X_405082

Advanced Computer Networks

Video Stream Analytics

Lin Wang (lin.wang@vu.nl)

Period 2, Fall 2020

Course outline

Warm-up

- Fundamentals
- Forwarding and routing
- Network transport

Data centers

- Data center networking
- Data center transport

Programmability

- Software defined networking
- Programmable forwarding

Video

- Video streaming
- **Video stream analytics** 👈

Networking and ML

- Networking for ML
- ML for networking

Mobile computing

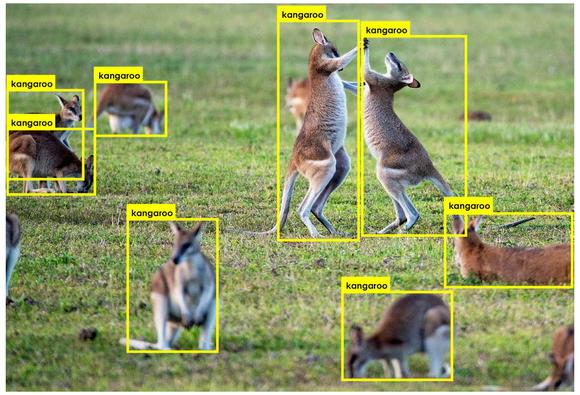
- Wireless and mobile

Learning objectives

What are the **goals** of video stream analytics?

How can we **optimize** for these goals?

Video analytics becomes pervasive



Wild-life cameras: learn about the habit of animals

Traffic cameras: monitor traffic conditions

Drone cameras: estimate the number of objects

Most of the time, Deep Neural Networks (DNNs) are used for object detection/recognition.

Question: given the wide-spread deployment of cameras, how to scale out video analytics?

Two general solutions

On-device analytics: using a light-weight DNN (e.g., SSD-MobileNet-v2)

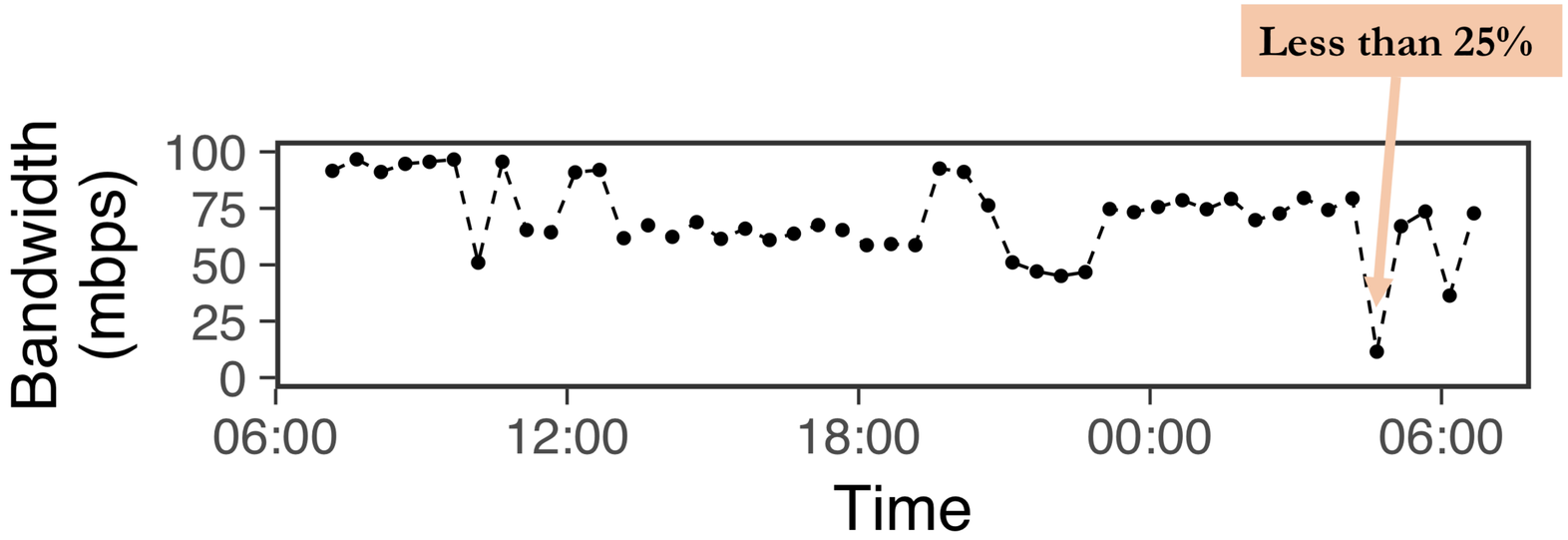
Pros: low latency, low network traffic
Cons: low analytics accuracy

