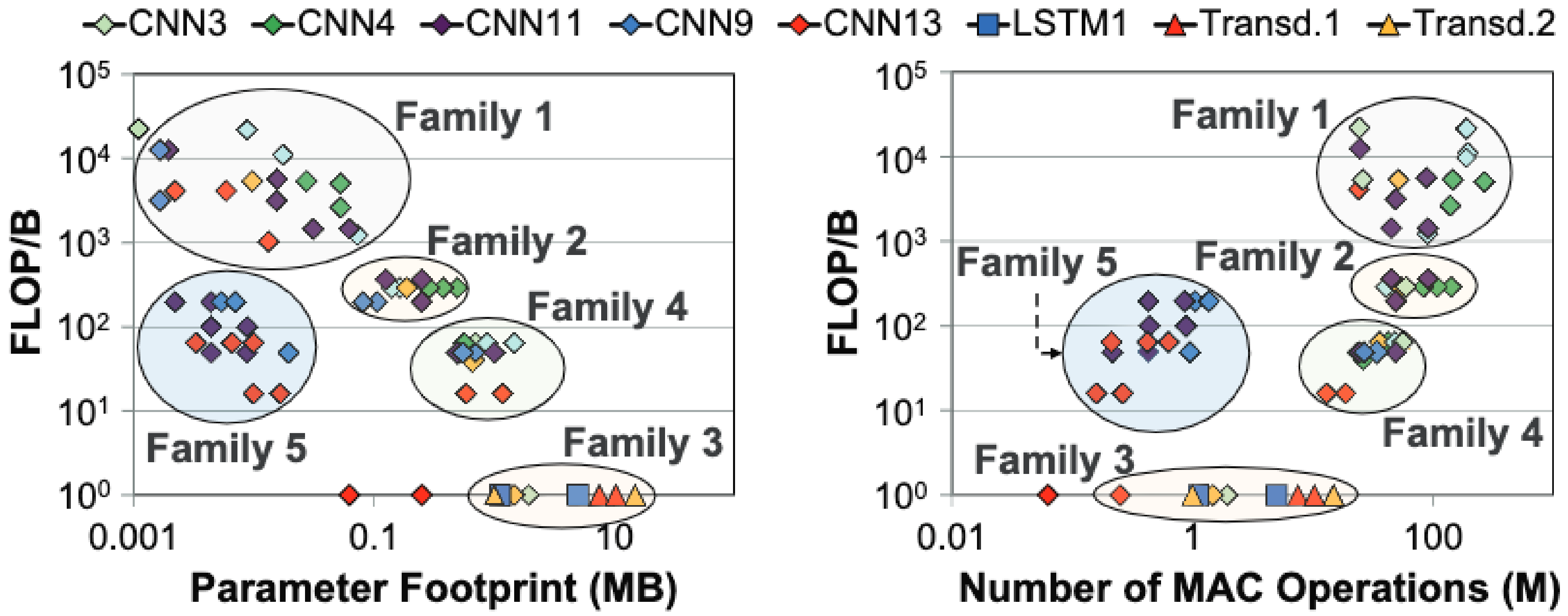
Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottleneck

Amirali Boroumand, Saugata Ghose, Berkin Akin Ravi Narayanaswami,
Geraldo F. Oliveria, Xiaoyu Ma, Eric Shiu, Onur Mutlu

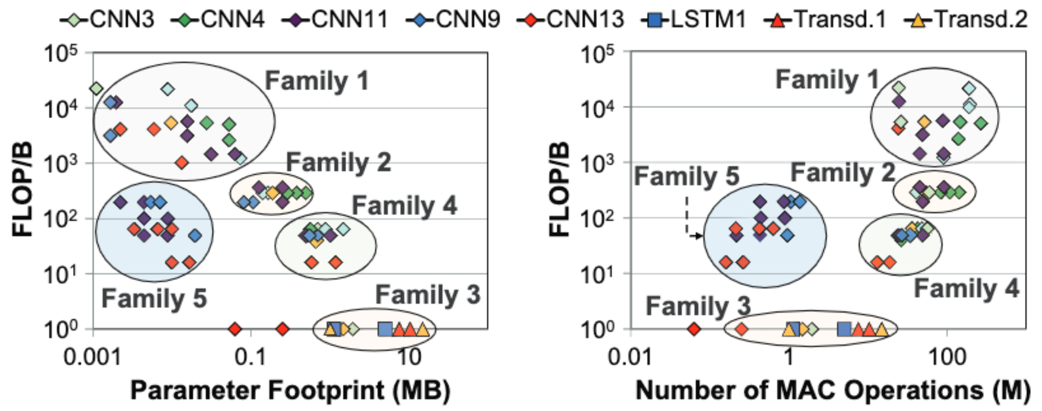
Made by Yongwon Shin | Computing Systems Lab

