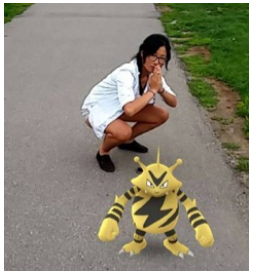
Delivering Deep Learning to Mobile Devices via Offloading

Xukan Ran*, Haoliang Chen*, Zhenming Liu[1], **Jiasi Chen***

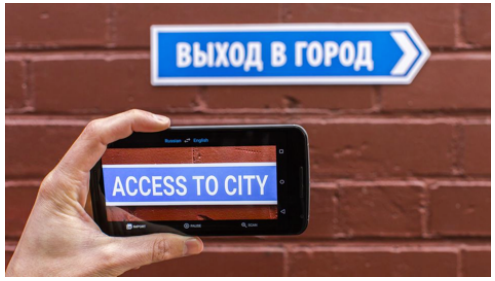
*University of California, Riverside [1]College of William and Mary

Deep learning on mobile devices

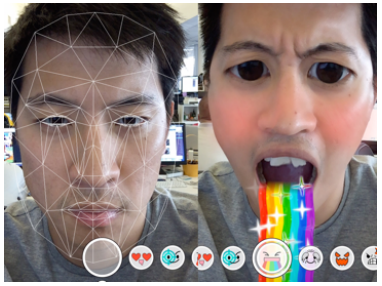
- Augmented reality (AR) is the next “killer app”



Pokemon Go



Google Translate (text processing)



Snapchat filters (face detection)

- Fast object recognition is key for general AR applications
- Deep learning is a popular technique for object recognition

Problem

- Current approaches for deep learning on mobile devices
  1. Local-only processing
     - Apple Photos, Google Translate
     - GPU speedup [1]

     → Slow! (~600 ms/frame)

  2. Remote-only processing
     - Apple Siri, Amazon Alexa

     → Doesn't work when network is bad

- Goal: Develop a framework to intelligently offload to nearby edge devices for real-time video analysis using deep learning.
- Cannot use general offloading techniques. Need to specifically account for:
  - Characteristics of the video
  - Characteristics of the deep learning models
  - Application requirements

[1] L. Huynh, Y. Lee, R. Balan, "DeepMon: Mobile GPU-based Deep Learning Framework for Continuous Vision Applications", *ACM MobiSys*, 2016.

Design space

Degrees of freedom

- Video characteristics
  - Frame rate
  - Resolution
  - Bit rate
- Deep learning characteristics
  - Model size
  - Model latency / energy
  - Model accuracy

Constraints

- App requirements
  - Latency
  - Accuracy

Metrics

- Accuracy
- Frame rate
- Energy

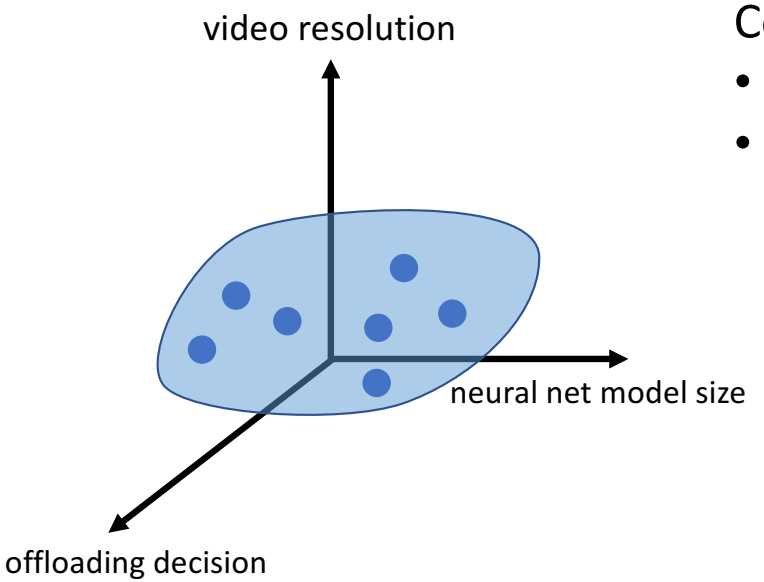
Complex interactions between these degrees of freedom and metrics

- *e.g.*, high bit rate when offloading → high accuracy, high energy
- *e.g.*, small deep learning model → high frame rate, low accuracy

How to decide?

