Microsoft

Flexible High-resolution Object Detection on Edge Devices with Tunable Latency

Shiqi Jiang, Zhiqi Lin, Yuanchun Li, Yuanchao Shu and Yunxin Liu

2022.2

Video Analytics on Edge Era

- Video analytics at scale
  - E.g., Public safety, urban planning, business
  - Advances in DL and hardware

- Call for Edge Intelligence
  - E.g., smart cameras,
  - Easier deployment, lower price, higher availability, better privacy

The Gap

Resource constrained edge devices

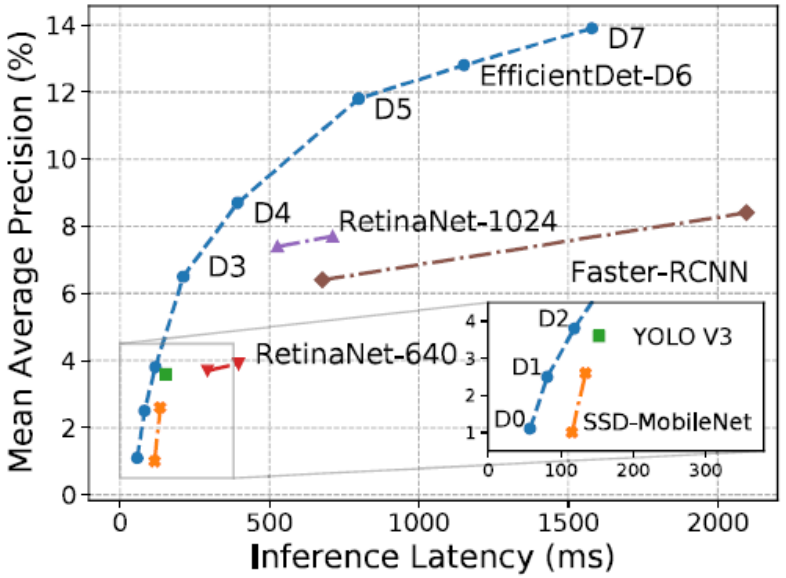
Compute intensive workload

High resolution cameras, i.e., 4K & 8K

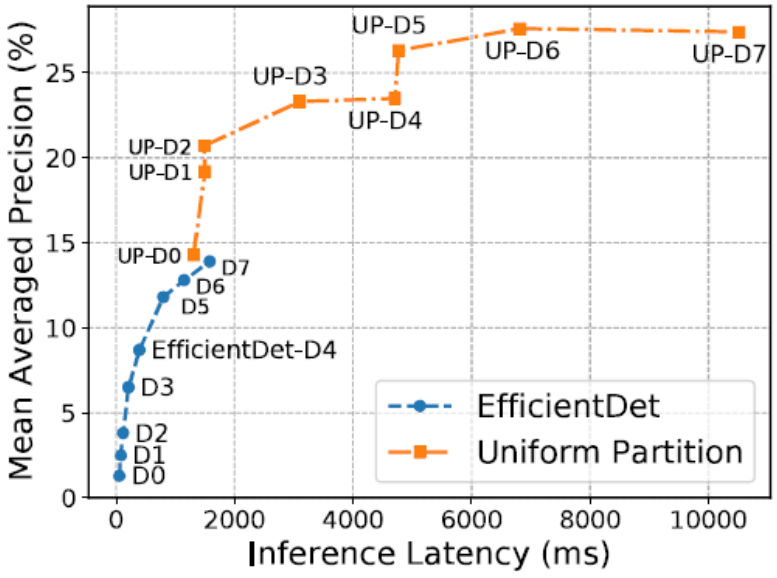
A Running Example - High-Res. Object Detection

- 4K Cam.
- Multiple platforms
  - Nvidia Jetson, Kirin, Snapdragon, etc.
- Improved performance
  - Compared to SOTA DNN models (1-7s)
  - Up to 8.1X speedups
  - 65.3% accuracy improvement on average
- Tunable latency
  - Achieve best accuracy within a configured latency budget

