Member Brief: SEI, AI Division, Advanced Computing Lab

Overview of Brief

- **Topic**: Optimization of ML Algorithms for Constrained SWAP Edge Computing in support of Decision Advantage
- **Purpose**: Seeking Mission Partners
- **Description**: Introduce the PHITE project, FY22/23 $1.5M internally funded applied R&D

- **Host Organization**: Software Engineering Institute (DOD FFRDC at Carnegie Mellon University)
- **Briefer**: Dr. Scott McMillan, Principal Research Engineer (smcmillan@sei.cmu.edu)

Objectives, Tasks, and Deliverables

- AIA SWG Objectives/Tasks
  - #3: Coordinate/Maintain: DoD AI Portfolio
    - Greater understanding of R&D underway with opportunity to leverage existing investment
  - #4: AI Guidance
    - Potential opportunity to inform AIA roadmap through lessons learned and ability to exercise use cases
  - #5: Validate through Partnerships
    - Serves as a Pathfinder for JADC2

Summary of Brief

Discussion of the PHITE (**P**ortable **H**igh-performance **I**nference at the **T**actical **E**dge) project

- Motivation
- High-level technical approach
- Campus collaboration
- Key Dates & Milestones

Working Group Discussion

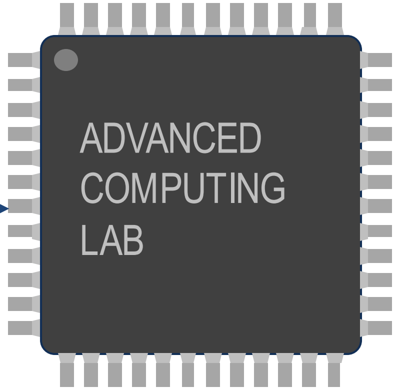
- What groups/missions would benefit from PHITE technology? E.g., force protection, ISR, humanitarian operations, predictive maintenance, autonomous platforms.

- What ML algorithms and hardware platforms do these groups/missions deploy?

About us

Mission
Apply our world class expertise to solve advanced computing problems in artificial intelligence at all scales – edge, cloud, and HPC – that enable continuous improvement of the warfighter's ability to defend our nation.

AI Division



Vision
To provide the warfighter artificial intelligence solutions that use the latest and most advanced computing technology at all scales in a timely, reliable, and safe and secure manner that maximizes warfighter effectiveness.

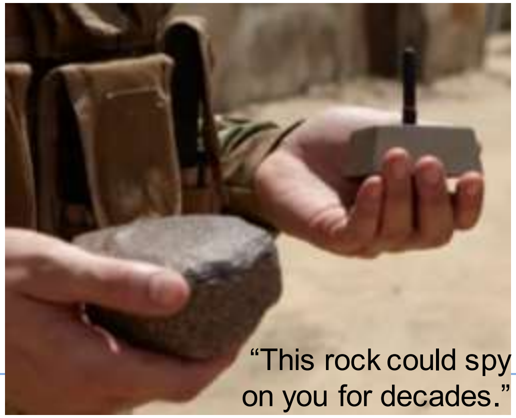
AI at the Tactical Edge

PHITE: Portable High-performance Inference at the Tactical Edge

- Wearable technology
- Autonomous UAVs/UGVs for
  - ISR
  - HADR
- Ground/Unattended Sensors
- Smart/Cognitive Radio
- TAK Display

Key challenges

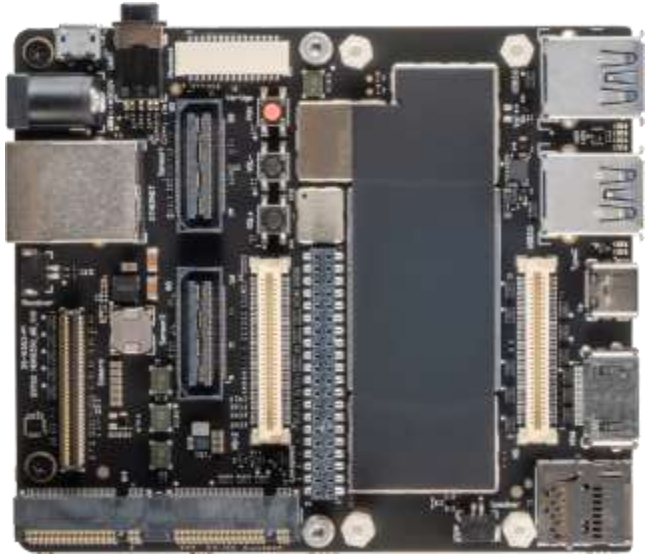
- SWAP
- Battery limitations
- Mission time



"This rock could spy on you for decades."

PHITE Goal: do more with less.

