Distributed Deep Learning Inference On Resource-Constrained IoT Edge Clusters

Kamyar Mirzazad Barijough, Zhuoran Zhao, Andreas Gerstlauer

System-Level Architecture and Modeling (SLAM) Lab

Department of Electrical and Computer Engineering

The University of Texas at Austin

`https://slam.ece.utexas.edu`

The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering

Background

- **Internet-of-Things (IoT)**
  - Large scale data processing under real-time constraints
  - Deep learning techniques for IoT applications
    - Computationally and memory-intensive
- **Cloud-based vs. fog/edge computing**
  - Privacy
  - Unpredictable remote server and communication latency
  - Computational resources near the sources
  - Clusters of edge and gateway devices

➢ **Distributed deep learning inference in IoT edge clusters**
  - ➢ Efficient deployment on resource-constrained IoT devices
  - ➢ Provide real-time guarantees in wireless edge communication

Distributed Deep Learning on the Edge

- **Partition neural network into tasks**
  - Enable distributed execution
  - Optimize memory and communication
- **Adapt to changing computational resources**
  - Dynamic task mapping and offloading
  - Lightweight and low overhead middleware
- **Provide real-time guarantees**
  - Bound communication latency
  - Distributed real-time scheduling

Related Work

- **Cloud-assisted inference [Kang’17, Teerapittayanon’17]**
  - ➢ Unpredictable cloud status and communication latency
  - ➢ Privacy issues and scalability
- **Lightweight deep neural network models**
  - Sparsification and pruning [Bhattacharya’16, Yao’17]
  - Compression [Iandola’16, Howard’17, Zhang’17]
  - ➢ Loss of accuracy and application-/scenario-dependent
- **MoDNN [Mao’17]**
  - Static partition and local distribution on mobile devices
  - MapReduce-like programming model in mobile cluster
    - – Bulk-synchronous and lock-step fashion
    - – Layer-by-layer synchronization
  - Limitation in scalability

Outline

- ✓ **Introduction**
  - ✓ Background
  - ✓ Related work

- **Fused Tile Partitioning (FTP)**
  - Input/output partitioning
  - Layer fusion

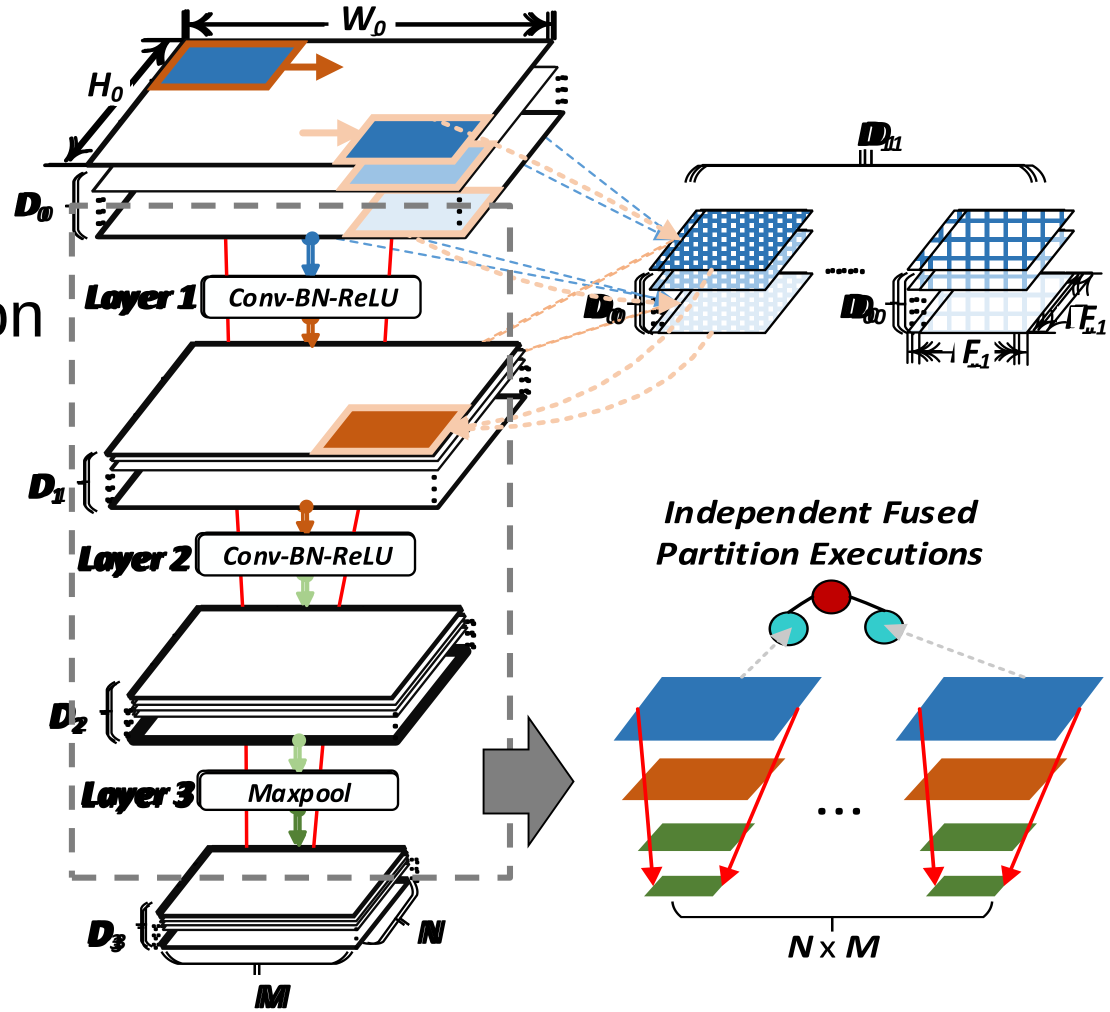
- **DeepThings middleware**
  - Distributed work stealing
  - Data reuse-aware work scheduling and distribution

