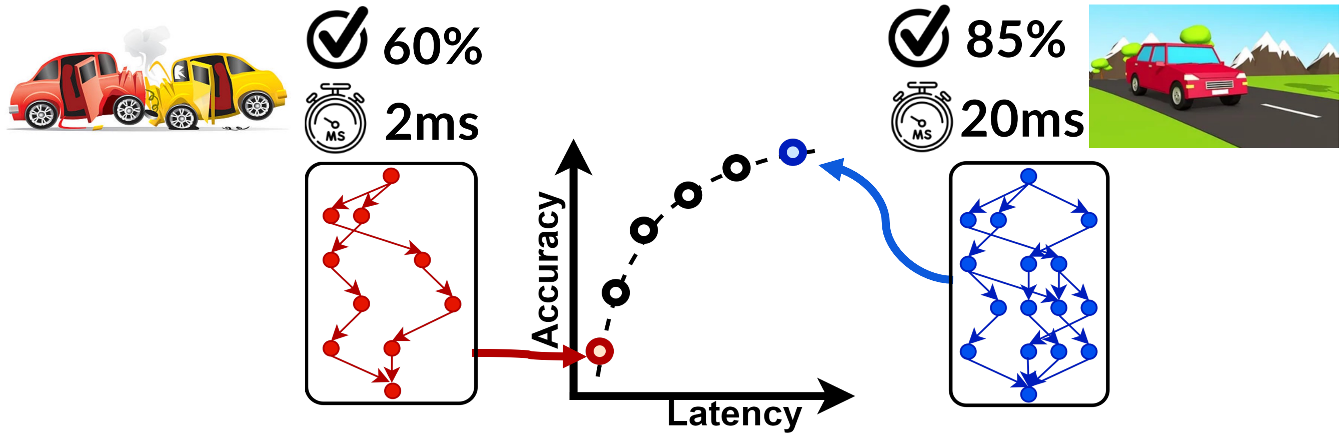
SubGraph Stationary HW-SW Co-design for ML Inference

Payman Behnam*[1], Jianming Tong*[2], Alind Khare [1], Yangyu Chen [2], Yue Pan [2], Pranav Gadikar [1], Abhimanyu Bambhaniya [2], Tushar Krishna [2], Alexey Tumanov [1]

[1]SAIL Lab, School of Computer Science, [2]Synergy Lab, School of ECE
Georgia Institute of Technology
* Equal Contribution

Motivation for Serving Multi-capacity Model

- ML applications are increasingly deployed in dynamic and unpredictable conditions.
  - Both Latency and accuracy constraints are critical.
- ML applications running on resource constrained devices forced to choose between accuracy and latency, which must be done rapidly to avoid missing deadlines.

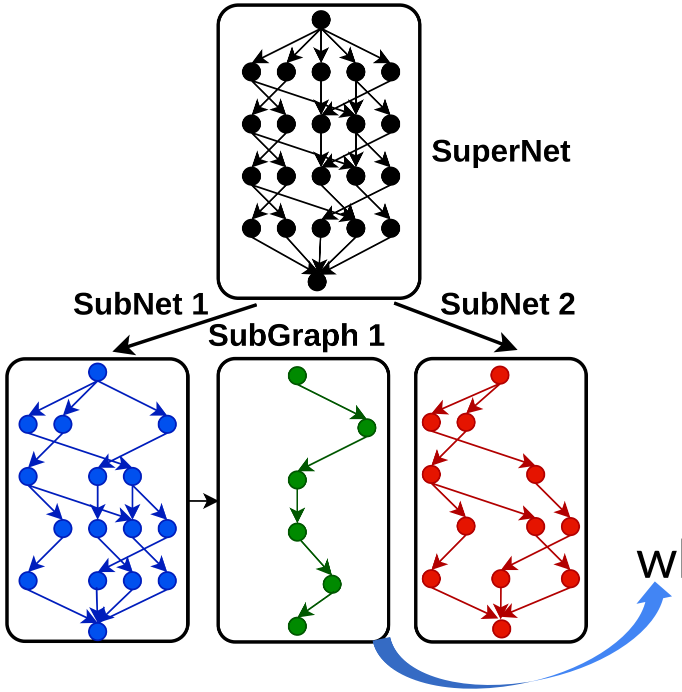


- No single point is sufficient while SOTA focuses on optimizing for a single point in the latency/accuracy tradeoff space.

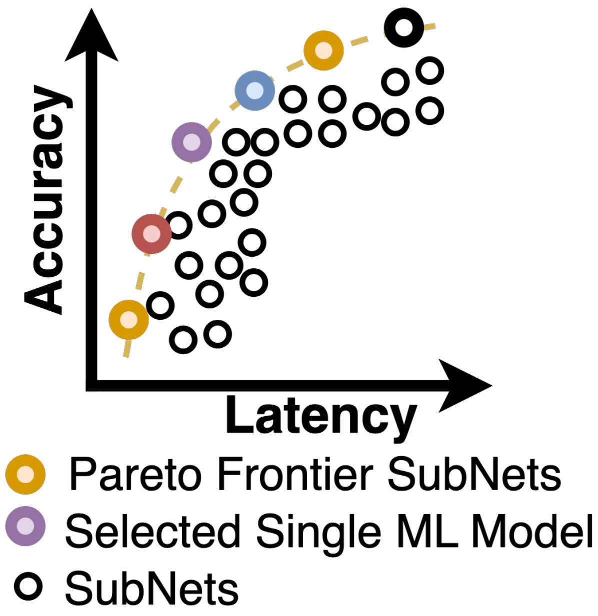
Takeaway: Serve a set of latency/accuracy options simultaneously, switching between them rapidly as needed. ***but how*?**

Weight-Shared (WS) DNNs

- WS-DNN is obtained by adding elasticity to the DNN dimensions.
- SuperNet forms a comprehensive and large model.
- SubNets which may differ in several elastic dimensions, partially share their weights as part of a single large DNN (SuperNet).
- SubNets can be directly used for predictions without any further re-training.
- Weight-shared DNNs induce a rich trade-off between accuracy and latency.



SubGraph is a subset of SubNet, which can't be used for Inference directly.



