THE UNIVERSITY of EDINBURGH
informatics

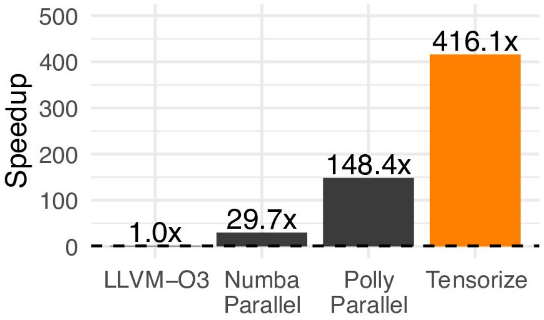
Tensorize: Fast Synthesis of Tensor Programs from Legacy Code using Symbolic Tracing, Sketching and Solving

Alexander Brauckmann, Luc Jaulmes, Jose W. de Souza Magalhães, Elizabeth Polgreen, Michael O'Boyle

Lifting to Tensor DSLs gives great Performance

Loop-level Program

```
for x in range(data.shape[1]):
  mean[x] = 0.0
  for k in range(data.shape[0]):
    mean[x] += data[k][x]
  mean[x] /= data.shape[0]

for i in range(data.shape[1]):
  for j in range(data.shape[1]):
    cov[i][j] = 0.0
    for k in range(data.shape[0]):
      cov[i][j] +=
          (data[k][i] - mean[i])
          * (data[k][j] - mean[j])
    cov[i][j] /= data.shape[0] - 1.
    cov[j][i] = cov[i][j]
```



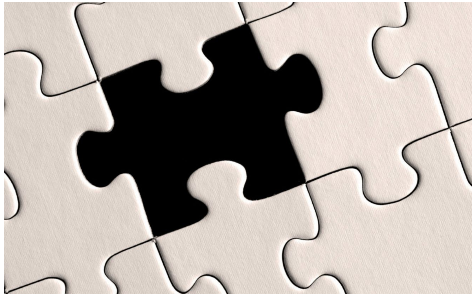
NumPy API

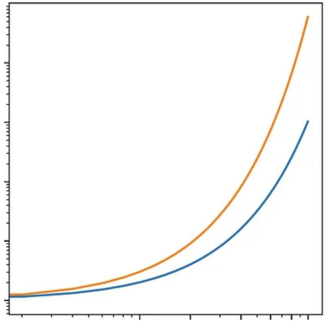
```
mean = np.sum(data, axis=0) / data.shape[0]
data = data - mean

cov = np.dot(data.T, data)
cov = cov / (data.shape[0] - 1.)
```

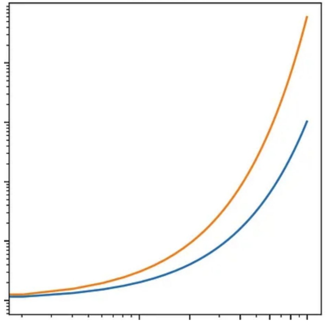
MLIR StableHLO

```
%0 = stablehlo.constant dense<0.000000e+00>
%1 = stablehlo.reduce(%arg1 init: %0) across 0
  : (tensor<1400x1200xf32>, tensor<f32>)
…
```

