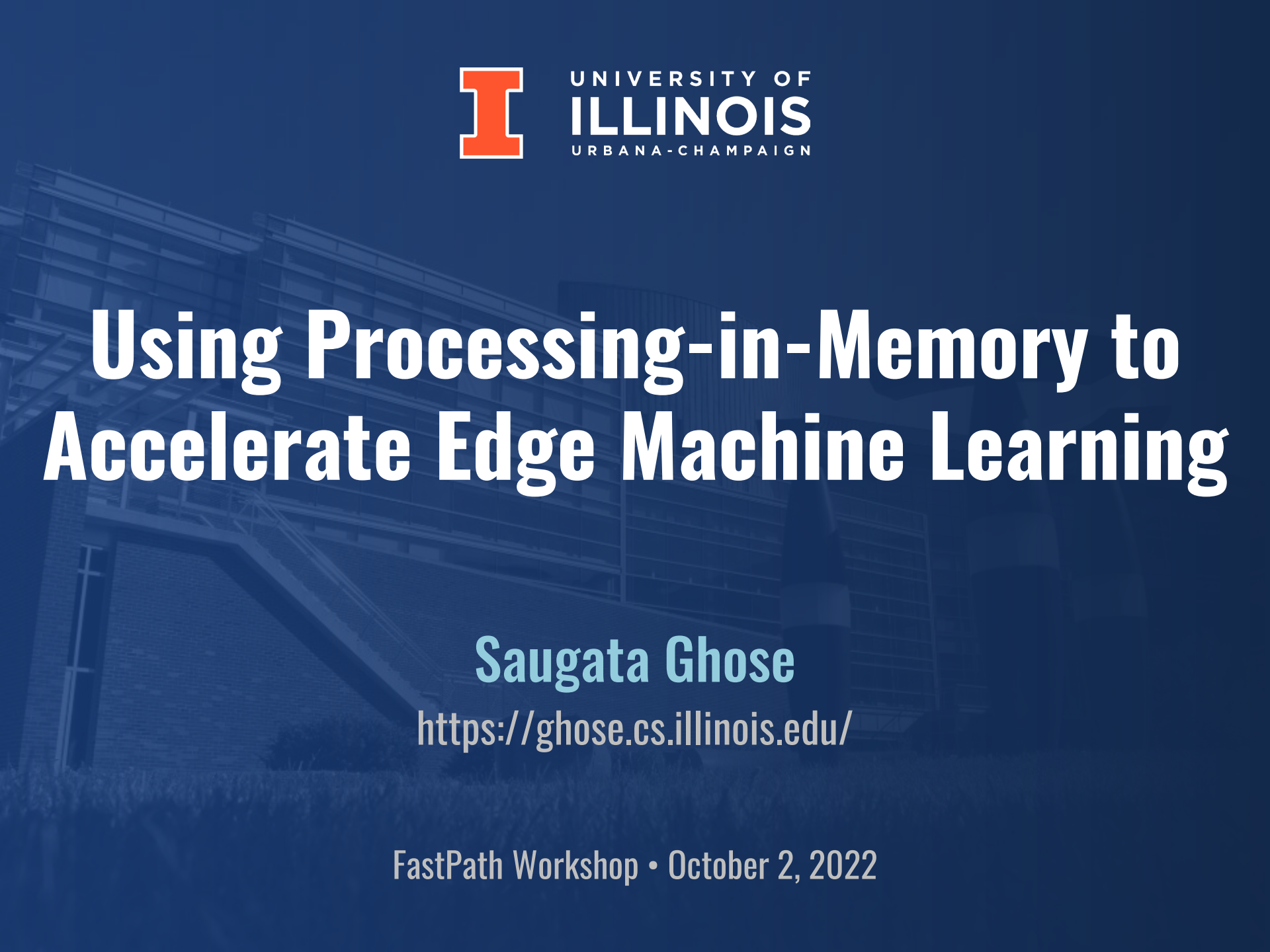


UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Using Processing-in-Memory to Accelerate Edge Machine Learning

Saugata Ghose
https://ghose.cs.illinois.edu/

FastPath Workshop • October 2, 2022

Executive Summary

- **Inference on edge devices is stressing accelerator capabilities**
  - We think of neural network (NN) models as computationally-intensive
  - Edge NN model footprints **exceeding the limited storage of accelerators**
  - We show this with a detailed characterization of **Google edge NN models**
- **Processing-in-memory can come to the rescue!**
  - New memory capabilities can overcome memory channel bottlenecks
  - Variants: **processing-near-memory** (PNM), **processing-using-memory** (PUM)
- **Mensa-G: heterogeneous edge NN accelerators using PNM**
  - **3.1x performance, 3.0x energy improvement** vs. Google Edge TPU
  - Full paper: Boroumand+ PACT 2021
- **RACER: data accelerator for edge computing using PUM**
  - **107x performance, 189x energy improvement** vs. 16-core Intel Xeon
  - Full papers: Truong+ MICRO 2021, Truong+ JETCAS 2022

Why Do Machine Learning (ML) on Edge Devices?

Significant interest in pushing ML inference computation directly to edge devices

Privacy

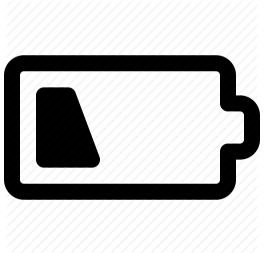
Connectivity

Latency

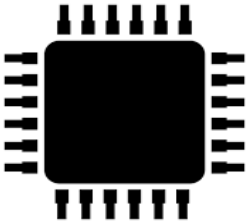
Bandwidth

Why Build Specialized ML Accelerators?

Edge devices have limited battery and computation budgets

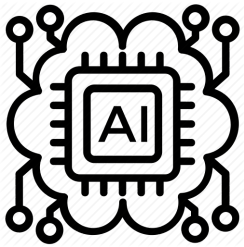


Limited Power Budget



Limited Computational Resources

Specialized accelerators can significantly improve inference latency and energy consumption



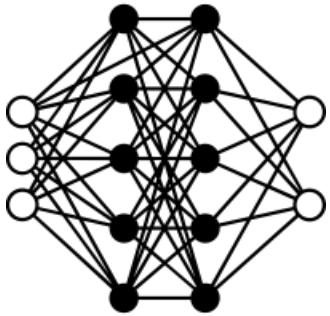
Apple Neural Engine (A12)

Google Edge TPU

Myriad of Edge Neural Network Models

Speech Recognition

RNN Transducers



LSTMs

Language Translation

Face Detection

CNN

RCNN

Image Captioning

Challenge: edge ML accelerators have to execute inference efficiently across a wide variety of NN models

Introduction

Characterizing Edge NN Models

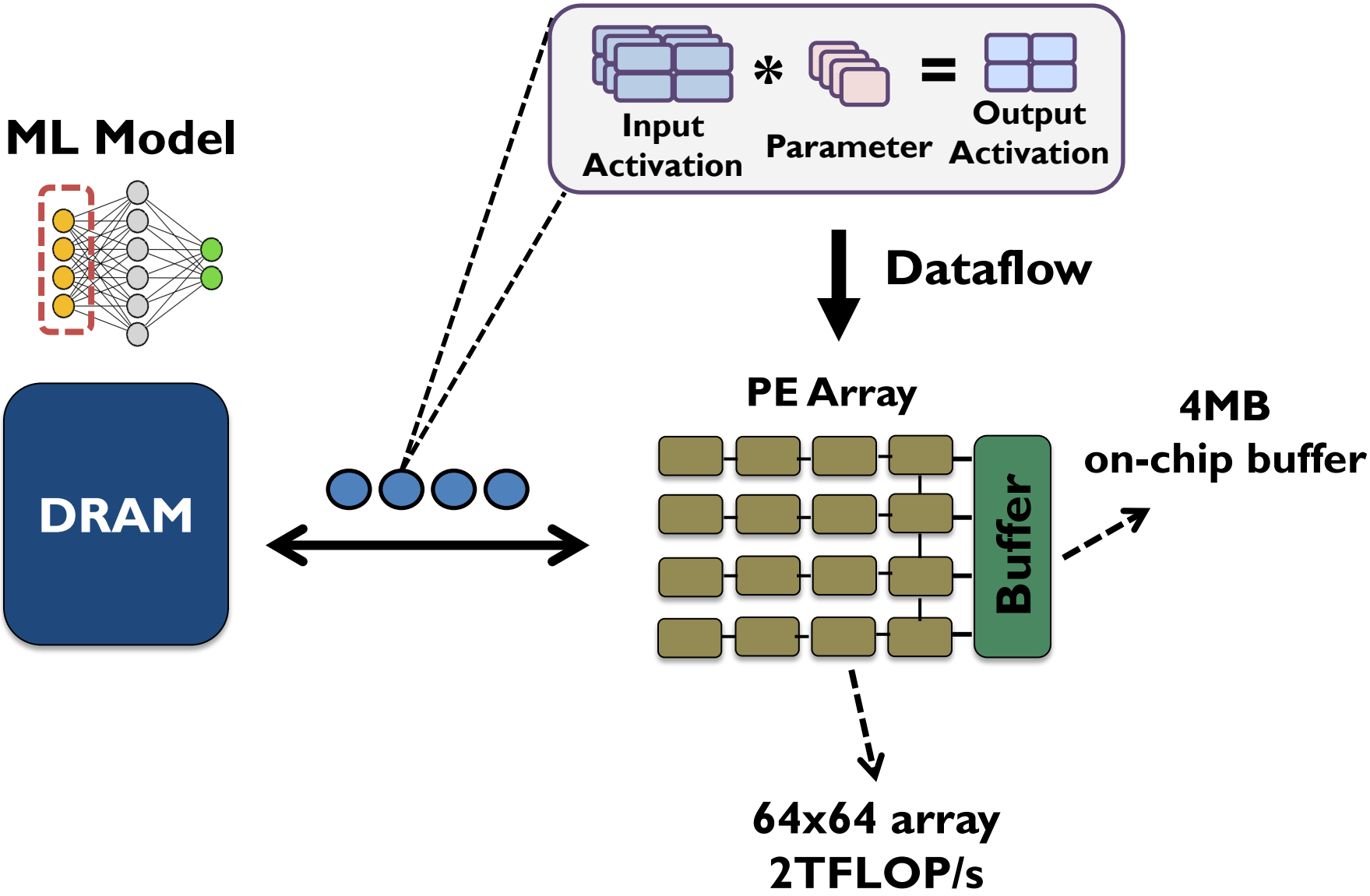
Alleviating Data Costs with Processing-in-Memory

