DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters

TEXAS The University of Texas at Austin

Zhuoran Zhao, Kamyar Mirzazad and Andreas Gerstlauer

Electrical and Computer Engineering

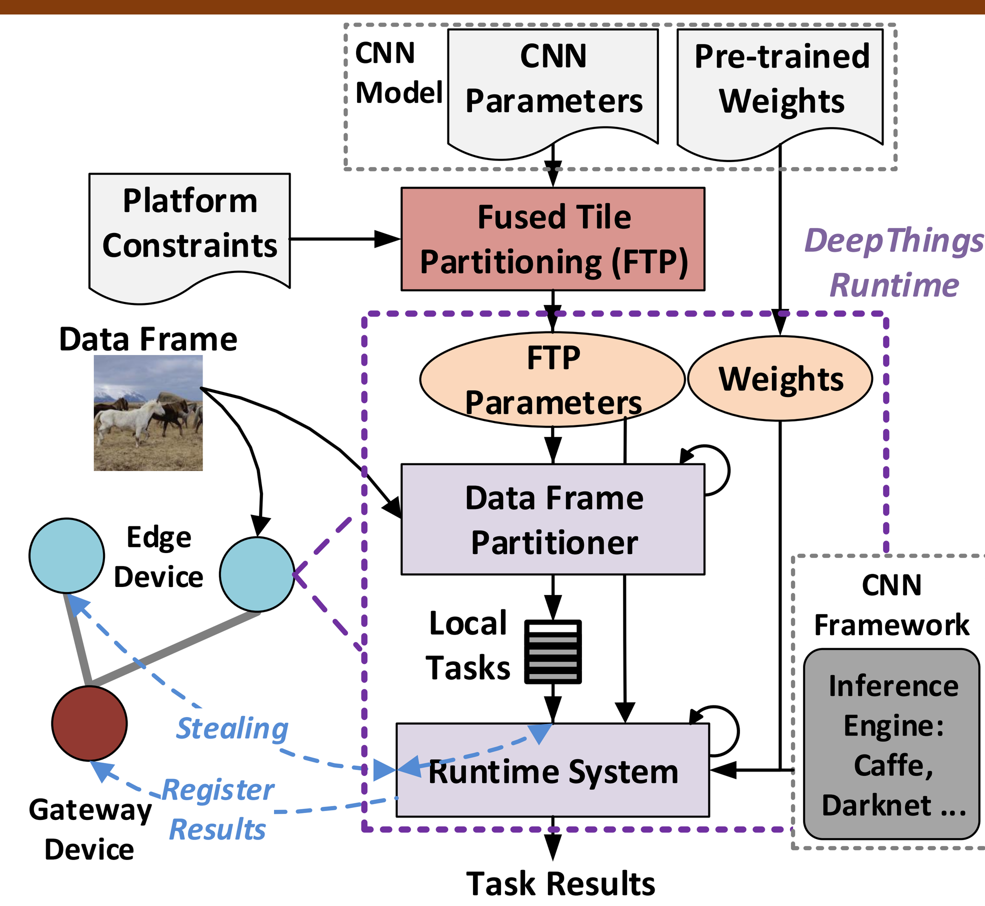
The University of Texas at Austin

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

Overview

Background and Motivation

- Internet-of-Things (IoT)
  - Complicated and noisy sensing scenarios
  - Large scale data processing & analytics
  - ➢ Deep learning (DL) techniques for IoT applications
    - Computational and memory-intensive
- Cloud-based vs. fog/edge computing
  - Privacy
  - Unpredictable remote server and communication latency
  - Computational resources near the sources
    - Edge and gateway devices
- ➢ Deep learning inference in IoT edge clusters
  - ➢ Efficient deployment on resource-constrained IoT devices