Measurements on Existing Art



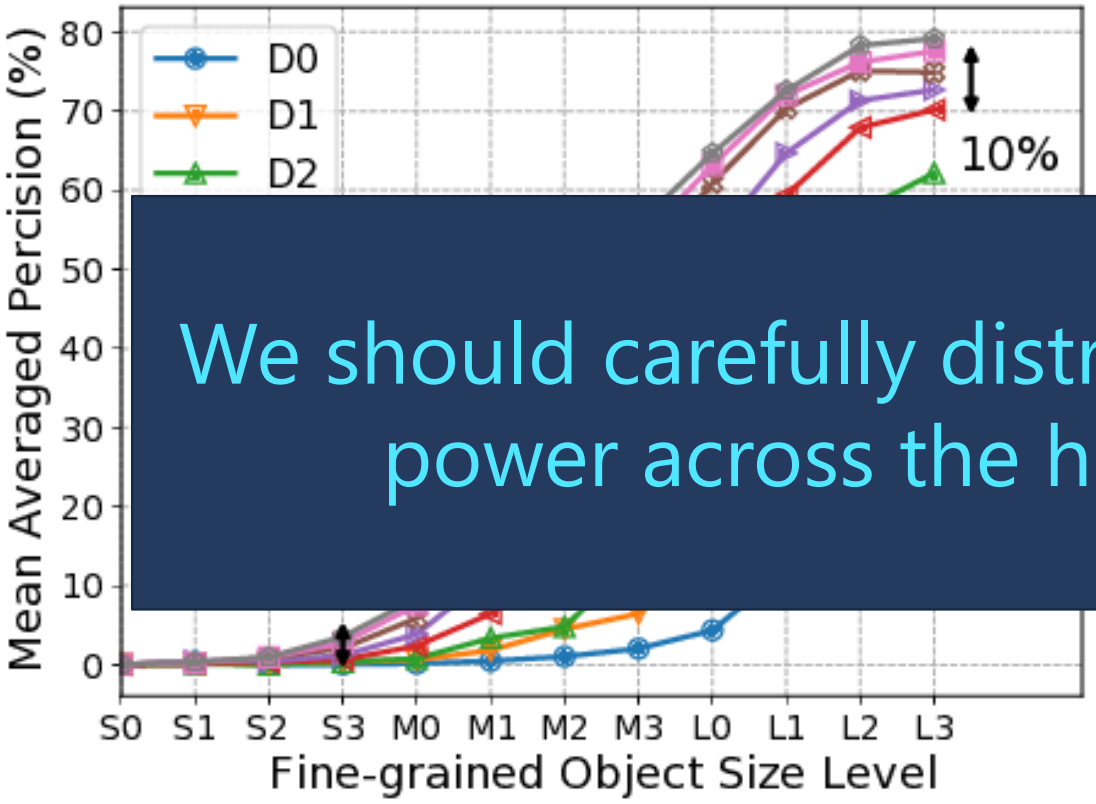
mAP and GPU inference latency of selected NNs on PANDA 4K dataset with Jetson



mAP and inference latency of uniform partitioning on PANDA 4K dataset with Jetson

- NNs are designed for low-resolution
  - E.g., 416x416 for YOLOv3
- Down-sampling inputs
  - Low accuracy
  - Only 0.8% for SSD-MobileNet
- Up-scaling NNs
  - High latency
  - High-cost low-gain
- Uniform partitioning
  - Better accuracy
  - Higher latency

Key Observations



mAP of EfficientDet NNs on objects with different sizes

- EfficientDet NNs
  - D0 (small) - D7 (large)

arge(L0-L3)