Mensa-G: Heterogeneous NN Acceleration with PNM

RACER: Edge Data Acceleration with PUM

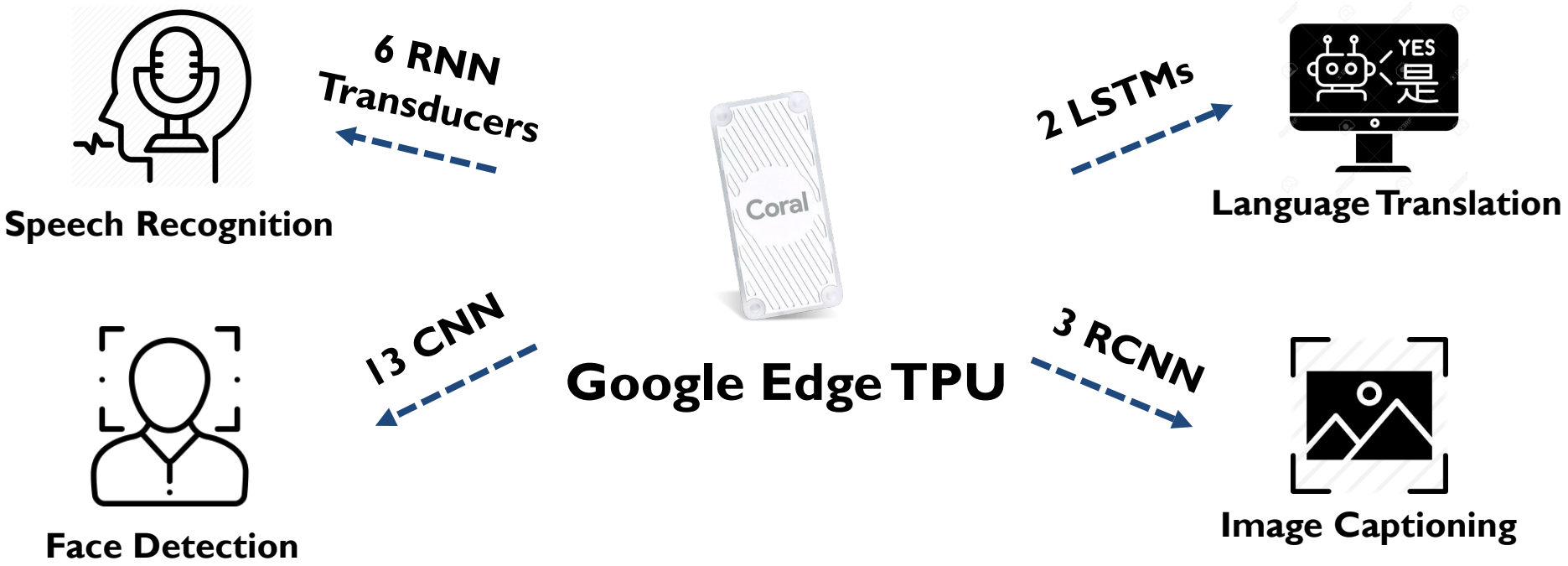
Closing Thoughts

Edge TPU: Baseline Accelerator



Google Edge NN Models

- We analyze inference execution using 24 edge NN models



The Edge TPU Suffers From Three Major Challenges

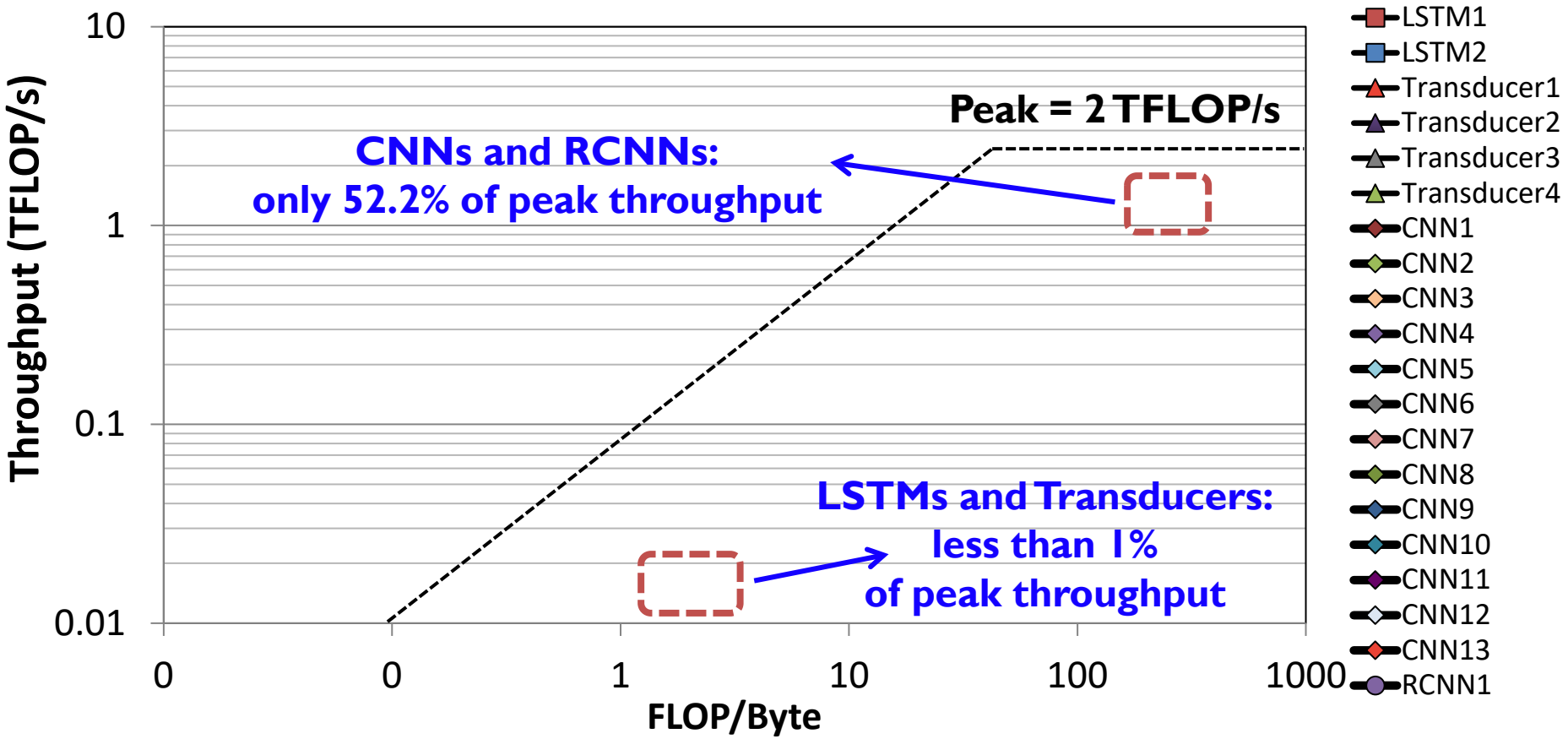
1. It operates significantly below its peak **throughput**

2. It operates significantly below its peak **energy efficiency**

3. It handles **memory accesses** inefficiently

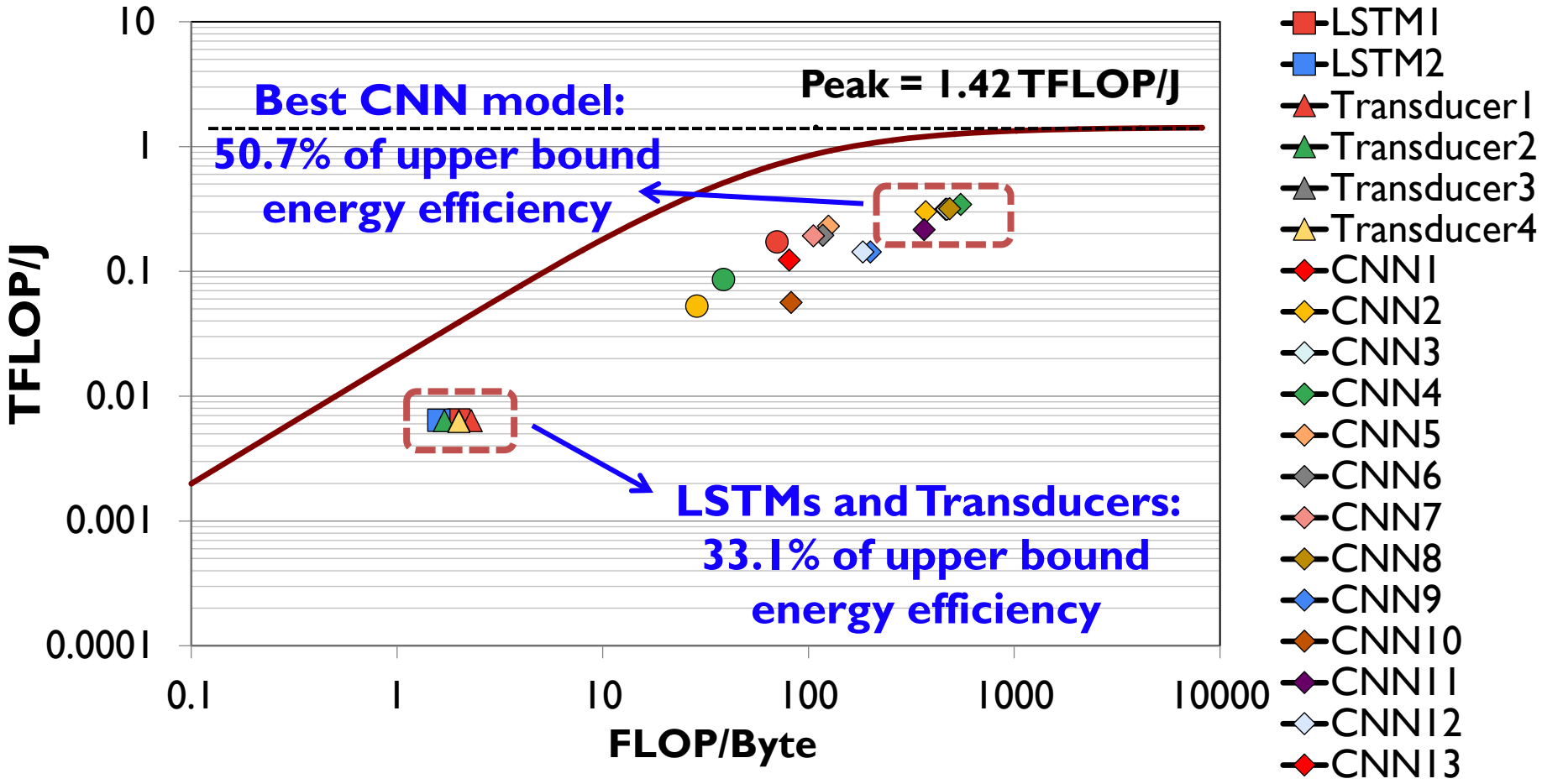
Challenge 1: High Resource Underutilization

We find that the accelerator operates significantly below its peak throughput across all models



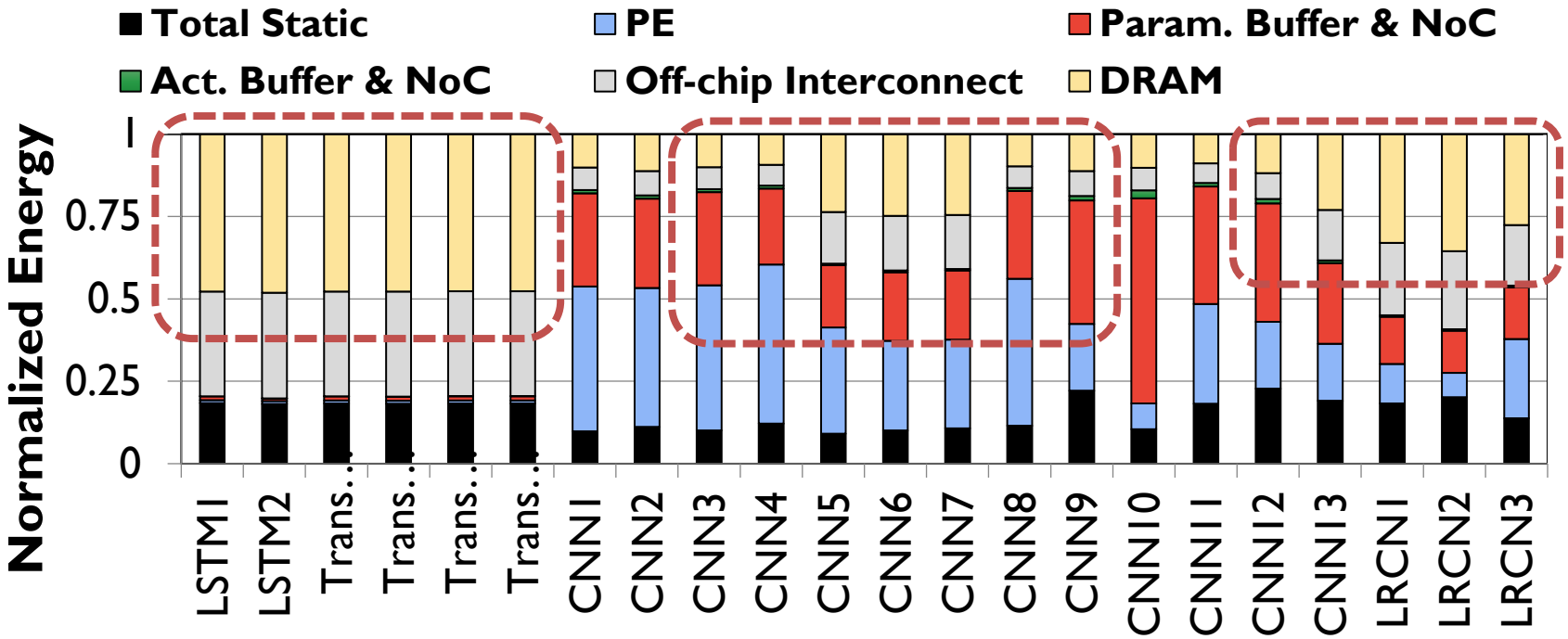
Challenge 2: Low Energy Efficiency

The accelerator operates far below its upper bound energy efficiency



Challenge 3: Inefficient Memory Access Handling

Parameter traffic (off-chip and on-chip) takes a large portion of the inference energy and execution time



46% and 31% of total energy goes to off-chip parameter traffic and distributing parameters across the PE arrays

The Edge TPU Suffers From Three Major Challenges

1. It operates significantly below its peak **throughput**

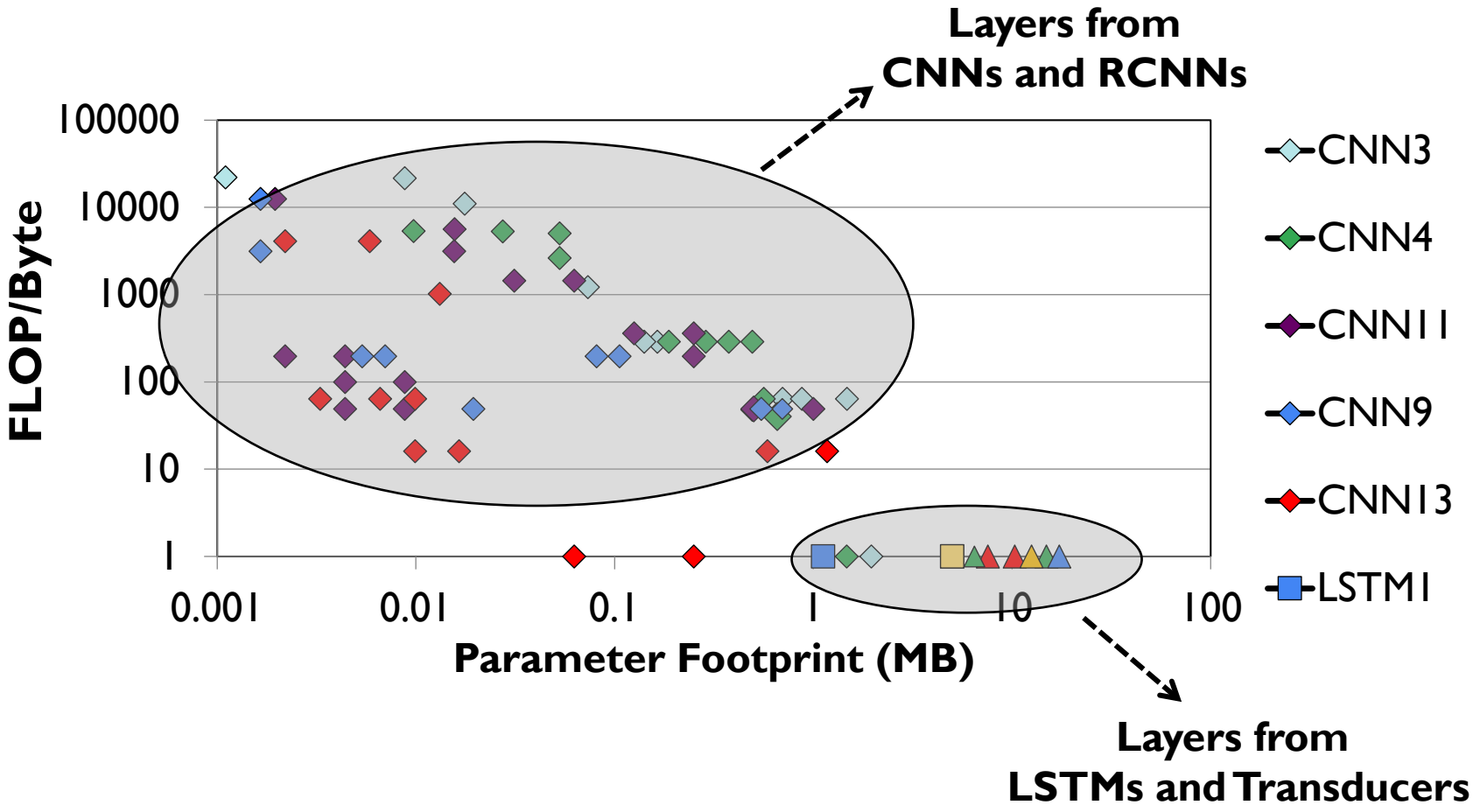
2. It operates significantly below its peak **energy efficiency**

3. It handles **memory accesses** inefficiently

Where do these challenges come from?

