Resilient Edge-Cloud Autonomous Learning with Timely inferences

Haider Abdelrahman, James Chang, Lakshya Gour, Tanushree Mehta, Shreya Venugopal

Advisor: Prof. Anand Sarwate

The Team

Haider Abdelrahman
Electrical & Computer Engineering, 2026

Yunhyuk Chang
Electrical & Computer Engineering, 2024

Lakshya Gour
Computer Science + Math, 2026

Tanushree Mehta
Electrical & Computer Engineering, 2026

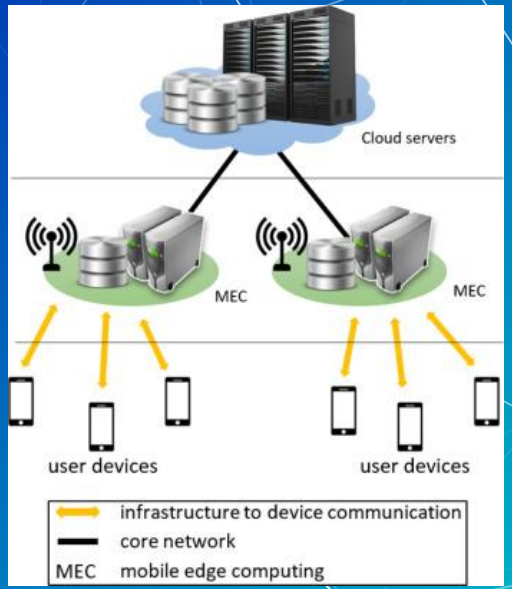
Shreya Venugopal
Computer Science, Grad student, 2024

The Problem

- Real-time machine learning models are getting more complex
- Running them on less powerful (mobile) devices is becoming difficult b/c of the need for lower latency
- Solution: MEC(Mobile-edge computing)

What is MEC (Mobile-Edge Computing)?

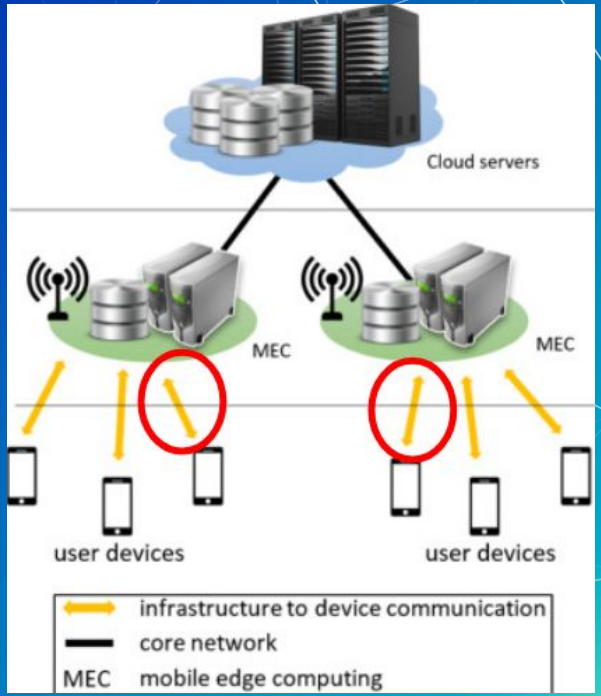
A network architecture that brings computation and storage capabilities closer to the end-users, reducing latency and improving real-time application performance.



Which part are we interested in?

- **Threshold:** Confidence level at which mobile asks for help
- **Asking for help:** Inference confidence < Threshold, request help
- **Average Latency:** Total time to perform task