Challenges in Existing Approaches





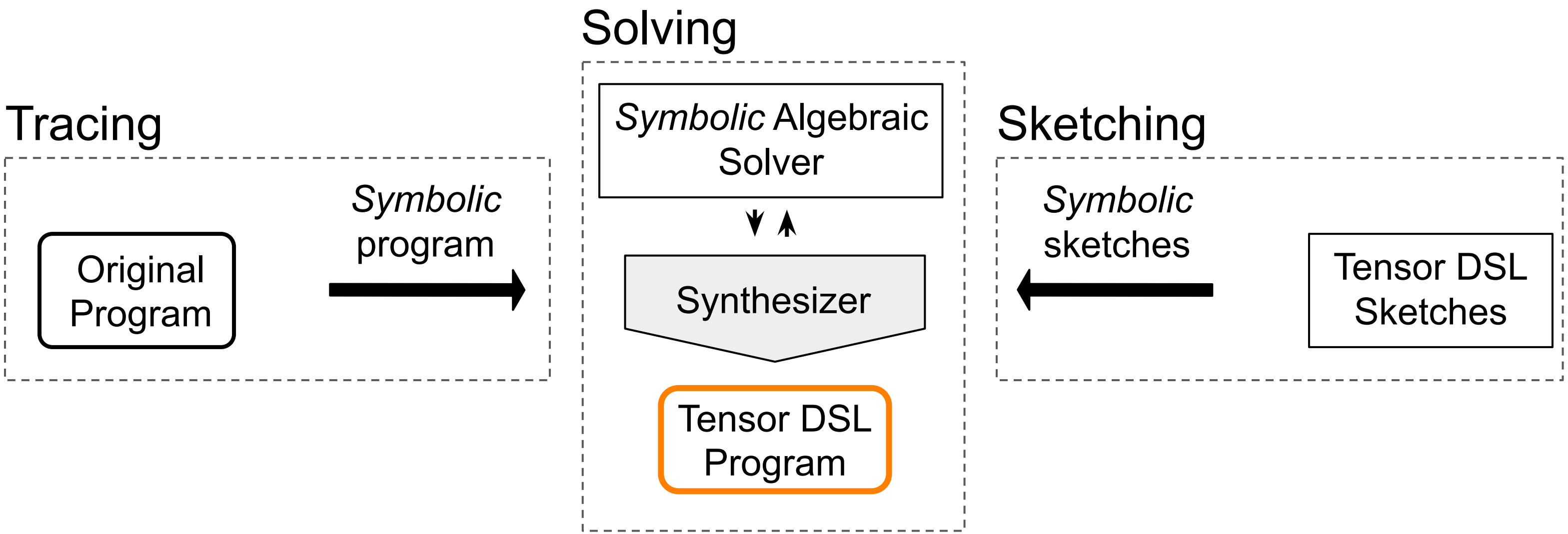


Pattern Matching
Robust ❌

Bottom-Up Program Synthesis
Robust ✅
Scalable ❌

Verified Lifting
Robust ✅
Scalable ❌

Tensorize – Decomposing the Synthesis Problem



- **Tracing:** Captures program semantics as symbolic equations
- **Sketching:** Generates sketches from Tensor DSL grammar
- **Solving:** Recursively simplifies the symbolic trace

Symbolic Tracing

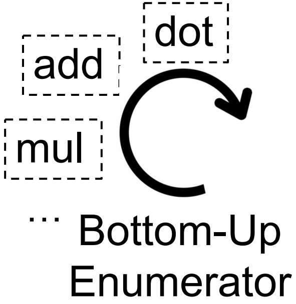
```
for i in range(data.shape[1]):
  for j in range(data.shape[1]):
    cov[i][j] = 0.0
    for k in range(data.shape[0]):
      cov[i][j] +=
          (data[k][i] - mean[i])
          * (data[k][j] - mean[j])
    cov[i][j] /= data.shape[0] - 1.
    cov[j][i] = cov[i][j]
```

$$\begin{pmatrix} cov_{0,0} & \cdots \\ cov_{1,0} & \cdots \\ \vdots & \ddots \\ cov_{m-1,0} & \cdots \end{pmatrix} =$$

$$\left(\begin{matrix} \frac{(A_{0,0}-B_0)(A_{0,0}-B_0)+(A_{1,0}-B_0)(A_{1,0}-B_0)+\cdots+(A_{n-1,0}}{C} & \\ \frac{(A_{0,1}-B_1)(A_{0,0}-B_0)+(A_{1,1}-B_1)(A_{1,0}-B_0)+\cdots+(A_{n-1,1}}{C} & \cdots \\ \vdots & \\ \frac{(A_{0,m-1}-B_{m-1})(A_{0,0}-B_0)+(A_{1,m-1}-B_{m-1})(A_{1,0}-B_0)+\cdots+(A_{n}}{C} & \end{matrix} \right.$$