Diversity Across the Models

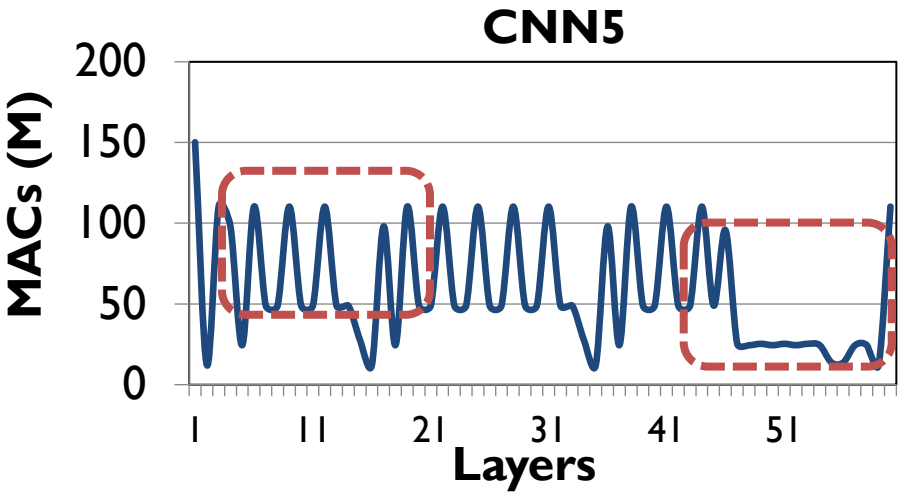
Insight 1: there is significant variation in terms of layer characteristics across the models

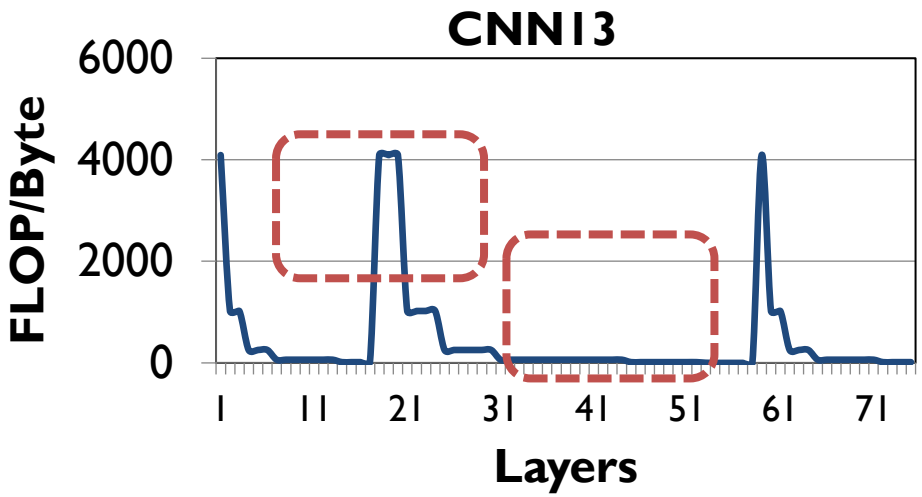


Diversity Within the Models

Insight 2: even within each model, layers exhibit significant variation in terms of layer characteristics

For example, our analysis of edge CNN models shows:



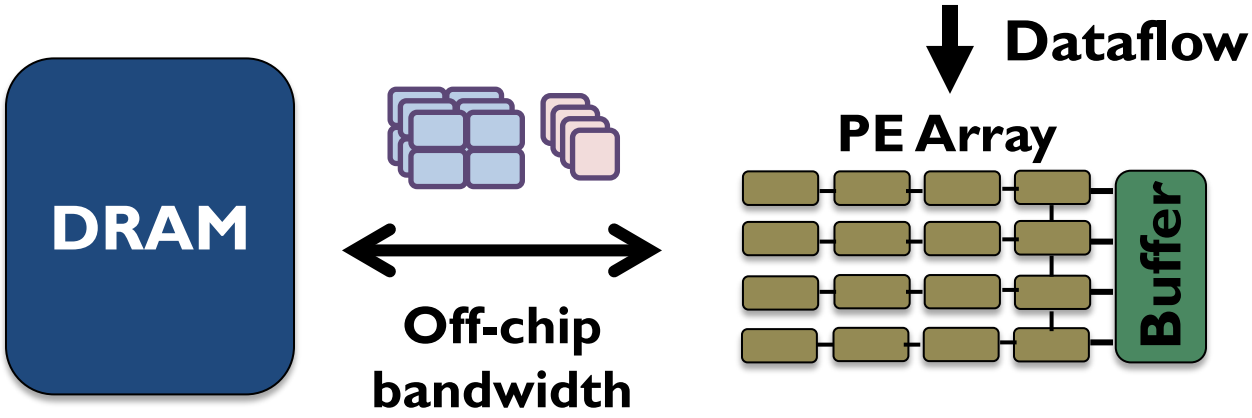


Variation in MAC intensity: up to 200x across layers

Variation in FLOP/Byte: up to 244x across layers

Root Cause of Accelerator Challenges

The key components of Google Edge TPU are completely oblivious to layer heterogeneity



Edge accelerators typically take a monolithic approach: equip the accelerator with an over-provisioned PE array and on-chip buffer, a rigid dataflow, and fixed off-chip bandwidth

While this approach might work for a specific group of layers, it fails to efficiently execute inference across a wide variety of edge models

Introduction

Characterizing Edge NN Models

Alleviating Data Costs with Processing-in-Memory

Mensa-G: Heterogeneous NN Acceleration with PNM

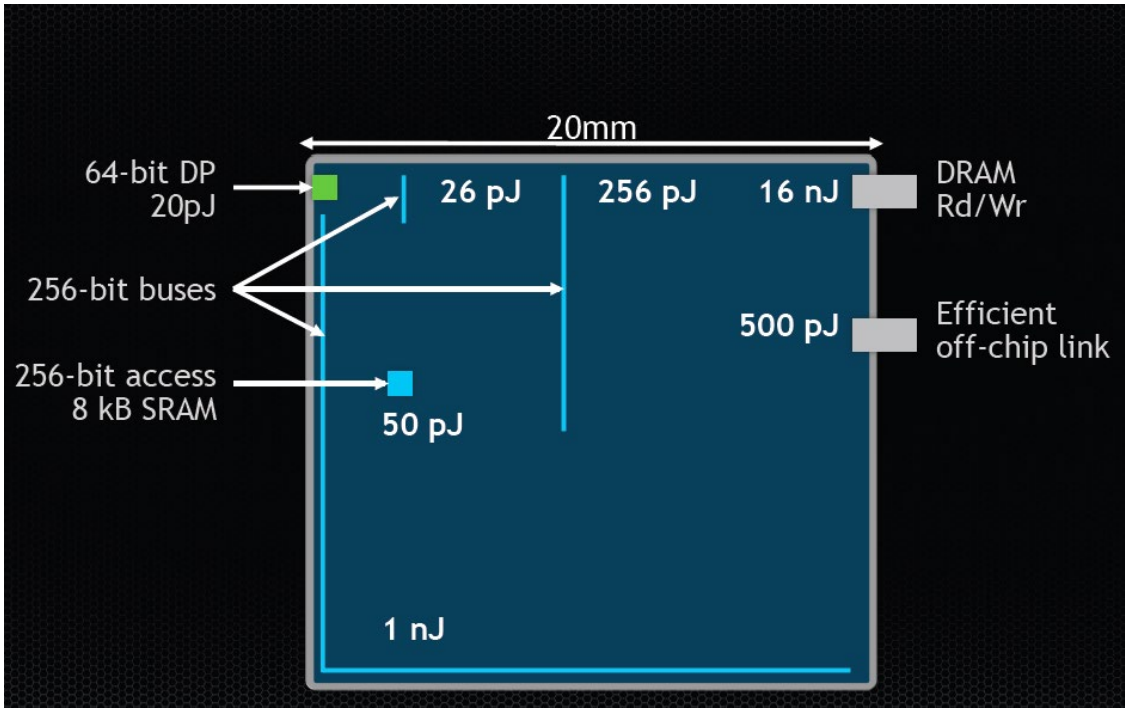
RACER: Edge Data Acceleration with PUM

Closing Thoughts

The Cost of Data Movement in Modern CPUs

- **In terms of energy costs, data movement dominates computation**
  - 10–50x the energy to move data on-chip than to do a floating-point compute
  - Almost 1000x to DRAM!
- **Data movement is a major bottleneck in modern systems**
  - High energy spent on off-chip communication
  - Pin-limited bandwidth
  - High latency
  - Identified as the **von Neumann bottleneck** by John Backus in 1977

Source: Dally, keynote at HiPEAC 2015



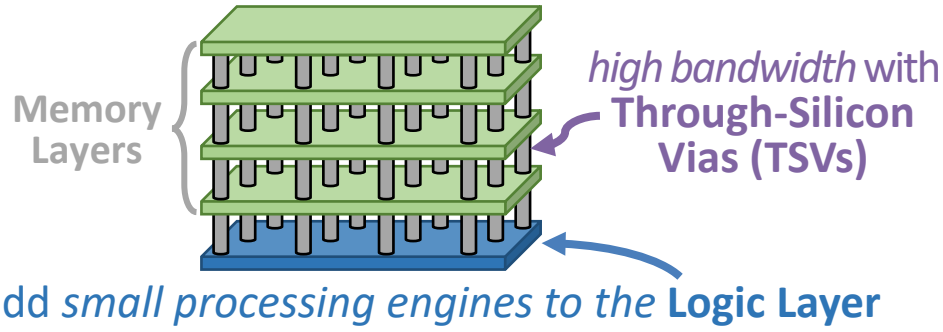
Can We Avoid Moving Data Around?

- **Processing-in-memory (PIM), a.k.a. near-data processing (NDP)**
  - Add some compute capability to memory
  - No need to move data across memory channel

- **PIM is not a new concept**
  - First works proposed as early as 1970 (Harold Stone's Logic-in-Memory Computer)
  - Lots of work in the 1990s, but became dormant shortly after
  - New memory design innovations are providing new opportunities for PIM

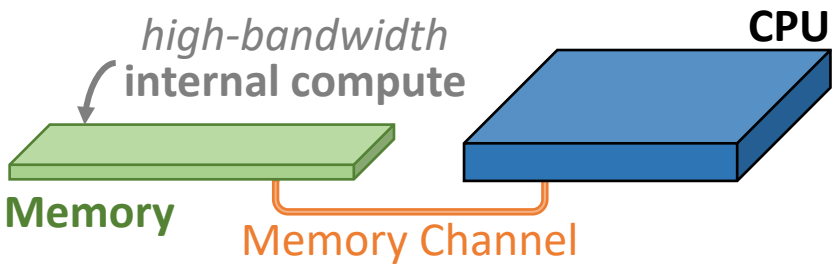
- **We'll look at two variants of PIM today**

PIM Variant 1: Processing-Near-Memory (PNM)



- **Let's take advantage of the high internal bandwidth of 3D-stacked memory**
  - Chips can have as many as 2K TSVs (essentially vertical wires) internally
  - Memory channel has only 64 wires
- **Discrete logic in either**
  - Dedicated logic layer inside a 3D-stacked chip
  - Separate die connected with high-bandwidth interface to the memory (e.g., Si interposer, Intel EMIB)

PIM Variant 2: Processing-Using-Memory (PUM)



- **Take advantage of memory technologies that can perform logic**
  - Many **non-volatile memory** cells can use analog properties to perform bitwise operations
  - Can be done in traditional memory technologies (e.g., SRAM, DRAM) with additional logic
- **Little to no additional logic**
  - Takes advantage of inherent circuit-level properties of memory
  - Bandwidth, potential dictated by memory structure (e.g., crosspoint memory arrays – a 2D grid of memory cells)

Introduction

Characterizing Edge NN Models

Alleviating Data Costs with Processing-in-Memory

Mensa-G: Heterogeneous NN Acceleration with PNM

RACER: Edge Data Acceleration with PUM

Closing Thoughts

Mensa Framework

Goal:

