An Analytical Model for Distributed Video Analytics in Mobile Computer Vision-based Systems

Michael Lesko-Krleza[1], Rodolfo W. L. Coutinho[2] and Yousef R. Shayan[3]

[1] ECE, Concordia University, Montreal, Canada

m_lesk@live.concordia.ca, rodolfo.coutinho@concordia.ca, yousef.shayan@concordia.ca

UNIVERSITÉ Concordia UNIVERSITY

Outline

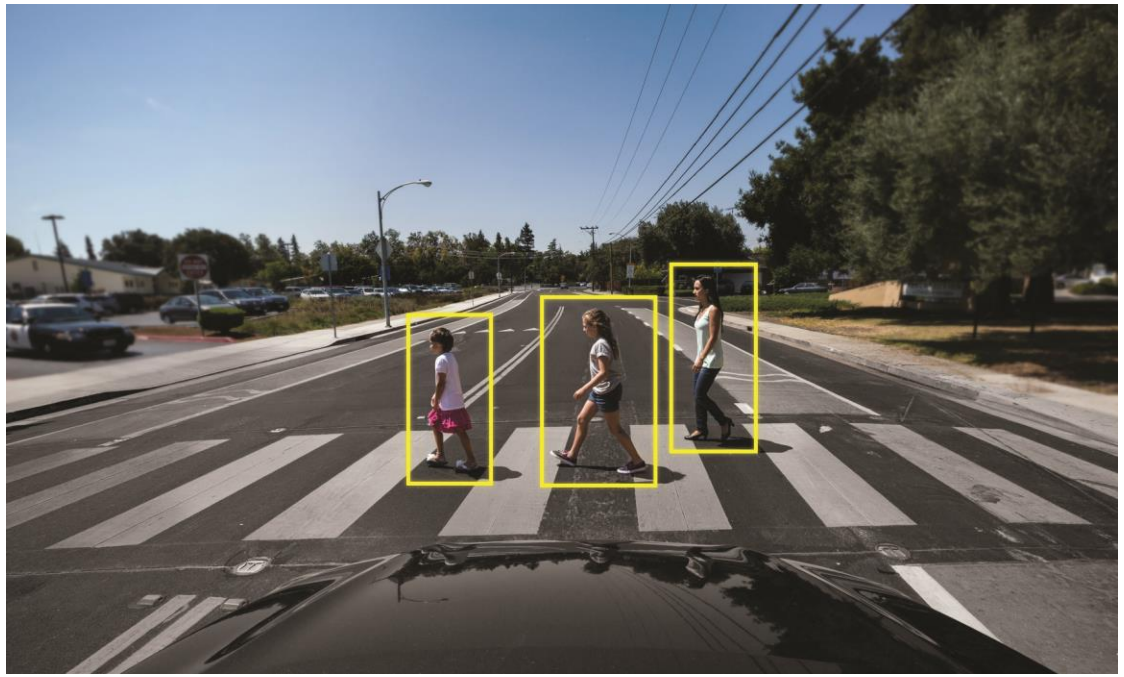
- Introduction and Motivation
- Research Challenges
- Related Work
- Objectives
- Preliminaries
- Proposed Model
  - Logic
  - Equations
- Results
- Conclusion and Future Work

Introduction and Motivation

- Challenging frame processing on mobile IoT systems:
  - Limited local device resources for local processing
  - Time-consuming offloading for remote processing
  - Factors to optimize:
    - Total processing latency: Processing and offloading latency
    - Energy cost
  - Composed of:
    - Mobile cellular devices
    - Cloud servers
- Potential applications:
  - Autonomous factory robots
  - Self-driving vehicles
  - Surveillance security cameras

Research Challenges

- Provide video analytics in a timely and efficient manner
  - Limited computational resources
  - Limited energy