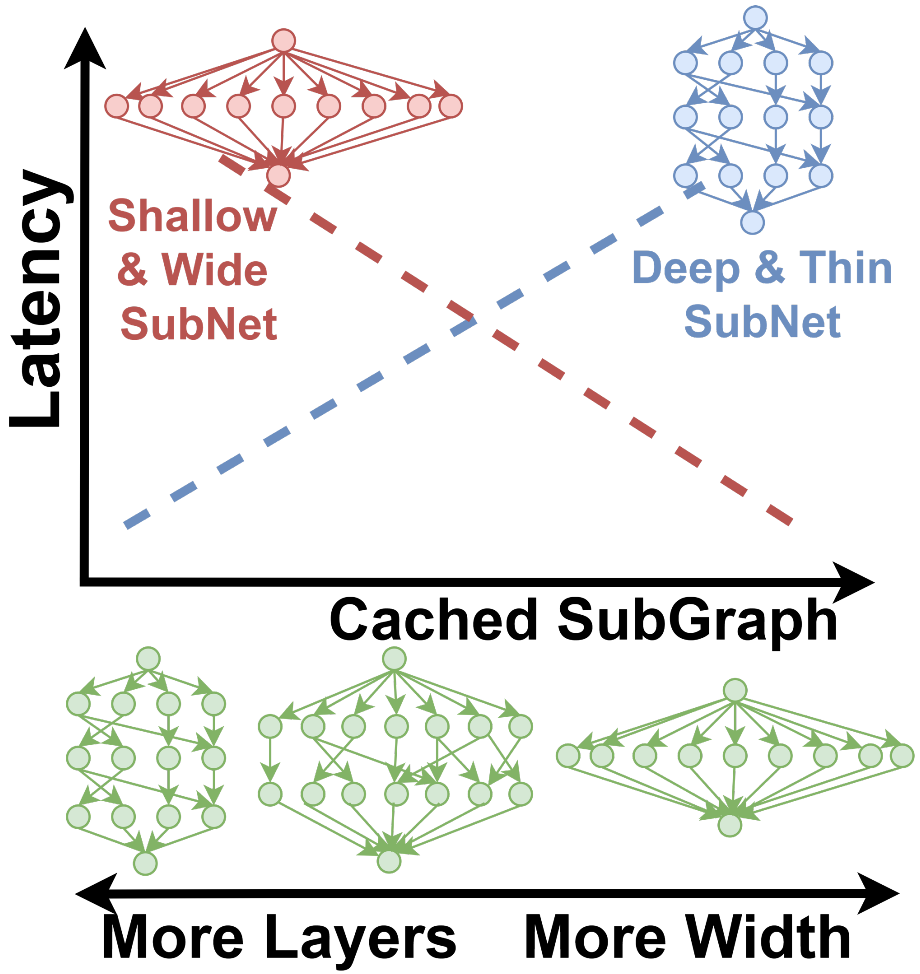
Proposed Solution: SUSHI

- Serving WS-DNNs: Vertically integrated HW-SW co-design
  - SUSHI enjoys co-design the accelerator and scheduling policy.
- HW (accelerator): latency pareto frontier navigation mechanism
- SW (Scheduler): latency pareto frontier navigation policy
  - What SubNets to serve
  - What SubGraph to cache
- Latency/accuracy navigation across a stream of queries

SUSHI Challenges

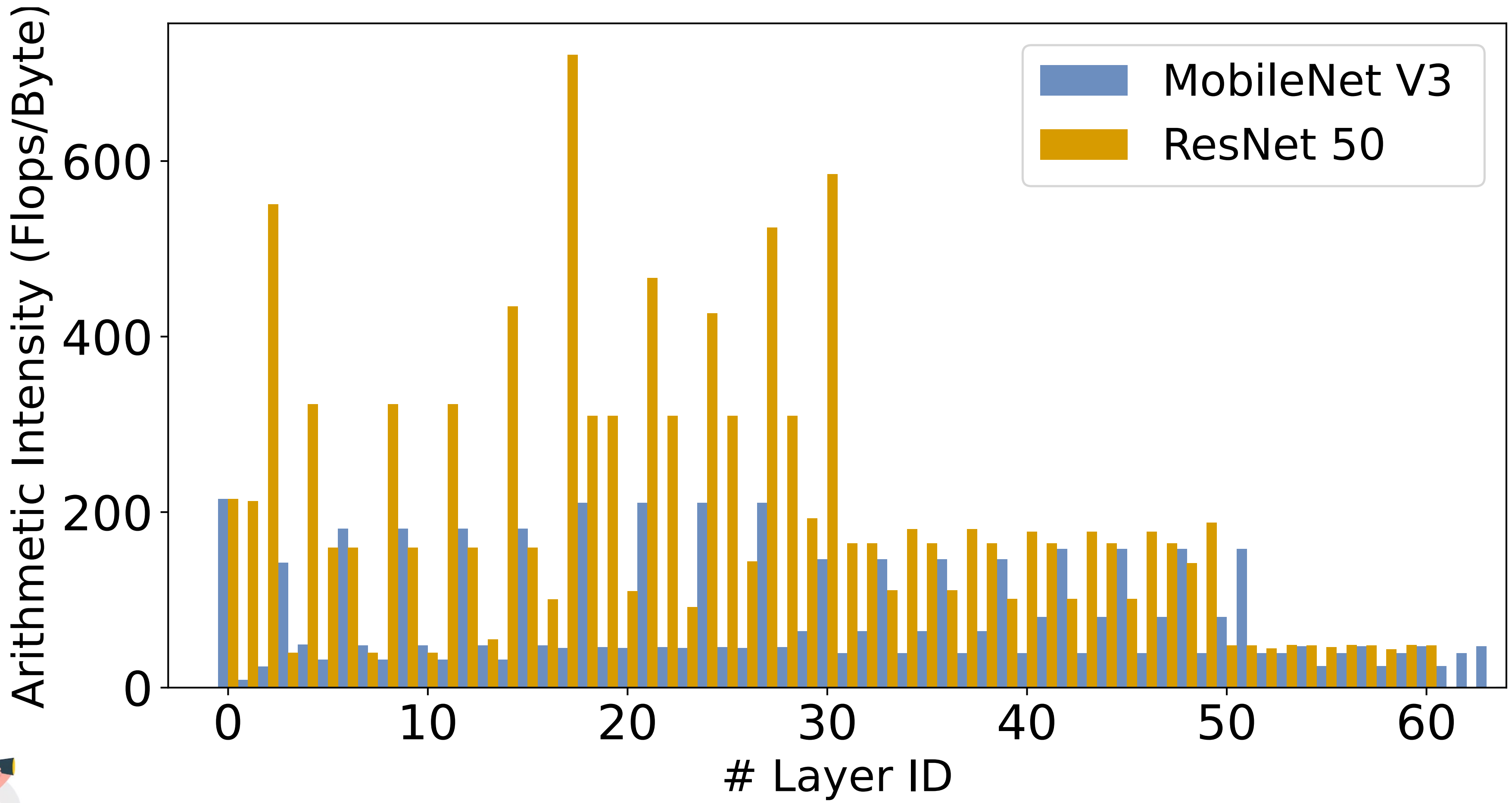
- Hardware: Support rapid switching between SubNets
- Caching: Best cache size for SubGraph caching
- Scheduling: What to serve & What to cache
- Abstraction: Generalize to any hardware