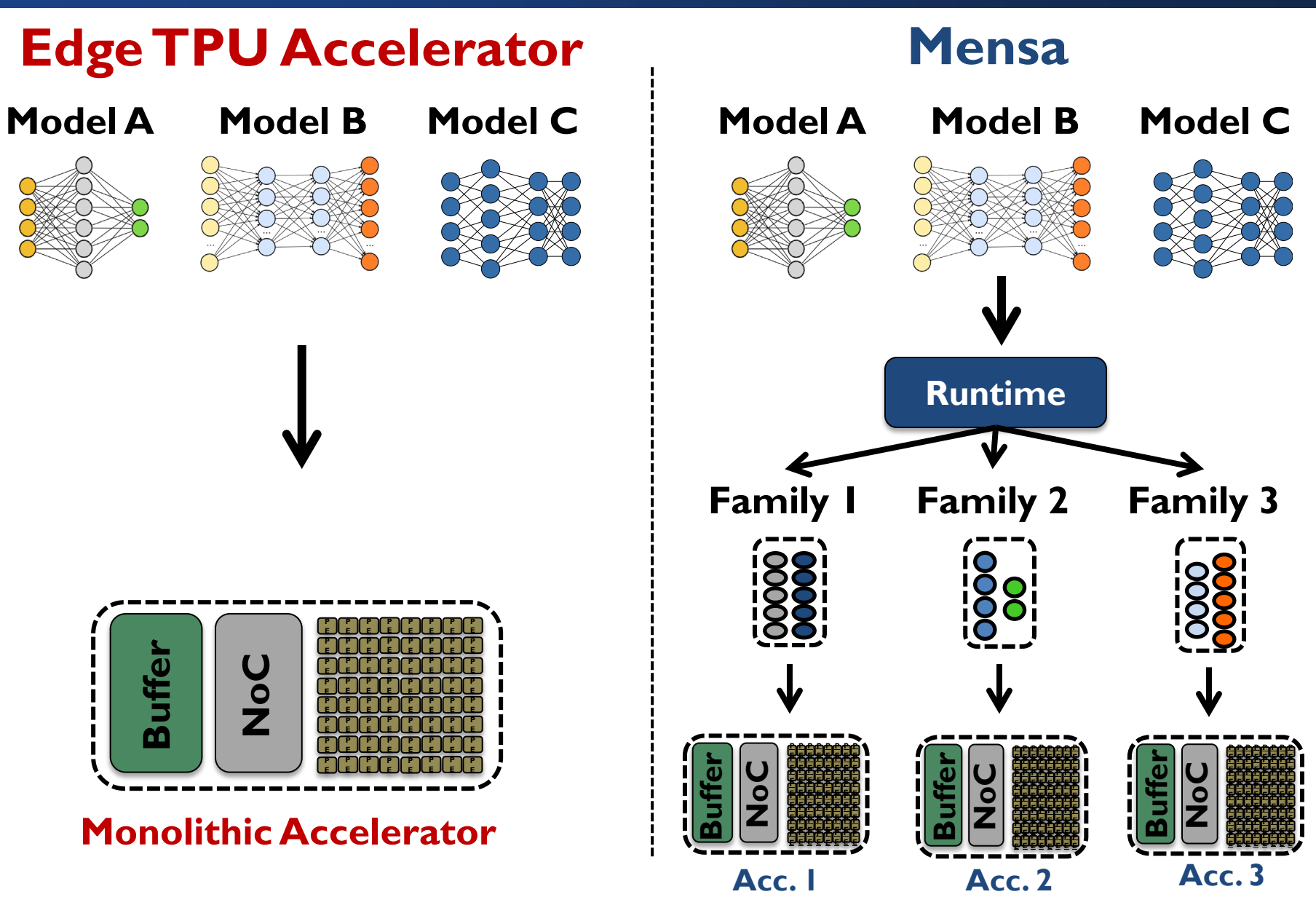
Design an edge accelerator that can efficiently run inference across a wide range of different models and layers

Instead of running the entire NN model on a monolithic accelerator:

↓

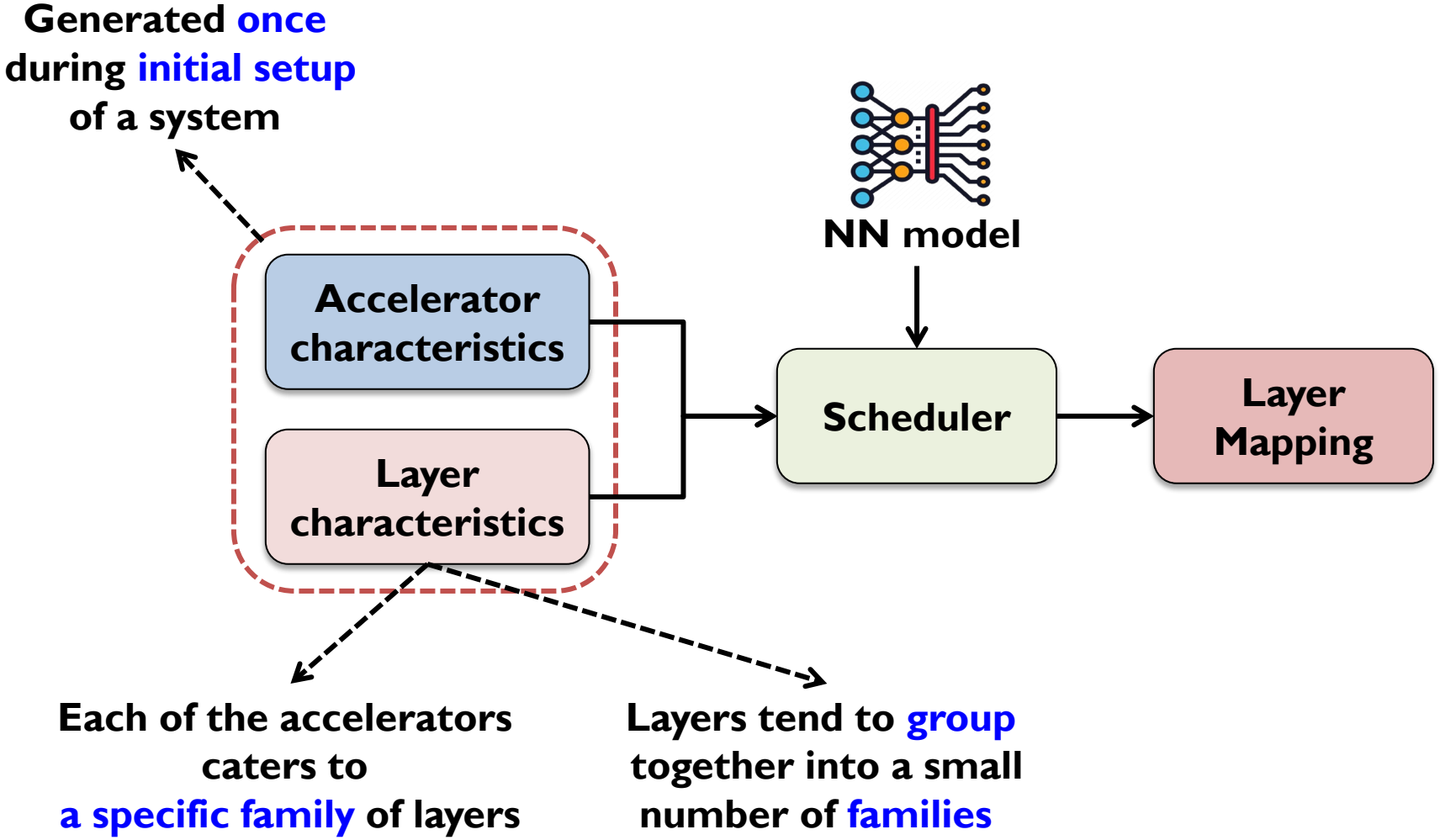
Mensa: a new acceleration framework for edge NN inference

Mensa High-Level Overview



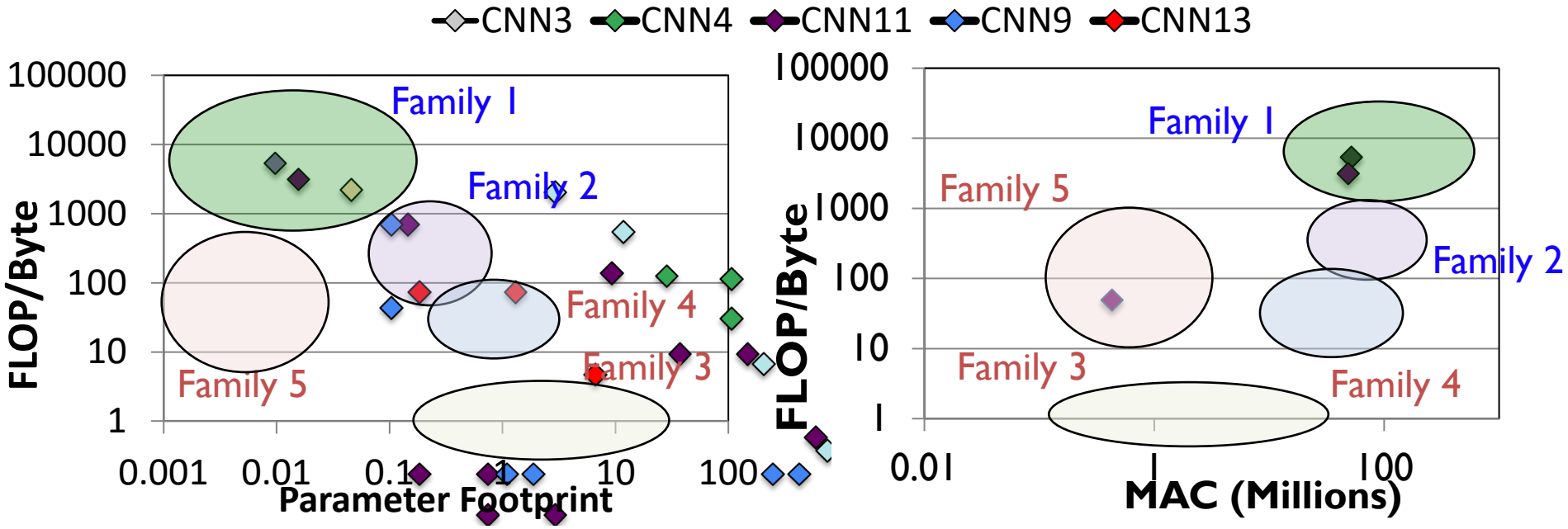
Mensa Runtime Scheduler

Software runtime scheduler identifies *which accelerator* each layer in an NN model should run on



Identifying Layer Families

Key observation: the majority of layers group into a small number of layer families

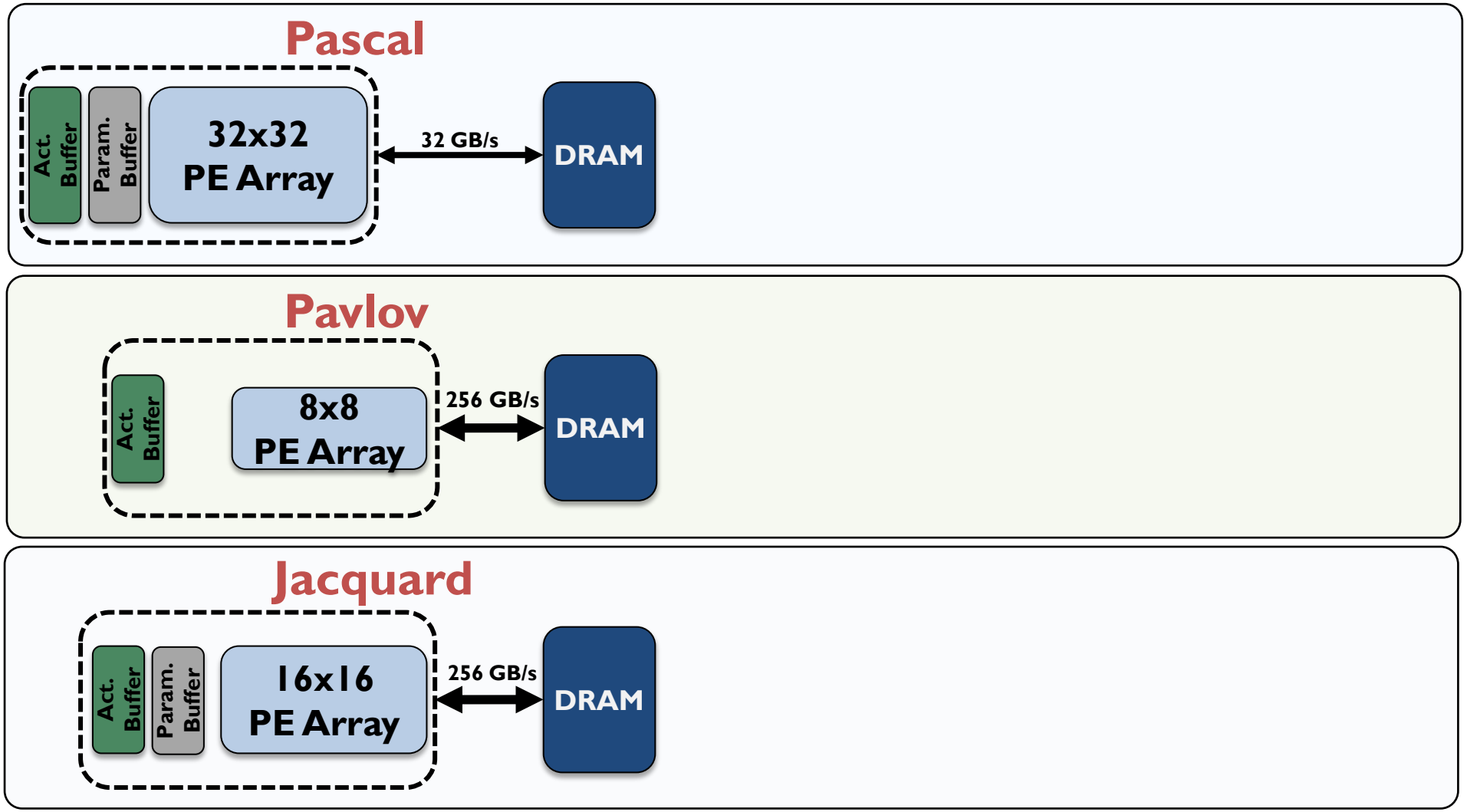


Families 1 & 2: low parameter footprint, high data reuse and MAC intensity
→ compute-centric layers

Families 3, 4 & 5: high parameter footprint, low data reuse & MAC intensity
→ data-centric layers