Cloud-based analytics: use a sophisticated DNN (e.g., FasterRCNN-ResNet101)

Pros: high analytics accuracy
Cons: high latency, high network traffic

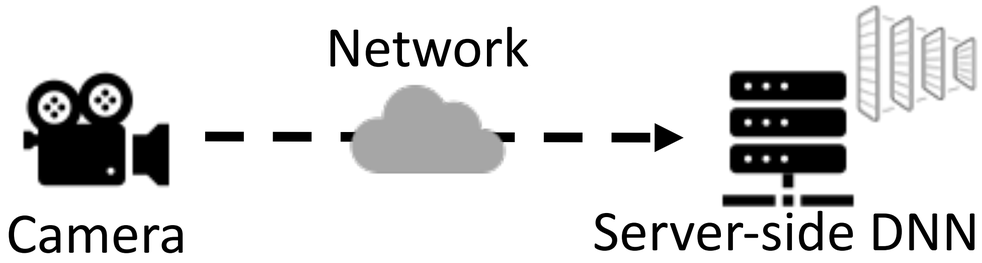
Current typical solution is to **stream** the video to the cloud over the Wide Area Network (WAN) while **addressing the challenges on latency and network traffic**

WAN bandwidth



WAN bandwidth is **scarce** (15x smaller than LAN on average for Amazon EC2 cloud, 60x worst case), **expensive** (38x more than the servers), and **dynamic**.

