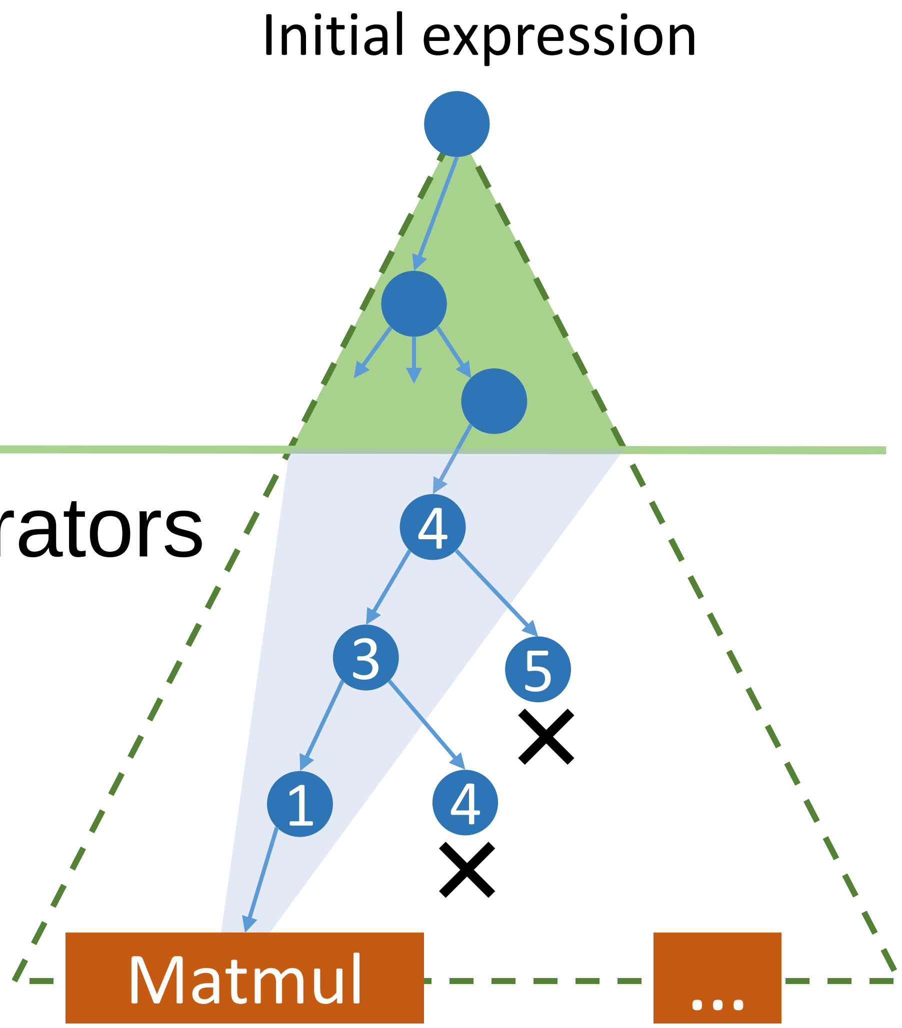
- Measure similarity between expressions

# Expression-Distance-Guided Search

- Two search stages

- Stage I: explore search space
  - Apply all possible derivations

- Stage II: converge to performant operators
  - Guided by expression distance



# Evaluation

**Platforms:**

Nvidia A100 & V100 GPUs

**Backends:**

cuBLAS + cuDNN, AutoTVM, Ansor

**Models:**

Language model: Longformer

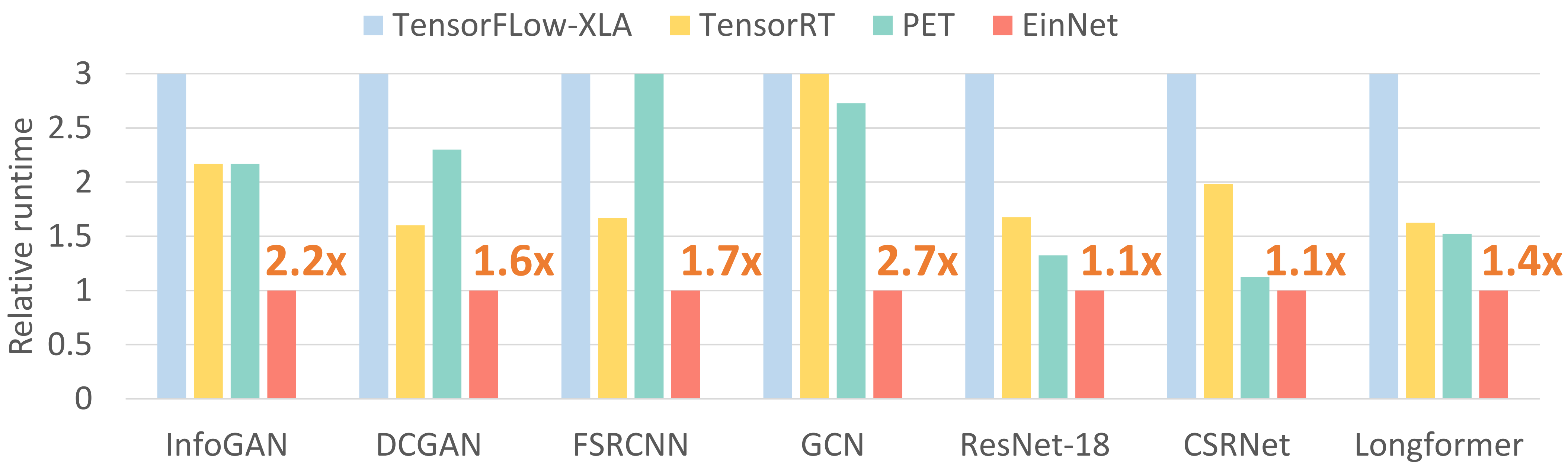
Image generation: InfoGAN, DCGAN, FSRCNN

Image understanding: GCN, ResNet-18, CSRNet

**Baseline:**

TensorRT, PET, Tensorflow, Tensorflow-XLA, Nimble, TVM

# End-to-End Inference Evaluation (Nvidia A100 GPU)




**EinNet outperforms existing optimizers by up to 2.7x**

# Conclusion

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

**EINNET** is a **derivation-based** tensor program optimizer

**Proposed technique:** expression derivation

**Larger search space:** general tensor algebra transformations

**Better performance:** up to 2.7x speedup

**Available at: https://github.com/InfiniTensor/InfiniTensor**

# Q&A

## Thank you!

Acknowledgement: Deng Feng Fund