Mensa-G: Mensa for Google Edge Models

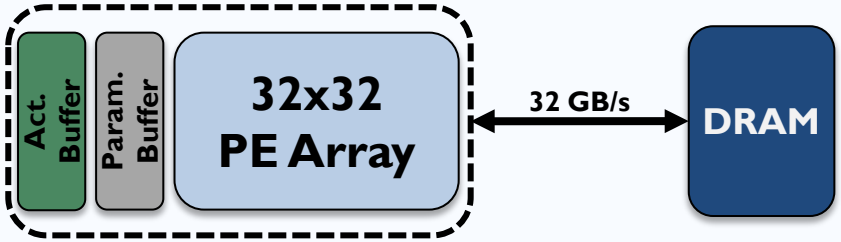
Based on key characteristics of families, we design three accelerators to efficiently execute inference across our Google NN models



Mensa-G: Mensa for Google Edge Models

Based on key characteristics of families, we design three accelerators to efficiently execute inference across our Google NN models

Pascal



Families 1&2 → compute-centric layers
- 32x32 PE Array → 2 TFLOP/s
- 256KB Act. Buffer → 8x Reduction
- 128KB Param. Buffer → 32x Reduction
- On-chip accelerator

Pavlov

Jacquard

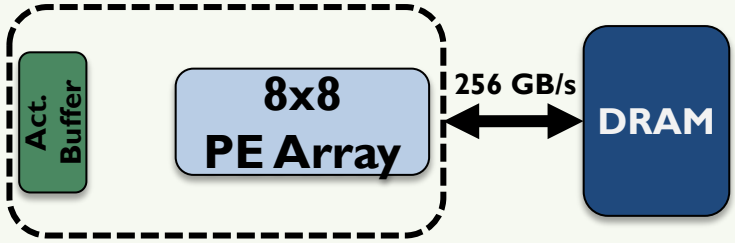
Mensa-G: Mensa for Google Edge Models

Based on key characteristics of families, we design three accelerators to efficiently execute inference across our Google NN models

Pascal

Families 1&2 → compute-centric layers
- 32x32 PE Array → 2 TFLOP/s
- 256KB Act. Buffer → 8x Reduction
- 128KB Param. Buffer → 32x Reduction
- On-chip accelerator

Pavlov



Family 3 → LSTM data-centric layers
- 8x8 PE Array → 128 GFLOP/s
- 128KB Act. Buffer → 16x Reduction
- No Param. Buffer → 4MB in Baseline
- PNM accelerator

Jacquard

Mensa-G: Mensa for Google Edge Models

Based on key characteristics of families, we design three accelerators to efficiently execute inference across our Google NN models

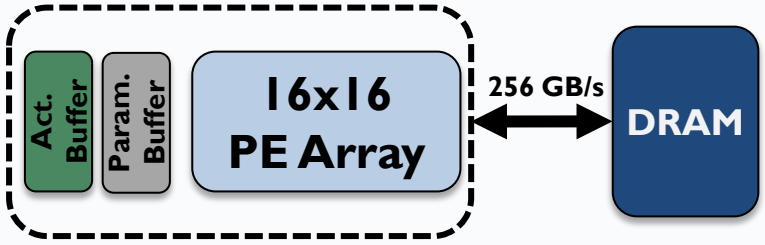
Pascal

Families 1&2 → compute-centric layers
- 32x32 PE Array → 2 TFLOP/s
- 256KB Act. Buffer → 8x Reduction
- 128KB Param. Buffer → 32x Reduction
- On-chip accelerator

Pavlov

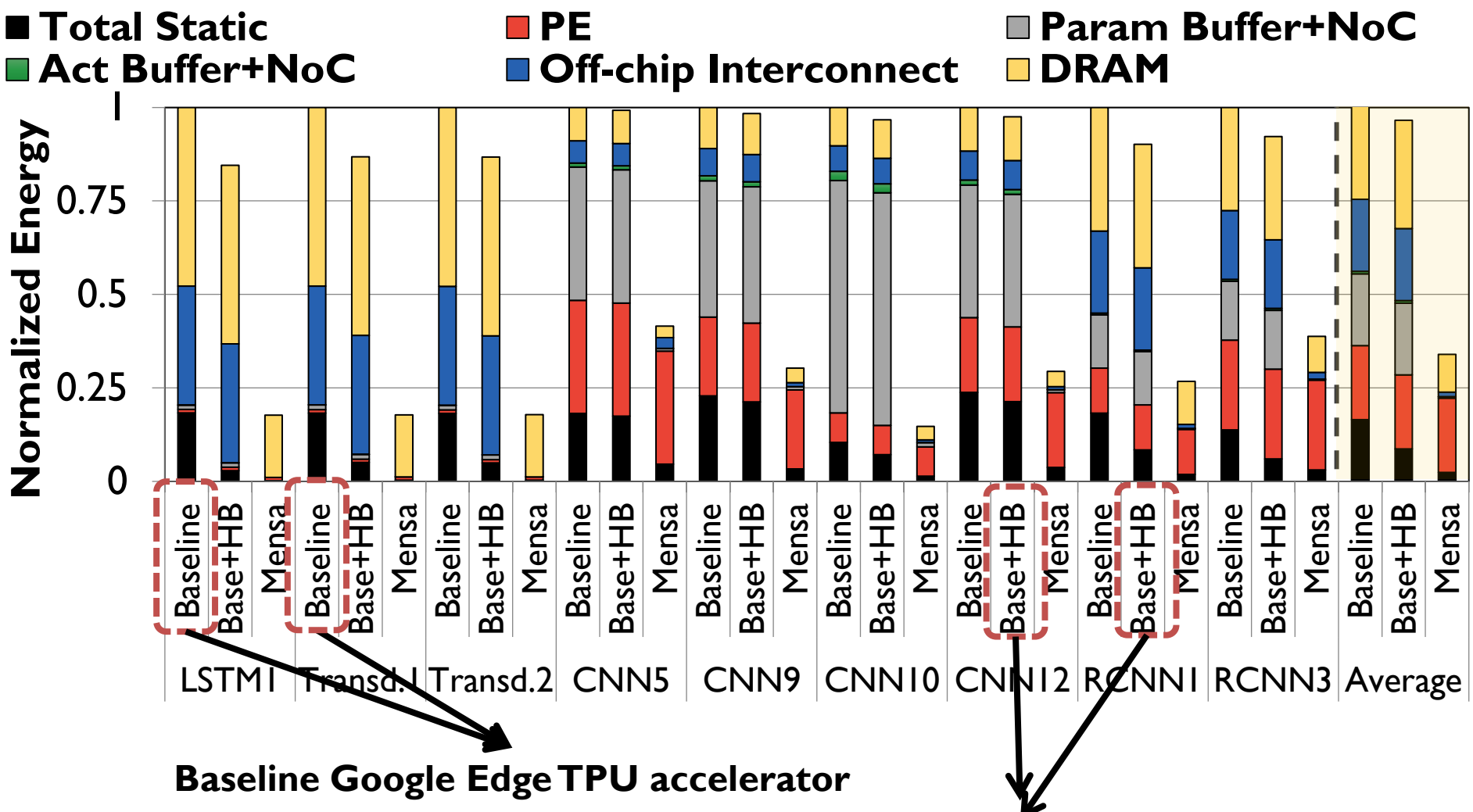
Family 3 → LSTM data-centric layers
- 8x8 PE Array → 128 GFLOP/s
- 128KB Act. Buffer → 16x Reduction
- No Param. Buffer → 4MB in Baseline
- PNM accelerator

Jacquard

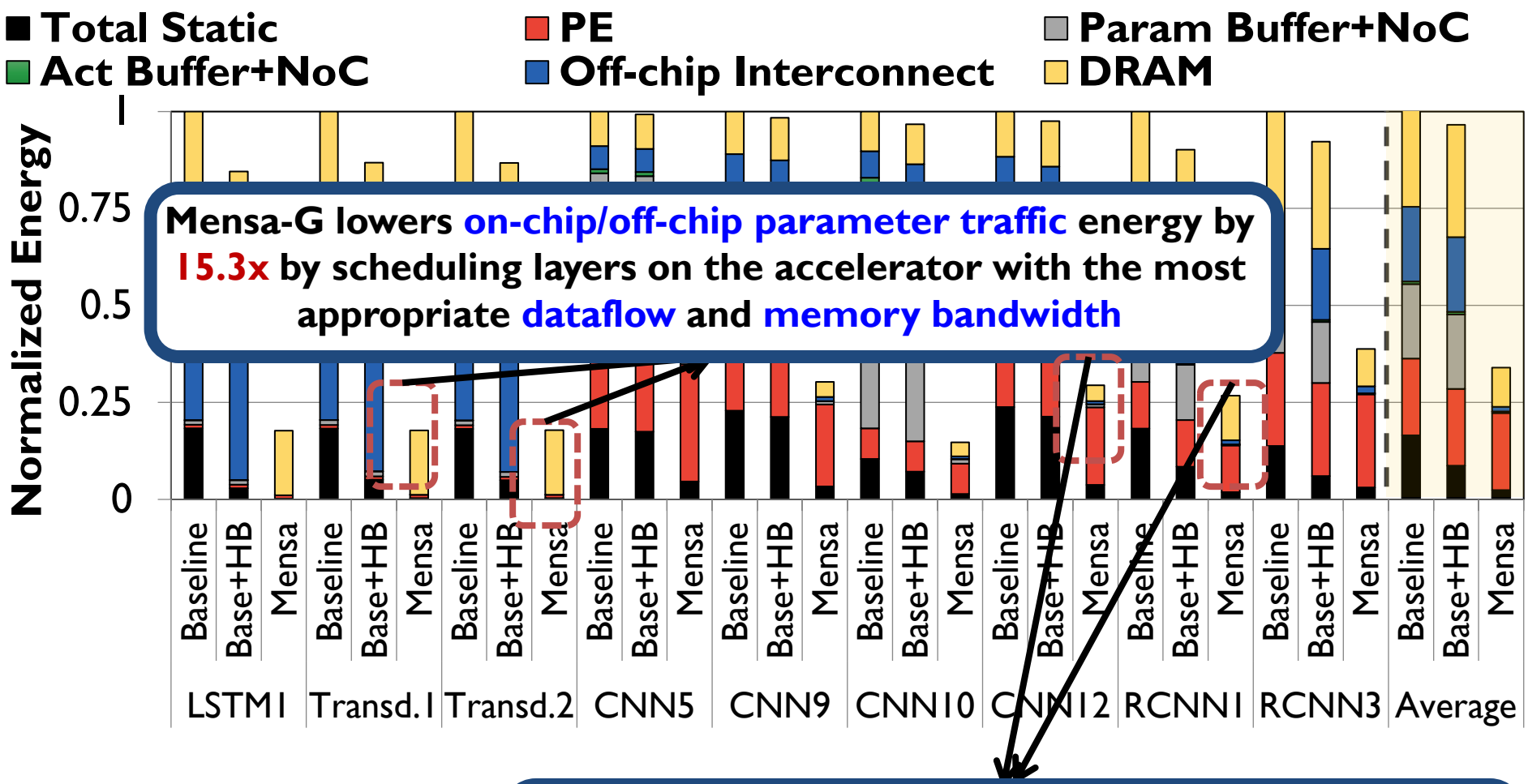


Families 4&5 → non-LSTM data-centric layers
- 16x16 PE Array → 256 GFLOP/s
- 128KB Act. Buffer → 16x Reduction
- 128KB Param. Buffer → 32x Reduction
- PNM accelerator

Energy Analysis

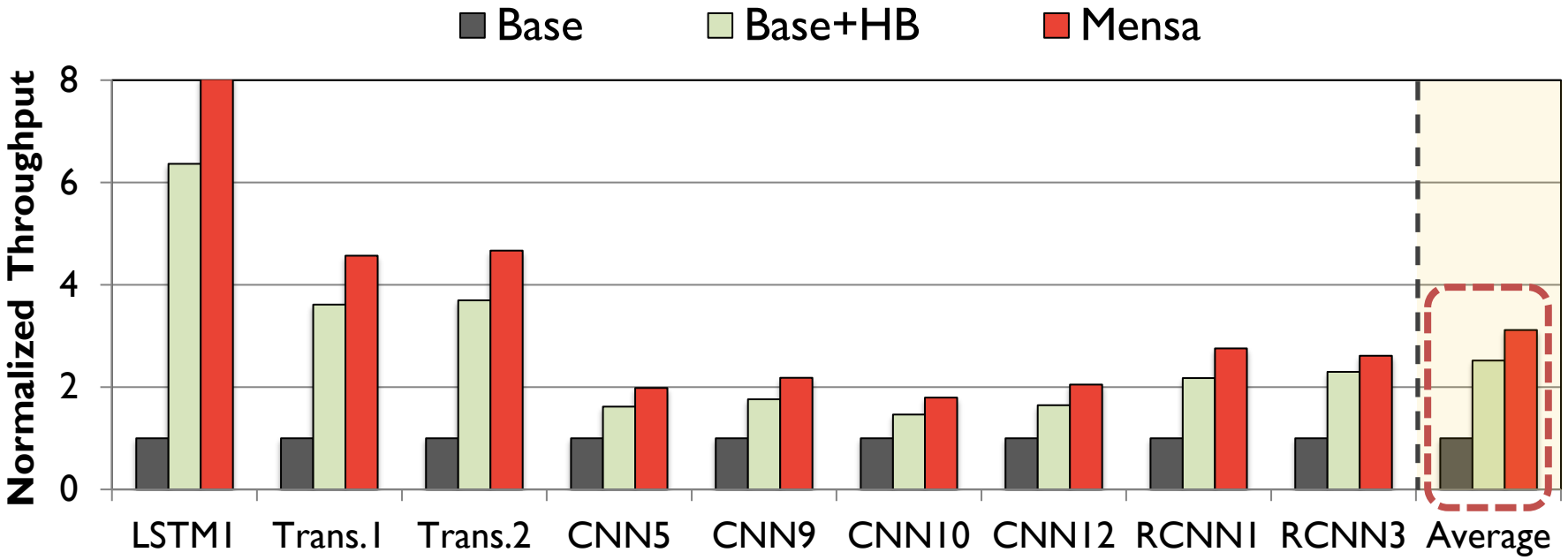


Energy Analysis



Mensa-G improves energy efficiency by 3.0x compared to the baseline Edge TPU

Throughput Analysis



Mensa-G improves throughput by 3.1x compared to the baseline Edge TPU

More in Our PACT 2021 Paper

Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand[†◇] Saugata Ghose[‡] Berkin Akin[§] Ravi Narayanaswami[§]
Geraldo F. Oliveira[⋆] Xiaoyu Ma[§] Eric Shiu[§] Onur Mutlu[⋆†]