Video stream analytics design goals



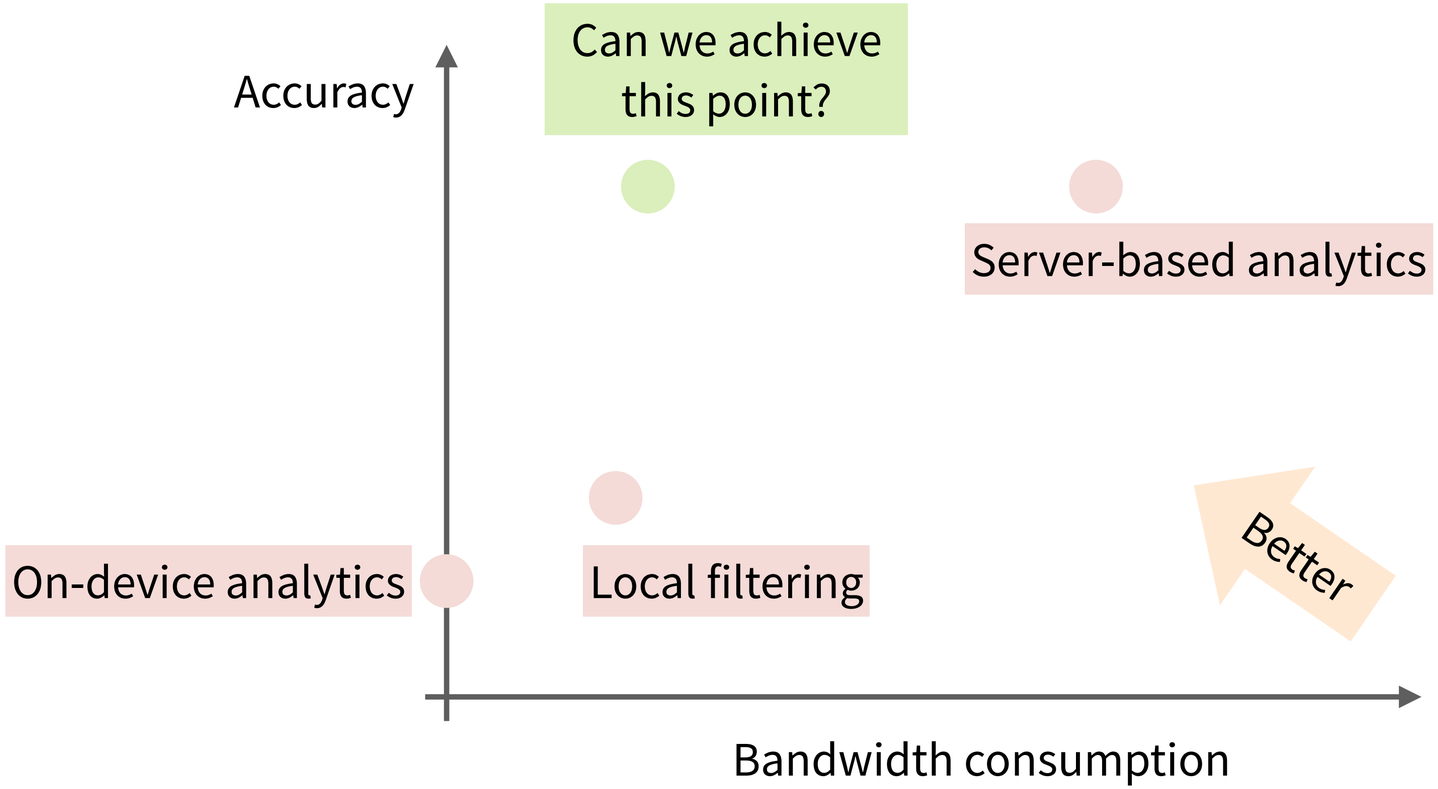
Preserve high **analytics accuracy**

- As the server-side DNN is directly inferring on the raw video

Save **network bandwidth**

- Reduce bandwidth cost
- Reduce response delay (as a result of reduced bandwidth per frame)

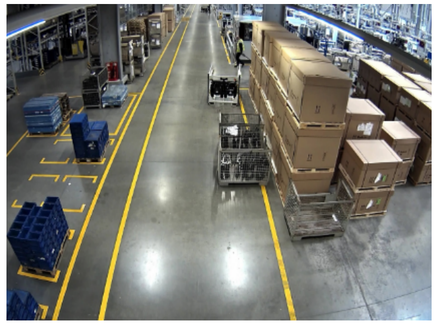
Bandwidth-accuracy tradeoff landscape



Opportunities

Video compression

- Reduce the video frame resolution or frame rate
- Use different codecs to achieve higher compression ratio



Compression

Content-based filtering

- **Early discard:** filter out irrelevant frames and drop them on device
- **Object filtering:** filter out the relevant objects and send only the part with objects

Filtering

Challenges with compression-based approaches

Network bandwidth is highly **variable**, so video streaming should be **adaptive** to bandwidth changes. How?

Manual policy: "If bandwidth is insufficient, switch to sending images at 75% fidelity, then 50% if still not enough."

Bandwidth-accuracy tradeoffs are not simply linearly

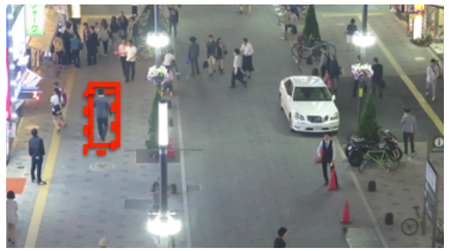
Application-specific optimizations: DASH prioritizes frame rate over resolution

Not generalizable to other applications

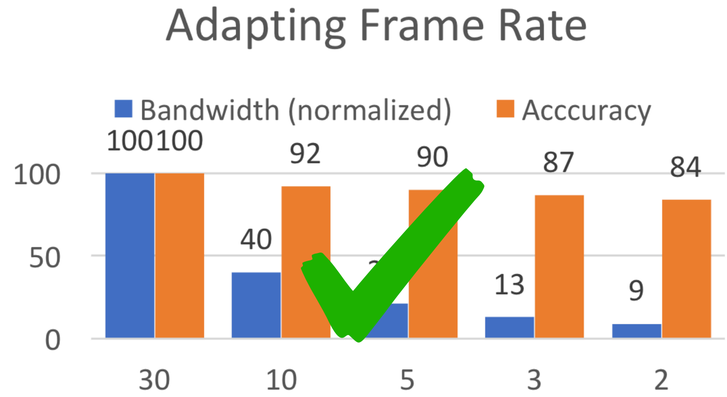
Poor generalization of application-specific solutions