Decision framework

Degrees of freedom:

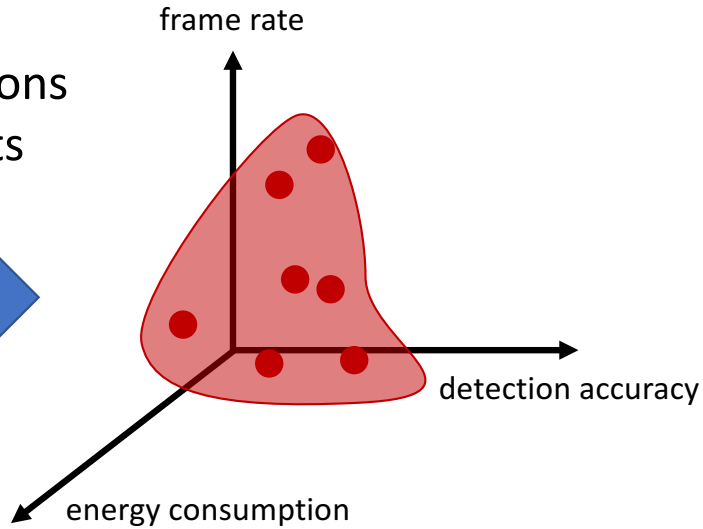


Constraints:

- Current network conditions
- Application requirements

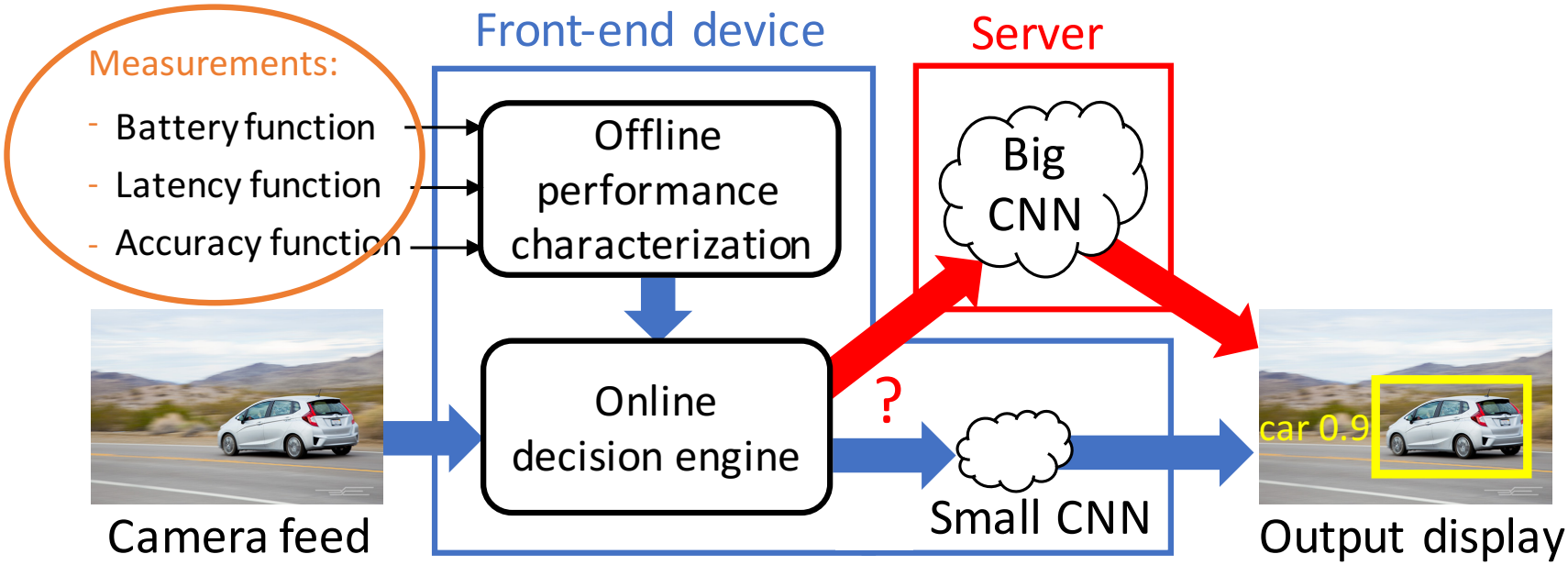
Optimize decision

Metrics:



Relation between the degrees of freedom on the metrics cannot be analytically understood → need measurements!