Symbolic Sketching

Tensor DSL Grammars
NumPy,
StableHLO,
…



Depth 1

add(A, B) $\rightarrow A_1 + B_1$

mul(A, B) $\rightarrow A_1 * B_1$

add(A, ??) $\rightarrow A_1 + ??$

add(??, B) $\rightarrow ?? + B_1$

mul(A, ??) $\rightarrow A_1 + ??$

mul(??, B) $\rightarrow ?? + B_1$

Depth 2

add(mul(A, B), C) $\rightarrow A_1 {*} B_1 + B_1$

…

add(mul(??, B), C) $\rightarrow ??_1 {*} B_1 + C_1$

add(mul(A, ??), C) $\rightarrow A_1 {*} ??_1 + C_1$

add(mul(A, B), ??) $\rightarrow A_1 {*} B_1 + ??_1$

…

Synthesis through Symbolic Simplification

Spec: $\boxed{A_1{*}B_1 + A_2{*}B_2 + A_3{*}B_3 + C}$

$A_1{*}B_1 + A_2{*}B_2 + A_3{*}B_3 + C = A_1 + \text{??}$
$\Leftrightarrow A_1{*}B_1 + A_2{*}B_2 + A_3{*}B_3 + C - A_1 = \text{??}$

→ More complex than spec.
→ Prune

add(A, ??)
$\rightarrow A_1 + \text{??}$

add(??, C)
$\rightarrow \text{??} + C$

$A_1{*}B_1 + A_2{*}B_2 + A_3{*}B_3 + C = \text{??} + C$
$\Leftrightarrow A_1{*}B_1 + A_2{*}B_2 + A_3{*}B_3 = \text{??}$

→ Simplifying the spec.
→ Interesting to explore

Simplified Spec: $\boxed{\text{add}(A_1{*}B_1 + A_2{*}B_2 + A_3{*}B_3\text{, C)}}$

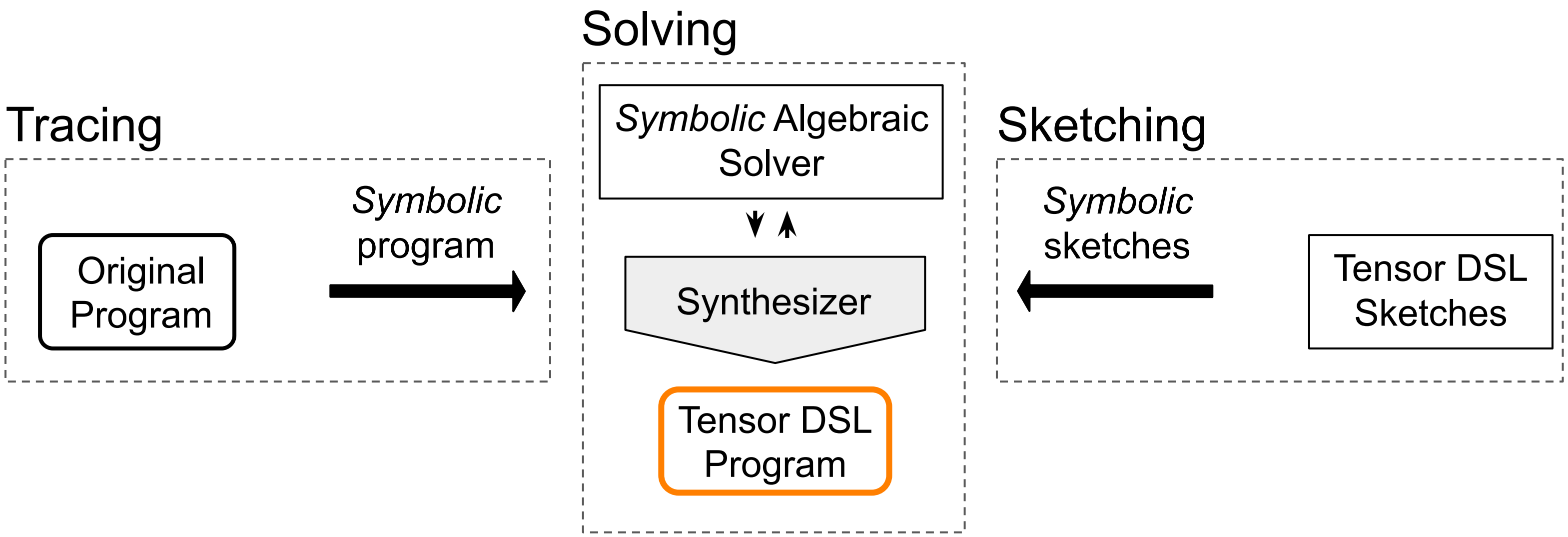
add(A, ??)
$\rightarrow A_1 + \text{??}$