SUSHI Challenges: Caching



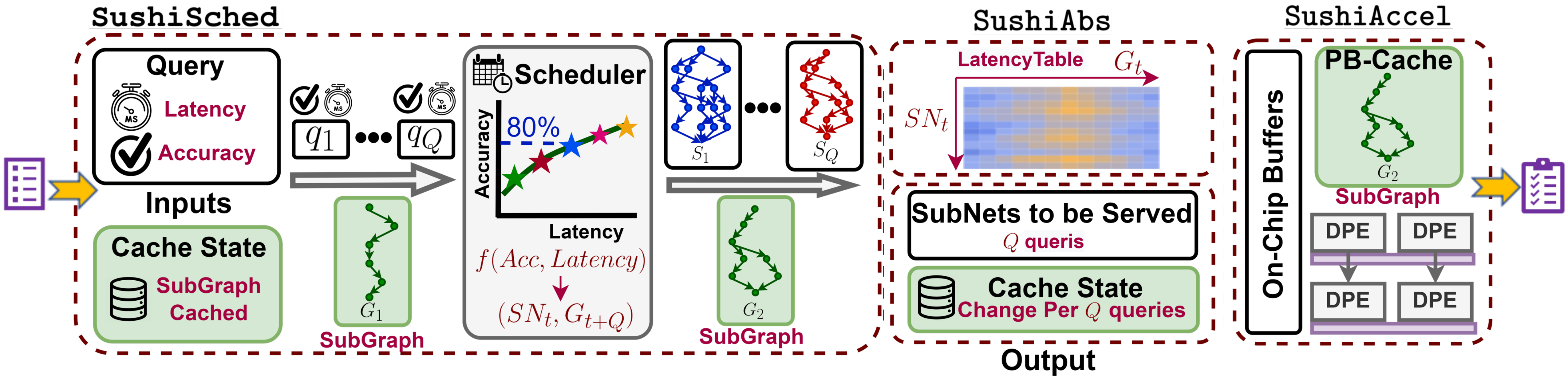
Latency of different SubNets is a function of different cached SubGraphs

SUSHI Challenges: Memory-boundedness



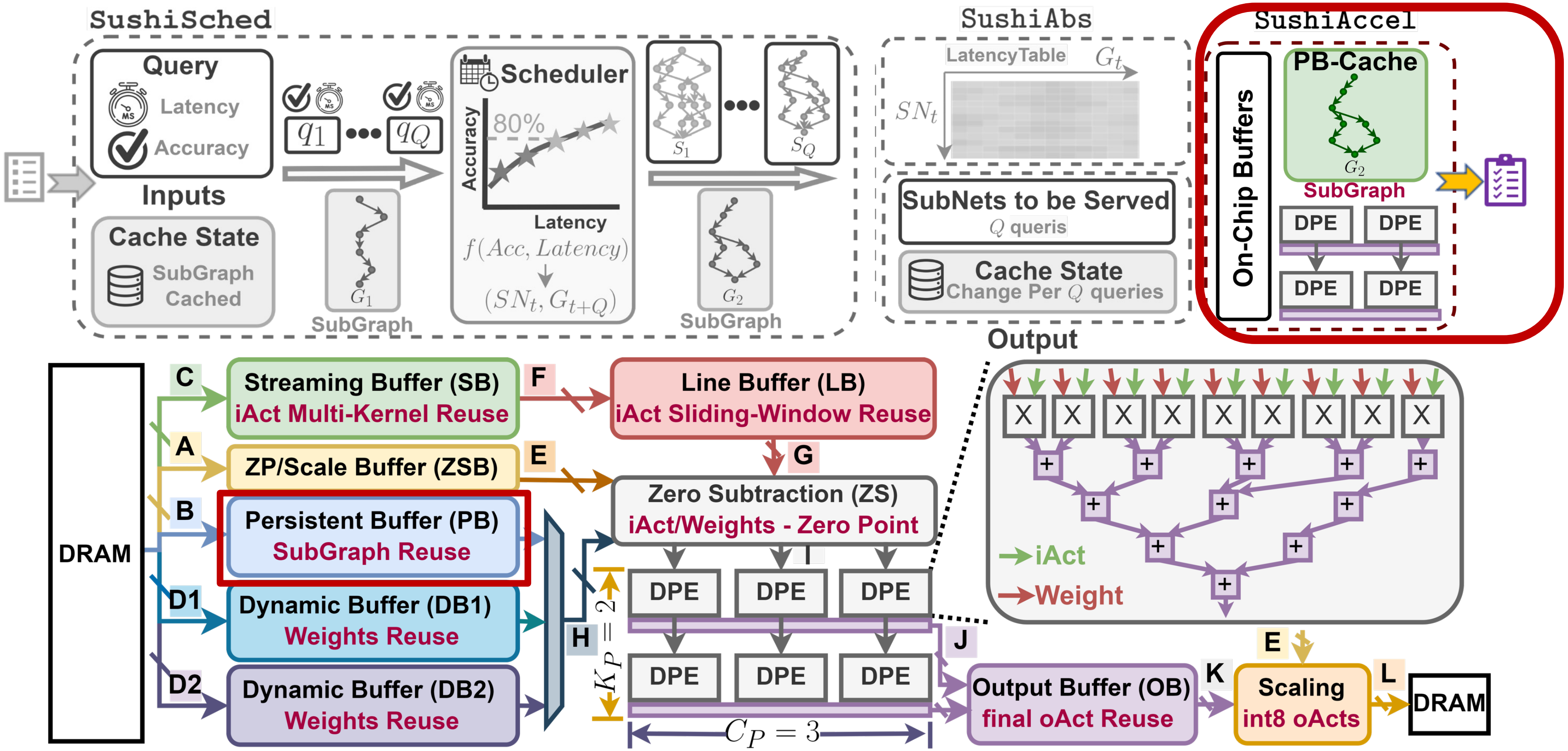
Some WS-DNN convolutional layers are memory- bound.