System design

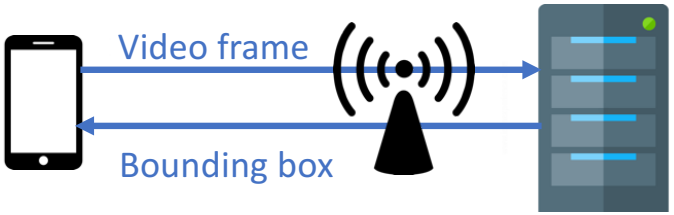


Experimental setup

- Deep learning model: YOLO built on Tensorflow [2]
  - tiny-yolo: 9 convolutional layers
  - big-yolo: 22 convolutional layers

- Local processing: OnePlus 3T Android phone with quad-core CPU, 6 GB RAM
- Remote processing: Server with quad-core CPU, 8 GB RAM, NVIDIA GeForce GTX970 graphics card with 4GB of RAM

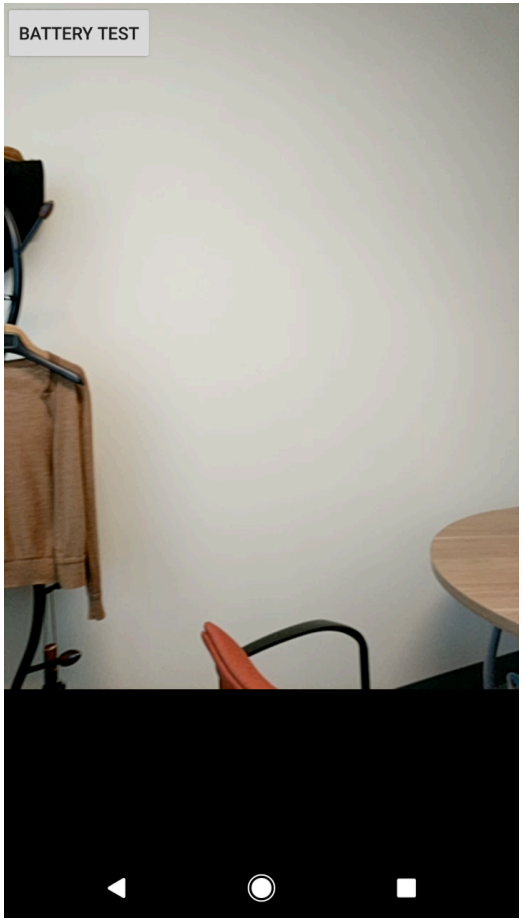
Developed app to implement offloading:



[2] Joseph Redmon, Ali Farhadi, "YOLO9000: Better, Faster, Stronger", *CVPR*, 2017.