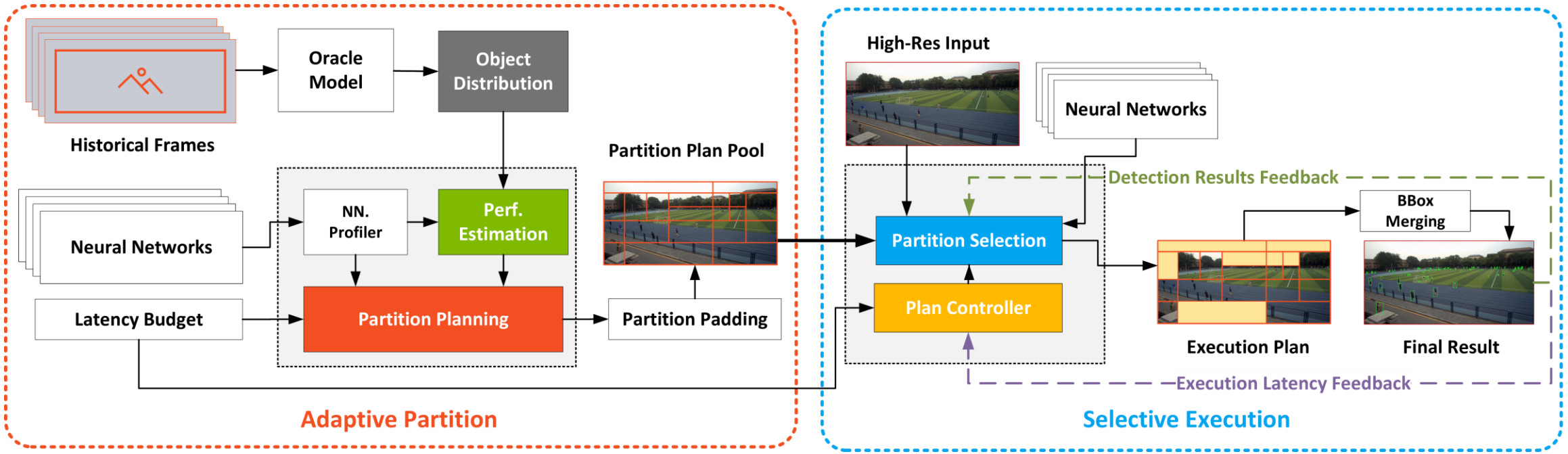
- D6 v.s. D3, ~6X higher latency
- On M2, 160% acc. improvement
- On L3, 10% acc. improvement

We should carefully distribute the limited compute power across the high-resolution image

Our Idea

- Do not design another NN, but
  - Explore existing diverse models
- Do not distribute equal compute power to every pixel, but
  - Prioritize blocks and distribute adaptively
  - By assigning a proper NN
- Tunable performance
  - Achieve best accuracy within (any) latency budgets on (any) hardware