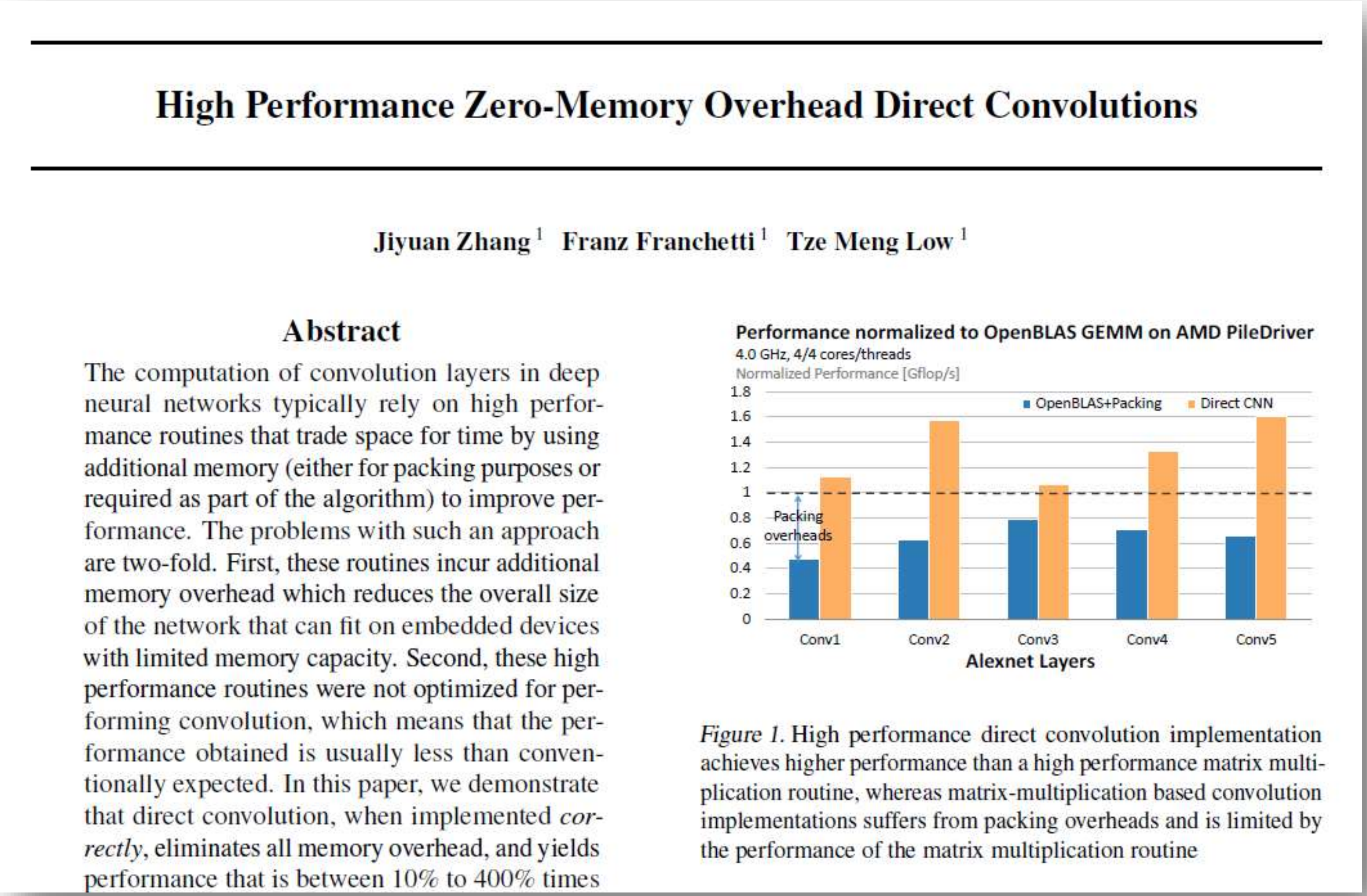
Hardware at-a-glance




System:	**Qualcomm Snapdragon 888 Development Kit**	**Raspberry Pi Model 4-B SOC**	**Raspberry Pi PICO Microcontroller**	**Your hardware here…**
Power	3-8W platform, <TBD>W Hexagon	~3W (idle) – 7W (4 cores)	6mW - 330mW	
Cost	$1,349 (for a dev board)	$35-$75	$4	
Memory	<TBD>GB dedicated to Hexagon	2/4/8 GB SDRAM	264KB RAM (2MB flash)	
Processor	Kryo 680 CPU, Adreno 660 GPU, Hexagon 780 AI Accelerator	Broadcom BCM2711, ARM v7l, Cortex-A72 (4 cores)	RP2040: ARM Cortex-M0+ (dual core)	
Peak	26 TOPS (Hexagon)	13.5 gigaFLOPS	266 megaFLOPS	
Model/Size	<TBD>	AlexNet / 60M parameters	MobileNet V2 / 3M parameters	

Background

- Much of our efforts are targeted at extending the 2018 research on direct convolutions.
- Lower memory footprint (no overhead)
- More efficient computation



High Performance Zero-Memory Overhead Direct Convolutions

Jiyuan Zhang [1] Franz Franchetti [1] Tze Meng Low [1]

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

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

In *International Conference on Machine Learning*, pp. 5776-5785. PMLR, 2018.

