PHITE: Portable High-performance Inference at the Tactical Edge

November 14, 2022
Scott McMillan
Principal Research Engineer
Copyright 2022 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM22-0873
PHITE: Enabling AI for Decision-Making Advantage at the Tactical Edge

**Problem:** Today’s AI software is computationally expensive and requires extensive knowledge, skill, and effort to adopt on low-power devices at the tactical edge.

**Solution:** Develop an open-source library of machine learning (ML) algorithms optimized for low-power (100’s mW—0’s W) embedded devices.

**DoD Benefit**
- Aid deployment of ML across a spectrum of edge-based applications.
- Enable rapid adoption of new/novel embedded hardware architectures.
- Provide efficient use of limited hardware for performance gains in AI/ML applications.
- Enable a wider range of applications at the tactical edge through portable and more capable software foundations.

**Areas of Opportunity**
- Soldier-borne sensors
- Predictive maintenance
- Unattended sensors
- IoT/Io(B)T devices

**Photo:** U.S. Army

Longer operational times ➞ Increased situational awareness/force protection ➞ Less weight ➞ Increased mobility
The Advanced Targeting and Lethality Aided System (ATLAS) is an emerging targeting technology being developed by DEVCOM’s C5ISR and Armaments Centers. ATLAS uses cutting-edge sensing technologies and machine-learning algorithms to automate manual tasks during passive target acquisition, allowing crews to engage three targets in the time it would normally take for them to engage one.

“Advancements from the PHITE project will improve mission critical parameters in current edge systems and make possible new edge systems.”

— Forrest Bussler, Chief, Embedded Hardware and Processing Branch, US Army DEVCOM C5ISR Center
Dataset: Automatic Target Recognition (ATR)

300 GB of full-motion video clips at multiple ranges and aspects (moving in circles, walking in figure eights):

• Tanks
• Armored vehicles
• Trucks
• People

https://dsiac.org/databases/atr-algorithm-development-image-database/
Our goal is to maximize analytic capability at the smallest scales.

System: DGX-2 | Jetson AGX Xavier | Raspberry Pi PICO Microcontroller
---|---|---
Power | 10kW | <30W | 330mW
Cost | $399,000 | $999 | $4
Memory | 1TB system/512GB GPU | 32GB 256-bit LPDDR4x | 264KB RAM (2MB flash)
Processors | Intel Platinum (24 cores) x 2 + NVIDIA Tesla V100 x 16 | 512-core Volta GPU w/ 64 Tensor Cores (8 Volta SMs) | RP2040: ARM Cortex-M0+ (dual core)
Peak | 2 petaFLOPS | 1.41 teraFLOPS | 266 megaFLOPS
Model/Size | BiT-M(ResNet) / 900M parameters | AlexNet / 60M parameters | MobileNet V2 / 3M parameters

Push analytics capability to the right.
Approach: Extend and Apply CMU’s Research on Direct Convolutions

High Performance Zero-Memory Overhead Direct Convolutions

Jiyuan Zhang¹  Franz Franchetti¹  Tze Meng Low¹

Abstract

The computation of convolution layers in deep neural networks typically rely on high performance routines that trade space for time by using additional memory (either for packing purposes or required as part of the algorithm) to improve performance. The problems with such an approach are two-fold. First, these routines incur additional memory overhead which reduces the overall size of the network that can fit on embedded devices with limited memory capacity. Second, these high performance routines were not optimized for performing convolution, which means that the performance obtained is usually less than conventionally expected. In this paper, we demonstrate that direct convolution, when implemented correctly, eliminates all memory overhead, and yields performance that is between 10% to 400% times

Performance normalized to OpenBLAS GEMM on AMD PileDriver 4.0 GHz 4 cores/threads Normalized Performance (GOp/s) 1.8

Packing Overhead

Conv1 Conv2 Conv3 Conv4 Conv5

Alexnet Layers

Figure 1. High performance direct convolution implementation achieves higher performance than a high performance matrix multiplication routine, whereas matrix-multiplication based convolution implementations suffers from packing overheads and is limited by the performance of the matrix multiplication routine

Our Team

CMU / SEI

Dr. Scott McMillan, Principal Investigator
Principal Engineer – MTS, AI Division, SEI