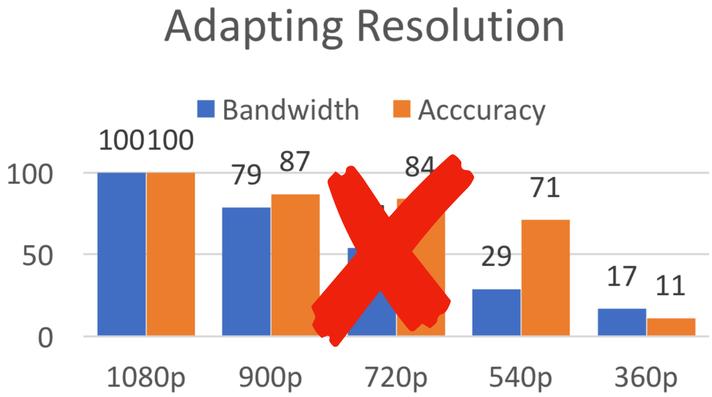
Scenario 1: A surveillance application that detects pedestrians on a busy street

t=0s, small target in far-field views



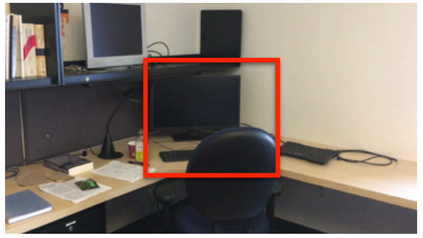
t=1s, small difference



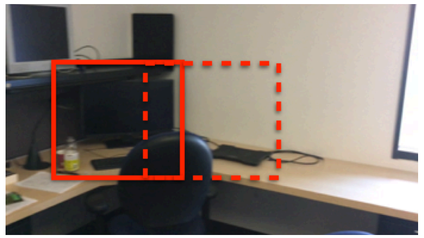


Poor generalization of application-specific solutions

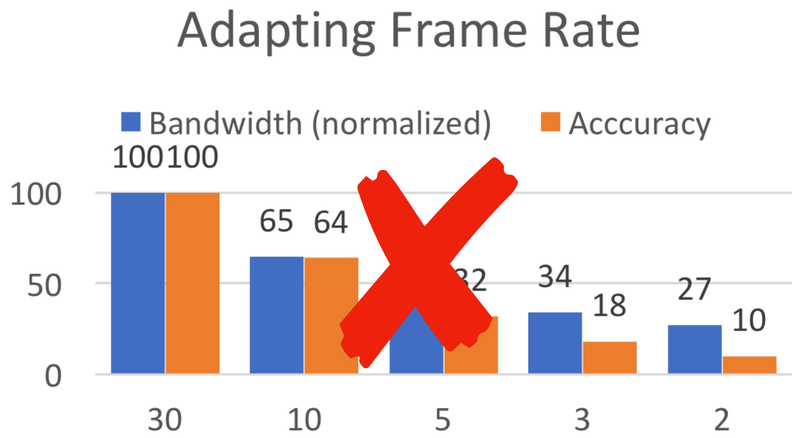
Scenario 2: An application that detects objects on a mobile phone