Model Families



- 5 groups
- Accelerator: F1,2 / F3 / F4,5

Model Families

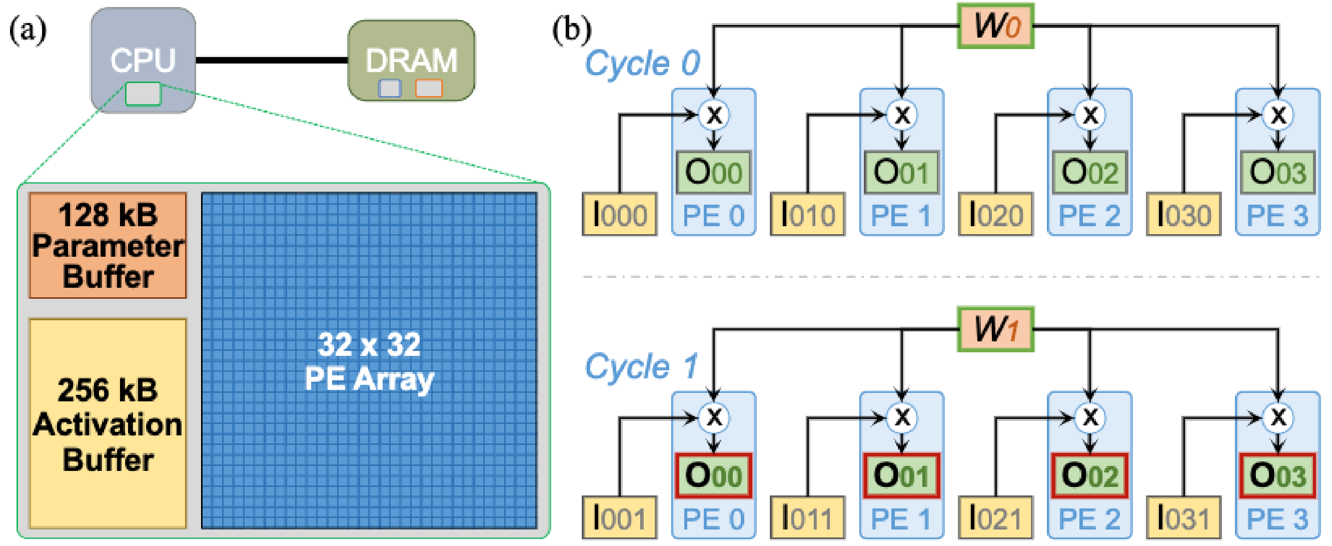


- Family 1
  - Small param. footprint, high AI for params., high MAC AI
  - e.g., early layers in CNN/RCNN
- Family 2
  - small param. footprint, moderate AI for params, high MAC AI
  - e.g., pointwise conv, middle
- Family 3
  - large param. footprint, minimal AI for params, low MAC AI
  - e.g., LSTM, Transducers
- Family 4
  - relatively large param. footprint, low-to-moderate AI for params, moderate MAC AI
  - e.g., deep input/output channels
- Family 5
  - very small parameter footprint, moderate AI for params, low MAC AI
  - e.g., depthwise conv

Pascal: Compute-Centric Accelerator Design

- Design point
  - Temporal reduction: mitigates the impact of large output activation footprint
  - Shoud avoid spatial reduction

Weight are supplied from parameter buffer due to reuse



Activations are accumulated across ***time***

Figure 7. (a) Pascal design; (b) Pascal dataflow for a pointwise layer.

* Note that Edge TPU has 64x64 PEs and 256kB params. buf.

Pavlov: LSTM-Centric Accelerator Design

- Design point
  - Exploit output activation reuse opportunities
  - Should reduce the off-chip memory BW required by params.