Remix System Overview

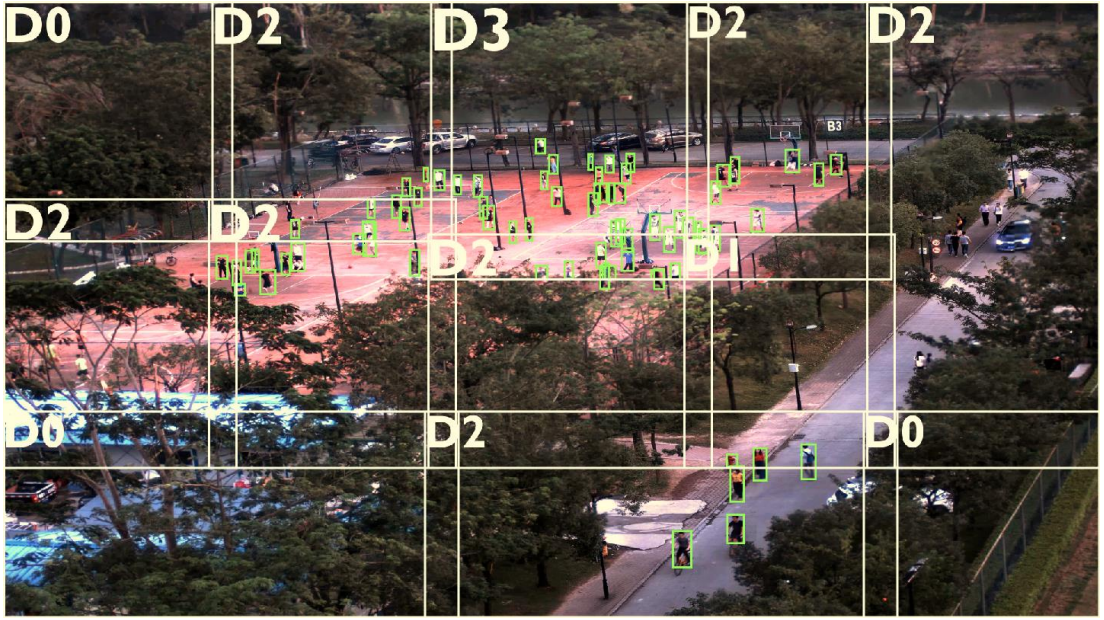


Offline, generate the nonuniform partition plan, considering long-term spatial characteristics

Online, determine which blocks can be skipped, considering short-term temporal dynamics

Adaptive Partition

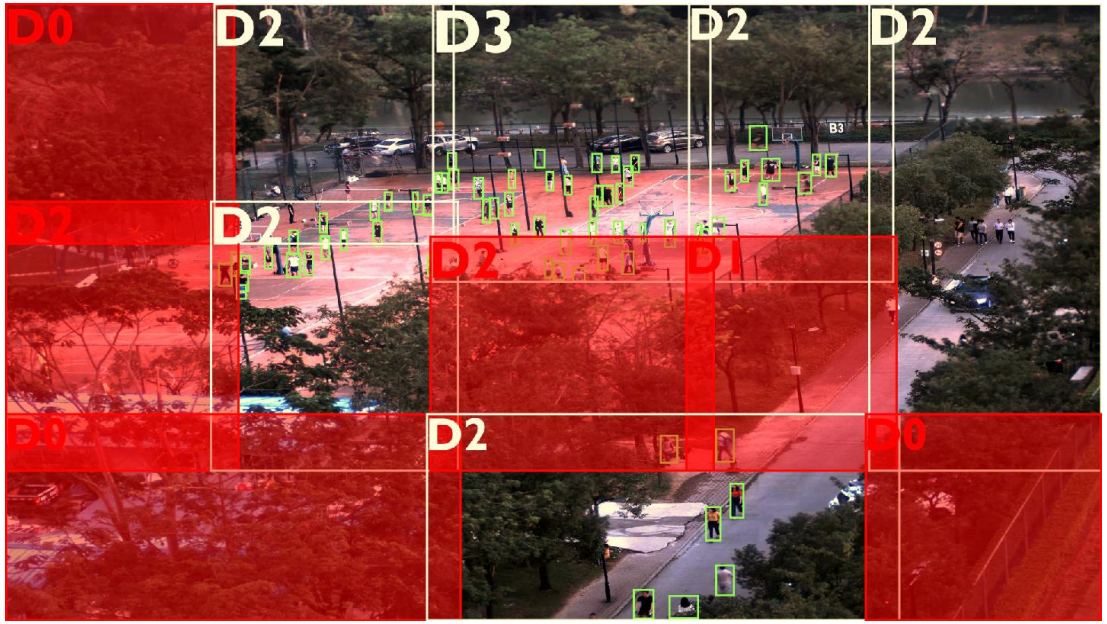
- Input:
  - A set of diverse NNs
  - A few (5-30) historical frames
  - A latency budget
- Output
  - Partition Plans
- Problems and submodules
  1. Profile NNs -> NN Profiler
  2. Estimate the accuracy and latency of candidates plans -> Perf. Estimation
  3. Enumerate all possible plans -> Partition Planning
  4. How to avoid objects being cut-off -> AC-Pad