Key Dates and Milestones

Major Milestones	Fiscal Year	
	22	23
Select and analyze three edge applications (at least one DoD problem)	X	
Select and analyze hardware capability of at least three HW devices	X	
Develop performance models for HW devices	X	
Develop optimized algorithms for targeted HW (at least 1 in FY22)	X	X
API and library development, application development on HW		X
Evaluation of applications against baseline system, benchmarking		X
Demonstration of DoD application with streaming data		X

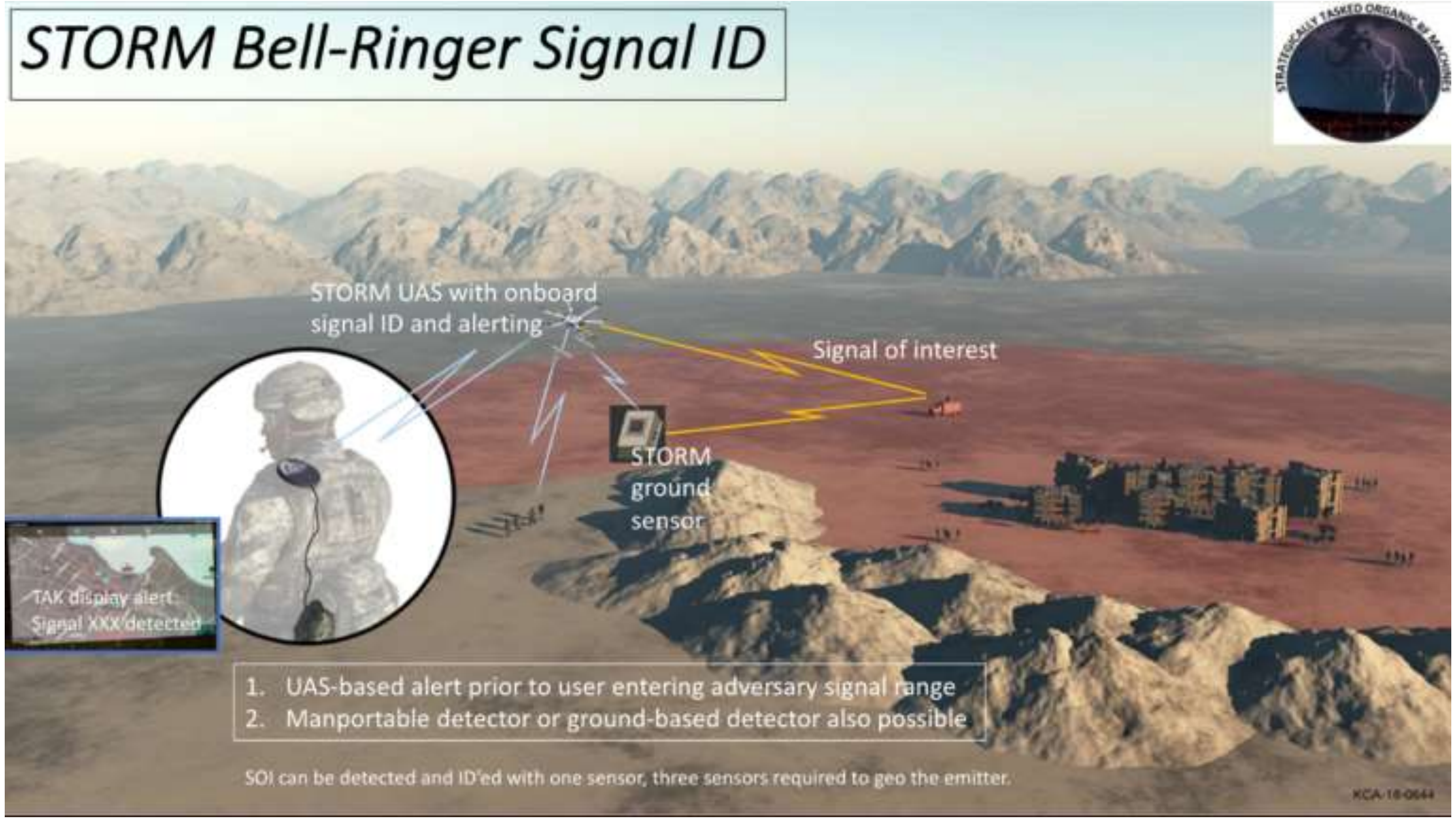
Hardware-Software Co-design for Edge AI Applications

"In general, ***compute*** can improve mission time and lower energy consumption by as much as 5X."

Boroujerdian, Behzad, et al. "Why compute matters for UAV energy efficiency?" (2018)

Objective 1: Application dependent SoC co-designs for stakeholders with accelerators that are 2x more energy efficient than GPU solutions and are 30% better in mission metrics.

Objective 2: A co-design flow methodology and **codriver** implementation that includes four tools per pipeline stage and enables co-design of RTL for an application that includes accelerator IP *within two months*.