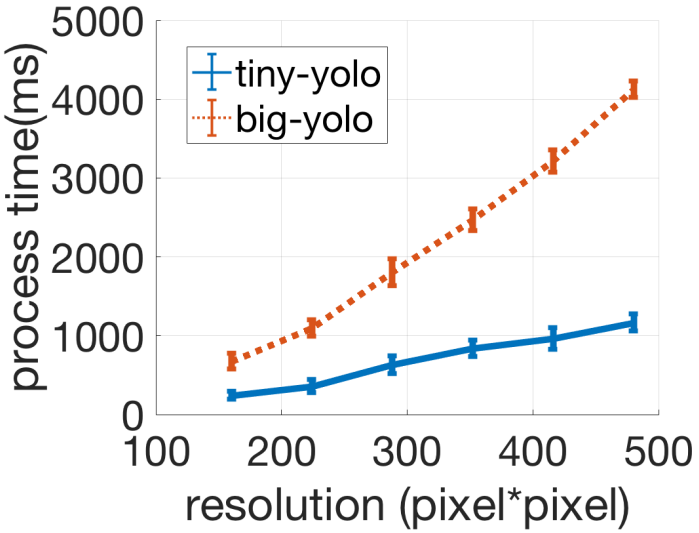
Remote-only processing

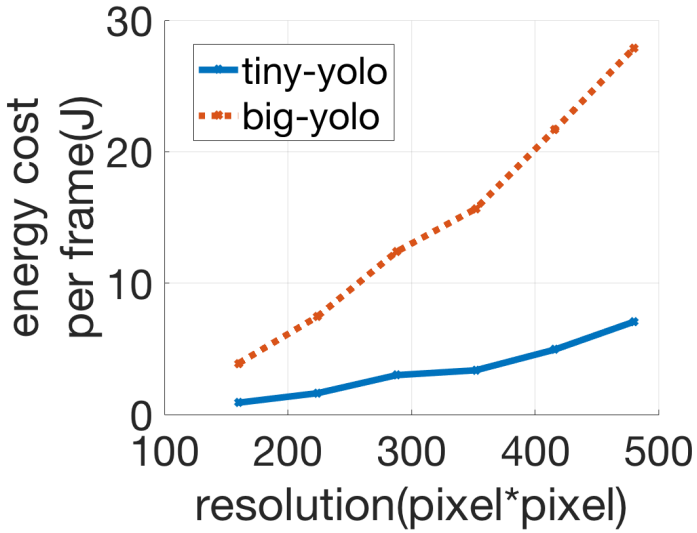
Local-only processing



How do latency and energy change with resolution?

- Encode a video frame at different resolutions
- Measure the processing time and energy usage in Android on the smartphone



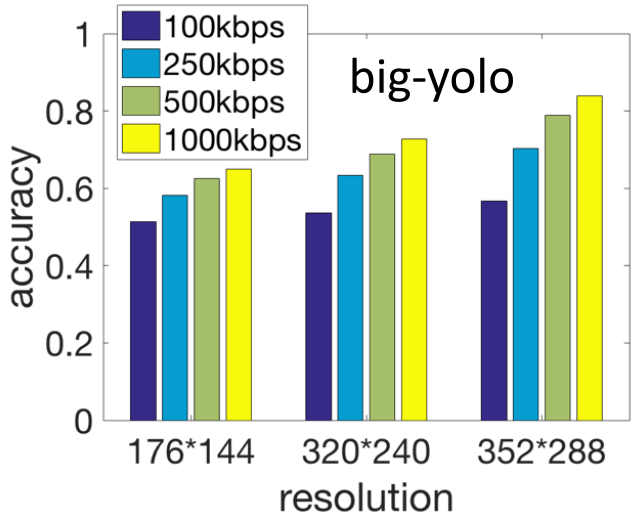


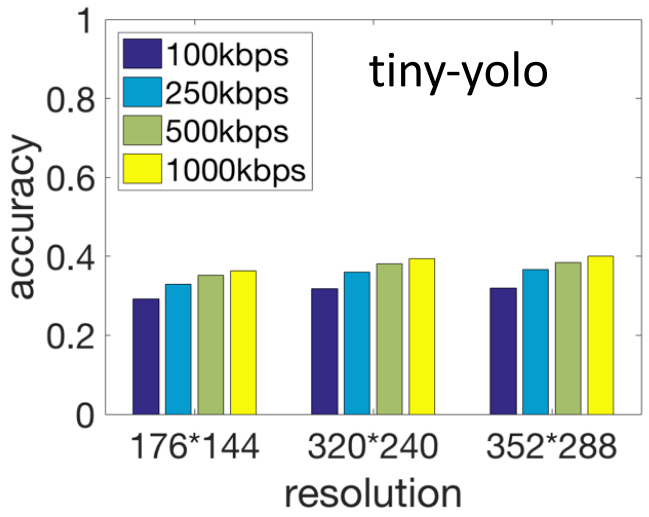
Energy and latency increase with pixels^2

How does accuracy change with bit rate and resolution?

- Encode 20 videos at different bit rate and resolutions
- Measure the accuracy (IoU) relative to the big-yolo + raw video