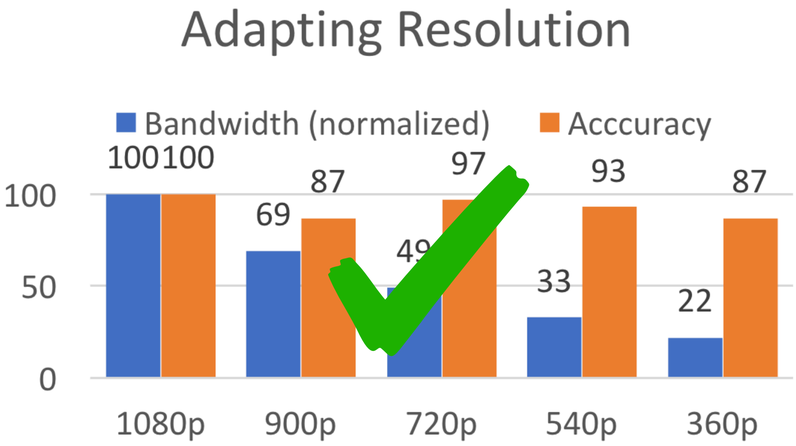


t=0s, nearby and large target



t=1s, large difference due to camera movement

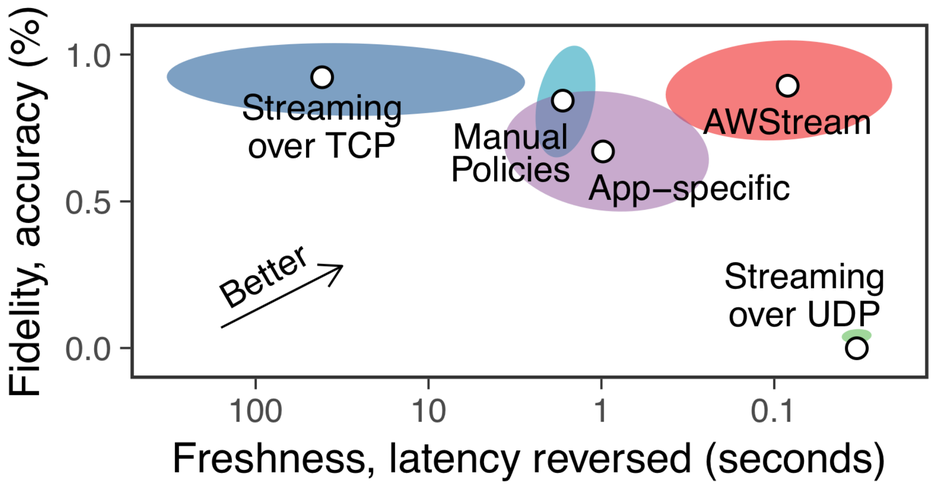


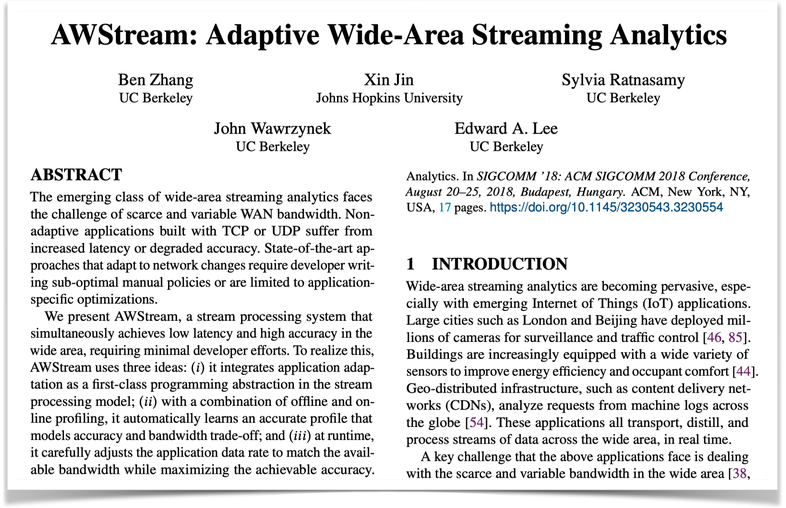


Adaptive video stream analytics

Adaptation policies must be

- Precise, automatically generated, for each application
- **Pareto-optimal profile:** maximizing application accuracy while satisfying bandwidth requirements





AWStream: Adaptive Wide-Area Streaming Analytics

Ben Zhang, UC Berkeley
Xin Jin, Johns Hopkins University
Sylvia Ratnasamy, UC Berkeley
John Wawrzynek, UC Berkeley
Edward A. Lee, UC Berkeley

ABSTRACT

The emerging class of wide-area streaming analytics faces the challenge of scarce and variable WAN bandwidth. Non-adaptive applications built with TCP or UDP suffer from increased latency or degraded accuracy. State-of-the-art approaches that adapt to network changes require developer writing sub-optimal manual policies or are limited to application-specific optimizations.

We present AWStream, a stream processing system that simultaneously achieves low latency and high accuracy in the wide area, requiring minimal developer efforts. To realize this, AWStream uses three ideas: (*i*) it integrates application adaptation as a first-class programming abstraction in the stream processing model; (*ii*) with a combination of offline and online profiling, it automatically learns an accurate profile that models accuracy and bandwidth trade-off; and (*iii*) at runtime, it carefully adjusts the application data rate to match the available bandwidth while maximizing the achievable accuracy.