The 'Ideal' mission partner

- Needs ML algorithms to run on low-power, mobile and embedded hardware devices (use case)
- Can provide access to algorithms and data to validate results
- Can provide access to hardware exemplars or specifications of COTS platforms
- Is engaged, focused and vested in outcome
- Is a fearless champion

Backup

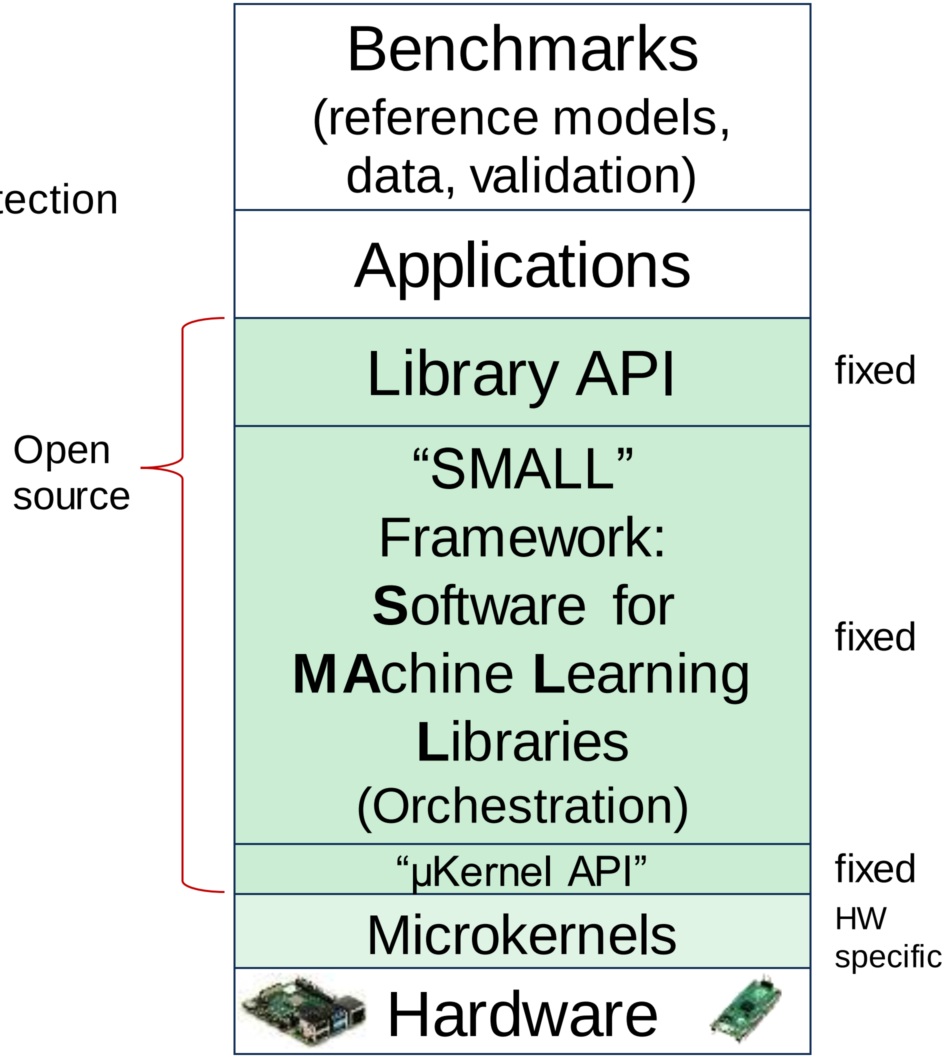
PHITE Software Architecture

Benchmark and application selection:

- **Open-source**: TinyMLPerf* benchmark applications:
  - image classification, visual wake words, audio wake words, object detection
- **CUI / ITAR**: <Your applications and data here>
  - E.g., target identification, anomaly detection.

Library development (open source): a BLIS†-like layered architecture

- **Top-level API**: Tensorflow 2 (as much as possible)
- **Middleware**: SMALL framework (under development)
- **Low-level**:
  - A microkernel API for small set of key machine learning primitives
  - Hardware specific codes, including assembly where necessary



- https://github.com/mlcommons/tiny
  https://mlcommons.org/en/inference-tiny-05/

† "BLIS: A Framework for Rapidly Instantiating BLAS Functionality", (TOMS) ACM Transactions on Mathematical Software, vol. 41, no. 3, June 2015, pp 1-33.

Team

Dr. Scott McMillan, CMU/SEI/AI, **PI**

Advanced computing, parallel and distributed algorithms, graph analytics, data-intensive computing, interactive data science at scale.

Prof. Tze Meng Low, CMU/ECE, **PI**

High-performance algorithms using formal methods and analytical models; performance portability through capture of interaction between software algorithms and hardware features; code generation and libraries for emerging domains.

Oren Wright, CMU/SEI/AI, CMU/ECE PhD

Applied AI/ML; graph signal processing; moving beyond ad hoc neural networks; all things Bayesian; "how to make AI fail gracefully…not catastrophically."