Example of generated partition plan adaptively

Selective Execution

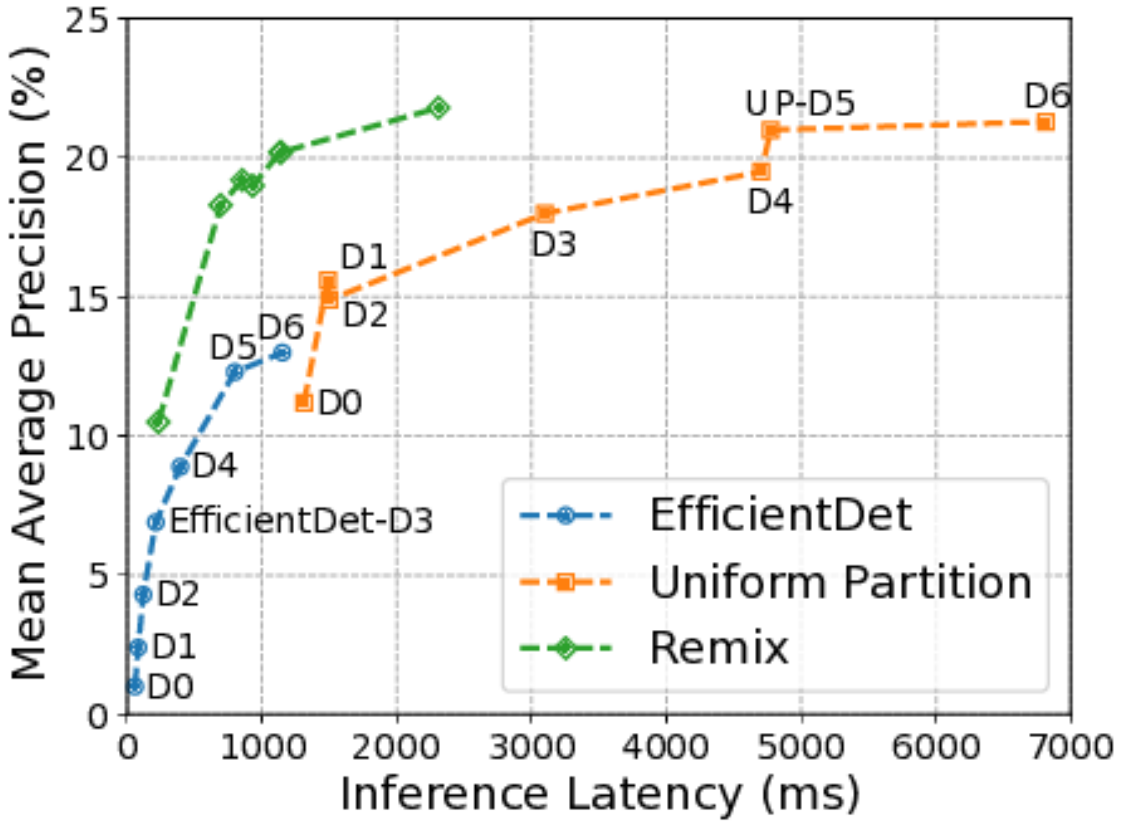
- Input:
  - Generated partition plans
  - The set of NNs
  - Live stream of 4K videos
- Output:
  - Detection results
- Problems and submodules
  - How to determine skipped blocks -> detection results feedback loop
  - How to best utilize edge resources -> execution latency feedback loop



Example of executed partition blocks selectively

A Live Demo

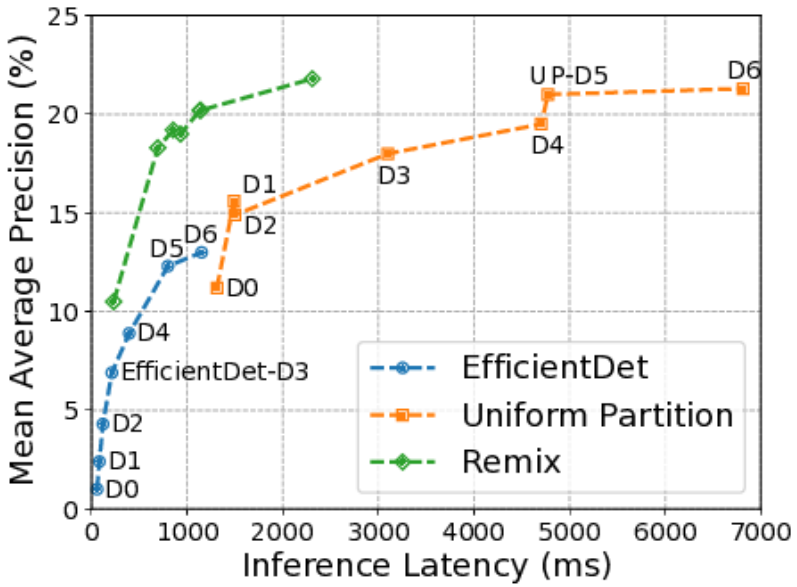
Evaluation - Faster Inference and Higher Precision