SUSHI System Overview



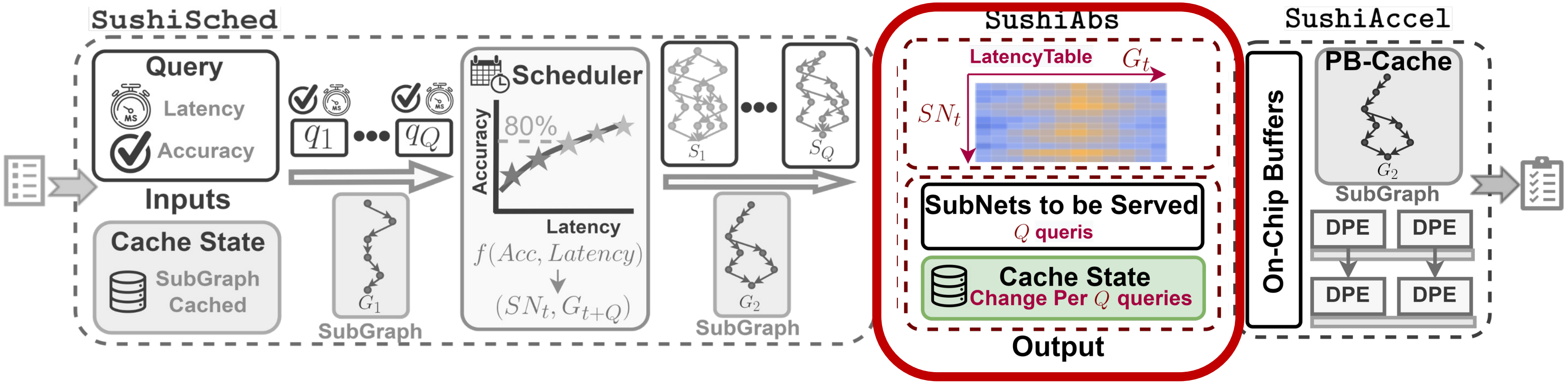
- Insight: Exploit temporal locality across weight shared inference queries.
- Key idea1: Decide what SubNet to serve (SN_t) based on what's cached.
- Key idea2: Decide what SubGraph to cache (G_t) based on the serving history.
- SubGraph Stationary (SGS) optimization across queries: Novel contribution.
- Generalizability: Abstracted accelerator state.

HW-SW Co-Design: FPGA Accelerator

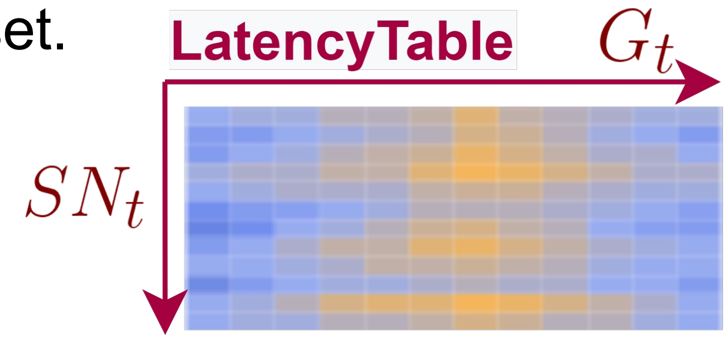


- SushiAccel supports input/weight/output and SubGraph Stationary.
- SushiAccel switches dataflows which are optimal for different layers.
  - Large kernels (kernel height ≥ 3) can be decomposed into serial of 3x3 to save computation & storage.
  - Small kernels (depth-wise conv) leverage channel level parallelism to increase resource utilization.