Analytics. In *SIGCOMM '18: ACM SIGCOMM 2018 Conference, August 20–25, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3230543.3230554

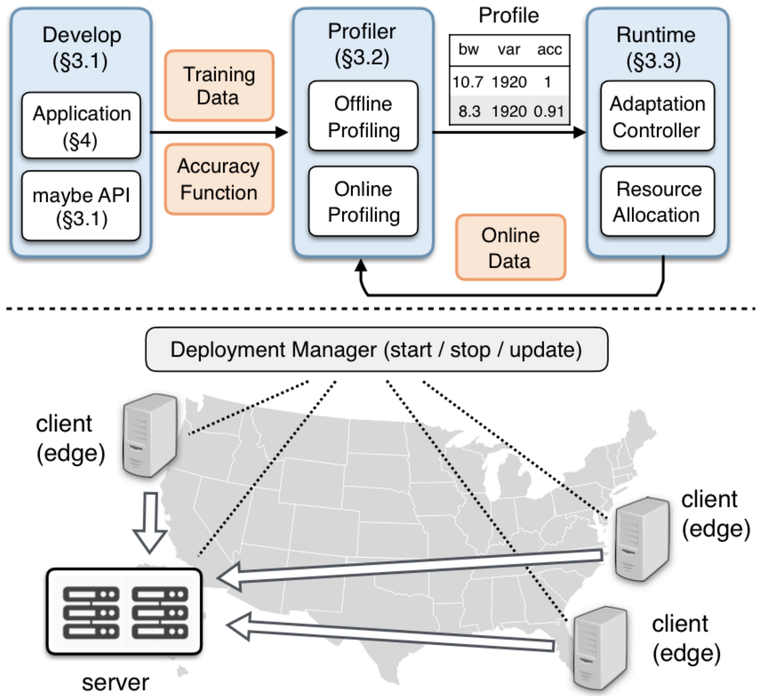
1 INTRODUCTION

Wide-area streaming analytics are becoming pervasive, especially with emerging Internet of Things (IoT) applications. Large cities such as London and Beijing have deployed millions of cameras for surveillance and traffic control [46, 85]. Buildings are increasingly equipped with a wide variety of sensors to improve energy efficiency and occupant comfort [44]. Geo-distributed infrastructure, such as content delivery networks (CDNs), analyze requests from machine logs across the globe [54]. These applications all transport, distill, and process streams of data across the wide area, in real time.

A key challenge that the above applications face is dealing with the scarce and variable bandwidth in the wide area [38,

ACM SIGCOMM 2018

AWStream system overview



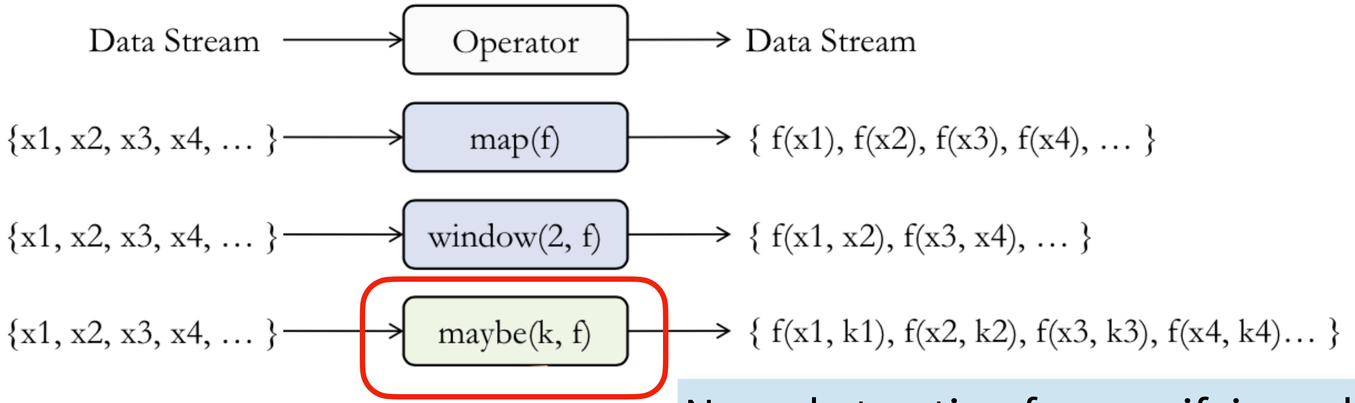
Systematic and quantitative adaptations: development, profiling, and runtime

AWStream contributions:

- New programming abstractions to express adaptation
- Automatic data-driven profile
- Runtime adaptation balancing freshness (latency) and fidelity (accuracy)