- **Real-time extensions**
  - Real-time guarantees
  - Real-time scheduling

Fused Tiled Partitioning

- **Convolutional operation**
  - Local connectivity between neurons of consecutive layers
  - ➢ Grid partitioning with boundary consideration
- **Chain of multiple convolutional layers**
  - Large amount of intermediate data
  - Boundary synchronization overhead per layer
  - ➢ Layer fusion

➢ **Independent execution stacks**



Outline

- ✓ **Introduction**
  - ✓ Background
  - ✓ Related work

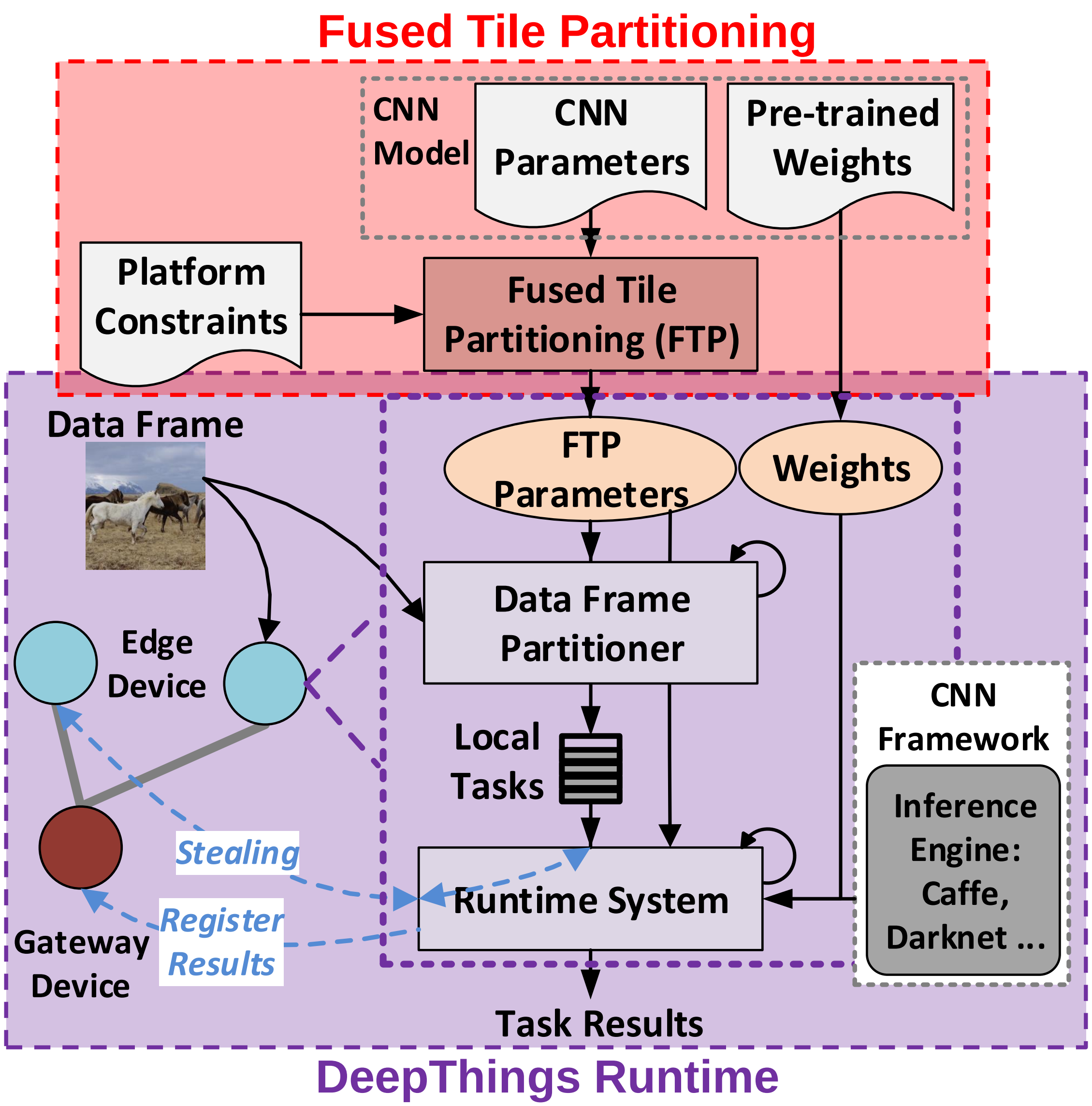
- ✓ **Fused Tile Partitioning (FTP)**
  - ✓ Input/output partitioning
  - ✓ Layer fusion