Upasana Sridhar, CMU/ECE PhD

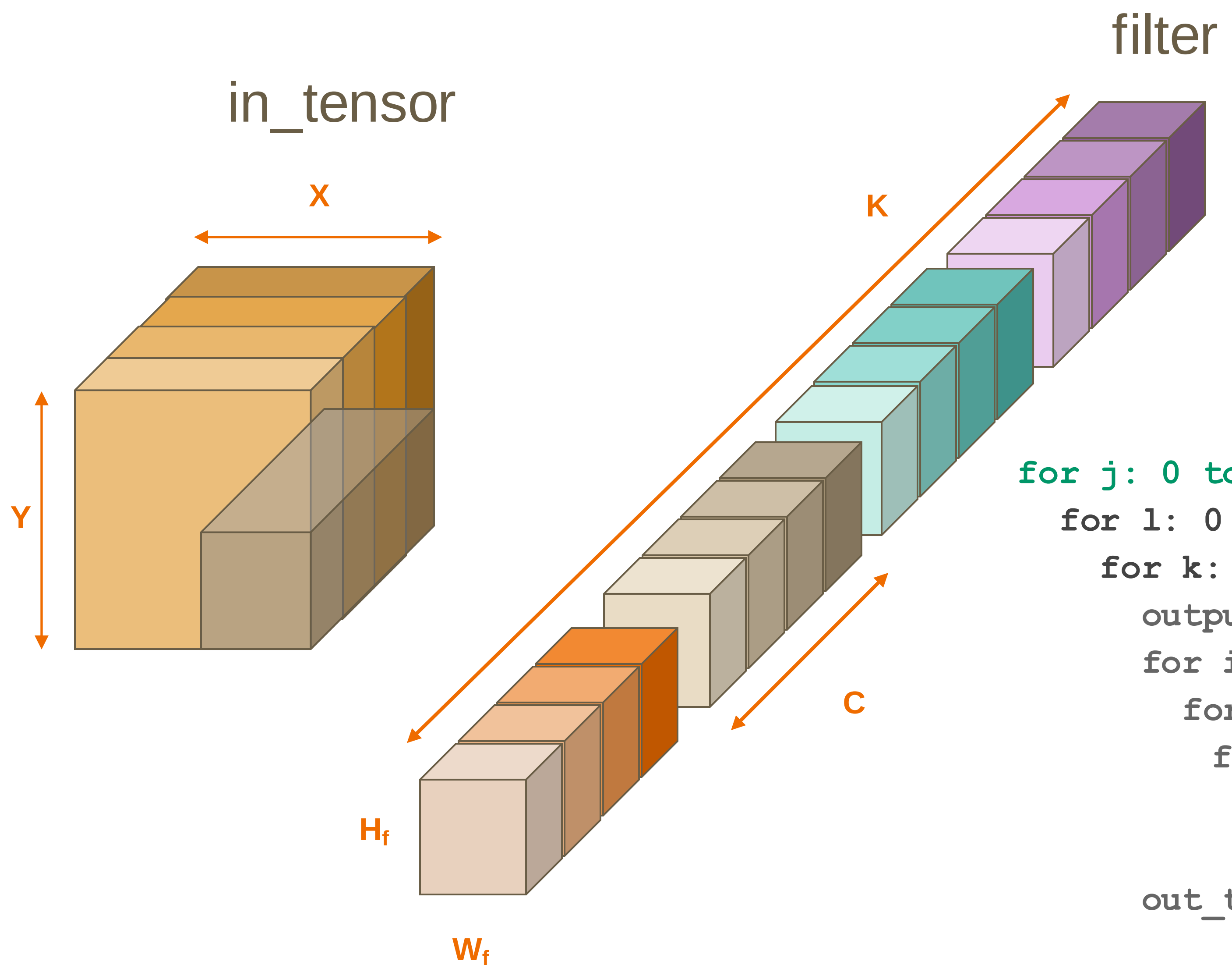
Analytical modelling for graph algorithms and machine learning inference; performance modelling, formal methods.

Nicolai Tukanov, CMU/ECE PhD

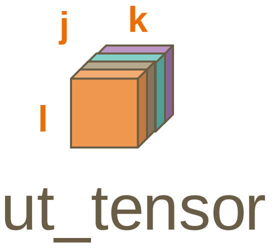
High-performance computing, performance modeling, hardware accelerators.

- Navya Chandra – ECE master's independent study: "Fused convolution on Pi Pico"
- Pankti Rajesh Shah – master's student started in Q2FY22

