TANGRAM: Optimized Coarse-Grained Dataflow for Scalable NN Accelerators

Mingyu Gao, Xuan Yang, Jing Pu, Mark Horowitz, Christos Kozyrakis

Stanford University
Tsinghua University
Google

ASPLOS – April 2019
Neural Networks (NNs)

- Unprecedented accuracy for challenging applications
  - Fully-connected (MLPs), Convolutional (CNNs), Recurrent (LSTMs) NNs

- Inference: layer-wise processing on direct acyclic graphs (DAGs)

Convolutional NN

LSTM Cell

Inception Module
NN Accelerators

- Domain-specific processing engine
  - An array of specialized processing elements (PEs)
  - On-chip register files and SRAMs
  - 100x performance and energy efficiency

- Diannao/Cambricon, Google TPU, Eyeriss, Cnvlutin, EIE, ...

![Diagram of NN Processing Engine with PE, Reg File, ALU, and Global Buffer]
Scaling NN Performance

- Use more PEs & more on-chip buffers

- **Monolithic** engine
  - ✗ Low resource utilization
  - ✗ Long array buses
  - ✗ Far from SRAM

- **Tiled** architecture—*focus of our work*
  - ✓ Mostly local data transfers
  - ✓ Easy to scale up/down
  - ? Dataflow scheduling
TANGRAM: Optimizing Coarse-Grained Dataflow

- **Intra-layer parallelism**
- **Buffer Sharing Dataflow**
  - Reuse data across engines → higher energy efficiency
  - Avoid on-chip data duplication → smaller buffer area
- **Inter-layer pipelining**
- **Fine-grained data forwarding & pipelining of complex DAGs**
  - Reduce pipeline stalls → higher throughput
  - Temporarily store forwarded data → smaller buffer area
Intra-Layer Parallelism
Parallelizing a Single Layer

- Inefficient buffer use for shared data
  - × Replicated buffered data (area)
  - × Data reuse limited within each tile (energy)
- ALL parallelization schemes share some data!
Optimizing Dataflow for Shared Data

- **Skew computation order of engines**
  - All engines start in parallel ➔ high throughput

- **Rotate buffered data between engines**
  - Fully reuse shared data ➔ low energy
  - No on-chip data duplication ➔ low area
Buffer Sharing Dataflow

- Unify distributed buffers as an ideal large buffer
  - Efficiently store and reuse data

- Formalize as loop transformations
  - (tile coordinate $x$, time step $t$) -> index of data to be buffered $i$
  - See paper for detailed maths

- Easy to implement
  - Buffer controller fetches from memory or other tiles
  - No changes for dataflow within a tile

- Support all parallelization schemes (including hybrid)
Inter-Layer Pipelining
Pipelining Multiple Layers

- Pros: avoid off-chip intermediate data accesses
  - Save DRAM bandwidth and energy

- Cons: utilize resources less efficiently
  - Long delays: pipeline filling/draining due to inter-layer data dependencies
  - Large SRAM buffers: store entire intermediate data
Fine-Grained Data Forwarding

- Forward each **subset** of data to the next layer as soon as ready
  - Reduce pipeline stalls: next layer starts earlier
  - Reduce buffer capacity: only store the subset currently being forwarded

- Require matched access patterns between adjacent layers

```plaintext
foreach b in batch Nb
    foreach ifmap i in Ni
        foreach ofmap o in No
            // 2D conv
            0[b][o] += I[b][i] * W[o][i]
```

No dependencies; trivially pipelined
Alternate Layer Loop Ordering (ALLO)

**Unoptimized**
- Buffer for ALL fmaps
- Delay for ALL fmaps

**Optimized**
- Buffer for ONE fmap
- Delay for ONE fmap
- Benefits apply to half of all layers
Layer Pipelining for Complex NN DAGs

- A dataflow tool explores pipeline schedules of multiple layers
- Subject to design rules due to data dependency constraints
  - E.g., no multiple predecessor layers on-chip
Evaluation Results
Modeling Methodology

- State-of-the-art NNs
  - CNNs: AlexNet, VGGNet, GoogLeNet, ResNet
  - MLPs & LSTMs: medium and large scales

- Hardware
  - Inference engine: Eyeriss [ISCA’16], 8 × 8 PEs, 32 kB buffer, 500 MHz
  - Off-chip memory: LPDDR3-1600, 4 channels
  - Overall chip: 16 x 16 tiles
    - 16384 PEs + 8 MB SRAM
    - 90 mm² at 28 nm
Overall Comparison

- Base tiled vs. monolithic: 3.6x performance, 7% worse energy
  - Less flexible and less efficient use of on-chip SRAM buffers
- TANGRAM: 2x over base tiled, outperforms monolithic
Intra- vs. Inter-Layer Optimizations

- **Intra-layer: Buffer Sharing**
  - AlexNet: fit large fmaps on-chip
  - MLP-L: enable weight pinning

- **Inter-layer: ALLO + complex DAGs**
  - AlexNet, GoogLeNet & LSTM-M
  - Linear NNs benefit less

![Energy Comparison Diagram](image-url)

- **Energy**
  - TANGRAM
  - w/o Intra
  - w/o Inter

<table>
<thead>
<tr>
<th>Network</th>
<th>TANGRAM</th>
<th>w/o Intra</th>
<th>w/o Inter</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>1</td>
<td>1.5</td>
<td>2.5</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>1</td>
<td>1.5</td>
<td>2.5</td>
</tr>
<tr>
<td>MLP-L</td>
<td>1</td>
<td>1.5</td>
<td>2.5</td>
</tr>
<tr>
<td>LSTM-M</td>
<td>1</td>
<td>1.5</td>
<td>2.5</td>
</tr>
</tbody>
</table>
Summary

- Efficiently scale NN acceleration
  - Coarse-grained parallel dataflow on tiled architectures
  - Optimized tiled architectures outperform monolithic engines

- TANGRAM: dataflow optimizations
  - Intra-layer buffer sharing
  - Inter-layer pipelining with fine-grained data forwarding
  - Pipelining complex NN DAGs

- Dataflow scheduling tool open sourced
  - https://github.com/stanford-mast/nn_dataflow

Thank you!