Related Works

- Select local or remote processing based on:
  - Run-time and memory demands to reduce latency [Tan and Cao]
  - Resolution to maximize accuracy and processing rate [Liu and Cao]
  - Frame size, priority, CPU and bandwidth usage to minimize latency and improve accuracy [Fu et al.]
  - Partitioned frames selected based network condition and frame states [Wang et al.]
- Optimizing computational resources:
  - Selecting edge server to minimize processing migration [Maand Mashayekhy]
  - Multiplexing local camera resources [Coutinho and Boukerche]

Related Works

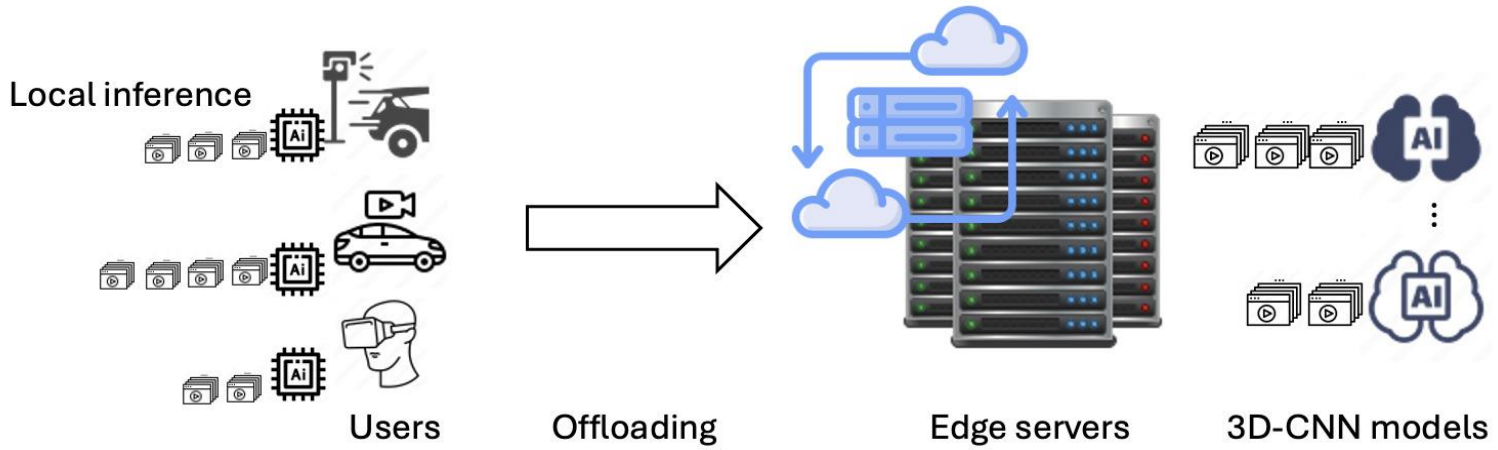
But…

1. How can we minimize energy and latency to best accommodate a given scenario?

2. Can we also vary hardware resources instead of assuming they are fixed?

Objectives

- Process frames locally or remotely
  - Energy-latency tradeoff
  - Mobile devices and edge server
  - Configured based on hardware
- Leverage the 5G-Advanced and 6G cellular network
- **Future goal:** Reinforcement learning for optimized solution



Preliminaries

- r_i: Video frame resolution of device 'i' (pixels)
- C: Pixel colour depth (bits/pixel)
- W: Bandwidth sub-band (Hz)
- p_i: Transmission power of device 'i' (W)
- h_i: Uplink channel gain between base station and device 'i'
- N_0: Noise power (W/Hz)
- t: Current time (s)
- t_p: Time instant of the arrival of frame 'p' to the offloading queue (s)
- t_c: Time instant of the arrival of frame 'c' in the local processing queue (s)
- Z_i: Number of Compute Unified Device Architecture (CUDA) cores on device 'i'
- Z_e: Number of stream processors on GPU at edge server 'e'