MVM requires V reused

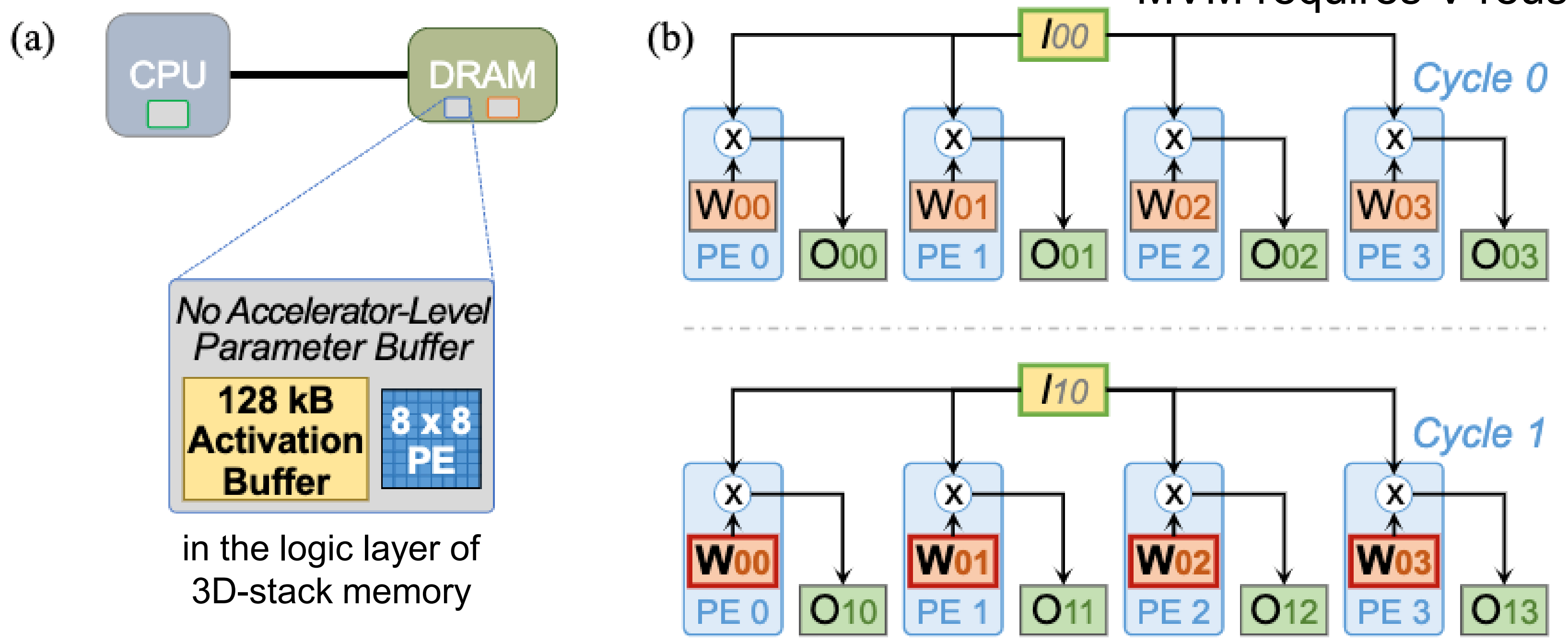


Figure 8. (a) Pavlov design; (b) Pavlov dataflow.

Weigh: few time use
→ No need buffer!

Jacquard Accelerator Design

- Design point
  - Should exploit temporal reuse opportunities for params.
  - Should provide high off-chip memory bandwidth