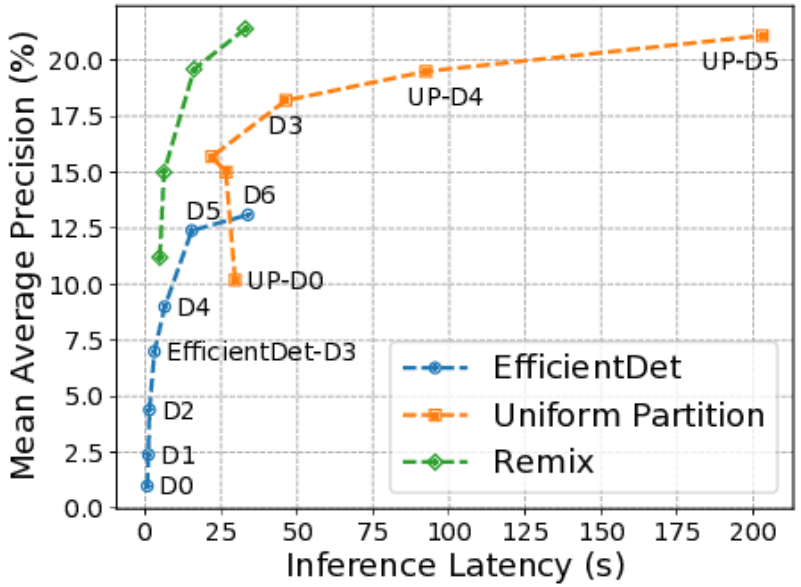
- 1.7x -8.1x speedups
  - With only accuracy drop of < 0.2%
  - Comparing to SOTA NNs
- 65.3% accuracy improvements
  - Under the same latency constraints
  - Comparing with SOTA NNs