- **DeepThings middleware**
  - Distributed work stealing
  - Data reuse-aware work scheduling and distribution

- **Real-time extensions**
  - Real-time guarantees
  - Real-time scheduling

DeepThings Overview

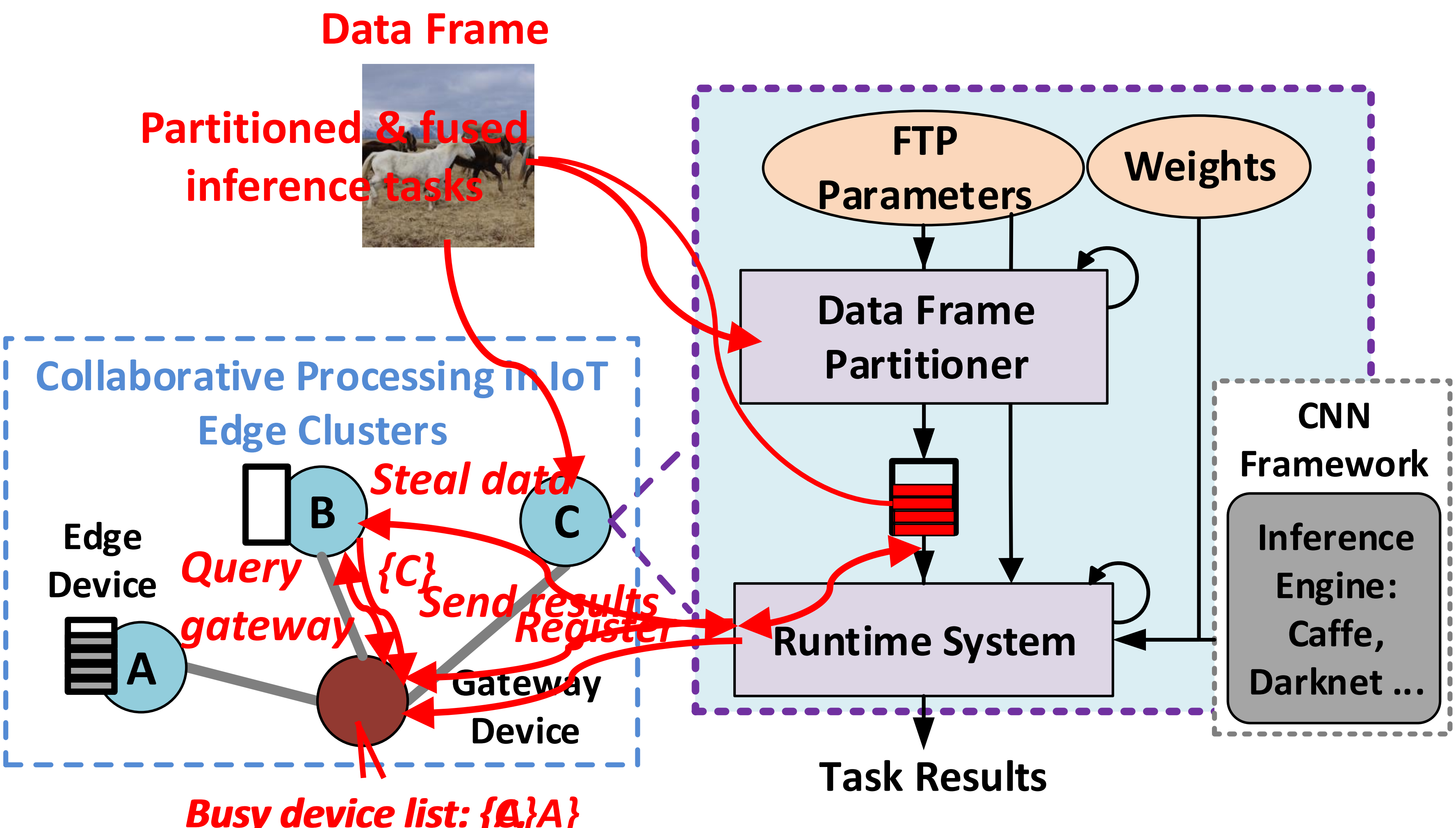
- **Generate independent tasks using FTP**
- **Runtime system**
  - Edge: peer-to-peer work stealing
  - Gateway: central coordination
  - ➢ Collaborative inference