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

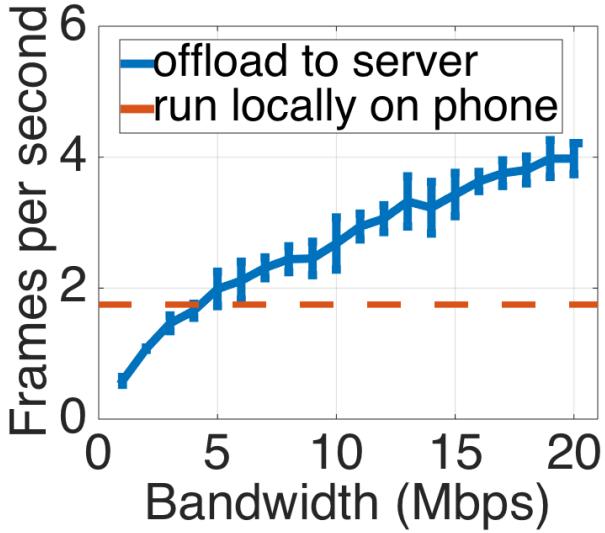


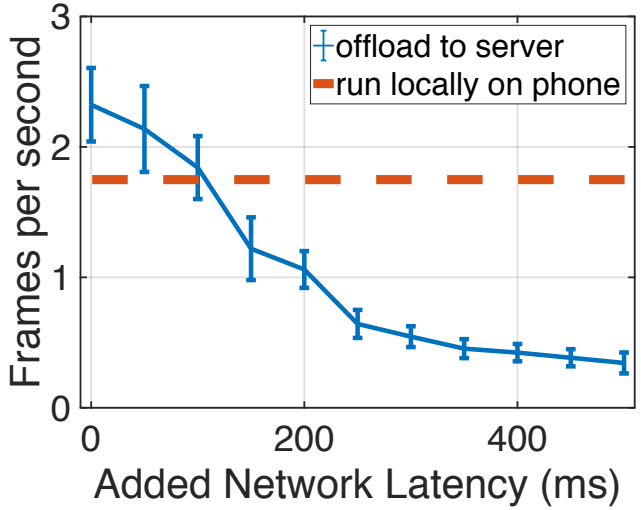


Accuracy increases with larger model, higher resolution, higher bit rate

How fast is deep learning, end-to-end?

- Measure # processed frames per second, under controlled network conditions
- *Caveat*: stop-and-wait for each processed frame



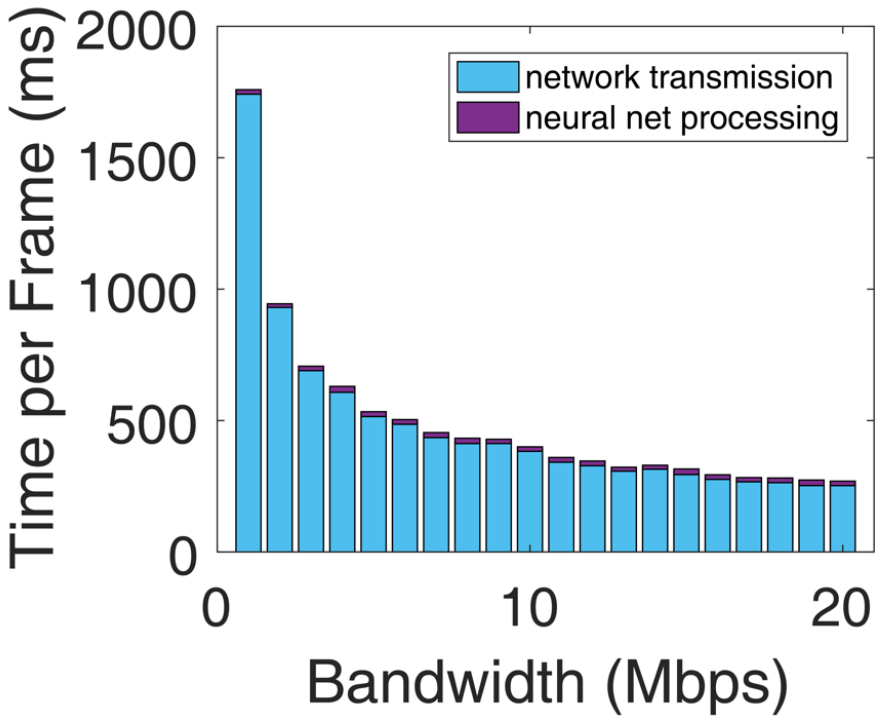


- Increased bandwidth → higher frame rate
- When bandwidth > 5Mbps, should offload

- Increased latency → lower frame rate
- When latency < 100ms, should offload

How much time is spent for communication?