Convolution Operation - End to End



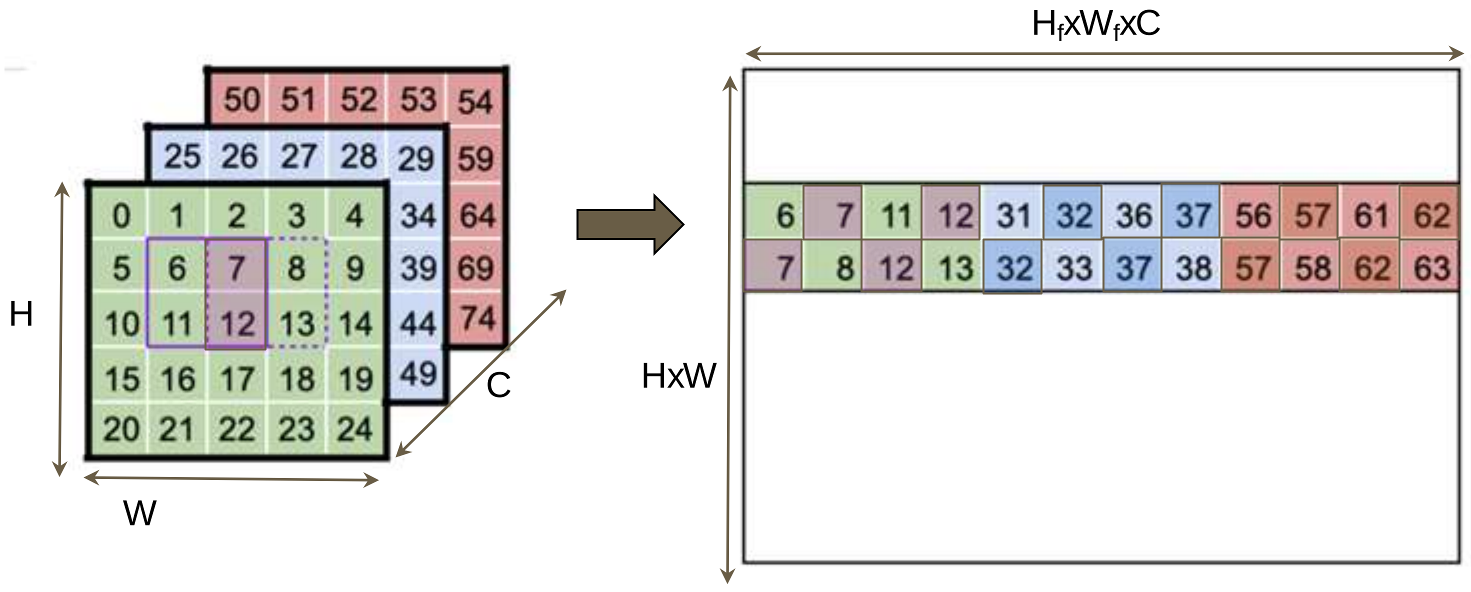

```
for j: 0 to K
  for l: 0 to Y
    for k: 0 to X
      output = 0
      for i: 0 to C
        for n: 0 to Hf
          for m: 0 to Wf
            output += (in_tensor[i][l * s + n][k * s + m]
                       * filter[j][i][n][m])
      out_tensor[j][l][k] = output
```