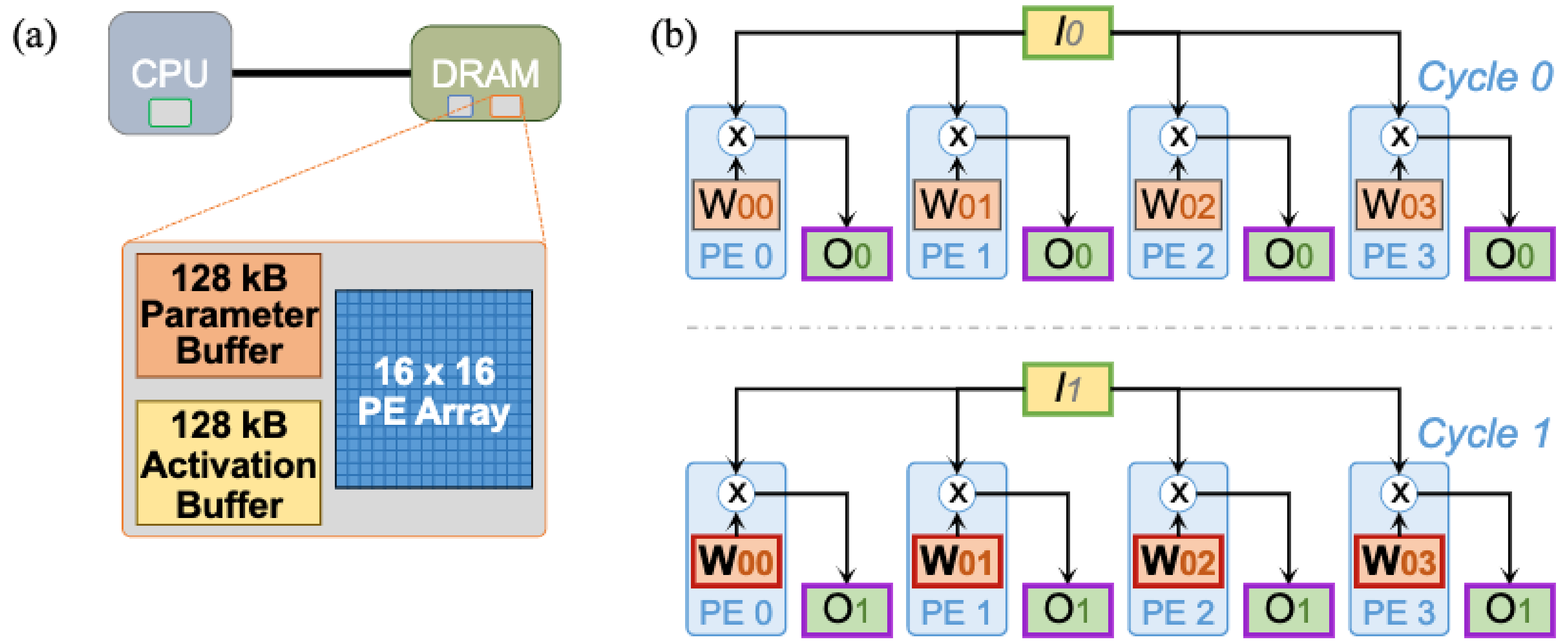


Figure 9. (a) Jacquard design; (b) Jacquard dataflow.

Smaller PEs and
Smaller Act. buf.
than Pascal

But dataflow is similar to Pavlov
→ due to layer characteristics

Mensa Runtime Schedular

- Assumption: No concurrent execution of layers
- Phase 1
  - Iterates through each layer in the model, and identifies the ideal hardware accelerator *in isolation*
  - To maximize throughput and energy efficiency
  - Not consider transferring act / comm. dependencies.
- Phase 2
  - Accounts for communication overhead using a simple cost analysis algorithm
  - Two cases
    - If the number of MAC operations required for layer *i* is 2x higher than the compute resources available in destination *i* – 1
    - If the amount of params. data (that destination *i* – 1 would need to fetch to run layer *i)* is grater than the amount of output activation data (that would have to be sent to the ideal accelerator) AND the opportunities for reusing the params. data in destination *i* – 1 are low (FLOP/B < 64)

→ Google edge models typically communicate between accelerators only 4-5 times during execution (implying one-time assignment is mostly optimal)

➔ Heuristic-based approach

Methodology

- Models
  - 24 Google edge NN models from Google mobile applications/products
  - TensorFlow Lite, 8-bit, QAT
  - Not disclosed
- Energy
  - PE and NoC (static/dynamic): In-house simulator
  - On-chip buffer: CACTI-P 6.5
  - MAC: 0.2pJ/bit
  - Memory access: According to LPDDR4 J/bit access in in-house simulator
- Performance
  - In-house simulator (Edge TPU)
  - On-chip buffer latency: CACTI-P 6.5
  - NDP: 256GB/s internal BW of HBM (8x external BW)
  - Edge TPU baseline: uses 2GB of HBM DRAM

Results

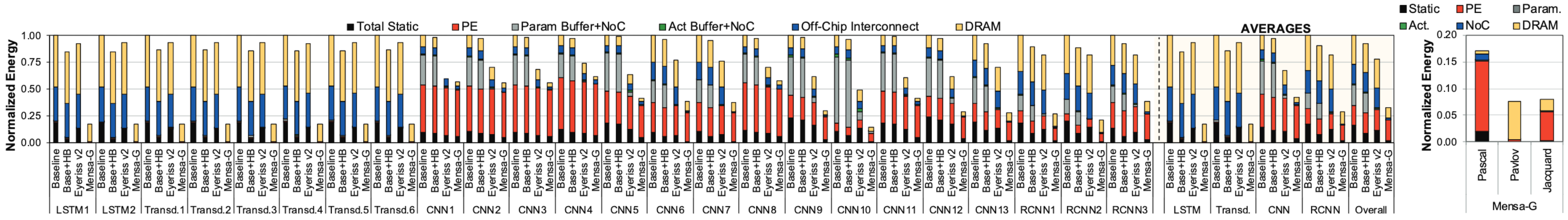


Figure 10. Inference energy across different models (left) and energy breakdown across our three proposed accelerators (right), normalized to Baseline.