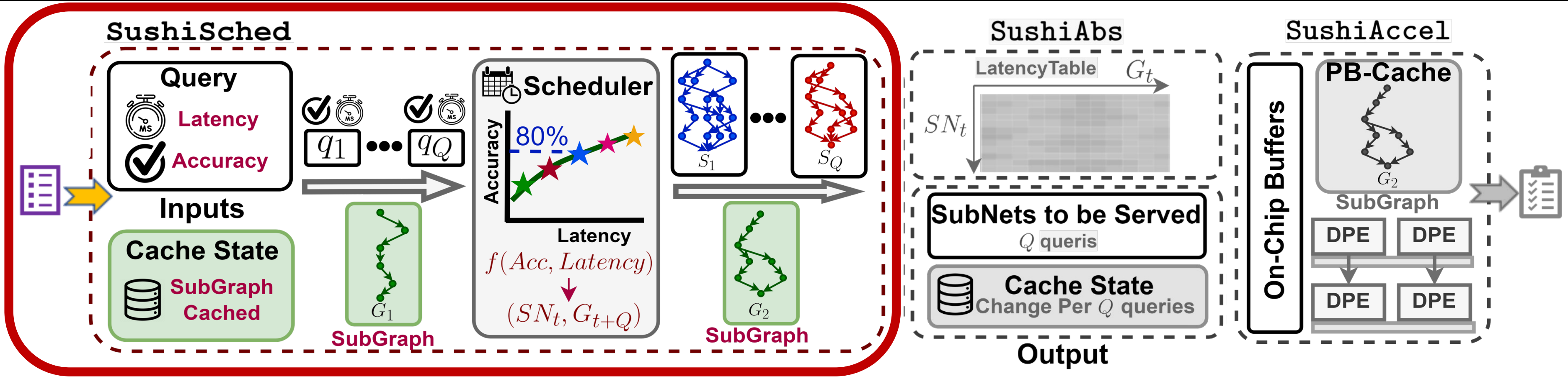
HW-SW Co-Design: Abstraction



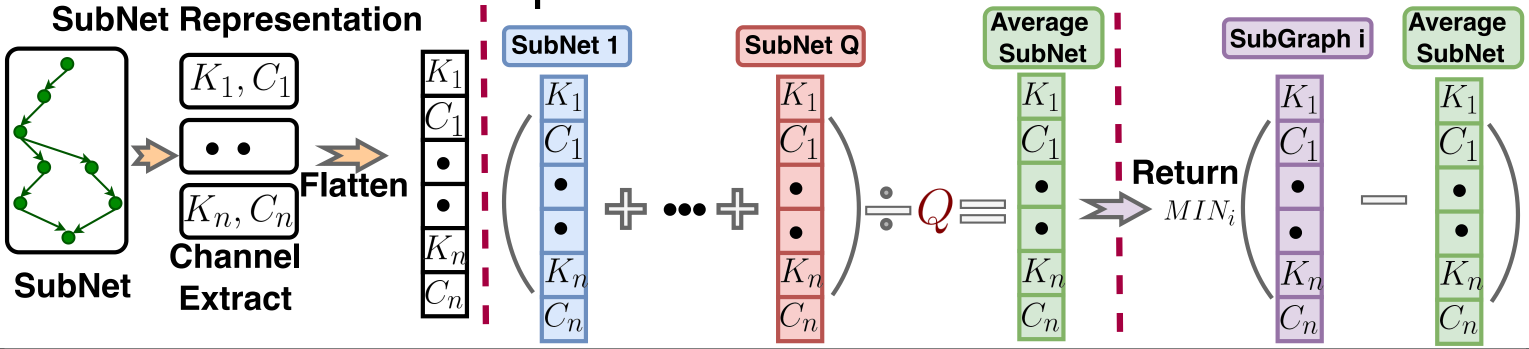
- Accelerator state awareness through abstraction:
  - Abstraction: $(SN_i, G_j) \rightarrow Latency_{ij}$
  - *Time efficient*: Abstraction employs a lookup table with SubNets as rows and SubGraphs as columns.
  - *Space efficient*: Abstraction limits the set of all possible cached SubGraphs to a significantly small set.



HW-SW Co-Design: Scheduler



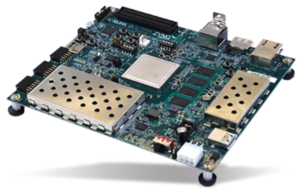
- Scheduler decides SubNets to serve as a function of cached SubGraph.
- The scheduler represents the SubNets and the SubGraphs as a vector.
- Proposed representation uses the number of kernels K_i and the number of channels C_i of every $layer_i$ to create a vector of size $2N$ for a N layered neural network.
- The scheduler keeps a running average of the past Q queries that were served by the scheduler to decide what SubGraph to cache.



EXPERIMENTAL Setup

- Workload: ResNet50 and MobV3
  - ResNet50 Sizes [7.58 MB, 27.47 MB]
  - MobV3 Sizes [2.97 MB, 4.74 MB]
- Simulators:
  - Architecture Analytic Model
  - Roofline Analysis
  - Scheduler
- Compare SushiAccel w/ PB and w/o PB with Xilinx DPU and CPU (Intel i7 10750H, 45 W).
- FPGA Deployment Platforms:

Edge



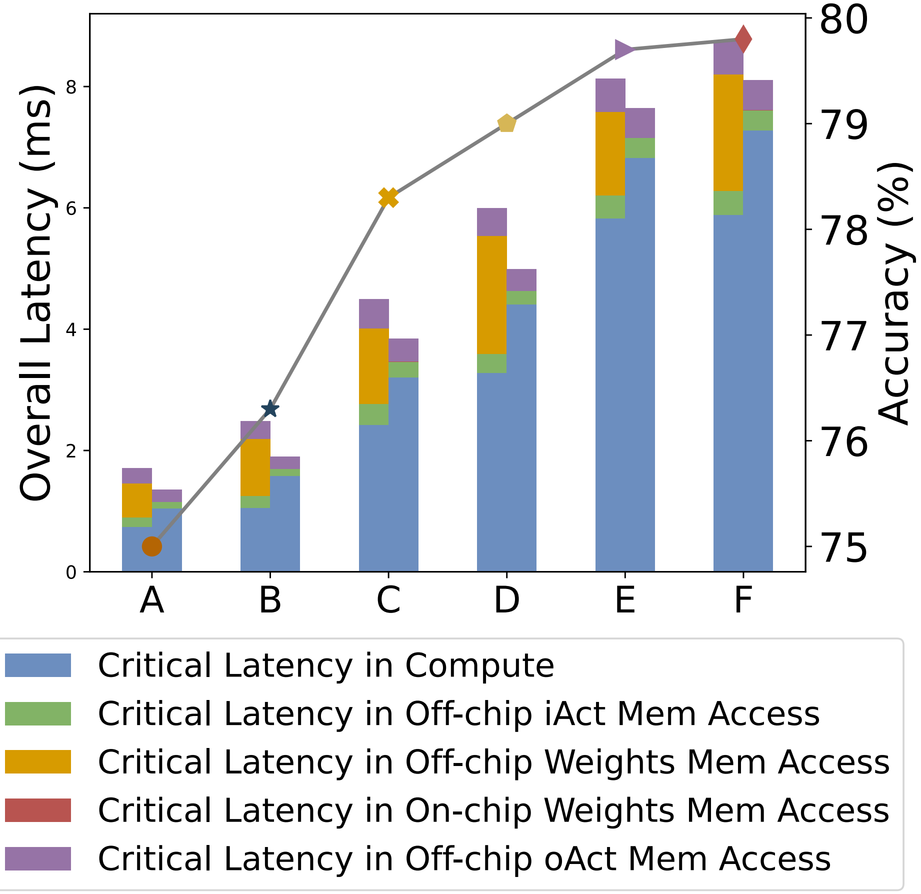
ZCU 104 (5W)