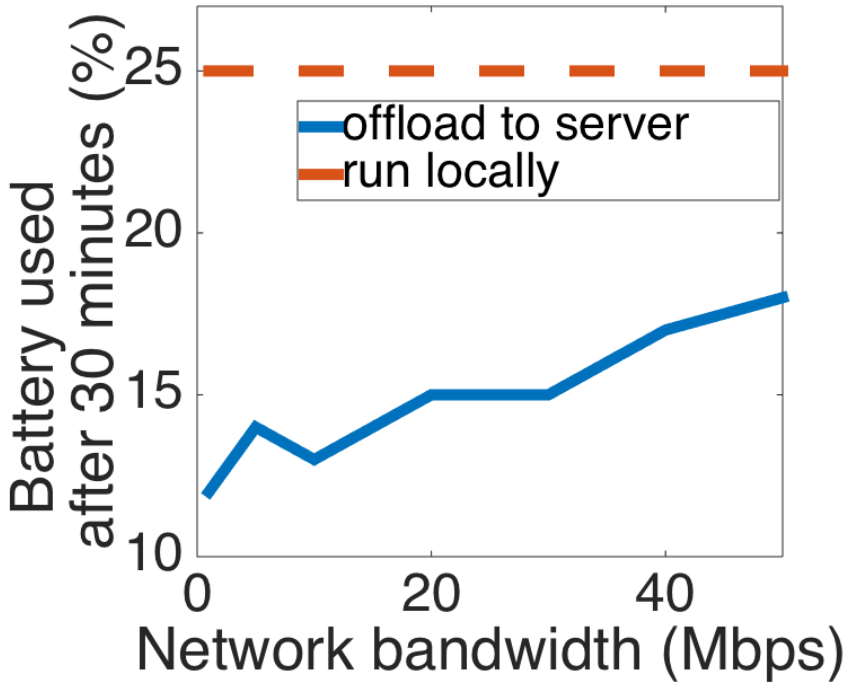
- Record timestamps as frame travels from phone to server and back



When offloading, majority of time is spent on network

How much battery is used from offloading deep learning?

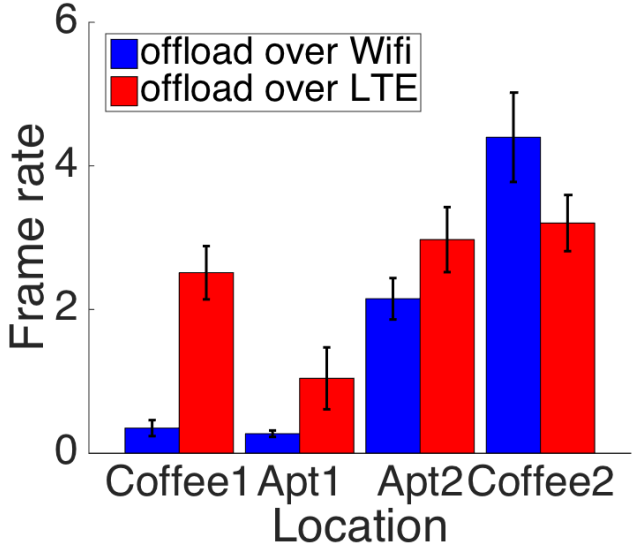
- Measure the battery drop after 30 seconds of continuous usage

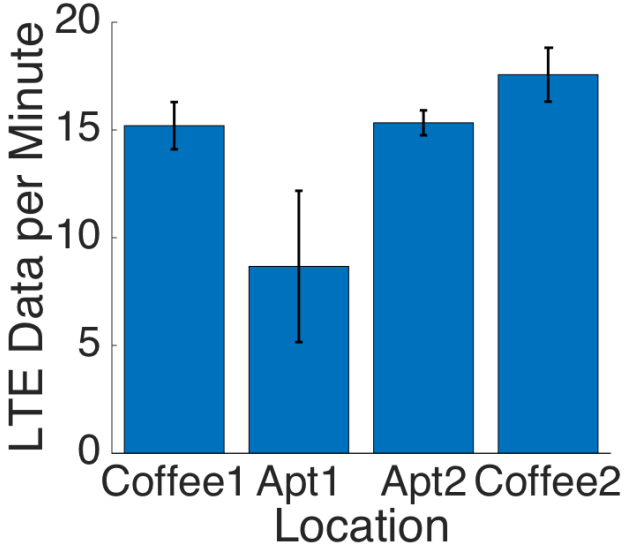


Higher bandwidth → more battery
Prefer to run locally to save battery

How well does offloading do in the wild?

- Perform 5 trials in public locations over LTE and WiFi
  - *Coffee shop 1*: Different city from server
  - *Coffee shop 2*: Same city, same subnet as server
  - *Apartment 1*: Different city than server
  - *Apartment 2*: Same city as server





Performance over LTE sometimes > WiFi

Higher frame rates over LTE at the expense of data cost

Key Take-Aways

Real-time video analysis using deep learning is slow (~600 ms/frame on smartphones)

Offloading can be beneficial (up to 2x frame rate), but optimal decision is unclear

In the wild, LTE sometimes > public WiFi