**dMazeRunner: Optimizing Convolutions on Dataflow Accelerators**

**Shail Dave¹, Aviral Shrivastava¹, Youngbin Kim², Sasikanth Avancha³, Kyoungwoo Lee²**

[¹] Compiler Microarchitecture Lab, Arizona State University  
[²] Department of Computer Science, Yonsei University  
[³] Parallel Computing Lab, Intel Labs

45th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2020)
Must-Accelerate Applications in ML Era

Widely Used ML Models

- Multi Layer Perceptrons
  - http://yann.lecun.com/exdb/lenet/
- Convolution Neural Networks
  - http://vision03.csail.mit.edu/cnn_art/index.html
- Sequence Models
  - https://deeplearning.mit.edu/
- Graph Neural Networks
  - Points of Interest
    - AlphaGo.
    - https://www.nature.com/articles/nature24270
  - Delaunay Triangulation
    - YOW! Data 2018 Conference.
    - https://www.youtube.com/watch?v=iDRb3CjESmM

Popular Applications

- Object Classification/Detection
- Media Processing/Generation
- Large-Scale Scientific Computing
- Designing Software 2.0
  - Google shrinks language translation code from 500k LoC to 500
  - Designing Computer Systems for Software 2.0.
  - Kunle Olukotun, NeurIPS 2018 Invited talk.
  - and more ...

Compute Intensive, Needs Energy-efficient Acceleration
Dataflow Accelerators: Promising Solution

Known variations include - Systolic Arrays,
  - Spatially Programmable Architecture,
  - Coarse-Grain Reconfigurable Array (CGRA)

- **Massive arrays of processing elements (PEs).**
  - **Simple:** absence of complex out-of-order pipeline and decoding.
  - **Programmable:** accommodate executing all operations within a loop.

- Private and shared memory for PEs **sustain data reuse.**

- PEs can be busy **performing computations while data is being communicated** from lower memories.
  - Taken care by effective data management i.e., software prefetching, data distribution, data allocation.

Mapping Computation in Space and Time

for m=1:2
  for oy=1:3
    for ox=1:3
      for fy=1:3
        for fx=1:3
          \( O[m][oy][ox] += I[oy+fy-1][ox+fx-1] \times W[m][fy][fx] \);
  
for m_L3=1:2
  for fy_L2=1:3
    access_SPM_and_comm_NoC()
  for fx_L1=1:3
    % filter row
    for oy_S=1:3
      % o/p row
      O[m_L3][oy_S][ox_S] += W[m_L3][fy_L2][fx_L1] \times I[oy_S+fy_L2-1][ox_S+fx_L2-1];
    for ox_S=1:3
      % o/p column

0 \times 3 \times 3 = \text{channel1}

Oy=3

Ox=3

Data in RF of PE(1,1)
I: 1x3, W: 1x1x3, O: 1x1x1

Execution on PE(1,1): \( O[m_L3][1][1] += W[m_L3][fy_L2][fx_L1] \times I[fy_L2][fx_L1] \)
Many many ways to execute nested loops (of DNN) on a dataflow accelerator

Both software and hardware design space

**Hardware**: Size, layout, and connectivity of PEs, size of on-chip buffer and registers, etc.

**Software**: loop mappings, e.g., **Spatial**: parallelism, data reuse, **Temporal**: ordering and tiling of the loops, data reuse, data buffering, etc.

Vast “Execution Method” Space

```
for n=1:N
    for m=1:M
        for c=1:C
            for ox=1:Ox
                for oy=1:Oy
                    for fx=1:Fx
                        for fy=1:Fy
                            O[n][m][ox][oy] +=
                                I[n][c][ox+fx-1][oy+fy-1]*
                                W[m][c][fx][fy];
                        end
                    end
                end
            end
        end
    end
end
```

4D Convolution:

- 1 ifmap
- * 512 filters
- 1 ofmap

**Concurrent Execution on PEs**

- L1 Accesses
- Scratch-Pad Memory
- L2 Accesses
- DRAM
- L3 Accesses

**Hardware Accelerator**

```
<,N,M,C,Ox,Oy,Fx,Fy> =
<1,512,256,7,7,3,3>
```

```
Conv5_1 [ResNet]
```
The problem of exploring the “execution methods” for mapping 7-deep nested loop onto dataflow accelerator becomes the problem of exploring all the possibilities of tiling and ordering factors in the 28-deep nested loop.
dMazeRunner Demo

Framework Features

1. Estimate performance and energy-efficiency of individual execution method for a specified accelerator hardware and layer dimensions

2. Optimize specific or all layers of common CNN networks

3. Explore efficient accelerator designs for DNNs

Release: https://github.com/MPSLab-ASU/dMazeRunner
Features with detailed modeling of:

✓ Analyze arbitrary perfectly nested loops.
✓ miss penalty and stall cycles (during PE execution and in managing PE’s local or shared memory).
✓ inter-PE communication.
✓ temporal/spatial data reuse.
✓ Integrated support common ML libraries MXNet/Keras/... (thanks TVM! – leveraging front-end)

Validation against DNN Optimizer of Yang et al.