Z. Zhao, K. Mirzazad and A. Gerstlauer, "DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters," CODES+ISSS, 2018.

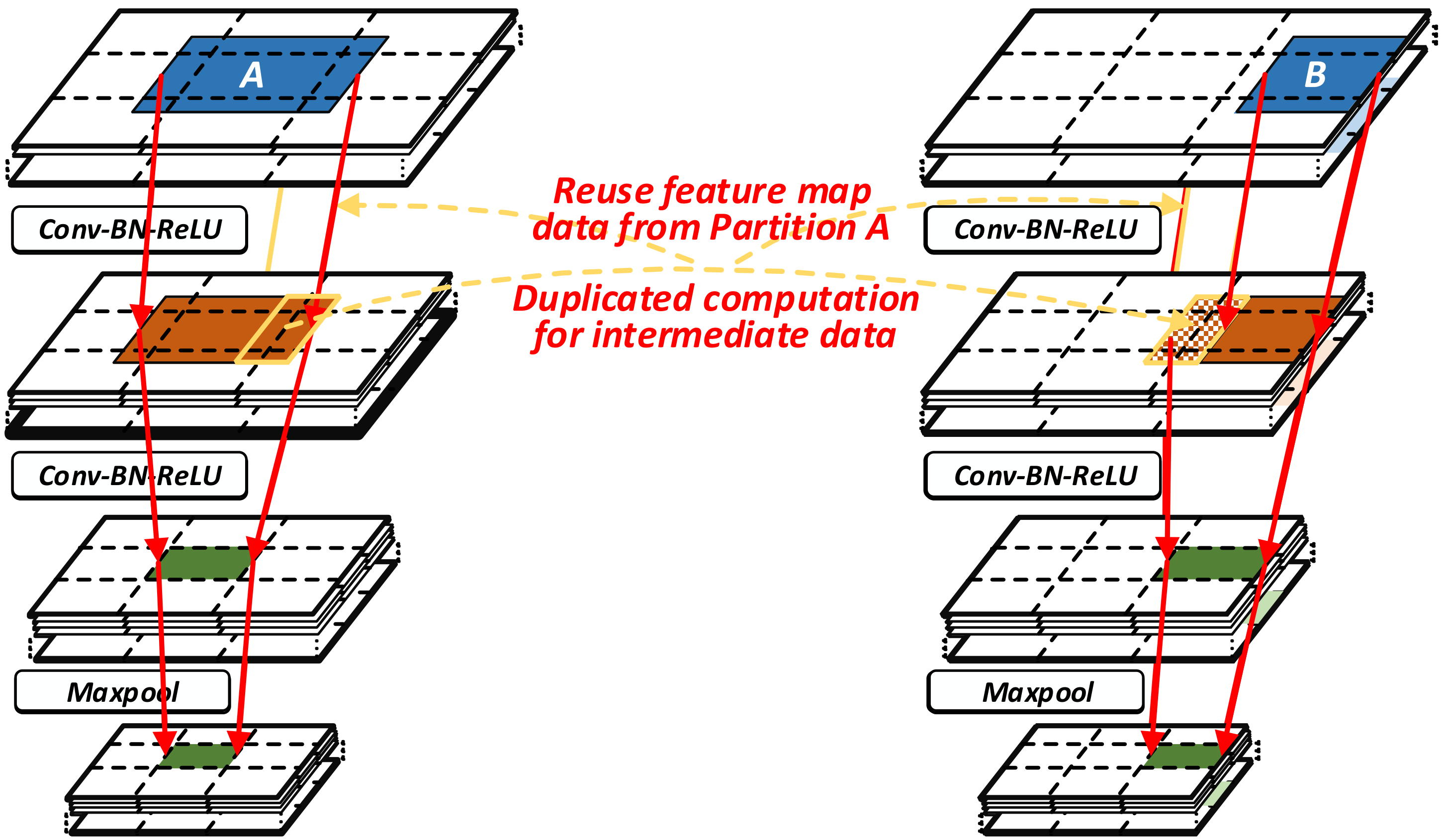
Work Stealing Approach

- **Message flow and data movement in DeepThings**
  - Peer-to-peer input data migration
  - Idle device steals from busy device



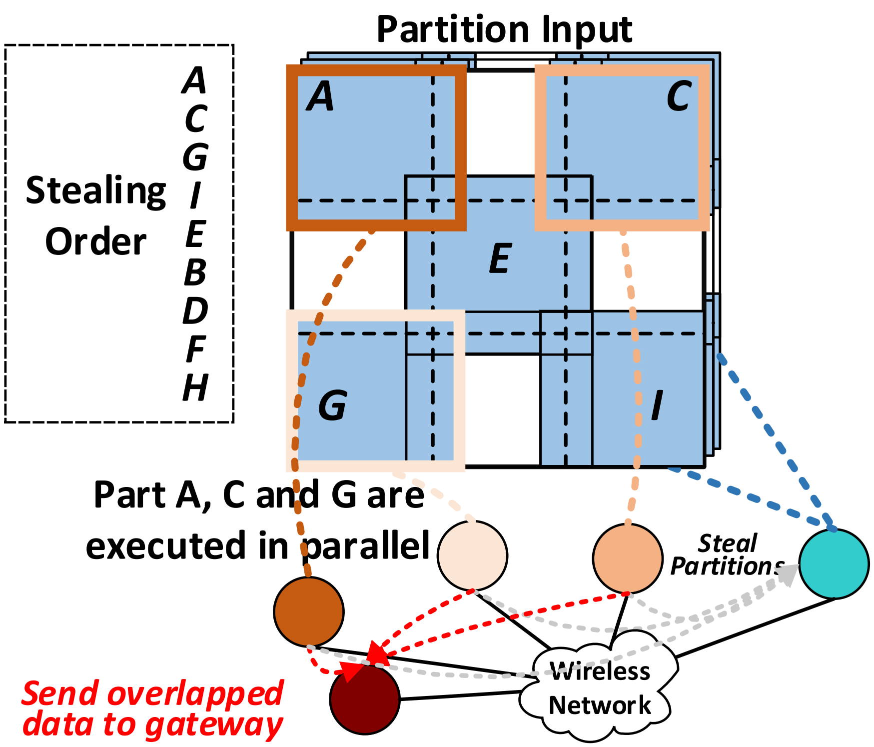
Data Reuse Opportunities

- **Redundancy in Fused Tile Partitioning**
  - Duplicate overlapped data for independent sub-tasks
  - Overlapped data amplified through many fused layers
  - ➢ Possible data reuse to reduce computation



Data Reuse-Aware Work Scheduling

- **FTP partition scheduling**
  - Minimize the partition dependency
    - Scheduling tasks to be stolen in dependency order
    - Caching overlapped reuse data in gateway



Experimental Setup (1)

- **DeepThings framework**
  - Retargetable implementation in C
  - Uses nnpack-accelerated Darknet as inference engine
  - TCP/IP socket APIs
  - Released in open-source form
    - https://github.com/SLAM-Lab/DeepThings
- **Experiment platform**
  - Up to 6 Raspberry Pi 3B connected with WiFi
- **Deep learning application**
  - You Only Look Once (YOLO) object detector
    - First 16 layers (12 convolutional and 4 maxpool layers)
    - More than 49% of computation and 86.6% of memory footprint
  - Multiple data sources
    - Emulate dynamic application scenarios