add(??, C)
$\rightarrow \text{??} + C$

dot(A,B)
$\rightarrow A_1{*}B_1 + A_2{*}B_2 + A_3{*}B_3$

$\boxed{\text{add(dot(A, B), C)}}$

Putting it Back Together



Comprehensive Benchmarks

Benchmarks

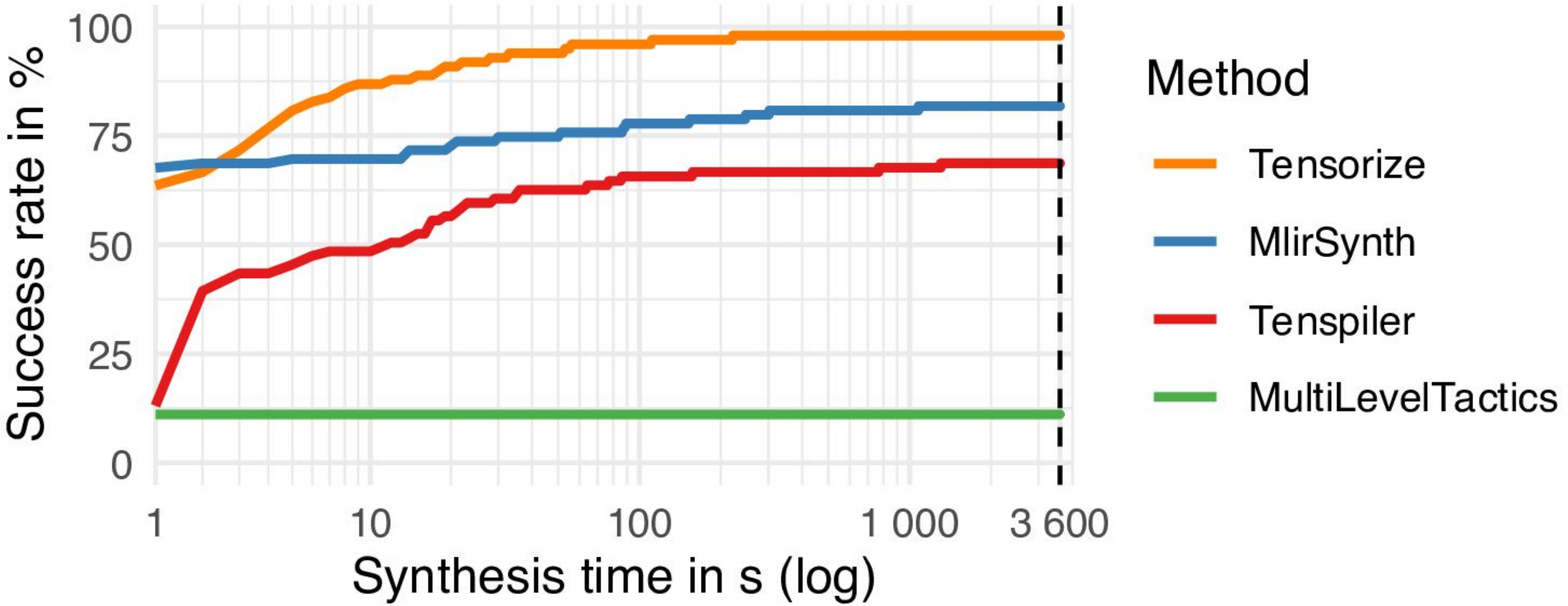
Suite	Workload	Benchmarks
blas [16]	Linear Algebra	3
blend [6]	Image Processing	12
darknet [45]	Machine Learning	14
dsp [29]	Image Processing	15
dspstone [58]	Signal Processing	5
llama [10]	Machine Learning	11
makespeare [46]	Linear Algebra	1
mathfu [11]	Math	12
polybench [42]	Data Mining, Lin. Alg.	15
simpl_array [50]	Array Programming	5
utdsp [47]	Signal Processing	6
TOTAL		**99**