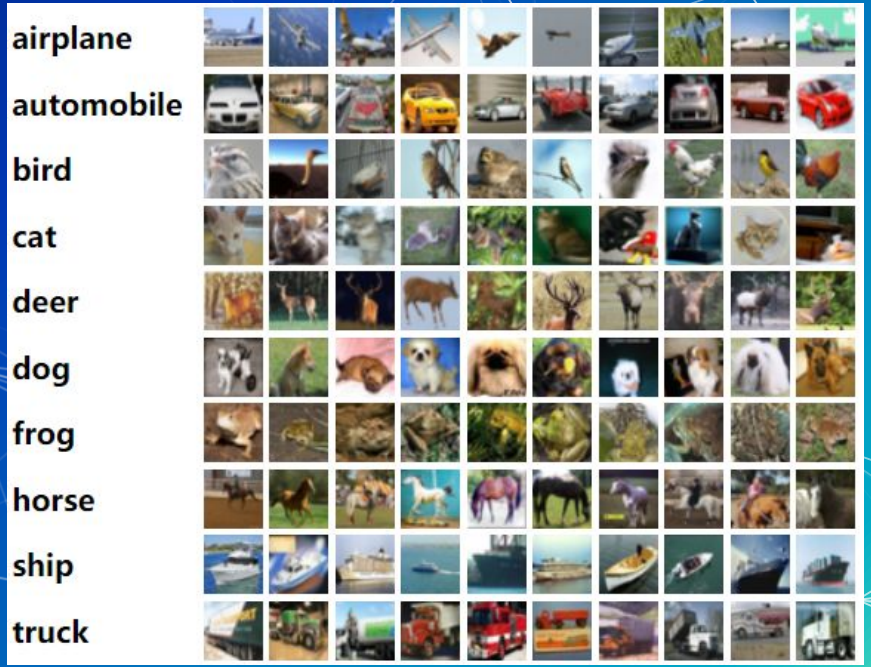
As you vary the threshold, how does the average latency change(over the dataset)?



Experimental Setup

Models and Datasets

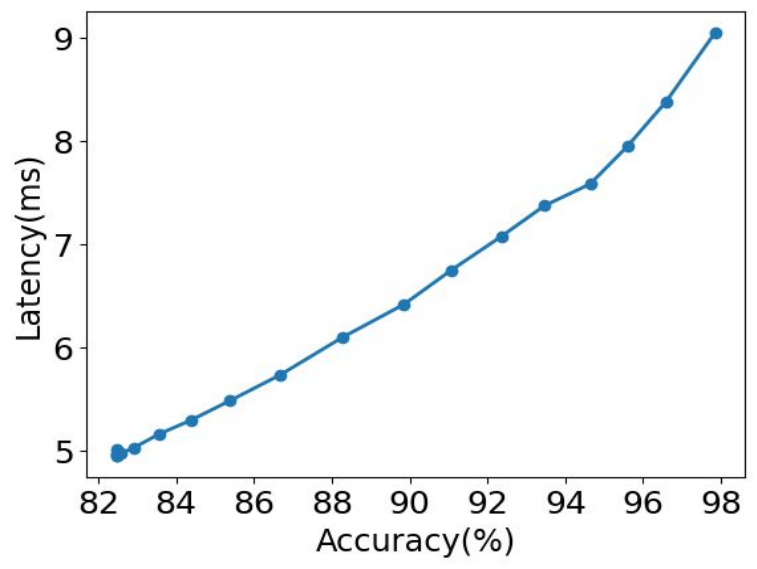
- Data:
  - CIFAR-10
  - 10 categories
  - Dataset size: 60,000
  - Test set size: 10,000
- Models:
  - MobileNetV2
  - DenseNet