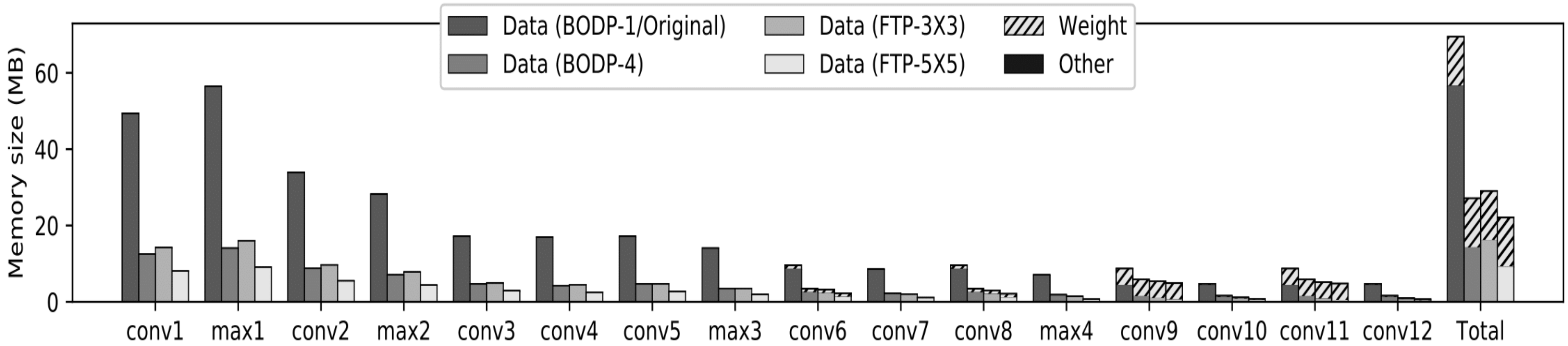
Experimental Setup (2)

- **DeepThings vs. MoDNN**
  - Work Sharing (WSH): Central data collection/coordination
  - Work Stealing (WST): Peer-to-peer data transmission
  - Data partitioning & synchronization
    - DeepThings (FTP): Overlapped data is duplicated/transmitted at input
    - MoDNN (BODP): Overlapped data is synchronized after every layer.

	DeepThings	**MoDNN**
Partition Method	Fused Tile Partitioning (FTP)	Biased One-Dimensional Partition (BODP)
Partition Dimensions	3x3 ~ 5x5	1x1 ~ 1x6
Distribution Method	Work Stealing (WST) Work Sharing (WSH)	Work Sharing (WSH)
Edge Node Number	1 ~ 6	