Cloud

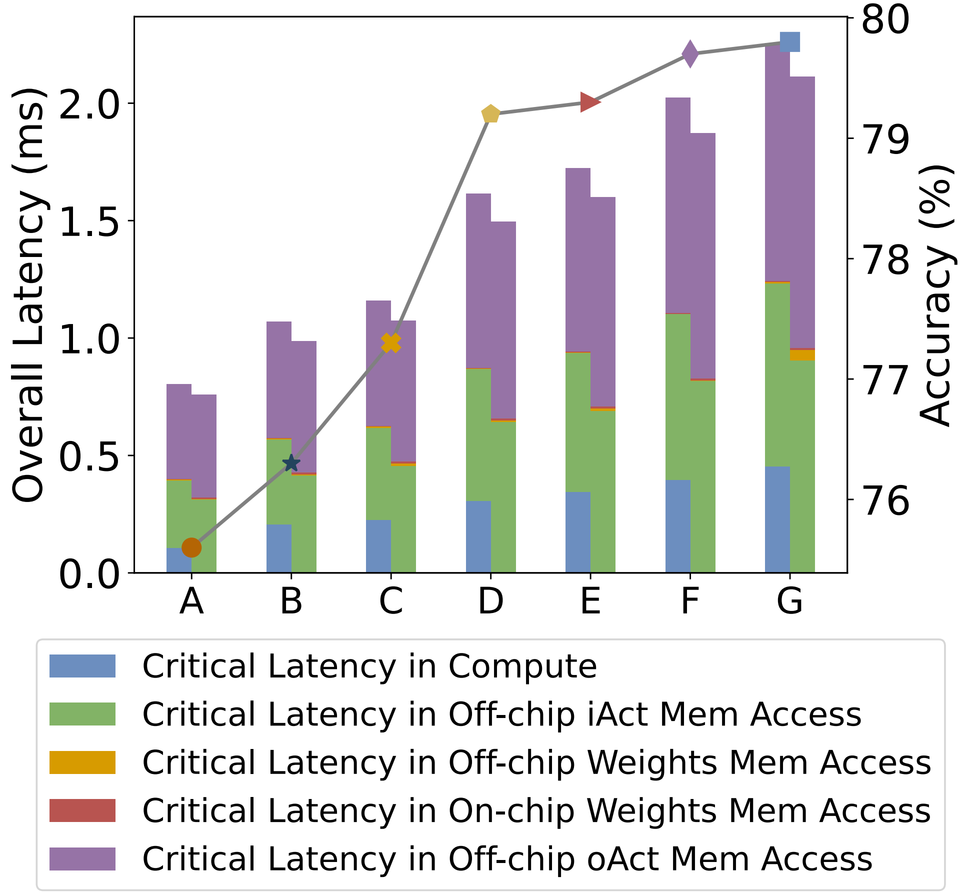
Alveo U50 (75W)

Potential Latency Reduction with SGS

- The latency of serving queries from pareto-frontiers could be reduced by [6%, 23.6%] for MobV3 and [5.7%, 7.92%] for ResNet50.



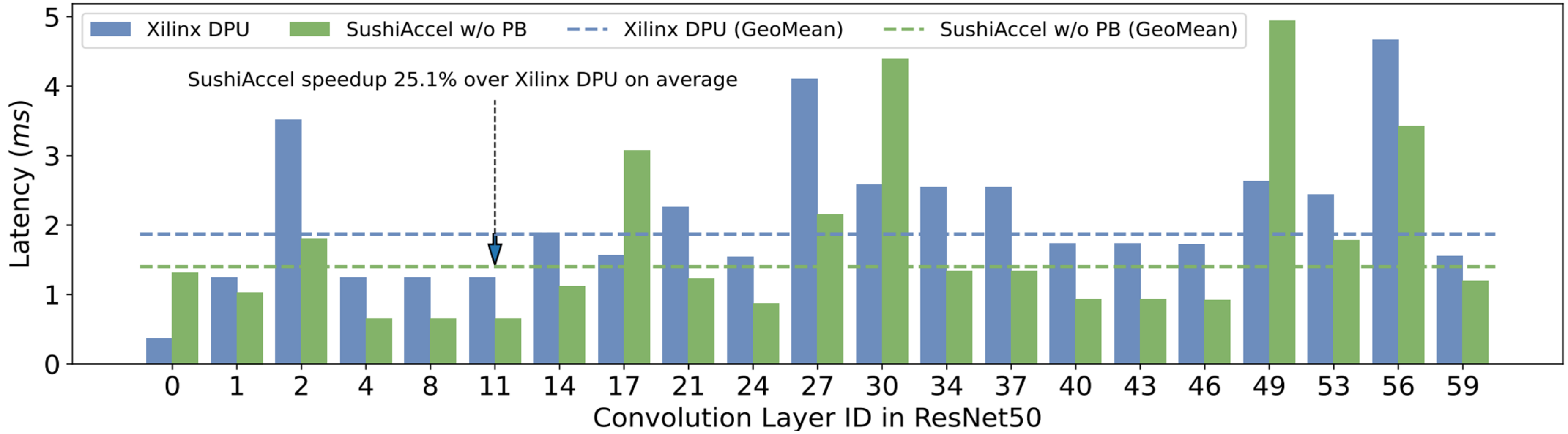
ResNet50



MobV3

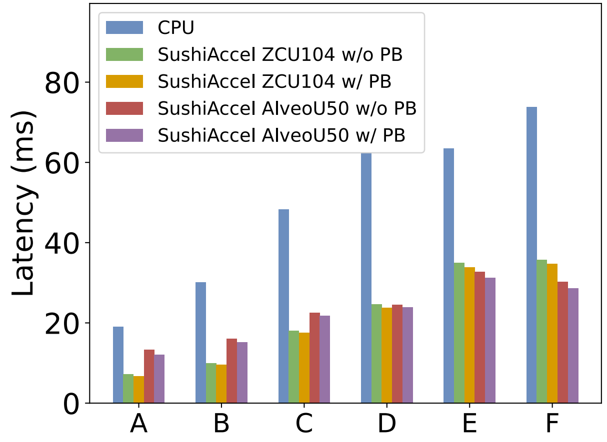
Takeaway: SUSHI eliminates extra off-chip access latency.