• Energy estimate differs by ~4.2% for variety of execution methods.
• For efficient mappings, major energy spent in RF accesses.

---

Shail Dave, Youngbin Kim, Sasikanth Avancha, Kyoungwoo Lee, Aviral Shrivastava, dMazeRunner: Executing Perfectly Nested Loops on Dataflow Accelerators [CODES+ISSS, TECS 2019].
Optimizing Multiple Dataflows

Executing ResNet layers on a 256-PE, 512B RF, 128kB SPM dataflow accelerator

![Diagram showing Energy-Delay Product (EDP) for different dataflow mechanisms.](image)
Adaptable Mappings Yield Better Results

- Adapts to layer / hardware architecture characteristics
  - Scales for layers and tensors of different shapes
- Finds non-intuitive mappings that are optimized for various key factors e.g.,
  - High Resource (PE/memories) Utilization
  - Maximized Reuse of Multiple Tensors
  - Minimized DRAM accesses
  - Hiding Communication Latency Behind Computations on PEs

**Example Mapping of ResNet Conv5_2 with Output Stationary Dataflow**

<table>
<thead>
<tr>
<th></th>
<th>MOC</th>
<th>dMazeRunner</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE Compute vs. Data comm. Latency:</td>
<td>144 vs. 648</td>
<td>576 vs. 576</td>
</tr>
<tr>
<td>Total cycles:</td>
<td>~10,616,832</td>
<td>~2,459,648</td>
</tr>
<tr>
<td>Ideal execution cycles for output-stationary:</td>
<td>2,359,296</td>
<td>2,359,296</td>
</tr>
<tr>
<td>Reduction in DRAM accesses (ifmaps, weights):</td>
<td>(1x, 1x)</td>
<td>(4.57x, 2x)</td>
</tr>
<tr>
<td>Perf. improvement (normalized to MOC):</td>
<td>1x</td>
<td>4.44x</td>
</tr>
<tr>
<td>Energy-Delay-Product reduction (normalized):</td>
<td>1x</td>
<td>9.86x</td>
</tr>
</tbody>
</table>

MOC: Simultaneous spatial processing of Multiple Output Channels
Achieving Close-to-Optimal Solutions in Seconds

Search Space Reduction for DNN Optimizations

<table>
<thead>
<tr>
<th>ResNet Conv Layers</th>
<th>Loop Tilings</th>
<th>Loop Orderings</th>
<th>dMzRnr explored Tilings</th>
<th>dMzRnr explored Orderings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.2E+08</td>
<td>7!×7!</td>
<td>46812</td>
<td>3×3</td>
</tr>
<tr>
<td>2.2</td>
<td>1.1E+09</td>
<td>7!×7!</td>
<td>122092</td>
<td>3×3</td>
</tr>
<tr>
<td>3.2</td>
<td>6.2E+08</td>
<td>7!×7!</td>
<td>53690</td>
<td>3×3</td>
</tr>
<tr>
<td>4.2</td>
<td>1.8E+08</td>
<td>7!×7!</td>
<td>10938</td>
<td>3×3</td>
</tr>
<tr>
<td>5.1</td>
<td>1.4E+07</td>
<td>7!×7!</td>
<td>877</td>
<td>3×3</td>
</tr>
<tr>
<td>5.2</td>
<td>1.7E+07</td>
<td>7!×7!</td>
<td>753</td>
<td>3×3</td>
</tr>
</tbody>
</table>

Search Space Exploration on an Intel i7-6700 Quad-core CPU

- **min:** ~1 second, ResNet conv5_2 (753×9 methods)
- **max:** ~122 seconds, ResNet conv2_2 (122092×9 methods)

Quick exploration:
- In-built support for a few common opt strategies.
- Implementation – multi-threaded, caches commonly invoked routines of analytical model.
- Enables effective Design Space Exploration of architecture, e.g., explore impact of scaling PEs and memory sizes on performance and energy efficiency.

Even domain non-experts can explore the space

```
python run_optimizer.py --frontend mxnet --model resnet18 --layer-index 0
```

Does not preclude experts/programmers from directing the search.

AlexNet and ResNet18 models in about 18 and 180 seconds, respectively
Conclusions

- **Dataflow accelerators**: promising for accelerating ML applications.
- Need to determine efficient “execution method” for spatiotemporal executions on dataflow accelerators.
- **dMazeRunner**: Automated, succinct, and fast exploration of mapping search space and architecture design space.
- Adaptive and non-intuitive mappings enable efficient dataflow acceleration.

[Release] [https://github.com/MPSLab-ASU/dMazeRunner](https://github.com/MPSLab-ASU/dMazeRunner)
[Project] [https://labs.engineering.asu.edu/mps-lab/ml-accelerators](https://labs.engineering.asu.edu/mps-lab/ml-accelerators)