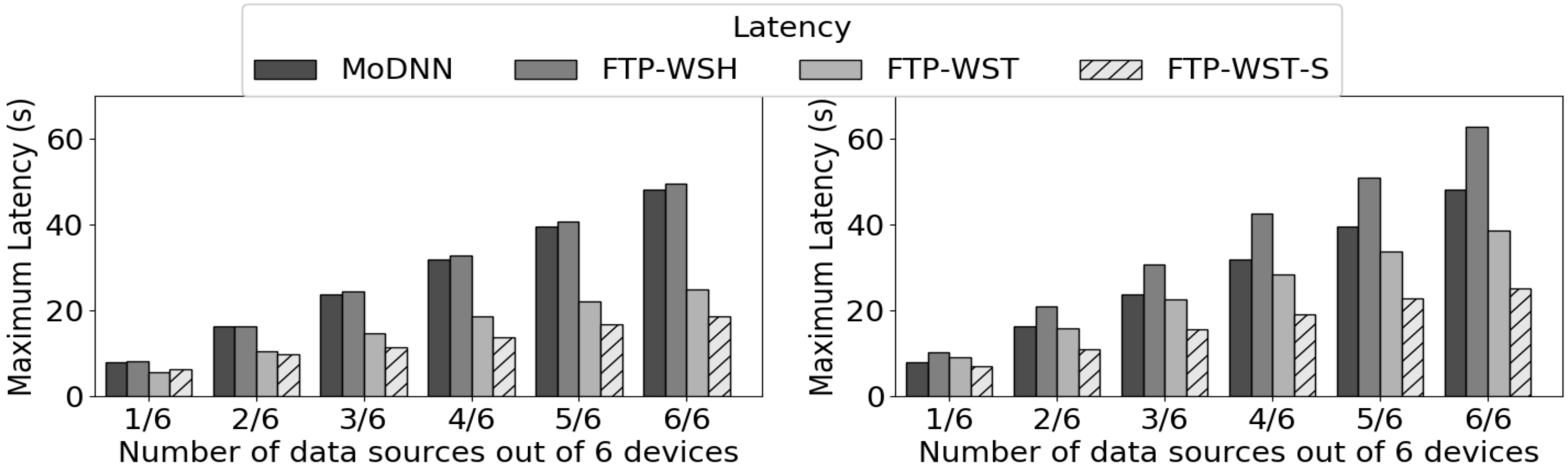
Memory Footprint

- **Per device memory footprints of each layer**
  - Memory reduction
    - Input/output feature map data is partitioned to save memory
    - Weight data is not partitioned and remains the same
  - Maximum memory usage reduction
    - 61% in 4-way BODP, 58% and 68% for FTP 3x3 and 5x5
  - Average memory footprint reduction per layer
    - 67% in 4-way BODP, 69% and 79% for FTP 3x3 and 5x5