- Superset of benchmarks from related works
- Diverse complexity
  - Number of Operations
  - Loop Depth
  - Operation Types
- Covering multiple application domains

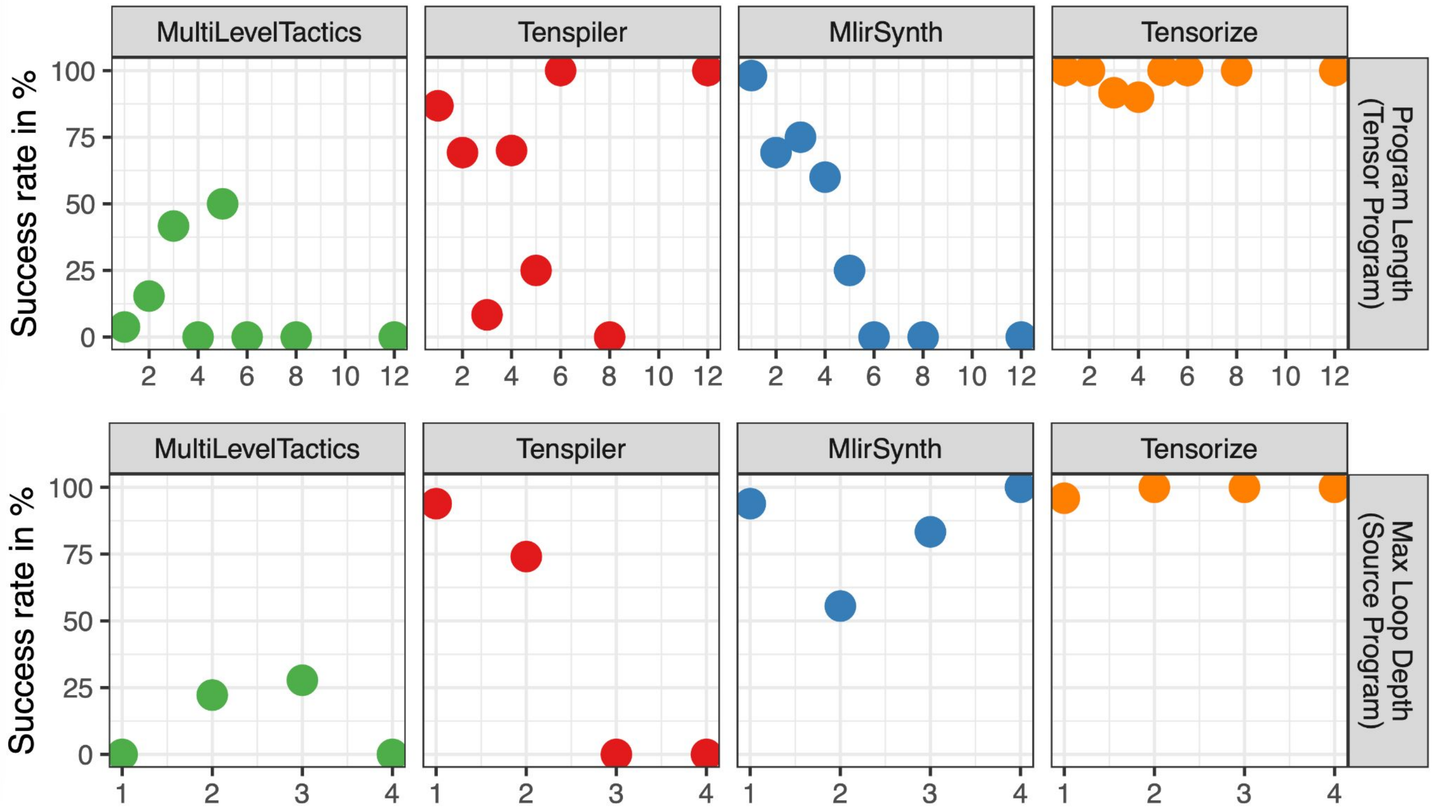
State-of-the-art Methods

Pattern Matching	Bottom-Up Program Synthesis	Verified Lifting
MultiLevel Tactics (CGO’21)	MlirSynth (PACT’23)	Tenspiler (ECOOP’24)