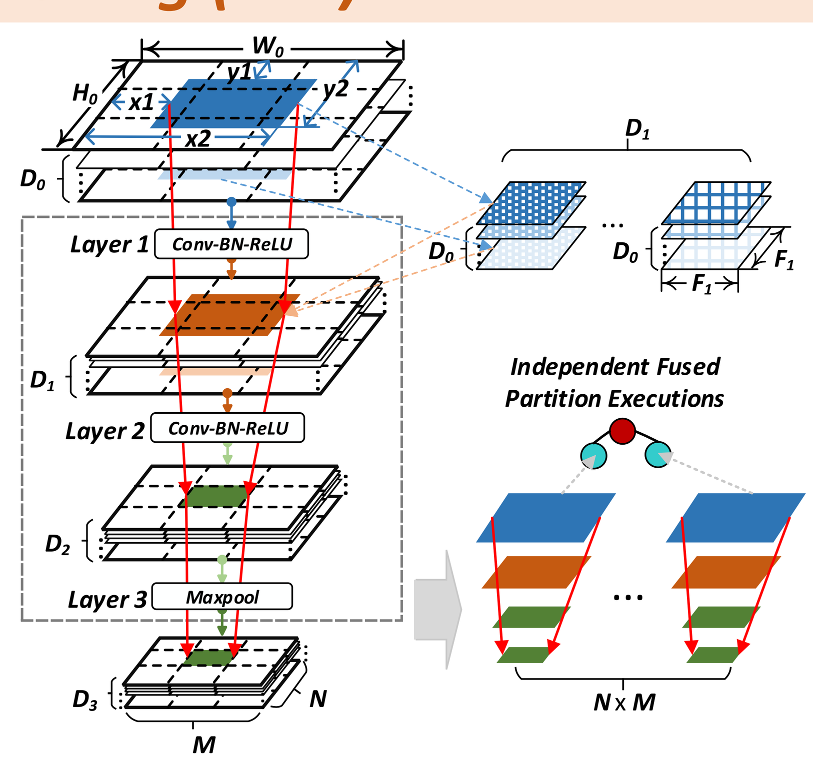
DeepThings Overview

- Fused Tile Partitioning (FTP)
  - Input data tiling
  - Layer fusion
  - ➢ Distributable tasks with lightweight data synchronization
- Distributed work stealing runtime system
  - Gateway: central coordination
  - Edge: peer-to-peer work stealing
  - ➢ Collaborative inference
- ➢ DeepThings: distributed adaptive deep learning inference on resource-constrained IoT edge clusters

DeepThings Framework

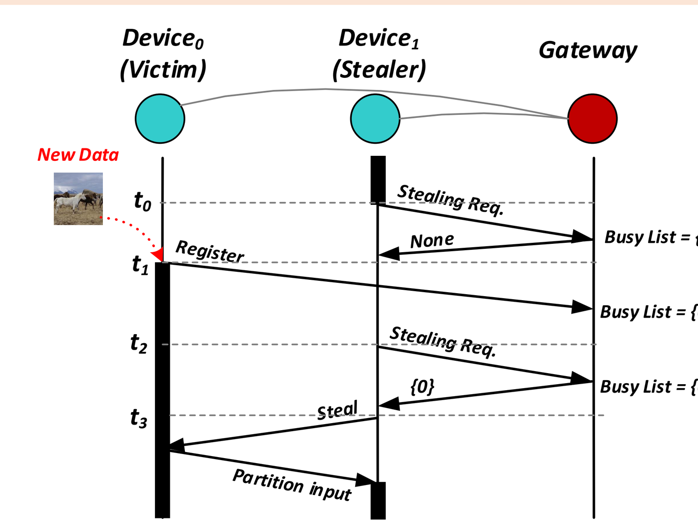
Fused Tile Partitioning (FTP)