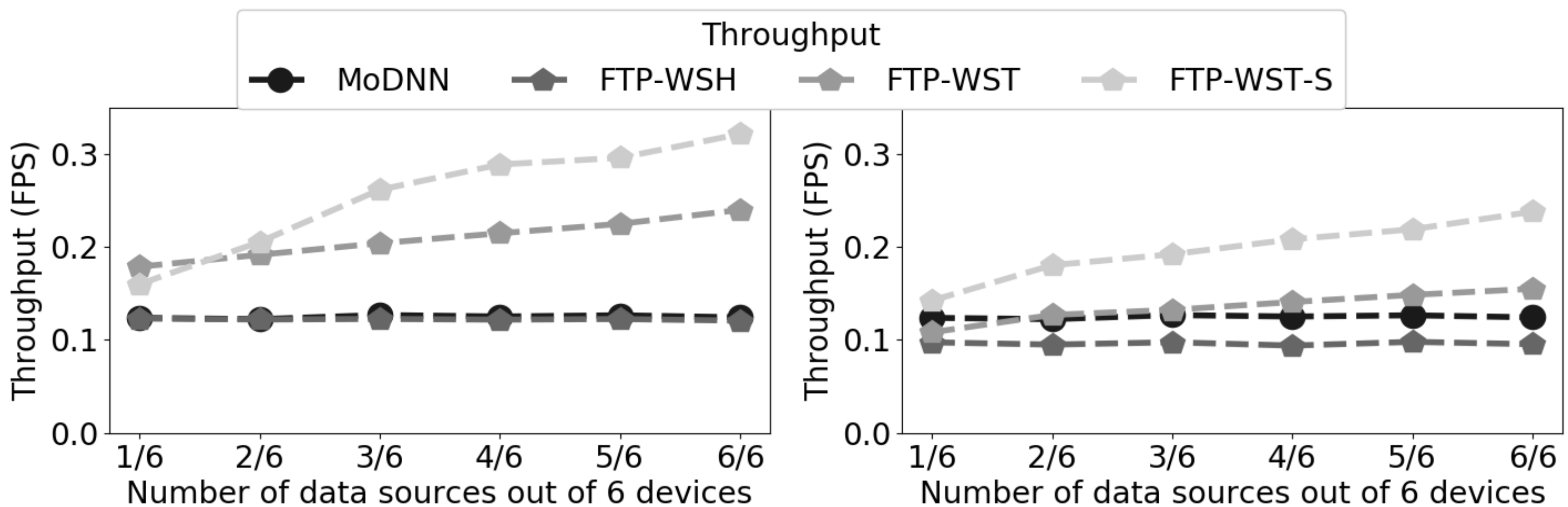
Multiple data sources

- **Maximum Latency**



FTP-3x3 **FTP-5x5**

- **Throughput**



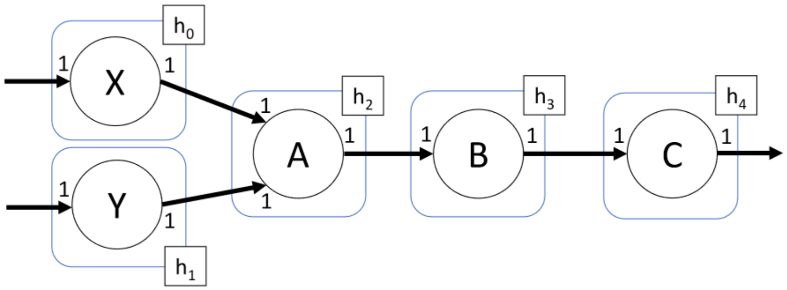
FTP-3x3 **FTP-5x5**

Outline

- ✓ **Introduction**
  - ✓ Background
  - ✓ Related work
- ✓ **Fused Tile Partitioning (FTP)**
  - ✓ Input/output partitioning
  - ✓ Layer fusion
- ✓ **DeepThings middleware**
  - ✓ Distributed work stealing
  - ✓ Data reuse-aware work scheduling and distribution
- **Real-time extensions**
  - Real-time guarantees
  - Real-time scheduling

Real-time Guarantees

- **No latency guarantee in open networks**
  - Network delay distribution
  - Well-known problem in VoIP, live streaming
- **Bound communication latency by enforcing timeouts**
  - Discard late packets
  - Smaller the timeout, more data losses
  - Trade-off between latency (timeout) and quality (losses)
- **Assign timeout to every network communication**
  - Easy to derive from the total latency in two-node system
  - Not trivial for larger systems
  - Which losses are more important?

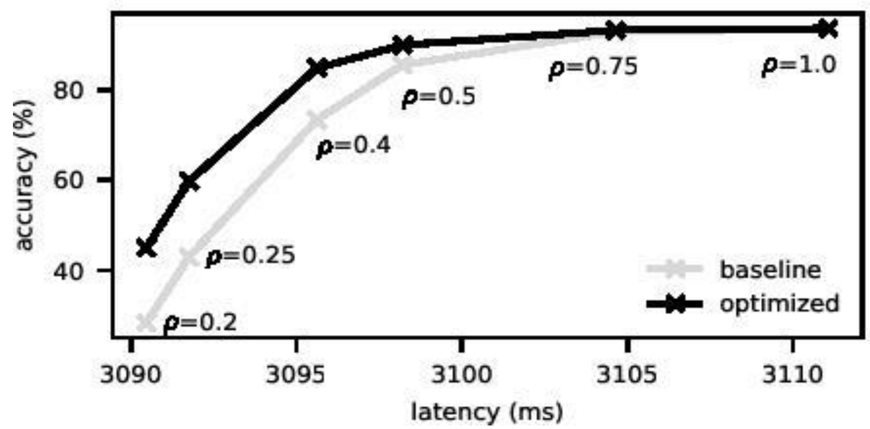


Real-time Scheduling

- **Differentiate between losses of nodes**
  - Represent the system as a dataflow with lossy communication
  - ➢ Quality model for lossy dataflow's schedule
    - – Assumptions: linearity, SNR metric

- **Formulate scheduling as an optimization problem**