Performance across Different Hardware

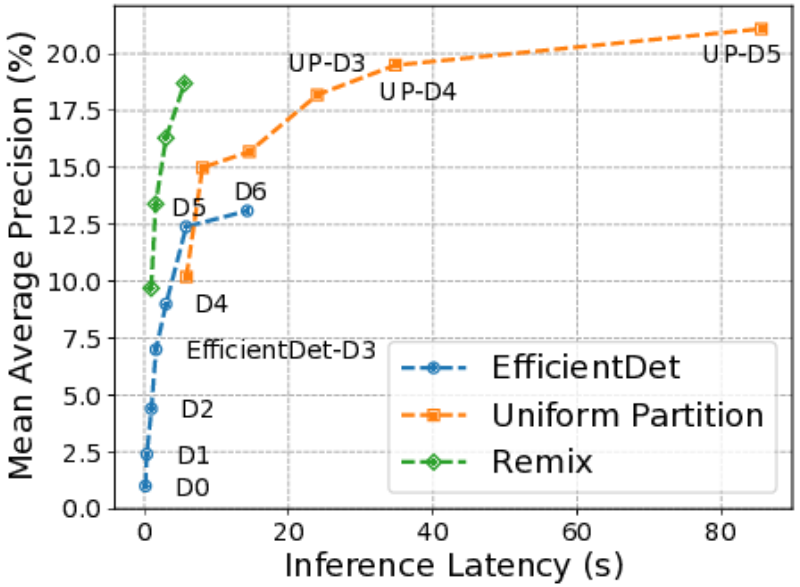
Jetson Xavier



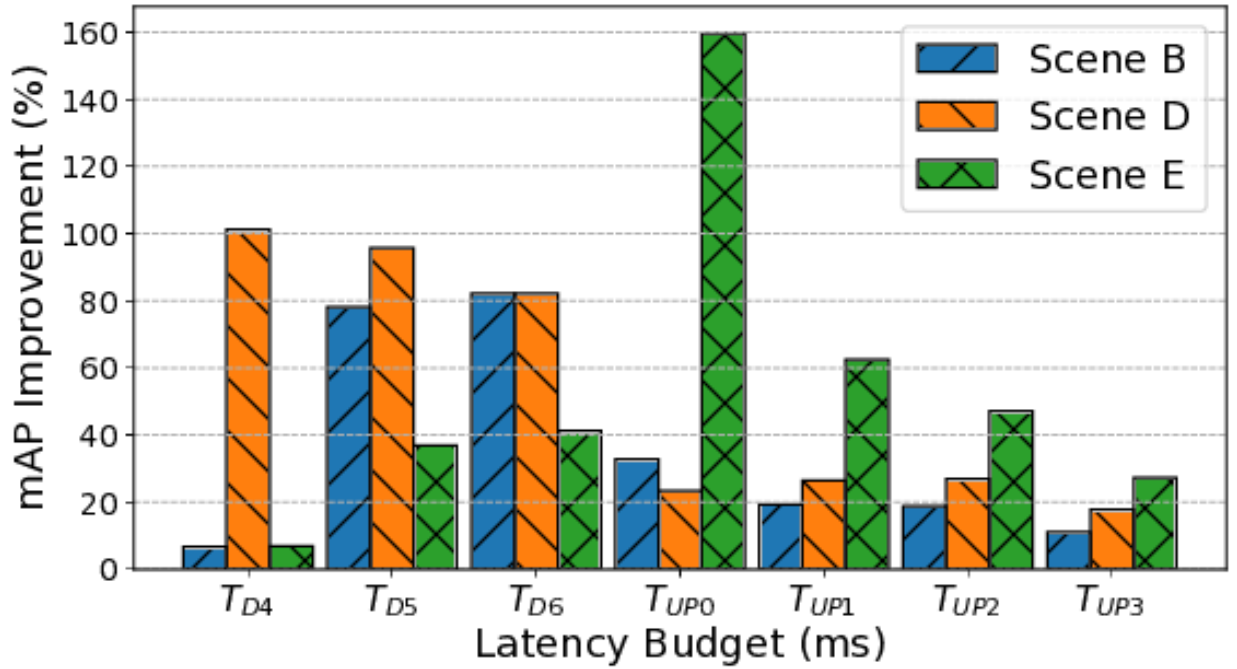
Kiren 970



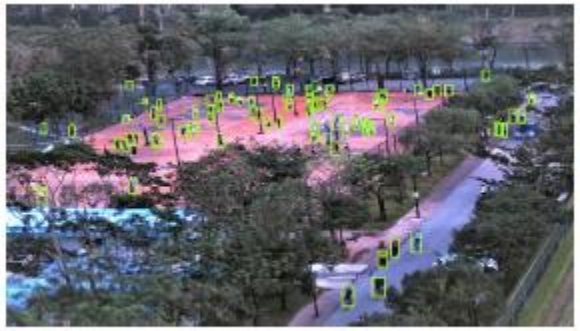
Snapdragon 855



Performance across Scenes



(a) Scene B: Plaza square.