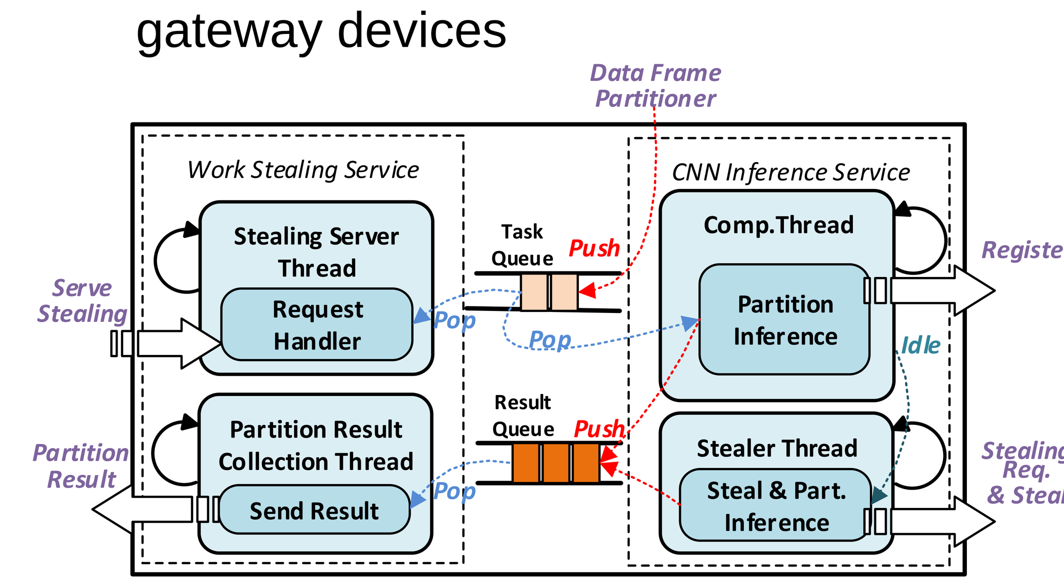
- Convolutional operation
  - Local connectivity between neurons of consecutive layers
  - ➢ Grid partitioning with boundary consideration
- Chain of multiple convolutional layers
  - Large amount of intermediate data
  - Boundary synchronization overhead per layer
  - ➢ Layer fusion
- ➢ Independent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



Distributed Work Stealing

- Distributed work stealing in local wireless area network
  - Source data partitioning & inference task generation
  - Peer-to-peer data & task migration
  - Gateway-managed load balancing
  - Local inference using external library
- Message flow between edge and gateway devices