Carnegie Mellon | University of Illinois Urbana-Champaign | Google | ETH zürich

- Details about the Mensa Runtime Scheduler
- Details about Pascal, Pavlov, and Jacquard's dataflows
- Energy comparison with Eyeriss v2
- Mensa-G's utilization results
- Mensa-G's inference latency results

Introduction

Characterizing Edge NN Models

Alleviating Data Costs with Processing-in-Memory

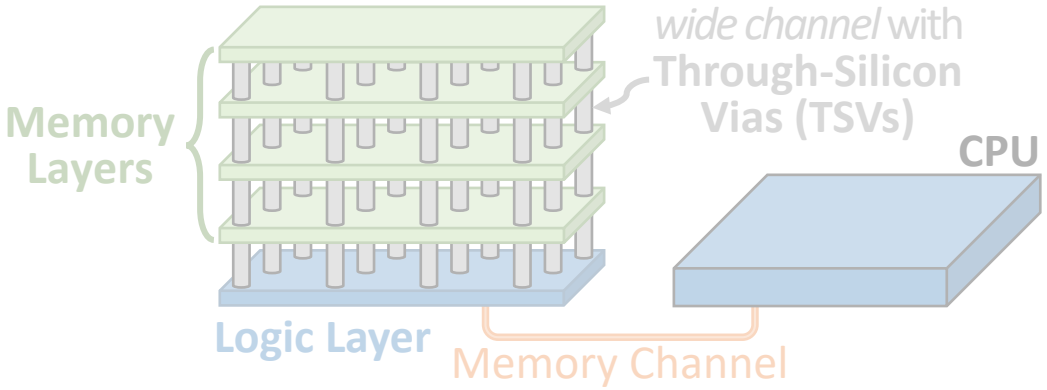
Mensa-G: Heterogeneous NN Acceleration with PNM

RACER: Edge Data Acceleration with PUM

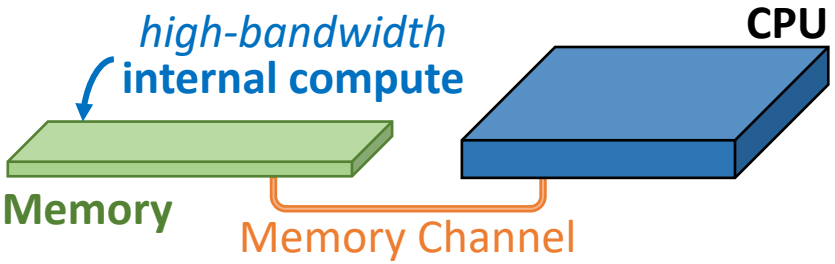
Closing Thoughts

Recall: Two Variants of PIM

- **Processing Near Memory (PNM)**
  - **Discrete logic** in or near the memory chip
  - Commercial examples
    - » UPMEM PIM–DRAM Big Data Accelerator
    - » Samsung Aquabolt-XL (a.k.a. HBM-PIM, AxDIMM)
    - » SK hynix GDDR6-AiM

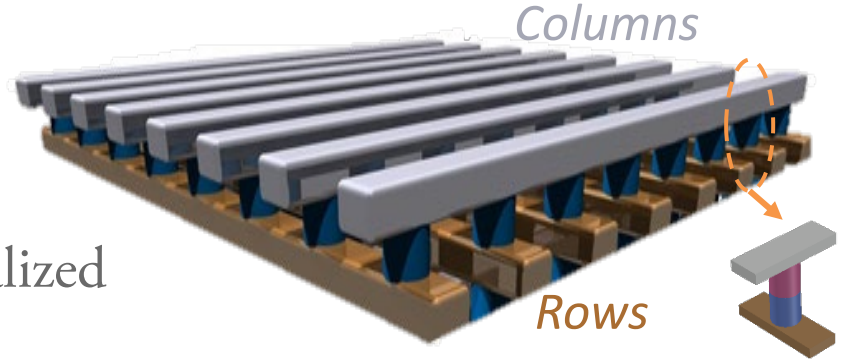


- **Processing Using Memory (PUM)**
  - **Electrical interactions between memory cells** w/ little additional logic
  - Commercial examples
    - » Mythic Analog Matrix Processor
    - » IBM Hermes

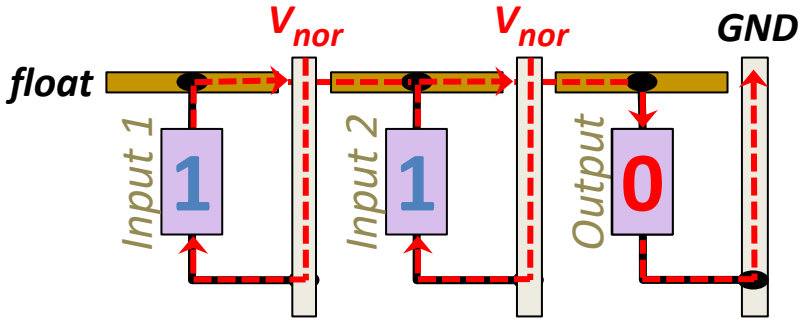


Exploiting Interactions Between Resistive RAM Cells

- **Data stored as a resistance**
  - 1S1R: a **selector device** in series with a **resistive memory switch**
  - Resistive switch programmed using localized thermal events
  - Typically organized as a **crosspoint array**

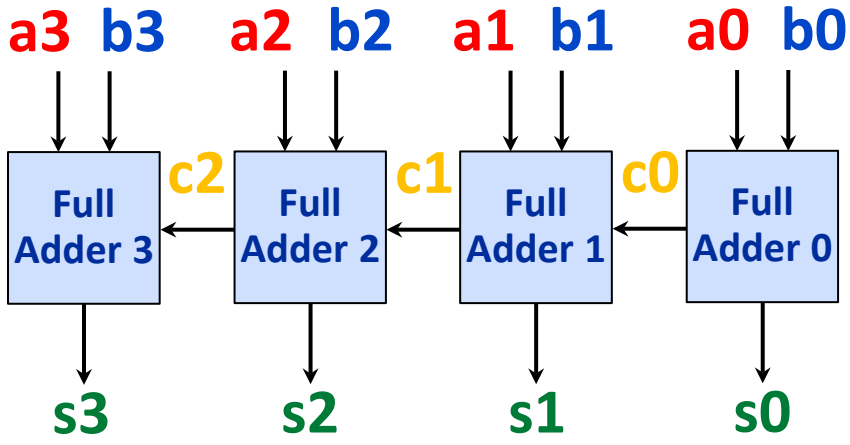


- **Example: turning on multiple ReRAM cells at once can perform several types of meaningful computation**
  - Multi-level ReRAM cells can perform dot product, low-precision multiply
  - **Single-level ReRAM cells can perform NOR**, NAND, OR, IMP



Digital PUM Architectures Enable Bit-Serial Compute

- **Bit-serial operations**
  - Perform operations one bit at a time
  - Example: ripple-carry add



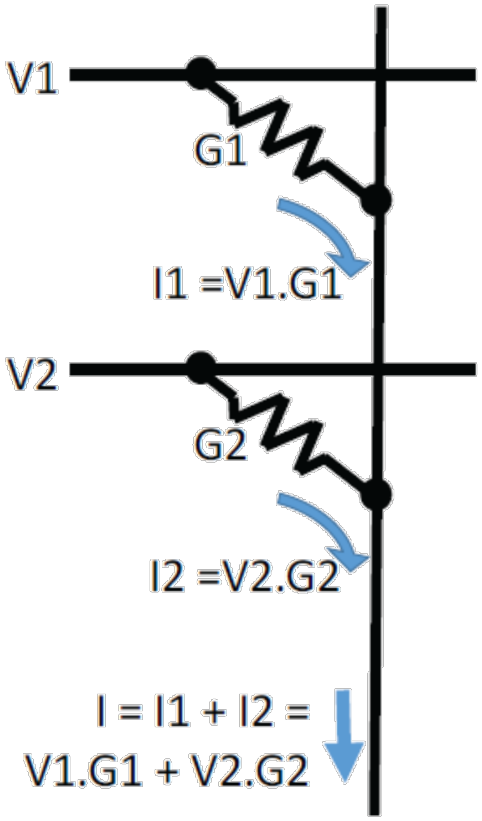
- **PUM architectures can perform many bit-serial functions**
  - Addition/subtraction
  - Content search (like a content-addressable memory, or CAM)

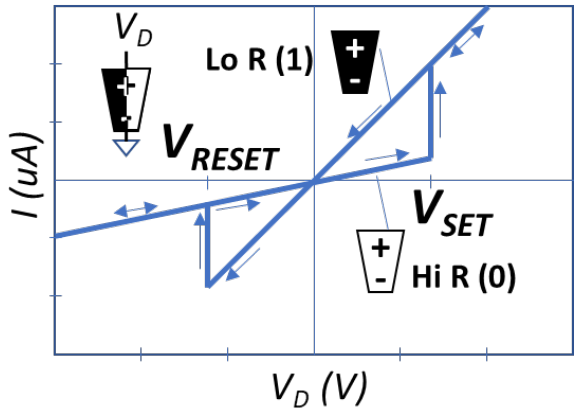
- **Bit-serial operations incur long latencies**

To compensate for long latencies, digital PUM architectures compute on whole columns of data to exploit data-level parallelism

Why Digital and Not Analog?

- **Analog PUM enables fast matrix multiply**
  - Store weights as resistance in **multi-level ReRAM cells**
  - Pass in inputs as variable voltages
  - Matrix multiply enabled by Kirchhoff's current law
- **Key drawbacks of analog PUM**
  - **Requires ADCs/DACs** that convert from/to currents for use with other logic
  - **Non-linearity of device** response makes it difficult to make multi-level cells





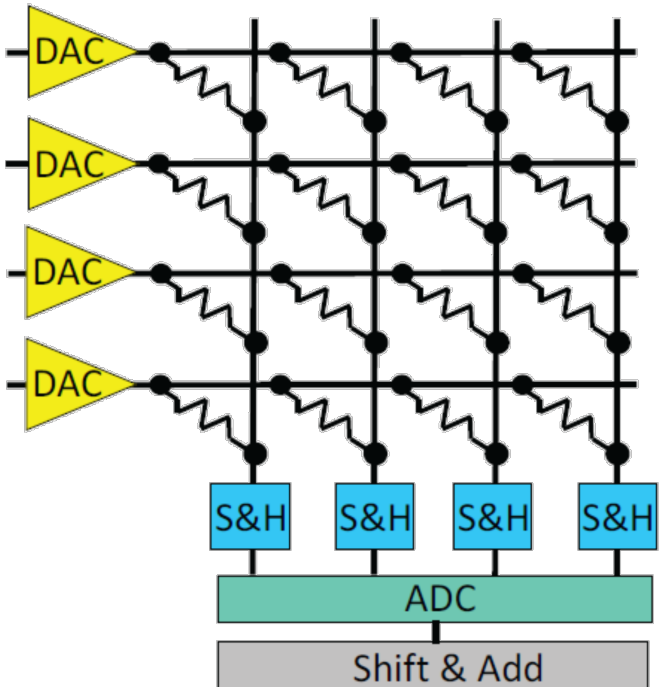


Image sources: Shafiee+ ISCA 2016

Issues with ReRAM-Based Digital PUM

- **The larger the crossbar size, the bigger the throughput for whole-column operations**

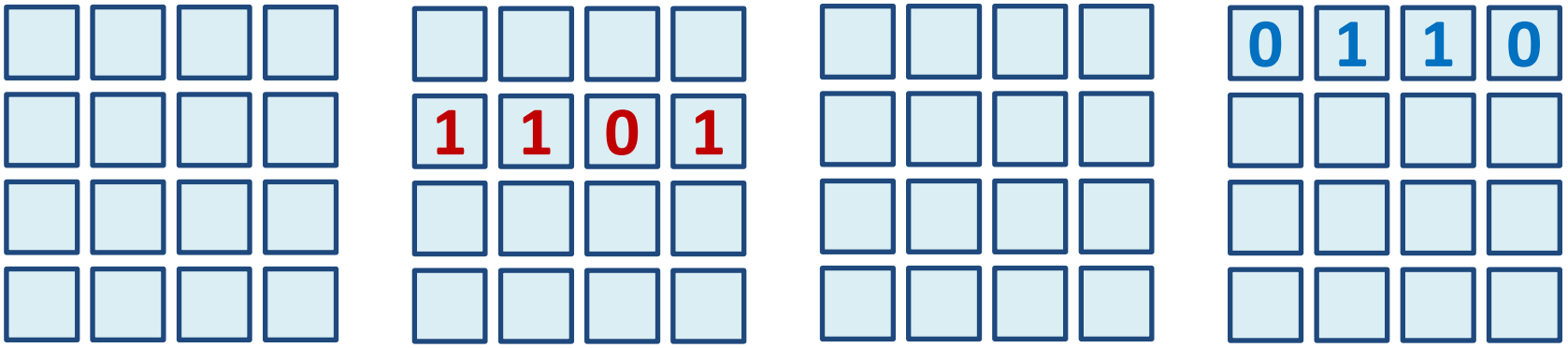
- **Whole-column operations limit the crossbar size**
  - **Each extra cell in a column adds current** → grows **proportionally** to crossbar size
  - Limited by the current carrying capacity of a wire
  - Large currents **permanently damage** the metal wires *(see MICRO 2021 paper for analysis)*

Whole-column operations are realistically possible only when column length < 200 cells!