Preliminaries

- $C(M_n)$: Inference complexity of model Mn on device 'i'
- $C(M_{e,i})$: Inference complexity of model M on edge server 'e' from device 'i'
- f_i: Mobile device 'i' processor frequency (Hz)
- $f_{e,i}$: Edge server 'e' processor frequency (Hz)
- κ: Device specific energy constant
- $I_{o,v,i}$: Local (0) or remote (1) indicator function for device 'i' producing video frame 'v'

Preliminaries

- Offloading Delay: $d_{v,i}^{o} = \frac{s_i}{R_i},$
- Frame Size: $s_i = r_i \times r_i \times C.$
- Uplink Data Rate: $R_i = W \times \log_2\left(1 + \frac{p_i \times h_i}{N_0}\right).$
- Offload Queue Waiting Time: $q_{v,i}^{w} = o_p + \sum_{\tau=0}^{|l_i^o(t)|} (q_{\tau,i}^{w} + d_{\tau,i}^{o}),$
- Offload Frame Remaining Waiting Time: $o_p = (t_p + q_{p,i}^{w} + d_{p,i}^{o}) - t,$

Preliminaries

- Local Processing Duration: $$d_{v,i}^{p} = \frac{\rho_i C(M_n)}{f_i},$$
- Local Processing Queue Waiting Time: $$q_{v,i}^{p} = p_c + \sum_{\tau=0}^{|l_i^p(t)|} (q_{\tau,i}^{p} + d_{\tau,i}^{p}),$$
- Local Frame Processing Remaining Waiting Time: $$p_c = (t_c + q_{c,i}^{p} + d_{c,i}^{p}) - t,$$
- Remote Server Processing Queue Waiting Time: $$q_{v,i,e}^{w} = o_{e,i} + \sum_{\tau=0}^{|I_i^e|} (q_{\tau,i,e}^{w} + d_{\tau,i,e}^{p}),$$
- Remote Processing Duration: $$d_{v,e}^{p} = \frac{\breve{\rho}_e C(M_{e,i})}{f_{e,i}},$$

Preliminaries

- Processor Unit Performance (Local & Remote): $\rho_i = \frac{1}{2Z_i}, \quad \rho_e = \frac{1}{2Z_e},$
- Local or Remote Total End-to-End Latency: $l_{v,i} = (1 - I_{o,v,i}) \times (d^p_{v,i} + q^p_{v,i}) + I_{o,v,i} \times (d^o_{v,i} + q^w_{v,i} + q^w_{v,i,e} + d^p_{v,e}),$
- Local Processing Energy Cost: $E^p_{v,i} = \kappa \rho_e C(M_{e,i}) f^2_{e,i},$
- Offloading Energy Cost: $E^o_{v,i} = \frac{s_i}{R_i} \times p_i,$
- Local or Remote Total End-to-End Energy Cost: $E^c_{v,i} = (1 - I_{o,v,i}) \times E^p_{v,i} + I_{o,v,i} \times E^o_{v,i},$

Proposed Model (Logic)

- Probabilistic Heuristic:
  - Local frames: (1 – α) × 100%
  - Offloaded frames: α × 100%
  - Baseline evaluation
  - α: Offloading rate

- The Probabilistic model serves as the baseline to compare against the Greedy approach.

- Greedy Heuristic:

Algorithm 1 Greedy approach

Require: video frame v to be processed

procedure LOCALORREMOTEPROCESSING(v)
  Estimate the latency when processing v locally
  Estimate the latency when processing v remotely
  Estimate the energy cost when processing v locally
  Estimate the energy cost when processing v remotely
  Calculate the local score S_{local}
  Calculate the remote score S_{rem}
  if $S_{local} > S_{rem}$ **then**
    Process the video frame v locally.
  else
    Offload the video frame v.
  end if
end procedure

Proposed Model (Equations)

- Local score and with min-max normalization:

$$S_{loc} = w_L n_{loc}(L_{v,i}^{loc}, L_{v,i}^{rem}) + w_E n_{loc}(E_{v,i}^{loc}, E_{v,i}^{rem}),$$

$$n_{loc}(loc, rem) = 1 - \frac{loc - \min(loc, rem)}{\max(loc, rem) - \min(loc, rem)},$$