Hardware Evaluation on Real Board (Single Query)

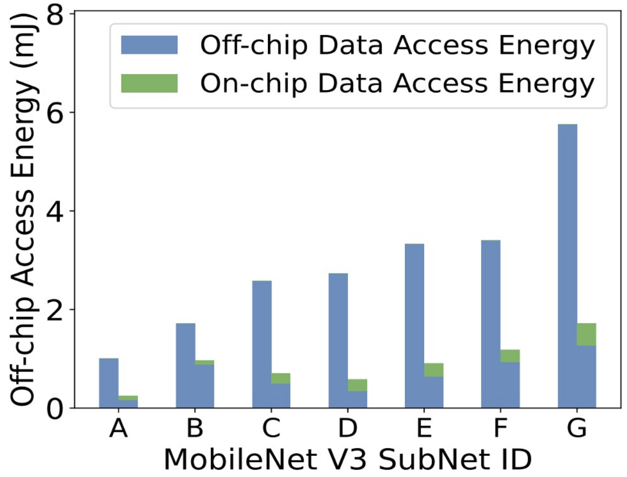


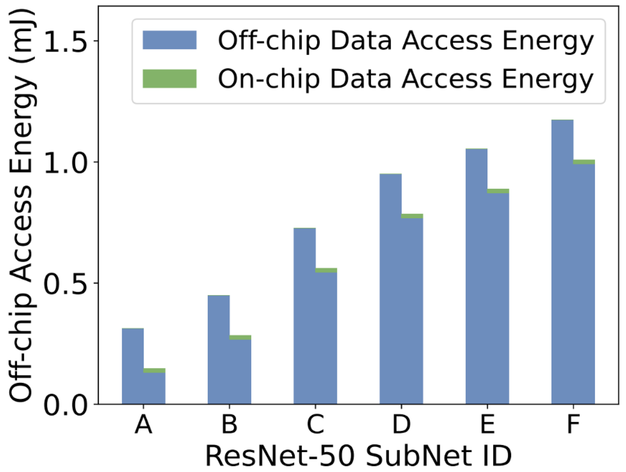
Comparison of SushiAccel w/o PB with SOTA Xilinx DPU (ZCU104)

Takeaway : SUSHI has an efficient dataflow (25% improvement over Xilinx DPU).



ResNet 50 SubNets Evaluation





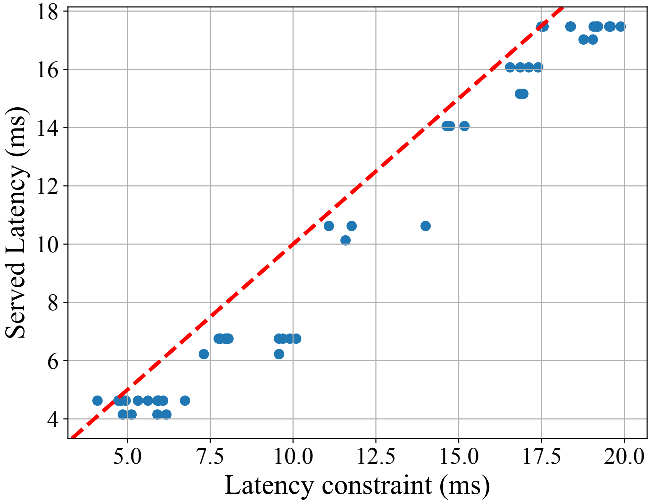
Off-chip DRAM Access Energy Saving [left: SushiAccel w/o PB; right: SushiAccel] (Alveo U50)

Takeaway: SUSHI further improves latency (additional 10%) and off-chip access energy (78%).

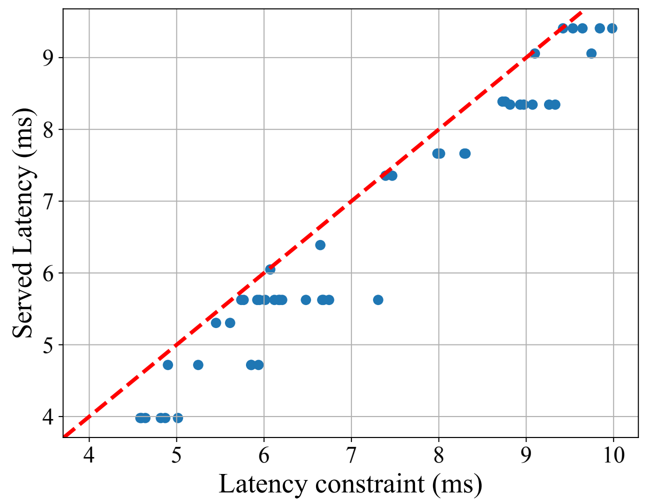
Accuracy/Latency Improvements (Stream of Queries)

SUSHI delivers lower latency while satisfying the accuracy requirements.

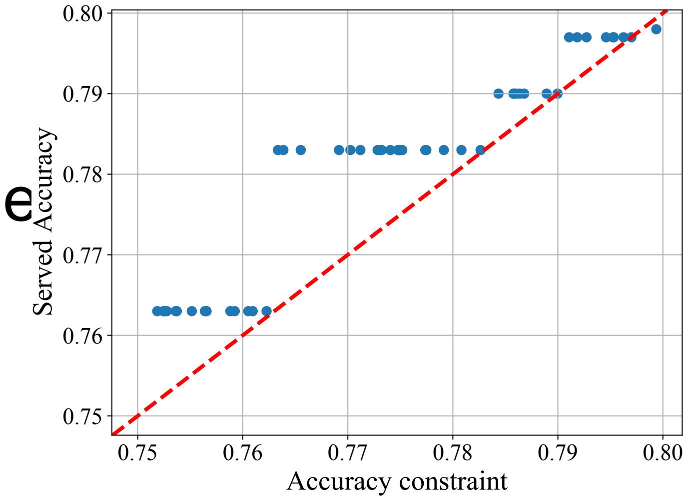
SUSHI delivers higher accuracy while satisfying the latency requirements.