AWStream programming abstraction: operators and APIs

Follow the convention of stream processing systems



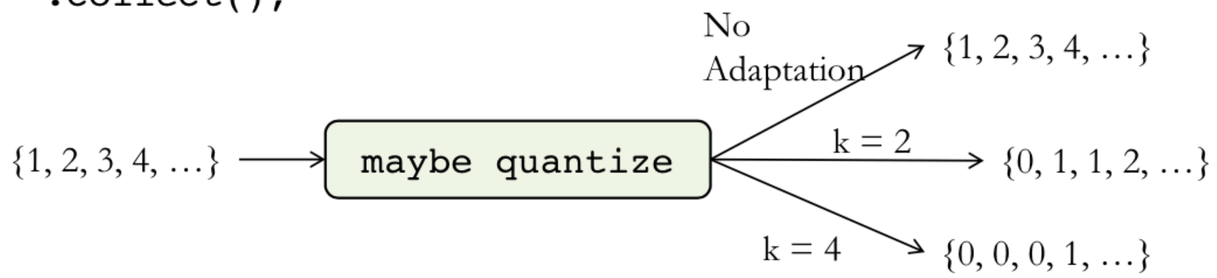
New abstraction for specifying adaptation

Normal Operators	*map* (f: I ⇒ O)	Stream<I> ⇒ Stream<O>
	skip (i: Integer)	Stream<I> ⇒ Stream<I>
	sliding_window (count: Integer, f: Vec<I> ⇒ O)	Stream<I> ⇒ Stream<O>
	…	…
Degradation Operators	*maybe* (knobs: Vec<T>, f: (T, I) ⇒ I)	Stream<I> ⇒ Stream<I>
	maybe_skip (knobs: Vec<Integer>)	Stream<I> ⇒ Stream<I>
	maybe_head (knobs: Vec<Integer>)	Stream<Vec<I>> ⇒ Stream<Vec<I>>
	maybe_downsample (knobs: Vec<(Integer, Integer)>)	Stream<Image> ⇒ Stream<Image>
	…	…

AWStream "maybe" operations

```
let quantized = vec![1, 2, 3, 4].into_stream()
    .maybe(vec![2, 4], |k, val| val / k)
    .collect();
```

Support user-defined functions




```
let app = Camera::new((1920, 1080), 30)
    .maybe_downsample(vec![(1600, 900), (1280, 720)])
    .maybe_skip(vec![2, 5])
    .map(|frame| pedestrian_detect(frame))
    .compose();
```

Modular (support combinations)

Data-driven profiling

```
let app = Camera::new((1920, 1080), 30)
    .maybe_downsample(vec![(1600, 900), (1280, 720)])
    .maybe_skip(vec![2, 5])
    .map(|frame| pedestrian_detect(frame))
    .compose();
```

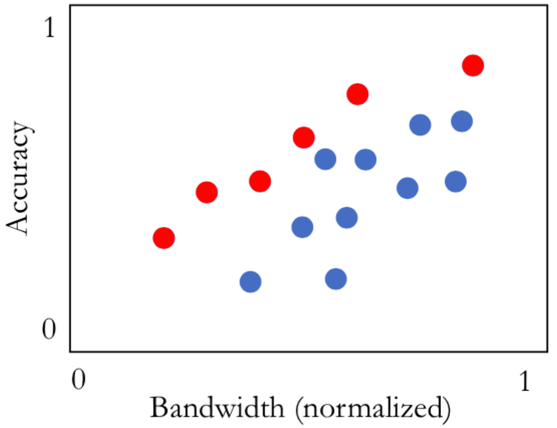
downsample	skip	bandwidth	accuracy
(1920, 1080)	0	10.7	1.0
(1600, 900)	0	8.3	0.88
(1280, 720)	0	6.3	0.87
(1920, 1080)	2	9.3	0.90
…	…	….	…

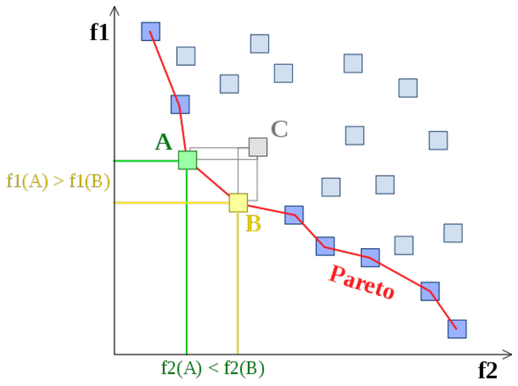
Training data: given different values for the control knobs, what are the bandwidth and accuracy?