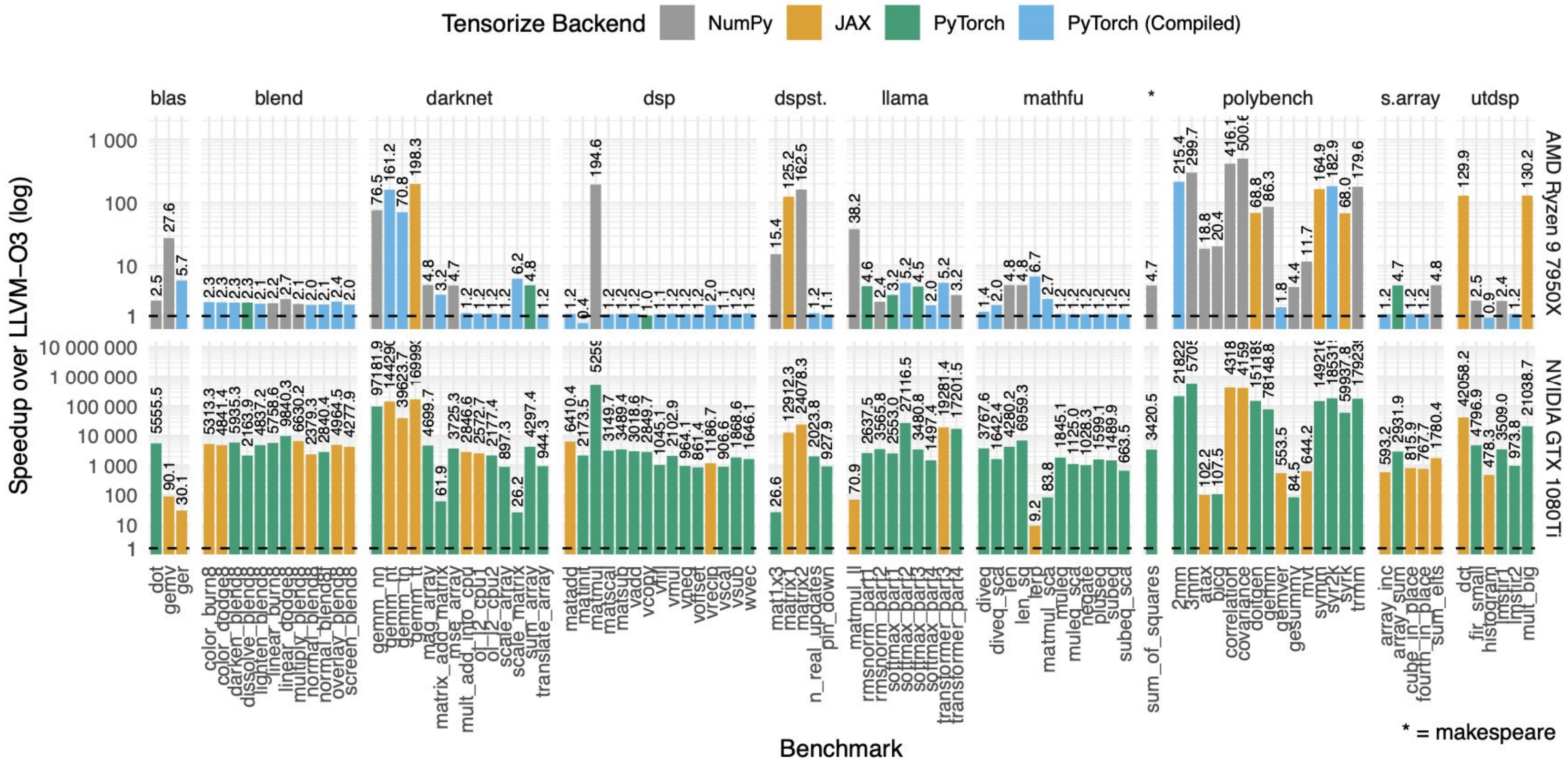
Tensorize Lifts More Benchmarks Faster



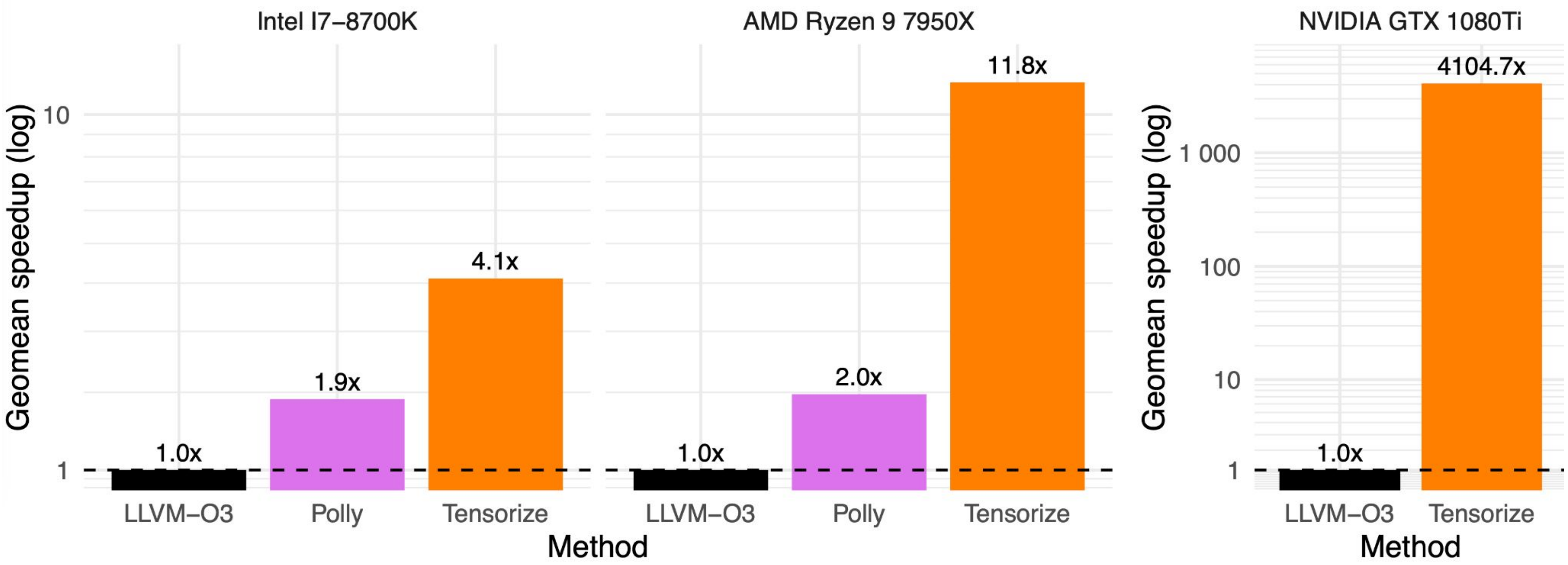
Tensorize Scales to Complex Programs



Tensorize Enables Various Tensor DSL Compilers



Tensorize Achieves Significant Speedups



Conclusions

- **Better scalability** than any previous tensor program lifting method through problem decomposition
- **Runtime speedup of 2.1x (Intel), 5.8x (AMD)** over state-of-the-art CPU compilers
- **Retargetability** to GPUs and hardware accelerators enable further speeups

Future Work

- **Faster synthesis** through C++ based implementation
- **More benchmarks** for more extensive evaluation

Tensorize is Open-Source!

- **Artifact:** https://zenodo.org/records/14095398
- **Development repo:** https://github.com/alexanderb14/tensorize

- **Implementation**: MLIR, JAX, SymPy

- **Setup**: Batteries-included
- **Usage**: Integrates in MLIR based flows

```
python tensorize/main.py --help
  --program    Source program in MLIR format.
  --synth_out  Output file of the synthesized program
               or stdout if empty.
  --target     Target Tensor DSL to lift to.
               Currently supported: numpy, hlo
  ...
```