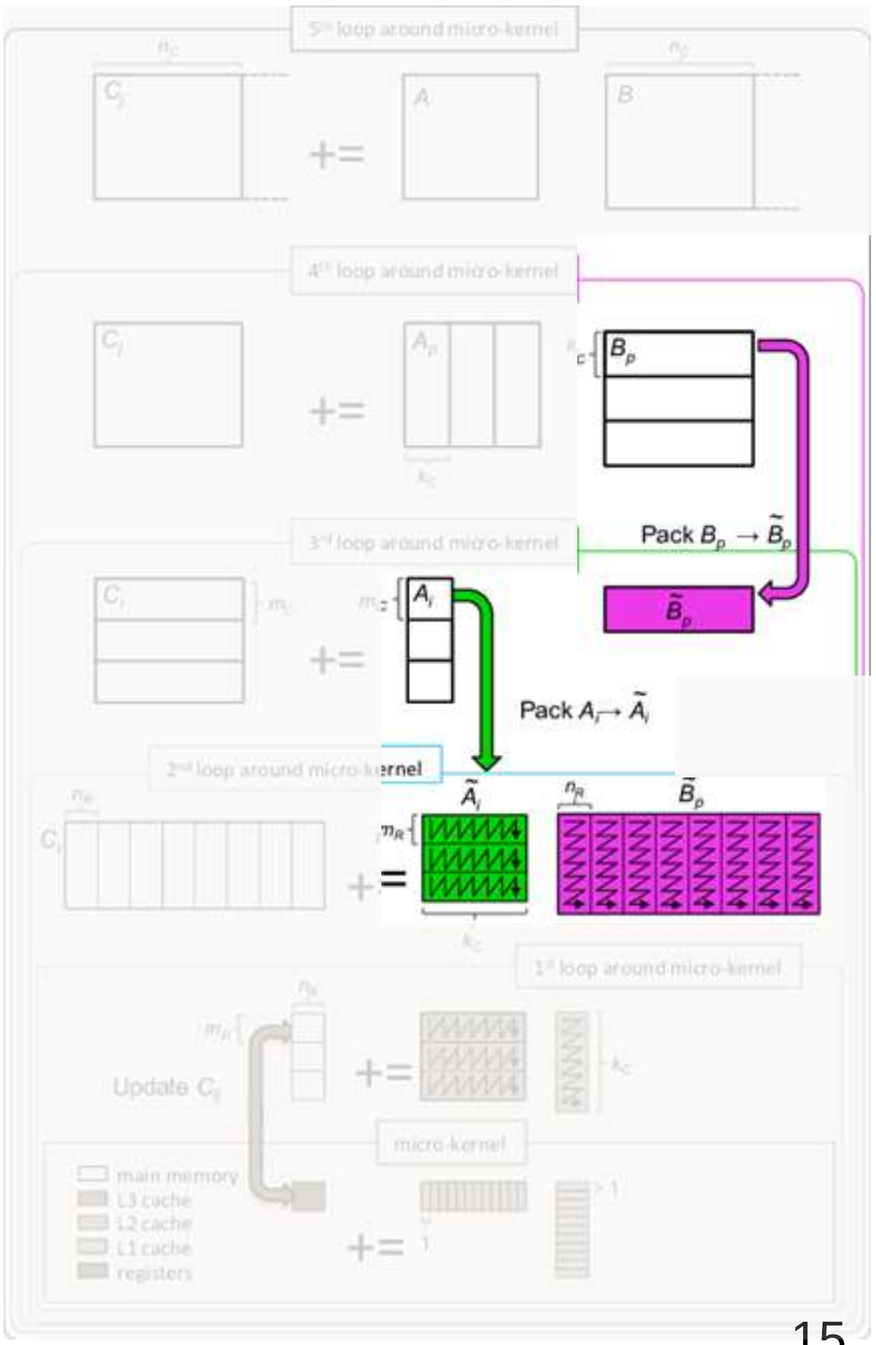
Convolutions as GEMM

- GEMM: BLAS dense **GE**neral **M**atrix-**M**atrix multiply
- Packing requires additional memory

GEMM packing

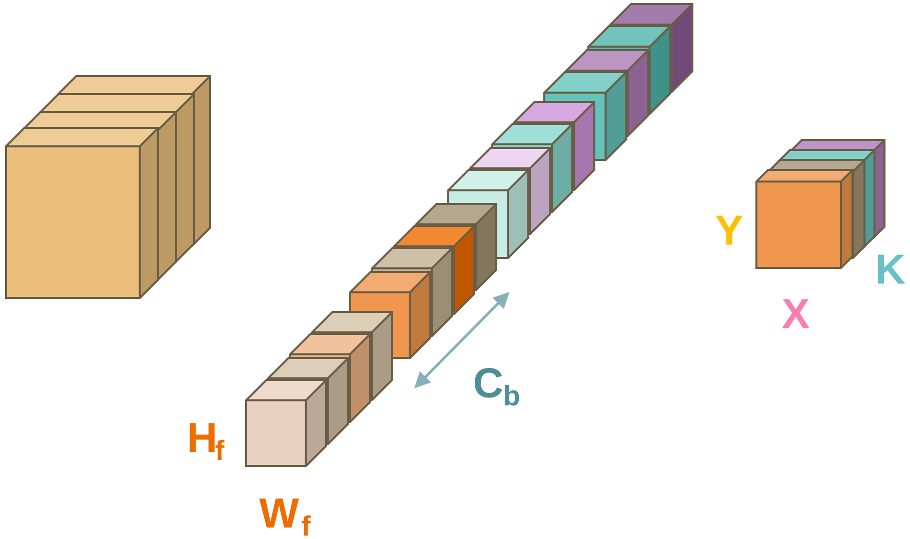


Data Duplication



Direct Convolution Loop Structure

- Loops follow Data Layout




```
for block in K; step Kb
  for i in C; step Cb
    for row in X
      for pixel_block in Y; step Wb
        for n in Hf
          for m in Wf
            for ii in Cb
              for pixel in pixel_block,Wb
                for channel in Cb
                  update(I, F, O)
```

Next step: fusing convolution layers

- Submitted to 2022 ACM International Conference on Supercomputing (ICS).
- In progress: Open-source software release on context dependent fusion.

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-elementwise 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.

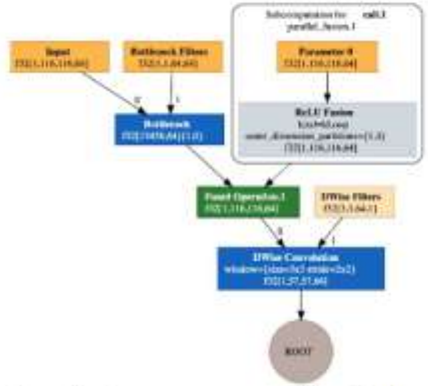
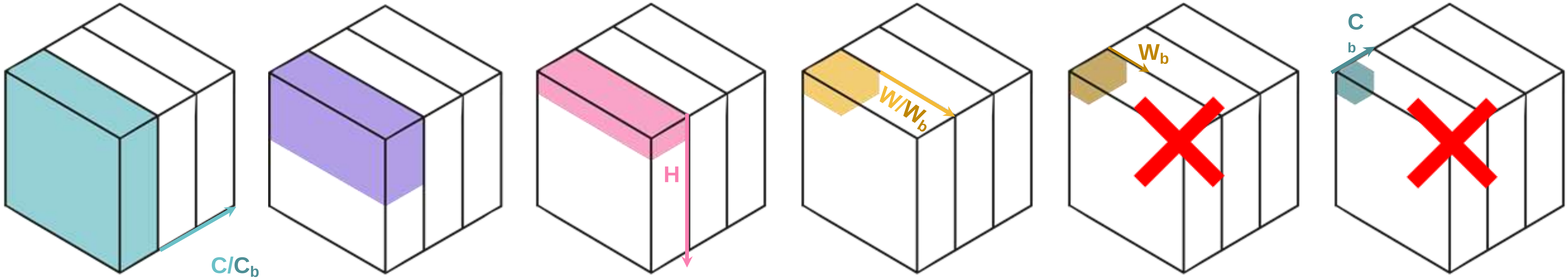