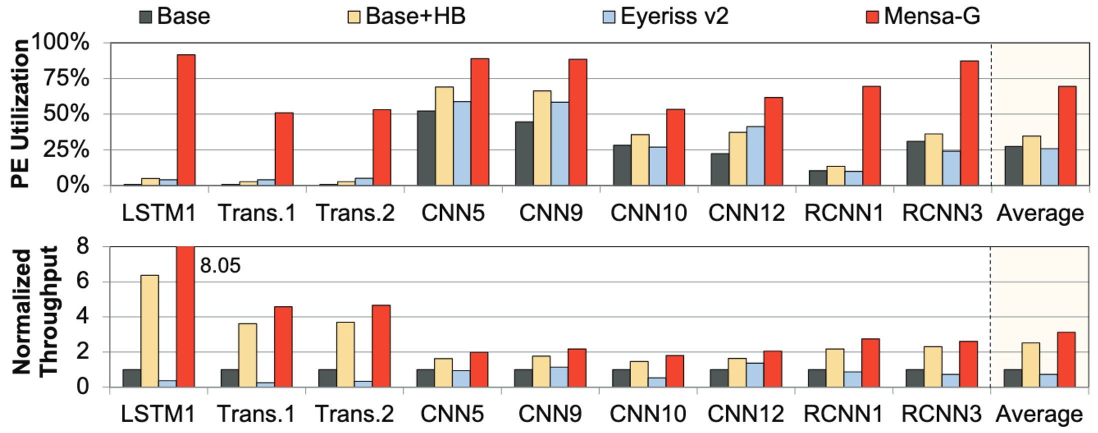


Figure 11. PE utilization (top) and Baseline-normalized throughput (bottom) across different models.

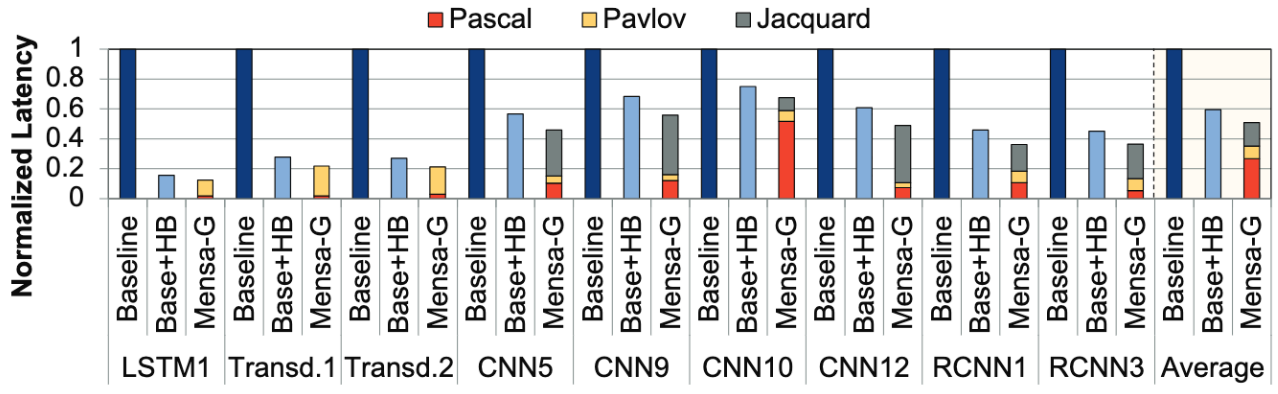


Figure 12. Inference latency, normalized to Baseline.

Interesting Points

- HW/SW co-design
  - Derived from intensive workload analysis
- Heterogeneous system with good scheduler
  - NDP is considered
  - Accelerators with multiple accelerators
- (In-house simulators)
  - Hard to develop new idea on top of this paper
  - Open-sourcing each simulator itself would have good impact

Future Work

- Concurrent execution and scheduling
  - NP-hard
  - c.f., MAGMA[1]
- Learning-based schedular
  - Currently hand-tuned
- Heterogeneity accelerator
  - Existing / future modes
  - Possibly more tight co-design of model families – scheduler?
- What about integration with PIM?
  - Enlarging search space with various constraints

[1] S. -C. Kao and T. Krishna, "MAGMA: An Optimization Framework for Mapping Multiple DNNs on Multiple Accelerator Cores," 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2022, pp. 814-830, doi: 10.1109/HPCA53966.2022.00065.