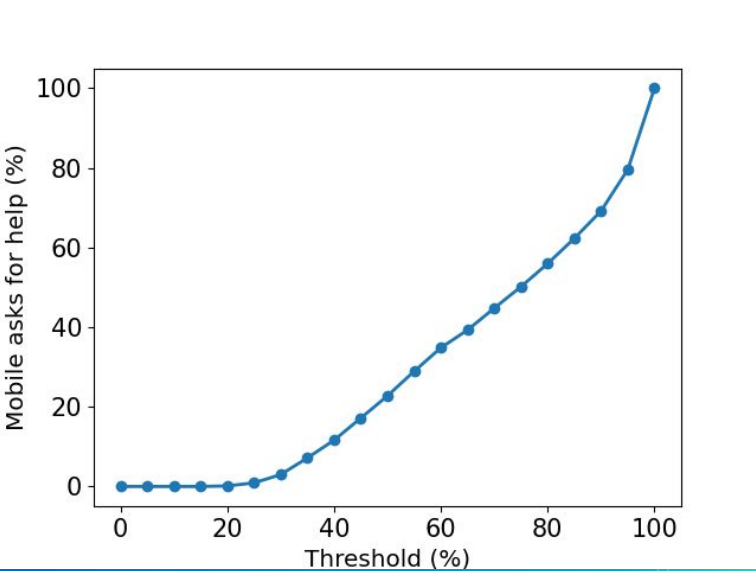


Findings

Baseline

Mobile Device: Small Neural Network (MobileNetV2)	Edge Device: Oracle (100% Accuracy)	Time Synchronization Protocol: PTP Network Connection: Ethernet





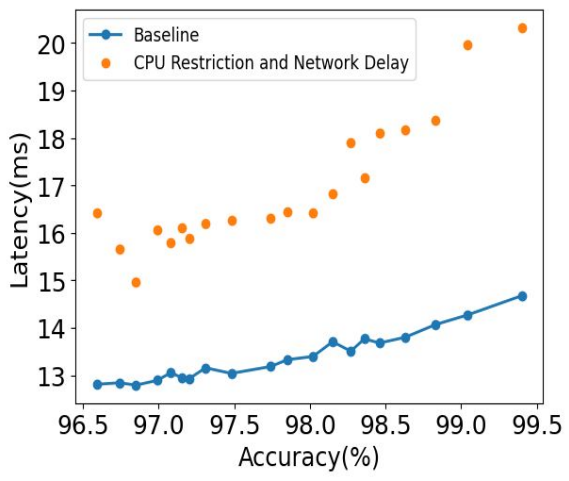
CPU Restriction and Network Delay

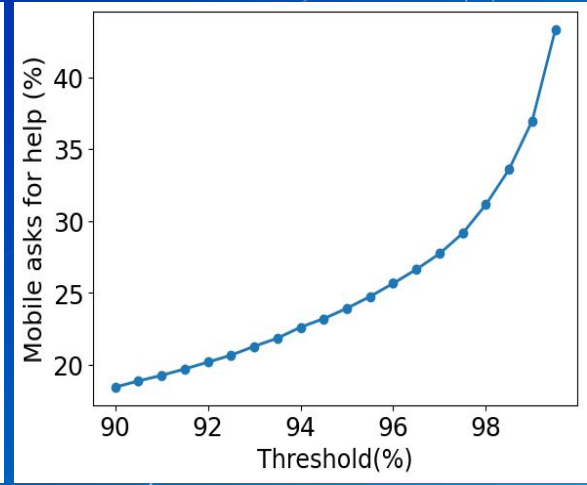
Mobile Device: Small Neural Network (MobileNetV2)

Edge Device: Oracle (100% Accuracy)