Prof. Tze Meng Low, Co-Principal Investigator
Assistant Research Professor, Electrical and Computer Engineering, CMU

Jay Palat
Senior Engineer, AI Division, SEI

Oren Wright
Senior Researcher – MTS, AI Division, SEI

Upasana Sridhar
PhD, Electrical and Computer Engineering, CMU

Nicolai Tukanov
PhD, Electrical and Computer Engineering, CMU

CMU / ECE

Pankti Rajesh Shah
ECE master's student

Navya Chandra
ECE master's independent study:
“Fused convolution on Pi Pico”
SMaLL: Software for Machine Learning Libraries

**Approach:** Two APIs:

- **Usability:** A high-level CNN API provides common functionality for machine learning developers.

- **Performance-portability:** A low-level micro-kernel API defines a small number of primitives to be hand-optimized by hardware experts for specific hardware.
SMaLL: **Software for Machine Learning Libraries**

### Approach: Prioritizing support for object detection and image classification models

<table>
<thead>
<tr>
<th>Applications &amp; Benchmarks (reference models, data, validation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN API</td>
</tr>
<tr>
<td>Orchestration: data layout and loop structure</td>
</tr>
<tr>
<td>μKernel API</td>
</tr>
<tr>
<td>Microkernels (HW-specific)</td>
</tr>
<tr>
<td>Hardware</td>
</tr>
</tbody>
</table>
SMaLL: Software for Machine Learning Libraries

Neural network layers currently implemented in the high-level SMaLL Library API:

- Convolution, partial and group
- 1x1 Convolution
- Depth-wise Convolution
- Max Pooling
- Activation (ReLU)
- Fully Connected (FC)—implemented as GEMM (or MMM) or 1x1 direct convolution
Coverage of MLPerf ‘Tiny’ and ‘Mobile’ Benchmarks

**Approach:** Prioritizing the object detection and image classification models

- **Yellow** → some layers not yet supported (e.g., Upsampling Convolution)
- **Red** → model type requires more study (e.g., Embedding Layers, Attention)

<table>
<thead>
<tr>
<th>Task</th>
<th>Models</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyword Spotting</td>
<td>DS-CNN</td>
<td>Speech Commands</td>
</tr>
<tr>
<td>Visual Wake Words</td>
<td>MobileNet</td>
<td>Visual Wake Words Dataset</td>
</tr>
<tr>
<td>Image Classification</td>
<td>ResNet</td>
<td>Cifar10</td>
</tr>
<tr>
<td>Anomaly Detection</td>
<td>Deep AutoEncoder</td>
<td>ToyADMOS</td>
</tr>
<tr>
<td>Image classification</td>
<td>MobileNetEdgeTPU</td>
<td>ImageNet</td>
</tr>
<tr>
<td>Object detection</td>
<td>MobileDETs</td>
<td>MS-COCO 2017</td>
</tr>
<tr>
<td>Segmentation</td>
<td>DeepLabV3+</td>
<td>ADE20K (32 classes, 512x512)</td>
</tr>
<tr>
<td>Segmentation</td>
<td>MOSAIC (U-Net)</td>
<td>ADE20K (32 classes, 512x512)</td>
</tr>
<tr>
<td>Language processing</td>
<td>Mobile-BERT</td>
<td>SQUAD 1.1</td>
</tr>
</tbody>
</table>

https://mlcommons.org
SMaLL: Software for Machine Learning Libraries

**Approach:** applying recent research advances in optimized computation
- “managing” the data
- “orchestrating” the computation

<table>
<thead>
<tr>
<th>Applications &amp; Benchmarks (reference models, data, validation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN API</td>
</tr>
<tr>
<td>Orchestration: data layout and loop structure</td>
</tr>
<tr>
<td>μKernel API</td>
</tr>
<tr>
<td>Microkernels (HW-specific)</td>
</tr>
<tr>
<td>Hardware</td>
</tr>
</tbody>
</table>
Convolution Operation—End to End

```plaintext
for j: 0 to K
    for l: 0 to Y
        for k: 0 to X
            for i: 0 to C
                for n: 0 to H_f
                    for m: 0 to W_f
                        output_tensor[j][l][k] +=
                        (in_tensor[i][l * s + n][k * s + m] * filter[j][i][n][m])
```

in_tensor

out_tensor

filter

X

Y

K

W_f

H_f

C

j

k

l
Much of our efforts are targeted at extending CMU’s 2018 research on direct convolutions.