- Remote score and with min-max normalization:

$$S_{rem} = w_L n_{rem}(L_{v,i}^{loc}, L_{v,i}^{rem}) + w_E n_{rem}(E_{v,i}^{loc}, E_{v,i}^{rem}),$$

$$n_{rem}(loc, rem) = 1 - \frac{rem - \min(loc, rem)}{\max(loc, rem) - \min(loc, rem)}$$

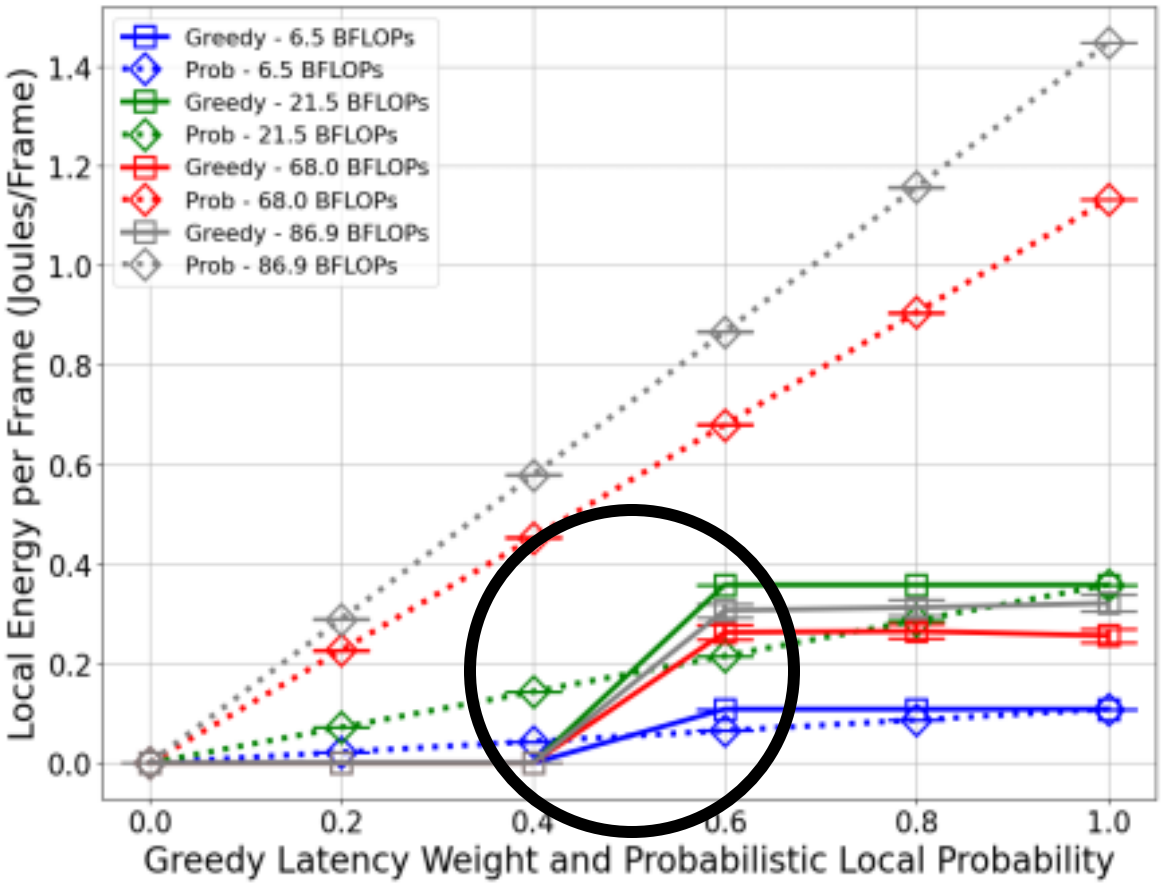
- $0 \leq \omega_L \leq 1$ and $0 \leq \omega_E \leq 1$ represent the latency and energy weights respectively
- L^{loc}, L^{rem} and E^{loc}, E^{rem} represent the latency and energy both locally and remotely