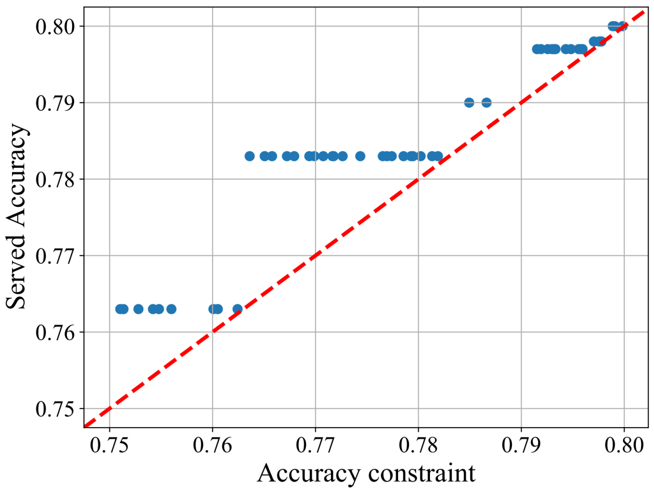
Lower Latency (ResNet50)



Lower Latency (MobV3)

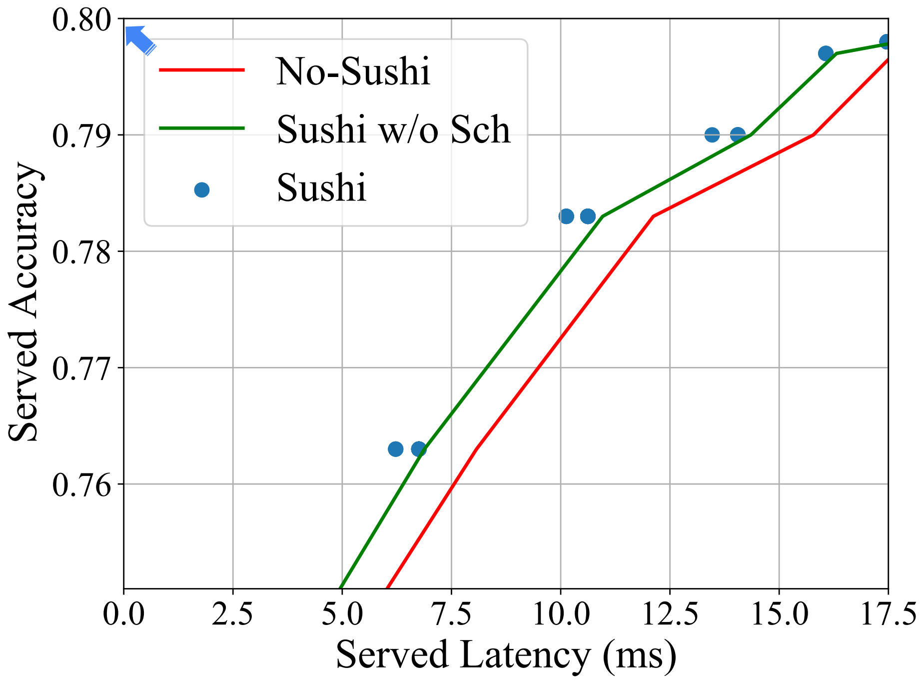


Higher Accuracy (ResNet50)

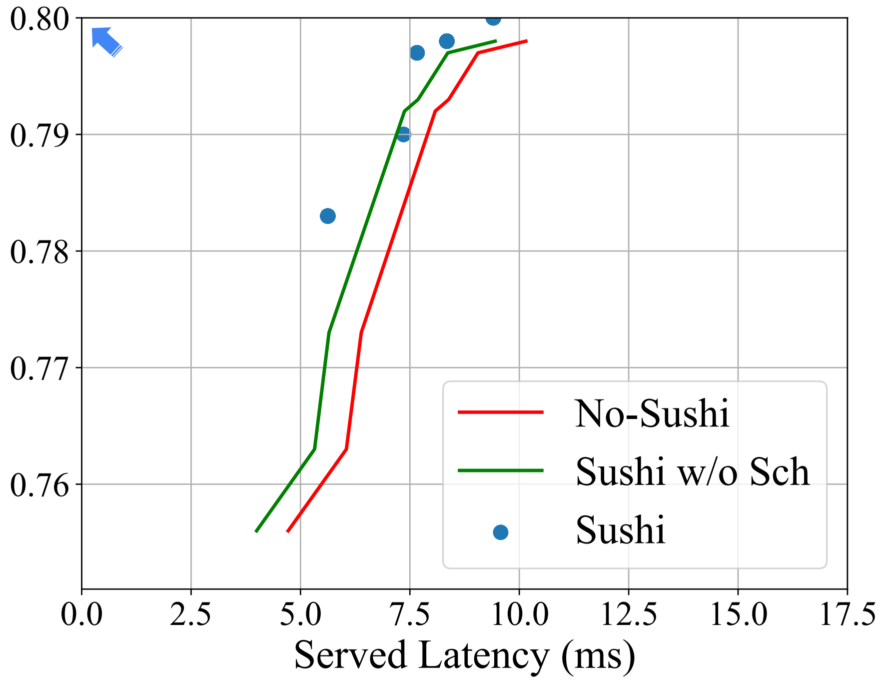


Higher Accuracy (MobV3)

Accuracy/Latency Improvements (Stream of Queries)



ResNet50



MobV3

Takeaway: SUSHI serves higher accuracy for the same latency (up to 0.98%).
Takeaway: SUSHI improves the average serving latency (25%(21%) for MobV3(ResNet 50)).

Conclusion

Problem:

- ML applications (AVs, ML4Health) must navigate latency/accuracy tradeoff space in soft-real time efficiently.

Proposed Solution:

- Vertically integrated HW-SW co-design for WS-DNN inference
- HW: SushiAccel: Switch among optimal dataflows developed for different layers.
- SW: Cross query SubGraph Stationary optimization
  - Choose SubNets /reuse SubGraph per query
- SushiAbs: Generalizable and efficient HW-SW interface
  - $(SN_i, G_j) \rightarrow Latency_{ij}$

Evidence:

- SUSHI improves average serving latency (25%(21%) for MobV3(ResNet50)).
- SUSHI increases serving accuracy for the same latency (up to 0.98%).
- SUSHI saves off-chip data access energy (up to 78%).