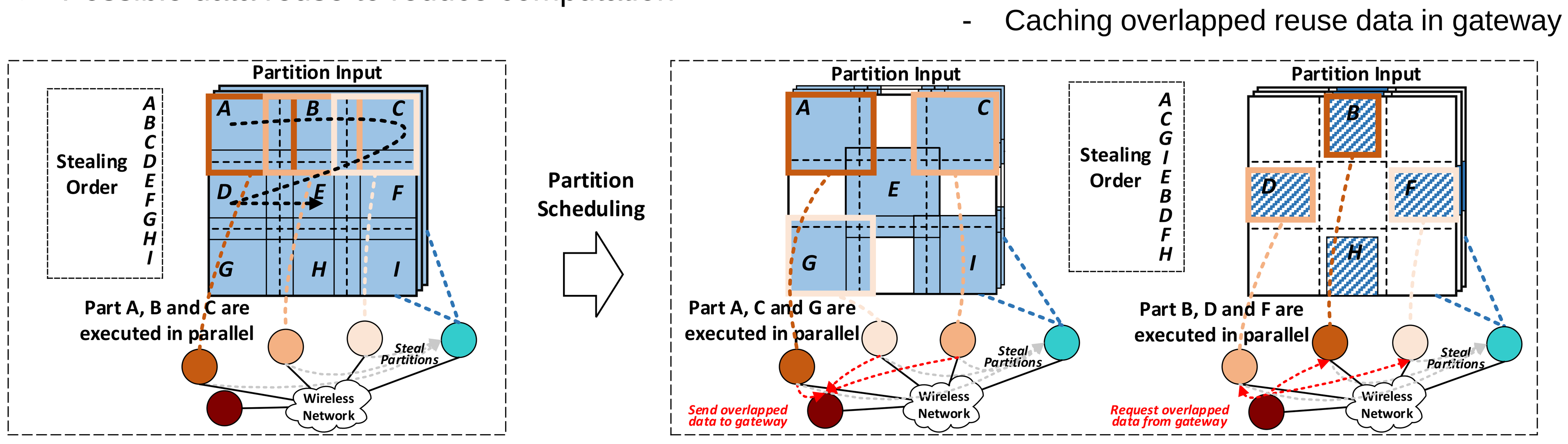




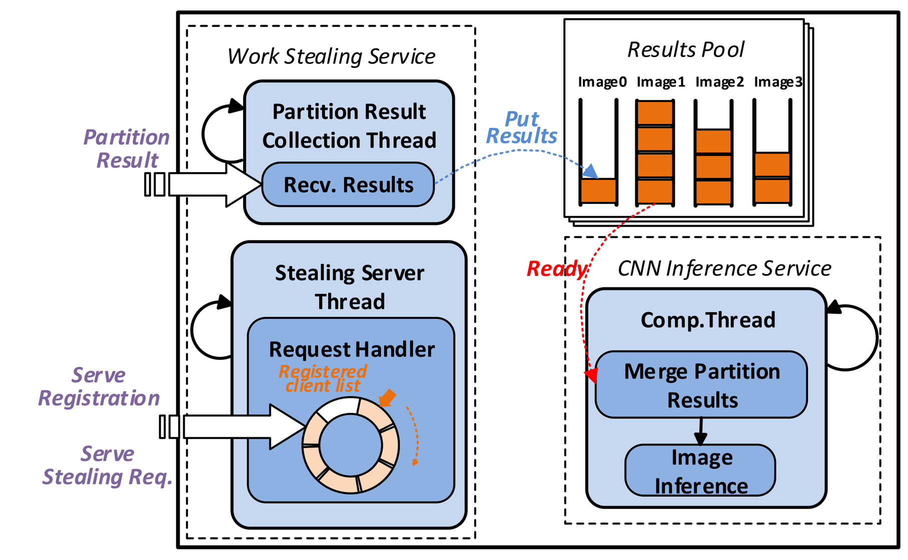
- Edge Node Runtime
  - Work stealing service
    - Serve work stealing
    - Send results to gateway
  - CNN inference service
    - Partition data frames
    - Process partitioned data
    - Perform work stealing

Data Reuse-Aware Work Scheduling

- Redundancy in Fused Tile Partitioning
  - Duplicated overlapped data for independent sub-tasks
  - Overlapped data amplified through many fused layers
  - ➢ Possible data reuse to reduce computation
- FTP partition scheduling
  - Minimize the partition dependency
    - Scheduling tasks to be stolen in dependency order
    - Caching overlapped reuse data in gateway



- Gateway Runtime
  - Work stealing service
    - Round-robin scheduling
    - Collect results from edge nodes
  - CNN inference service
    - Merge FTP partition results
    - Process remaining layers