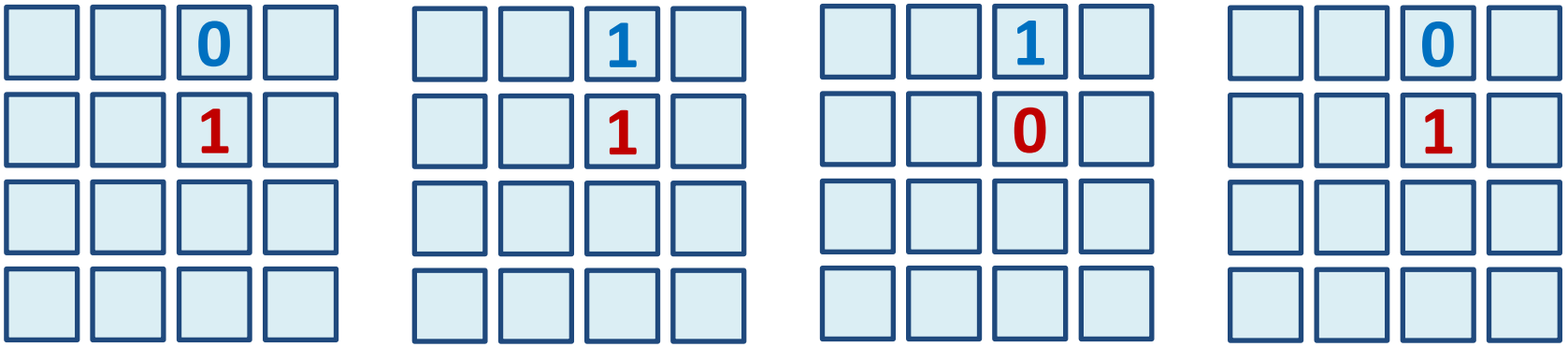
How can small tiles deliver high throughput at low overhead?

RACER: Optimizing PUM for Small Memory Tiles

- State-of-the-art processing-using-memory architectures keep whole chunks of a word in a single tile

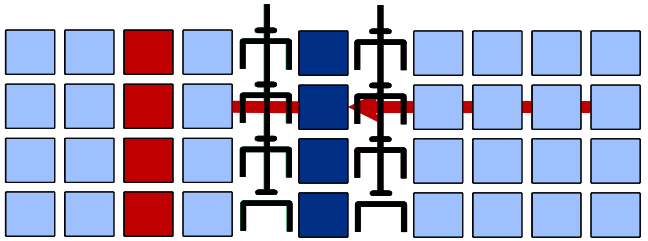


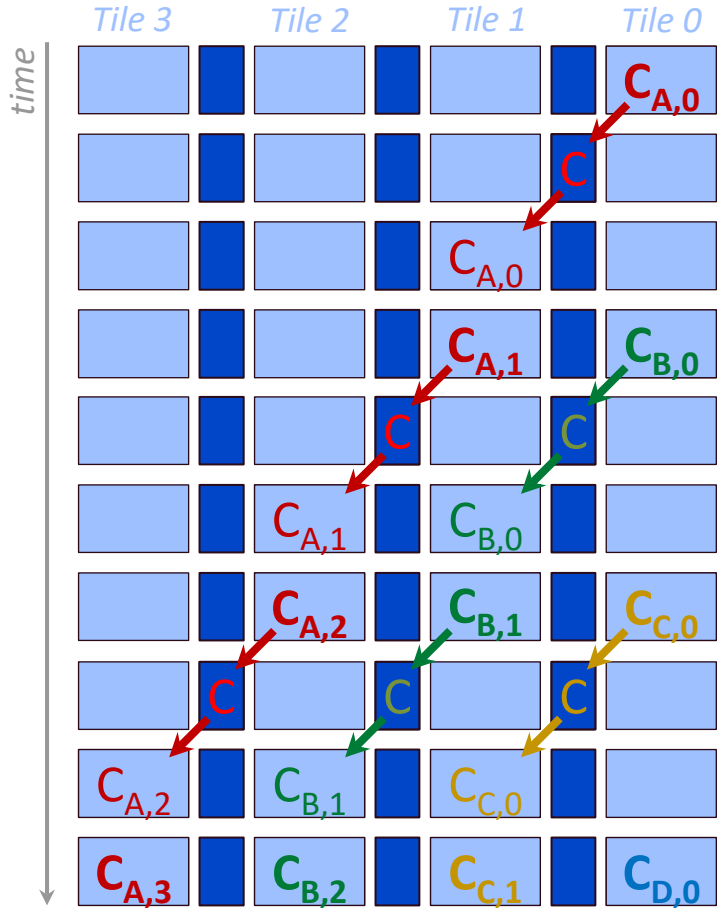
- In RACER, we distribute each bit of a word to a different tile



RACER: Dealing with Bit-Serial Ops Across Tiles

- **We add 1x64 ReRAM column buffers**
  - Enables tile-to-tile communication
  - Connects to an adjacent tile using pass gates

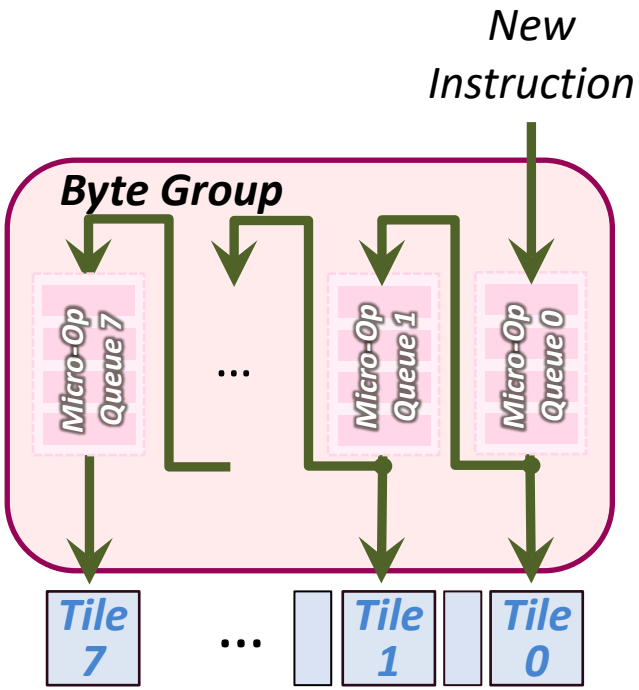




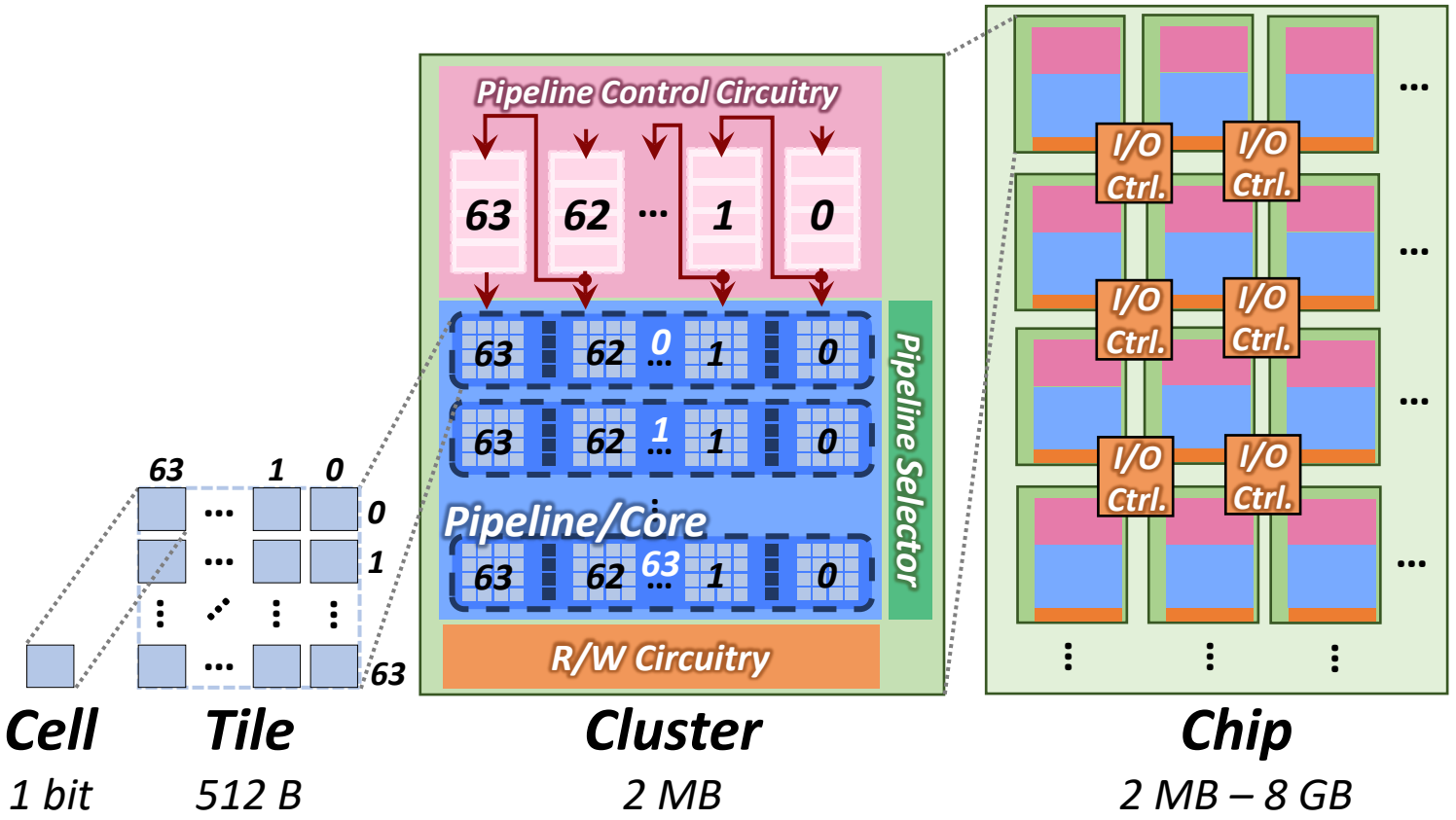
- **We enable a new technique that we call bit-pipelining**
  - **Treat each tile as a pipeline stage**
  - With *t* tiles, we can operate on *t* columns of words at once

Byte Group: Instruction Coordinator for RACER

- Core control circuitry for bit-pipelining
- Each bit (i.e., each tile) repeats the same exact operations
- NOR instructions (micro-ops) are stored in micro-op queues
  - Each tile has a dedicated queue
  - Queue *i* sends its micro-ops to Queue *i*+1
- Enables efficient support of 8-/16-/32-/64-bit operands



RACER Enables Scalable Edge Computing



- Each tile gets 1 bit of a word; pipelined across bits
- Cluster: 64 pipelines sharing one set of control/peripheral circuits
- Chip can contain however few/many clusters as needed

So What Can RACER Do With Bit-Serial Compute?

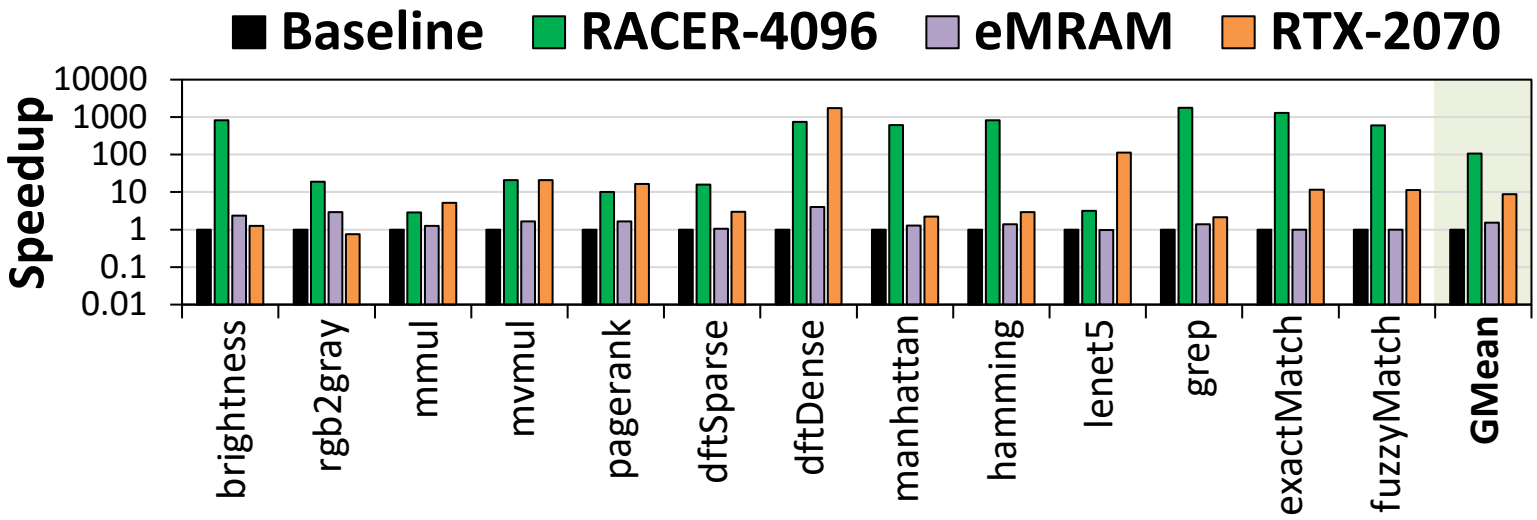
- **RACER core: pipeline w/ 32 kB of data**
  - Corresponds to 64 tiles connected w/ buffers
  - Each core has local access to data from 512 cores
  - Global network gives each core access to entire chip's data (up to 8 GB)
  - **Vectorized ISA** for easy programmability
    - » 64 words at once
    - » Can support non-bit-pipelined instructions†