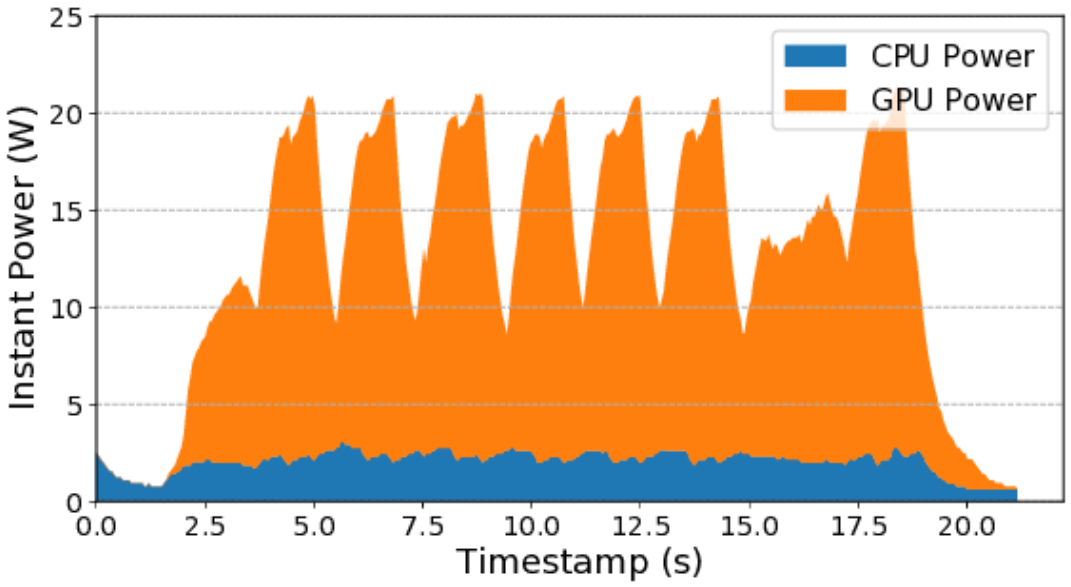


(b) Scene D: Sports ground.

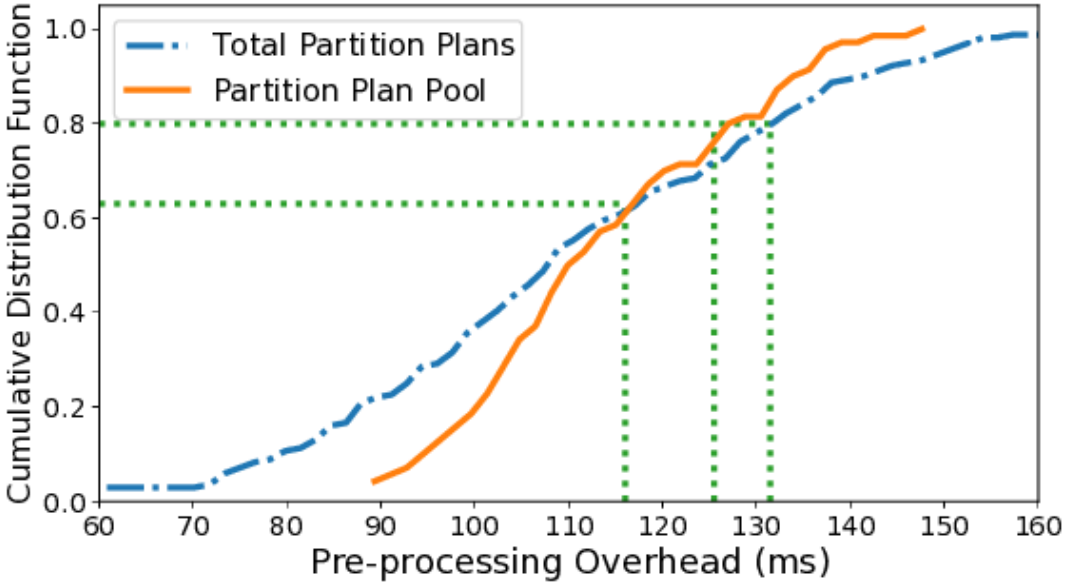
(c) Scene E: Street.

System Overhead

Energy Consumption



Pre-processing Overhead



Conclusion and Updates

- Remix, adaptive partition and selective execution
- Cloud-Edge co-design
- Tunable latency and elastic AI

- Updates
  - Triggering partition plans update automatically when scenes change
  - Utilizing heterogenous processing units on edge, i.e., GPU, CPU, NPU, etc.

Thank You

shijiang@microsoft.com