  - Find schedule to maximize quality and satisfy latency constraint
  - ➢ Quality/latency-aware scheduling

- **Improved trade-off between quality and latency**
  - Baseline: Uniform distribution of latency budget
  - ➢ Explored for a two-layer digit classification neural net with given mapping



K. Mirzazad, Z. Zhao and A. Gerstlauer, "Quality/Latency-Aware Real-time Scheduling of Distributed Streaming IoT Applications," CODES+ISSS, 2019.

Summary & Conclusions

- **Fused Tile Partitioning (FTP)**
  - Scalable and flexible partitioning method
  - Lightweight data synchronization
  - Independently distributable tasks
- **DeepThings middleware**
  - Distributed work stealing
  - Data reuse-aware work scheduling
  - Open-source framework in C code
- **Real-time extensions**
  - Provide latency guarantees via timeouts
  - Quality/latency-aware scheduling
  - To be open-sourced soon!

Q & A

Thank you!

Any Questions?

References:

[1] Z. Zhao, K. Mirzazad and A. Gerstlauer, "DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters," CODES+ISSS, 2018.

[2] K. Mirzazad, Z. Zhao and A. Gerstlauer, "Quality/Latency-Aware Real-time Scheduling of Distributed Streaming IoT Applications," CODES+ISSS, 2019.

[3] DeepThings on Github, https://github.com/SLAM-Lab/DeepThings

[4] https://slam.ece.utexas.edu/projects/NoS.html