Op.	Description/Notes	Op.	Description/Notes
Arithmetic Operations			
ADD	Two's complement add	ABS	Absolute value
SUB	Two's complement subtract	MUX	Multiplex (i.e., choose)
POPC	Population count	RELU	Rectified linear unit
CMPEQ	Check equality	LSHIFT	Left shift by 1
FUZZY	Fuzzy search	RSHIFT	Right shift by 1
MUL†	Multiply (only 8-/16-/32-bit)	SQRT†	CORDIC square root
MAC†	Multiply–accumulate	SIN†	CORDIC sine
DIV†	Division (returns quotient & remainder)	COS†	CORDIC cosine
MAX	Searches for the maximum number	EXP†	CORDIC exponent
MIN	Searches for the minimum number	CAS	Compare and swap
Boolean Operations			
NOR	Bitwise NOR	OR	Bitwise OR
NAND	Bitwise NAND	AND	Bitwise AND
NOT	Bitwise NOT	XOR	Bitwise XOR
Data Transfer Operations			
MOV	<**MOV** *buff*[*dst*] = *buff*[*src*]> Moves data stored in buffers of core *src* to buffers of core *dst*	SHIFT	<**SHIFT** *stride*> Parallel data shift *dst* = *src* + *stride*
Configuration Operations			
SET	<**SET** *start*, *stop*, *stride*> Turns on RACER core *i* for *i* ∈ *range*(*start*, *stop*, *stride*)	UNSET	Turns off all RACER cores that are active

Methodology

- **Iso-area comparisons to four state-of-the-art platforms**
  - **Baseline:** 16-core Xeon 8253 CPU + 8GB off-chip DRAM
  - **eMRAM:** 16-core Xeon 8253 CPU + 8GB on-chip MRAM
  - **RTX-2070:** GeForce RTX 2070 GPU
  - **DC:** Duality Cache, a compute-in-SRAM architecture
- **We model RACER at multiple levels of the stack**
  - Device-level ReRAM characteristics modeled using VerilogA with in-house device measurements
  - Control and peripheral circuits synthesized using FreePDK 15 nm
  - RACER ISA microbenchmarks executed using in-house simulator
  - Baseline modeled using MARSSx86 + DRAMSim2 + McPAT
- **Full paper:** https://ghose.cs.illinois.edu/papers/21micro_racer.pdf
- **Simulation framework open-sourced:** https://doi.org/10.5281/zenodo.5495803

RACER Increases Performance vs. CPU/GPU



107× speedup vs. CPU

thanks to RACER's tile-/pipeline-/cluster-level parallelism

71× speedup vs. eMRAM

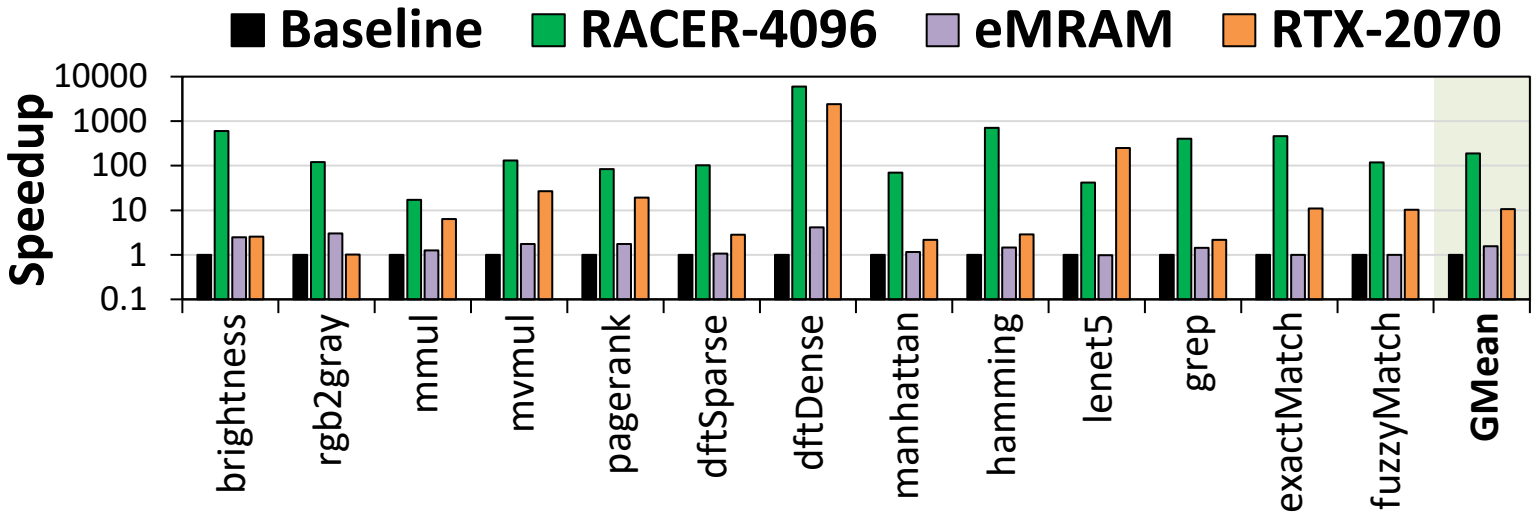
as embedded memory does not reduce frequent data movement

12× speedup vs. GPU

7× speedup vs. DC *(not shown)*

thanks to RACER's *in-situ* computation and tile-/pipeline-/cluster-level parallelism

RACER Significantly Reduces Energy



189× savings vs. CPU

thanks to RACER's *in-situ* computation and fast low-power circuitry

94× savings vs. eMRAM

as embedded memory mostly reduces only the energy used by off-chip network

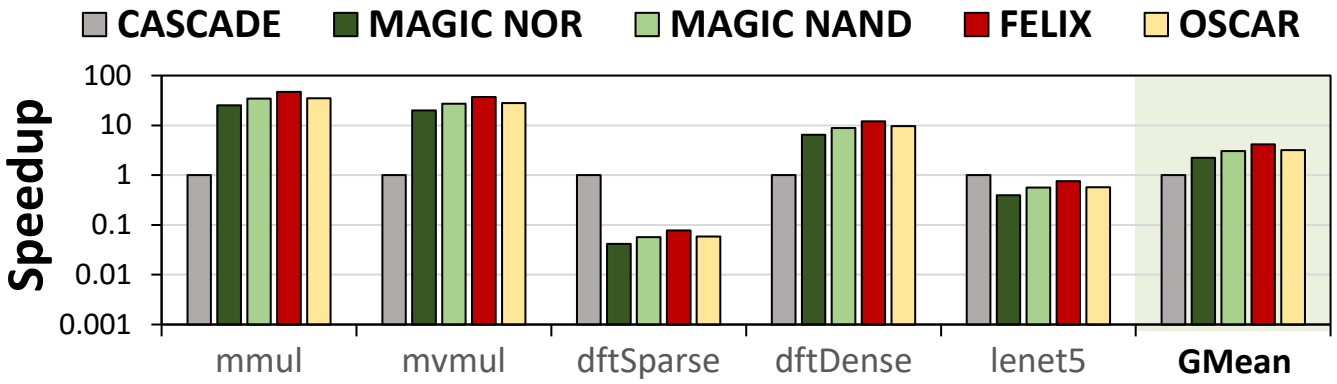
17× savings vs. GPU

1.3× savings vs. DC *(not shown)*

5× savings vs. DC for applications that trigger frequent data swapping

RACER Outperforms Analog Neural Net Accelerators

- **CASCADE**
  - State-of-the-art neural network (NN) accelerator [Chou+ MICRO 2019]
  - Over an order of magnitude throughput *and* energy improvements over CMOS-based NN accelerator (DaDiaNao)



- **RACER outperforms CASCADE**
  - RACER+OSCAR: **3.16x throughput improvement** on average
  - CASCADE outperforms RACER for sparse matrices
  - RACER is better for edge computing: can run many non-NN operations & microbenchmarks that CASCADE can't

Introduction

Characterizing Edge NN Models

Alleviating Data Costs with Processing-in-Memory

Mensa-G: Heterogeneous NN Acceleration with PNM

RACER: Edge Data Acceleration with PUM

Closing Thoughts

Conclusion

- **Inference on edge devices is stressing accelerator capabilities**
  - We think of neural network (NN) models as computationally-intensive
  - Edge NN model footprints **exceeding the limited storage of accelerators**
  - We show this with a detailed characterization of **Google edge NN models**
- **Processing-in-memory can come to the rescue!**
  - New memory capabilities can overcome memory channel bottlenecks
  - Variants: **processing-near-memory** (PNM), **processing-using-memory** (PUM)
- **Mensa-G: heterogeneous edge NN accelerators using PNM**
  - **3.1x performance, 3.0x energy improvement** vs. Google Edge TPU
  - Full paper: Boroumand+ PACT 2021
- **RACER: data accelerator for edge computing using PUM**
  - **107x performance, 189x energy improvement** vs. 16-core Intel Xeon
  - Full papers: Truong+ MICRO 2021, Truong+ JETCAS 2022

Thanks to My Collaborators

- **ARCANA Research Group:**
  https://arcana.cs.illinois.edu/

  - Amirali Boroumand
  - Minh S. Q. Truong
  - Eric Chen
  - Deanyone Su
  - Alex Glass
  - Ali Hoffmann

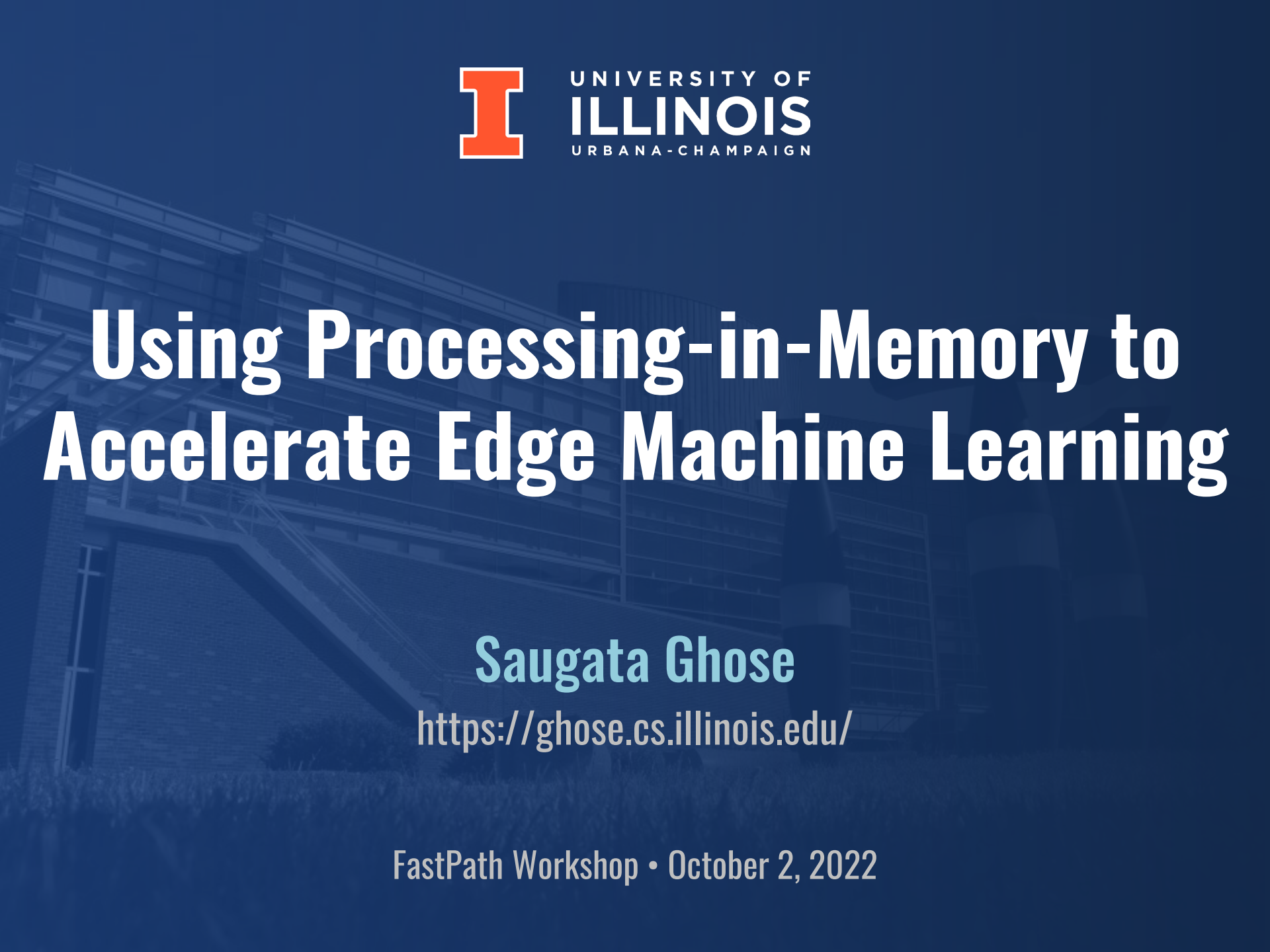
- **SAFARI Research Group:**

  - Onur Mutlu
  - Geraldo F. Oliveira
  - Juan Gómez-Luna
  - … and many others

- **CMU Data Storage Systems Center**

  - James A. Bain
  - L. Richard Carley
  - Marek Skowronski
  - Liting Shen

- **Supporters of ARCANA's PUM Research**
  - Scott Center for Energy Innovation
  - Sandia National Laboratories
  - Apple Ph.D. Fellowship for Minh Truong



UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Using Processing-in-Memory to Accelerate Edge Machine Learning

Saugata Ghose
https://ghose.cs.illinois.edu/

FastPath Workshop • October 2, 2022