Time Synchronization Protocol: PTP

Network Connection: Ethernet



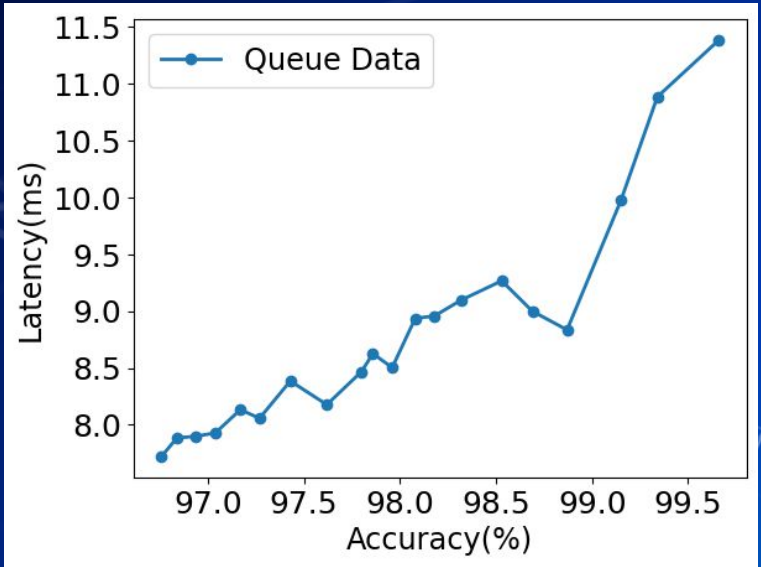


CPU Limit: 1.2 Ghz

Network: 8ms delay +/- 3ms

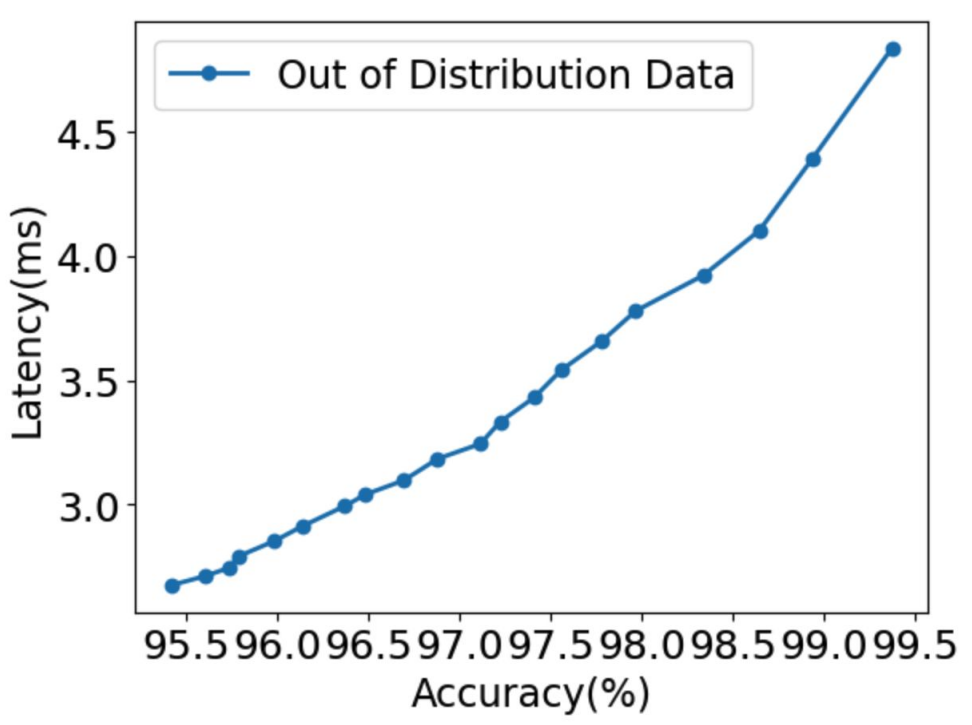
Queuing

Mobile Device: MobileNetV2 (85% Accuracy)	Edge Device: Oracle (100% Accuracy)	Time Synchronization Protocol: PTP Network Connection: Ethernet



- Queue at Edge
- Mobile continues to inference on next image(multithreading) as it waits for the Edge response
- Range of latency: 7-12ms

Out of Distribution Analysis



- Mobile has less class capability than Edge
- Separates unknown image to confused class
- Range of latency: 2.5-5ms

Conclusions

- Implementing threshold:
  - Lower latency inference than using only the Edge device
  - Higher accuracy inference than using only the Mobile device
- Emulating real life by restricting the CPU speed & network has high impact on latency
- Introducing parallelization (multithreading) during inference allows for lower latency and quicker predictions

Potential Next Steps

- Continue to better emulate real life scenarios
- Better automate testing and data collection
- Explore more complex problems
  - Split Computing
  - Early Exiting
  - Multiple Clients and Servers
  - Different Queuing Policies

Acknowledgements

Sponsor(s): nVerses Capital
Project head: Prof. Anand Sarwate
Special thanks: Prof. Waheed U. Bajwa, Ivan Seskar, Jenny Shane, Prof. Roy Yates, & all PhD students who helped!
This material is based upon work supported by the National Science Foundation under grant no. CNS-2148104 and is supported in part by funds from federal agency and industry partners as specified in the Resilient & Intelligent NextG Systems (RINGS) program.