Figure 1: Post-fusion compute graph for Bottleneck (1 × 1) Convolution + ReLu Activation + Depthwise Convolution block using XLA demonstrating that fusion in XLA is limited. The Relu layer has been fused with the Bottleneck convo-

Fusing Loop Nests

- Fusion may be possible at different levels



Decreasing (Working Set Size) WSS

Fuse each block

Fuse each row pool

Fuse each row

Fuse each pixel block

Fuse each pixel

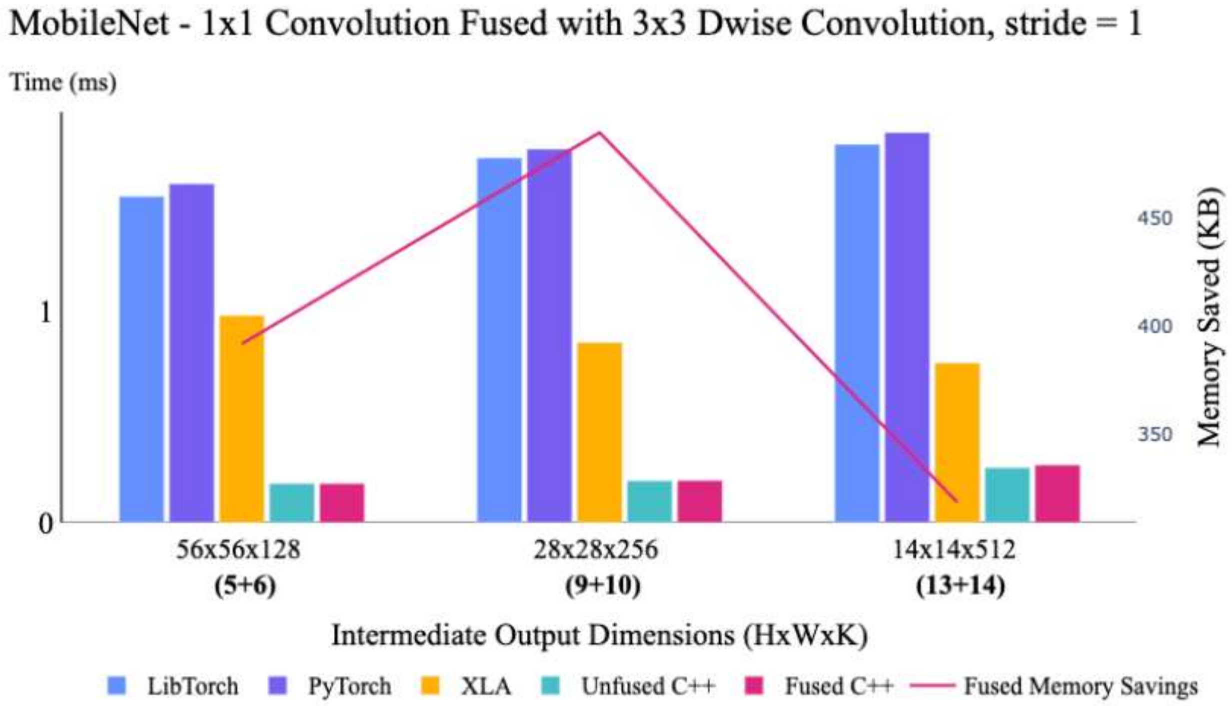
Fuse each channel

Highest Level (fuse outermost loop)

Lowest Level (fuse up to innermost loop)

Fused Loop Structure

- LOTS of knobs to tune: F_C, O_h, O_w, F_h, F_w, I_{cb}, O_{wb}, etc.



Algorithm 2 Fused abstract deep learning layer loop structure

```
for g ≤ G do
    for j ≤ K do
        for i ≤ F_C do                              ▷ Fusion in Last iteration
            for k ≤ O_h do                          ▷ Transformation Req for Fusion
                for l ≤ O_w do
                    for x ≤ F_h do
                        for y ≤ F_w do
                            for ii ≤ I_cb do
                                for ll ≤ O_wb do
                                    for jj ≤ O_cb do
                                        ReductionOp(O_0, I, F_0)
                                    end for
                                end for
                                                    ▷ Single Element Reduction
                                for ll ≤ O_wb do
                                    for jj ≤ O_cb do
                                        ReductionOp(O, O_0, F_1)
                                    end for
                                end for
                            end for
                        end for
                    end for
                                                    ▷ Channel Reduction
                    for x ≤ F^1_h do
                        for y ≤ F^1_w do
                            ii = 1
                            for ll ≤ O_wb do
                                for jj ≤ O_cb do
                                    ReductionOp(O, O_0, F_1)
                                end for
                            end for
                        end for
                    end for
                end for
            end for
        end for
    end for
end for=0
```