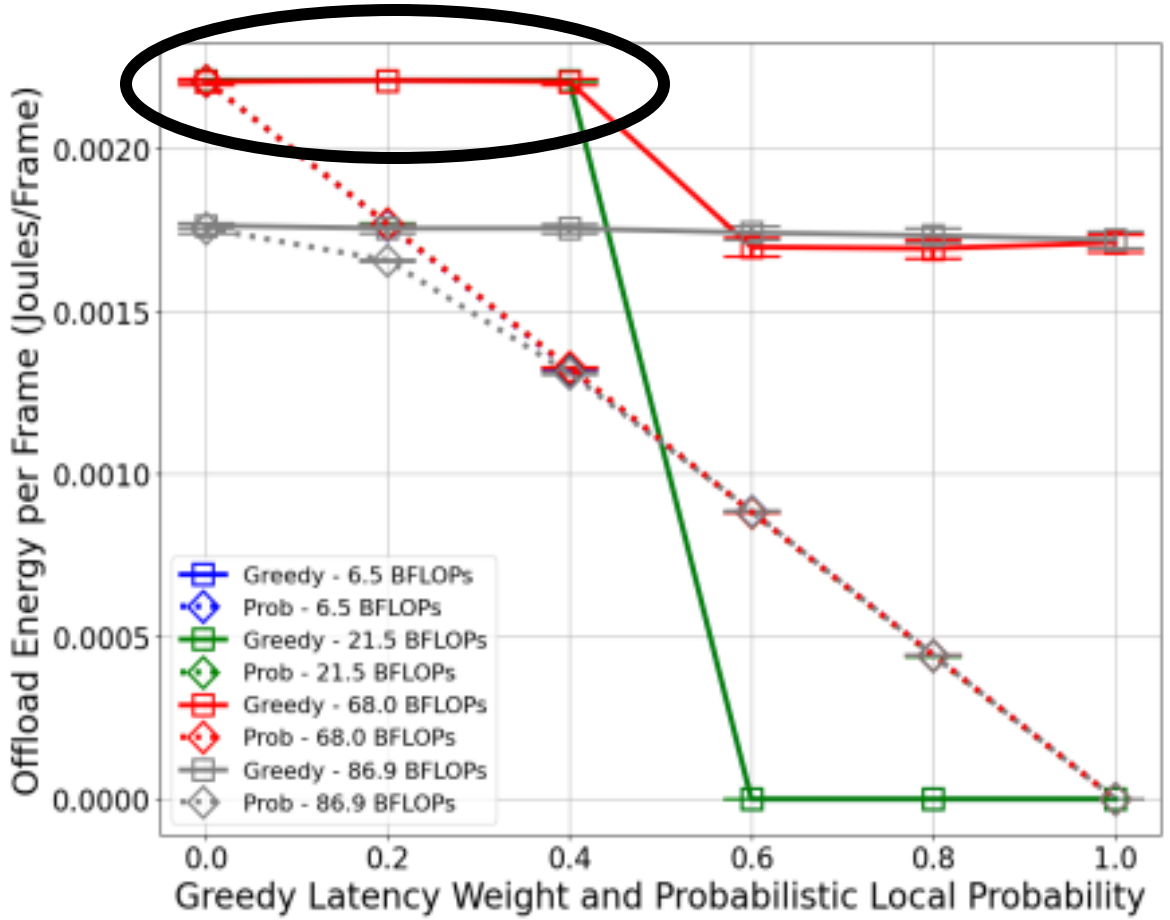
Results

- 10 mobile users
  - Using Nvidia Jetson Orin NX 16Gb with 1.173GHz at 40W
  - Produces frames at 27FPS
- 1 edge server
  - Single server serving all user requests
  - Using AMD Radeon RX 9070 XT with 2.5GHz at 304W
- 4 Computer Vision models per scenario with the same model locally and remotely
  - YOLO11N: 6.5BFLOPs
  - YOLO11S: 21.5 BFLOPs
  - YOLO11M: 68 BFLOPs
  - YOLO11L: 86.9 BFLOPs