Experimental Results

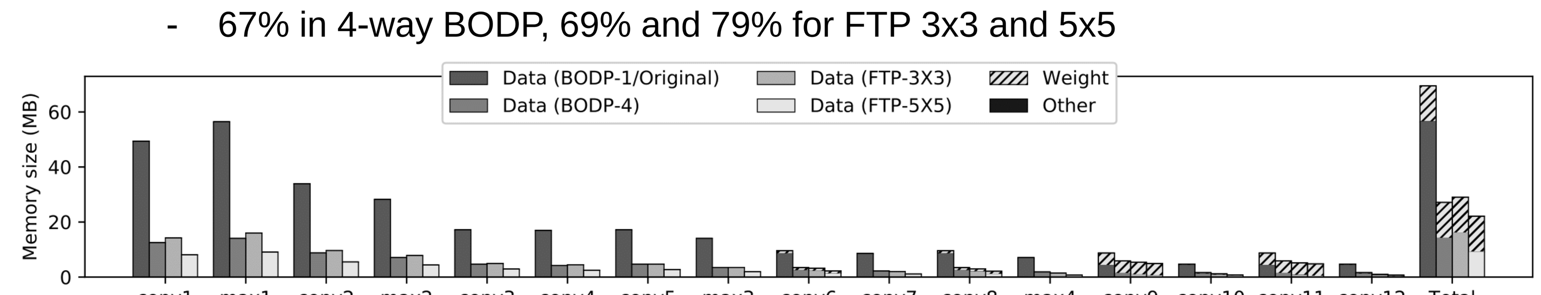
Experimental Setup

- Open-source framework
  - Retargetable implementation in C
  - TCP/IP socket APIs
  - Released in open-source form
- Experiment platform
  - Raspberry Pi 3 Model B
  - Up to 6 nodes in WLAN over WiFi
- Deep leaning application
  - You Only Look Once object detector
  - First 16 layers
  - Multiple data sources
- Experimental parameters

	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	

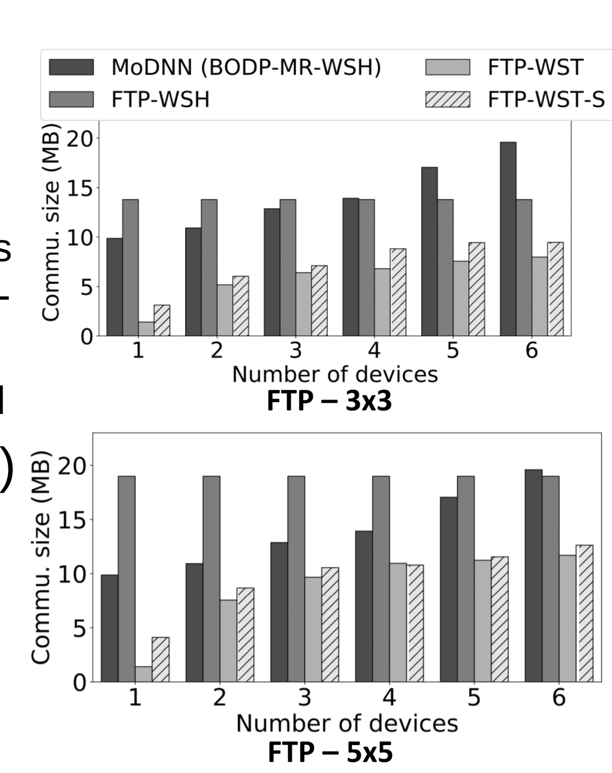
Memory Footprint

- Per device memory footprints of each layer
  - 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