- Custom data layout instead of packing
  - Saves memory
  - Blocks data for memory hierarchy
- Direct convolution loop nest is more computationally efficient

Orchestration: Data Layout and Loop Structure


```plaintext
for j': 0 to C_o/C_{ob} in parallel
  for i': 0 to C_i/C_{ib}
    for l: 0 to H_o
      for k': 0 to W_o/W_{ob}
        for n: 0 to H_f
          for m: 0 to W_f
            for ii: 0 to C_{ib}
              for kk: 0 to W_{ob}
                for jj: 0 to C_{ob}
                  out_tensor[j'*C_{ob}+jj][k'*W_{ob}+kk][l] +=
                  (in_tensor[i'*C_{ib} + ii][s*k'*W_{ob} + kk + m][l*s + n] *
                   filter[i'*C_{ib} + ii][j'*C_{ob} + jj][m][n])
Now 9 loops; outer loop is parallelized
```
Orchestration: Data Layout and Loop Structure

Three tuning parameters to block data for different hardware platforms

Beyond Element-wise Fusion for Reducing Convolutional Neural Nets Sizes

Upasana Sridhar*, Navya Chandra*, Martin Schatz+, Scott McMillan*, Tze Meng Low*
* Meta, Inc. (Facebook)
+ Software Engineering Institute, Carnegie Mellon University
* Dept. of Electrical and Computer Engineering, Carnegie Mellon University

ABSTRACT
Fusing multiple layers in a deep learning network is commonly recognized as an approach to improve performance and reduce the amount of memory required. However, current approaches to fused layers are often limited to those that contain element-wise operations, such as Activation and Batch Normalization. More complicated layers are often not fused as the indexing overheads are often considered to be more costly than the benefits of fused layers. In this work, we show that fusing non-element-wise operations can be beneficial. Fundamental to our approach is the ability to express CNN layers using the same loop nest; simplifying the analysis and thus making it easier to specify how to fuse layers together. We show that this fusion produces a 1.5-10x reduction in the memory requirement. Moreover, we show that the fused implementations also produce a runtime improvement on the order of 4.6x - 9.9x compared to PyTorch and 1.2x - 20.2x compared to Tensorflow compiled with XLA.

Figure 1: Post-fusion compute graph for Bottleneck (1 × 1) Convolution + ReLU Activation + Depthwise Convolution block using XLA demonstrating that fusion in XLA is limited. The Relu layer has been fused with the Bottleneck convo-
New research results on combining (or fusing) neural network layers.

- Specifically targeting convolution layers
- $1.5x-10x$ memory reduction
- $1.2x-20x$ performance gains over PyTorch and Tensorflow
**SMaLL: Software for Machine Learning Libraries**

**Innermost loops** define the microkernels the low-level API.

Microkernels are developed for specific targeted hardware (sometimes in assembly code).

**Performance models** developed from experiments using microkernels.

These models inform the selection of the **data blocking factors** in the orchestration layer.

<table>
<thead>
<tr>
<th>Applications &amp; Benchmarks</th>
<th>CNN API</th>
</tr>
</thead>
<tbody>
<tr>
<td>(reference models, data, validation)</td>
<td></td>
</tr>
<tr>
<td><strong>Orchestration:</strong></td>
<td></td>
</tr>
<tr>
<td>data layout and loop structure</td>
<td></td>
</tr>
<tr>
<td><strong>μKernel API</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Microkernels</strong> (HW-specific)</td>
<td></td>
</tr>
<tr>
<td><strong>Hardware</strong></td>
<td></td>
</tr>
</tbody>
</table>
Publications


What’s Next

• Developing microkernels for ATLAS hardware platform and benchmarking
• Adding support for more neural network layers
• Implementing all possible fused layers
• Open-source software release
• Explore integration with the MLIR ecosystem

For further information:
Scott McMillan
info@sei.cmu.edu