- Resolution 640 pixels x 640 pixels
- Tests performed for 60s each at 50 iterations with a confidence interval of 95%

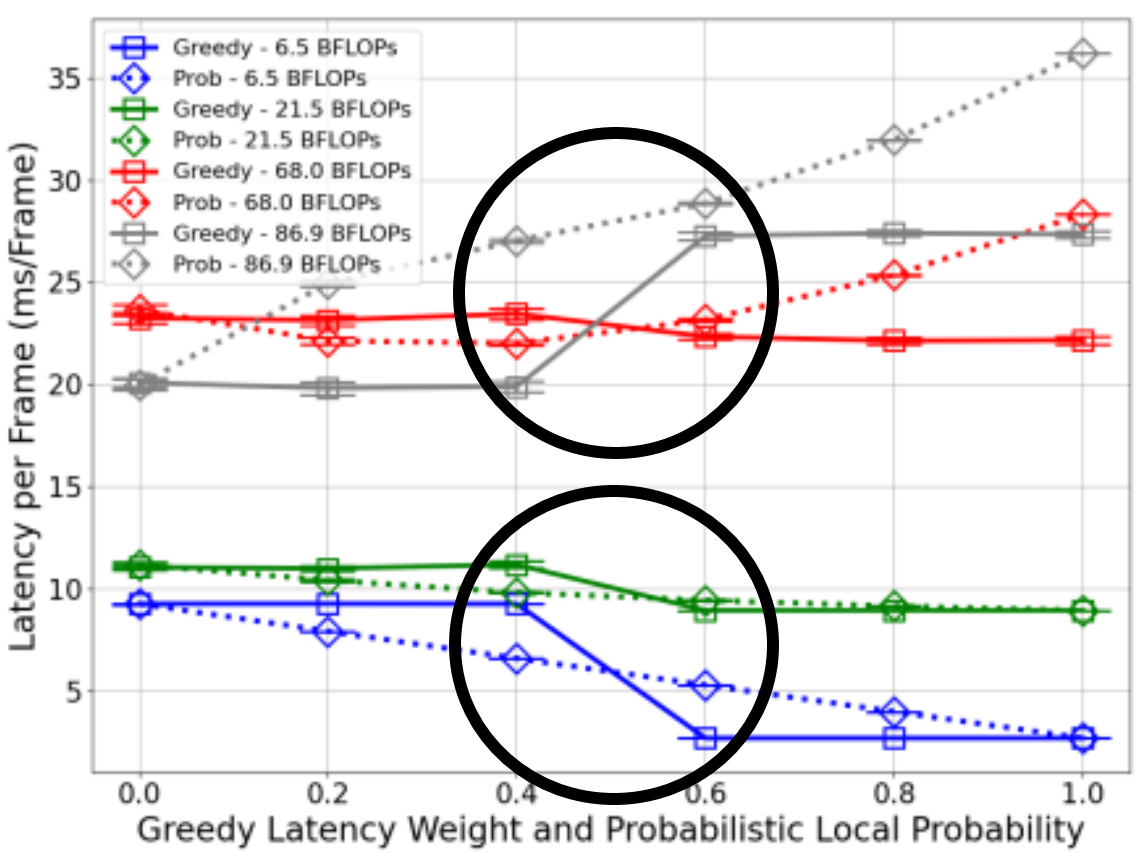
Results

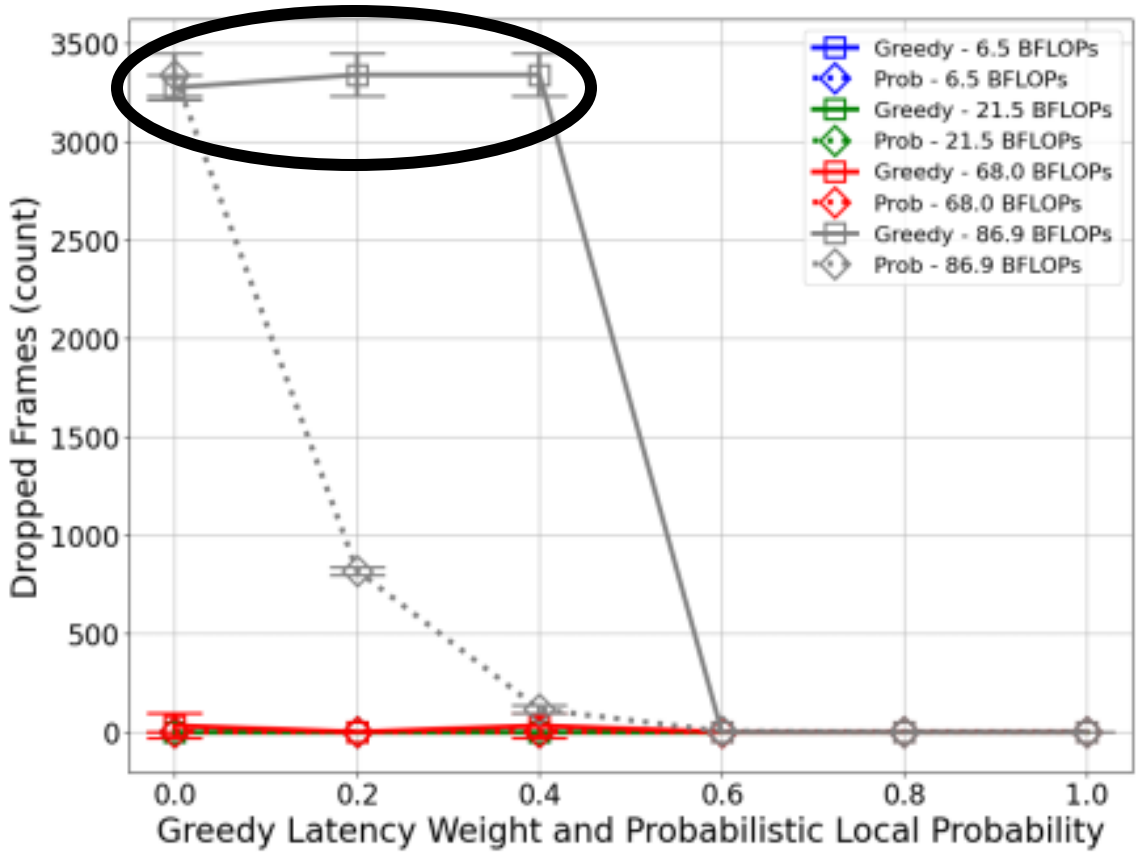




- Higher latency weight for reduced offloading
  - Lower energy weight for higher energy from local processing

Results





- Higher latency weight for lower latency from local processing
  - Lower energy weight for higher energy from local processing
- Drop frame if local end-to-end latency > single frame period (1/FPS) = Blowup!

Results

Main takeaways:

- Local Priority: Improved latency
  - Higher latency weight > 0.5
  - Possibly dropped frames

- Remote Priority: Decreased energy cost
  - Higher energy weight > 0.5
  - Reduced dropped frames (for greater remote resources)

Future Goal: Optimize the best of both for a given scenario

Conclusion and Future Work

- Mathematical framework for performance analysis while considering hardware
- Preferred processing may be assessed using estimation
- Weights may be tuned to prioritize desired goal
- Future work in progress:
  - Use RL agent for real-time optimization of parameters for given situation
  - Dynamic solution optimization based on system workload
  - Use additional parameters for performance optimization
    - Frequency
    - Device framerate
    - Offload rate
    - Chosen local model
  - Assess performance based on latency, energy **and** accuracy

Thank you!

Questions?