SparTA: Deep-Learning Model Sparsity via Tensor-with-Sparsity-Attribute

Ningxin Zheng, Bin Lin, Quanlu Zhang, Lingxiao Ma, Yuqing Yang, Fan Yang, Yang Wang, Mao Yang, Lidong Zhou

Microsoft Research, Tsinghua University
Computation Capacity vs DNN Model Size

The growth of DNN model size significantly outpaces the growth of modern accelerators [1]

[1] Harmony: Overcoming the hurdles of GPU memory capacity to train massive DNN models on commodity servers
Sparsity Commonly Exists

Sparsity commonly exists in DNN models

Researchers reveal that sparsity has orders of magnitude potential for computation and memory saving

Model compression algorithms
Evolving of Sparsity Pattern

Various approaches proposed to sparsify DNN models

- Unstructured/Structure pruning
- Single precision quantization
  - Layer 1: 4 bits
  - Layer 2: 4 bits
  - Layer 3: 4 bits
- Mixed precision across layers
  - Layer 1: 16 bits
  - Layer 2: 8 bits
  - Layer 3: 4 bits
- Mixed precision within a tensor
  - 32 bits
  - 8 bits

And many more...
Obstacles of Sparsity Optimization

Myth of the proxy metrics

Diminishing End-to-End Returns

Across-Stack Innovations in Silos

With so many advanced sparsity patterns, we still only see limited gain in practice.
The Myth of Proxy Metrics

- ML researchers use **proxy metrics** due to the difficulty of kernel optimization
- Proxy metrics (FLOPs) do not necessarily translate into **real latency**
  - Default sparse DNN library often leads to suboptimal performance
Diminishing End-to-End Returns

- Operator-centric sparsity research missing global optimization opportunities
- Sparsity propagates across the graph, leading to higher sparsity ratio
Across-Stack Innovations in Silos

• No mature end-to-end system that integrates various optimizations

• Models with different sparsity need to be optimized case by case

• Individual solutions/innovations are hard to be extended to/combined with other proposals
SparTA: An End-to-End Approach to Model Sparsity

**Treat sparsity as 1\textsuperscript{st}-class citizen in DNN compiler**

- TeSA, Tensor with Sparsity Attribute, the core abstraction of SparTA
  - Allow the specification of arbitrary sparsity pattern in any tensor
- TeSA propagation, exposing the full sparsity in an end-to-end manner
- Sparsity-aware execution plan transformation and code specialization
  - Generate high-quality codes given any sparsity pattern on any DNN model
Core Abstraction: TeSA

- Same shape as the original tensor, where each element represents a sparsity attribute
- Support element-wise sparsity specification to express arbitrary sparse pattern

### TeSA: Tensor with Sparsity Attribute

<table>
<thead>
<tr>
<th>Values</th>
<th>Sparsity Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

- 4: unit4
- 8: unit8
- 0: pruned
• Perform attribute propagation to infer the sparsity attributes of all other tensors
Perform attribute propagation to infer the sparsity attributes of all other tensors

Transform the execution plan accordingly to take advantage of the given sparsity
System Architecture

- Perform attribute propagation to infer the sparsity attributes of all other tensors
- Transform the execution plan accordingly to take advantage of the given sparsity
- **Perform the sparsity-aware code specialization**
TeSA Propagation

- Different operators have **different propagation behavior**
- A clear **interface** to register/expand propagation rules for customized operators
- **TeSA algebra** and **Tensor Scrambling** can infer the propagation rule automatically
**Execution Transformation**

- Transform the target pattern into one(some) pattern(s) that are easy to optimize
- **Integrating different optimizations** can achieve better performance

```c
void matmul_block_sparse(
    float *A, float *B, float *C)
{
}

void matmul_finegrained(
    float *A, float *B, float *C)
{
}
```
Code Specialization

- Kernel-level: eliminate dead computations by hardcoding sparsity pattern in code
- Instruction-level: replacing computation with hardware specific instruction (e.g., wmma)

```
for (m1: int, 0, 2)
  for (n1: int, 0, 2)
    for (k1: int, 0, 2){
      ... // [2,2] x [2,2]
    }
```

Eliminate the dead computations annotated by sparsity attribute
What SparTA Achieves

• support *various models, sparse patterns* and *their combinations*

• discover full *end-to-end* opportunity

• *integrates* different sparse optimizations systematically

• *real implementation* (not proxy metrics) for algorithm
Evaluation on Various Patterns & Models

SparTA supports popular sparse patterns on represented models in NLP/CV/speech.

<table>
<thead>
<tr>
<th></th>
<th>PyTorch (TorchScript)</th>
<th>TensorRT</th>
<th>TVM</th>
<th>TVM-S (TVM-Sparse)</th>
<th>Rammer</th>
<th>Rammer-S (Rammer + SOTA Sparse Kernels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SparTA’s speedup (up to) vs.</td>
<td>10.6x</td>
<td>5.0x</td>
<td>7.5x</td>
<td>20.1x</td>
<td>5.8x</td>
<td>5.6x</td>
</tr>
</tbody>
</table>

Test on Nvidia 2080Ti, Batchsize=32
End-to-end Opportunity

- Propagation automatically finds out more potential sparsity in the model with Tensor Algebra or Tensor Scrambling, e.g., from 50% to 89.7%

Average sparsity ratio: 89.7%
SparTA achieves the significant speedup by integrating/combining different sparse optimizations systematically.

As far as we know, SparTA is the first work that fully utilizes such complex sparse patterns.
SparTA provides the real latency for the compression algorithm to boost the algorithm performance.
Conclusion

• We treat sparsity as the first-class citizen in DNN frameworks to natively facilitate efficient training and inference of sparse models

• We propose an end-to-end sparsity optimization system called SparTA that
  • can integrates existing various sparsity optimizations systematically
  • provides real end-to-end speed up for different sparsity patterns
  • reveals new opportunities for sparsity at the graph-level
Thanks

Q&A

Artifact available at: https://github.com/microsoft/nni/tree/sparta_artifact/sparta
Formal repo: https://github.com/microsoft/SparTA.git (will open source soon)