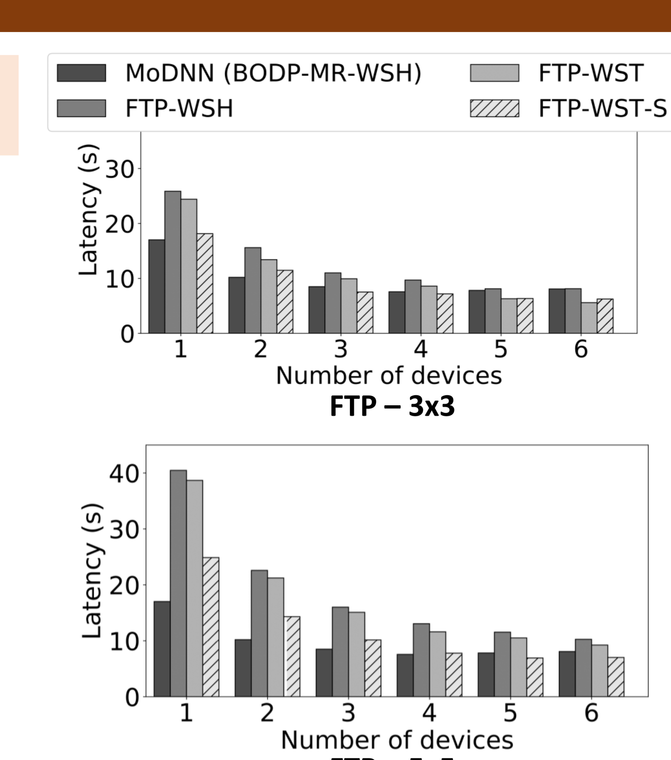
Communication Overhead

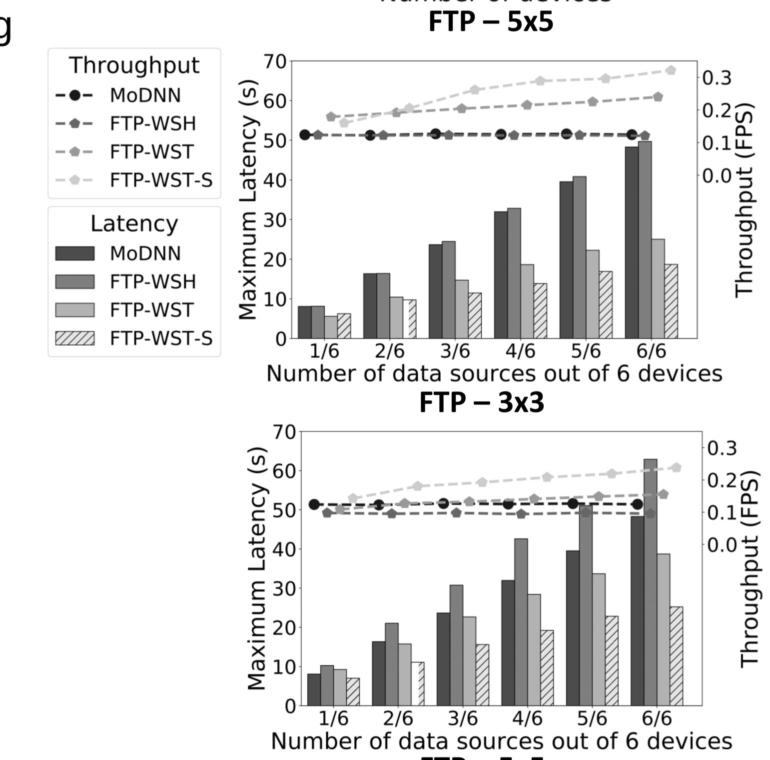
- Work sharing (WSH)
  - MoDNN: Communication overhead increases linearly with device number because of layer-based data exchange
  - FTP-WSH: Communication overhead is fixed
- Work stealing with scheduling (WST-S)
  - An average of 52% reduction comparing with WSH
  - Overhead in data reuse, amortized by smaller input data



Latency & Throughput

- Single data source
  - 6.8s with 3.5x speedup in FTP-WST-S, 6-device network
  - 8.1s with 2.1x speedup MoDNN, 6-device network
  - Scalability benefits in DeepThings
    - FTP: Avoid intensive intermediate data exchange
    - WST: Adaptively use communication bandwidth and exploit communication overhead
  - Data-reuse aware scheduling reduces 27% latency
- Multiple data sources
  - Maximum latency
    - MoDNN: proportional with number of sources
    - FTP-WST-S: 3.1x with data source(s) increasing from 1 to 6
  - Throughput
    - MoDNN: 0.12 FPS with 6 data sources
    - FTP-WST-S: 0.29 FPS with 6 data sources





Supported by National Science Foundation (NSF) Grant CNS-1421642