Profile: Pareto optimal

Training data points

configuration	bandwidth	accuracy
c1	10.7	1.0
c2	8.3	0.88
c3	6.3	0.87
c4	9.3	0.90
…	….	…



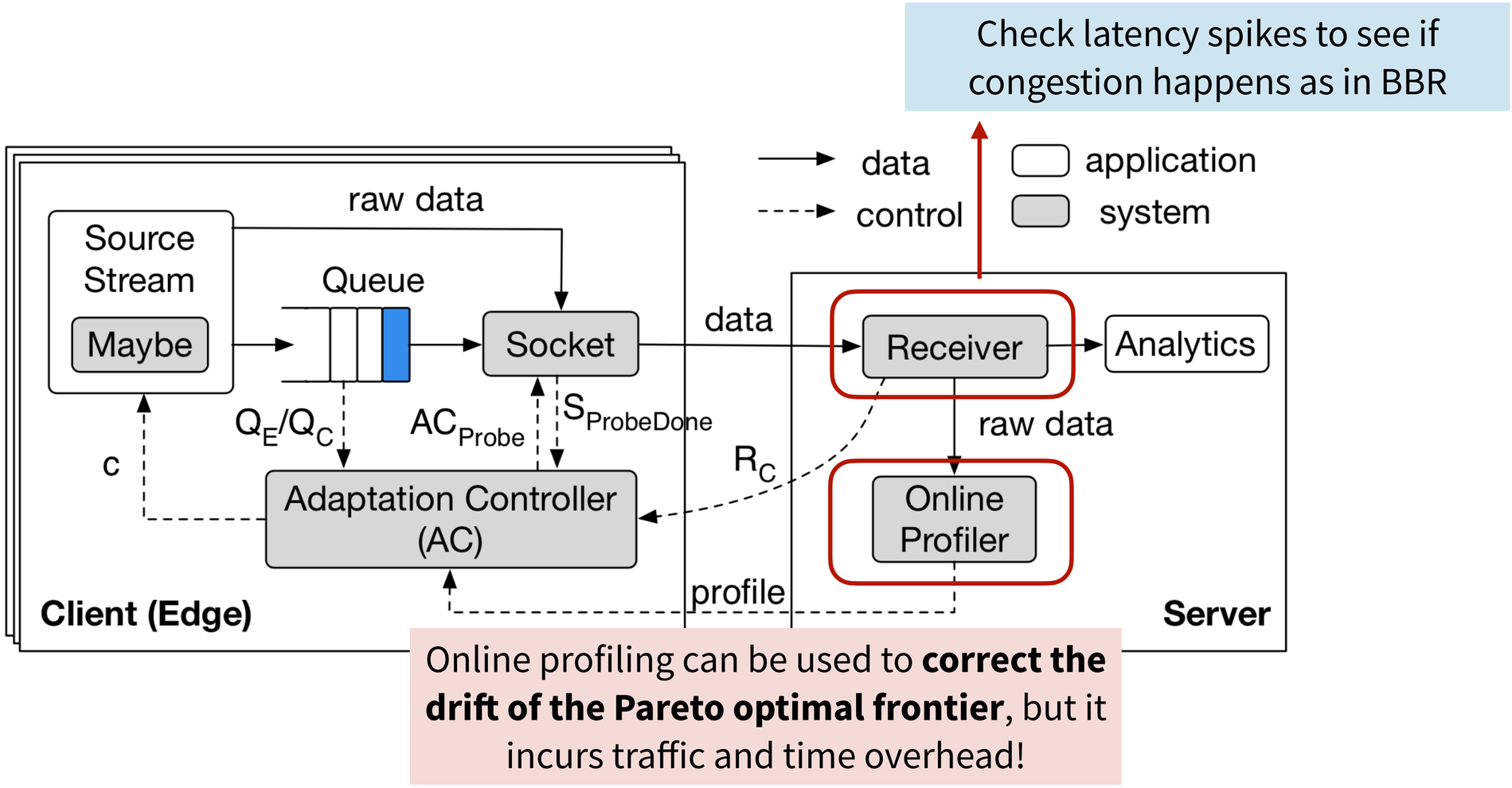


$$\mathbb{P} = \{c \in \mathbb{C} : \underbrace{\{c' \in \mathbb{C} : B(c') < B(c), A(c') > A(c)\}}_{\text{the set of better configuration c'}} = \varnothing\}$$

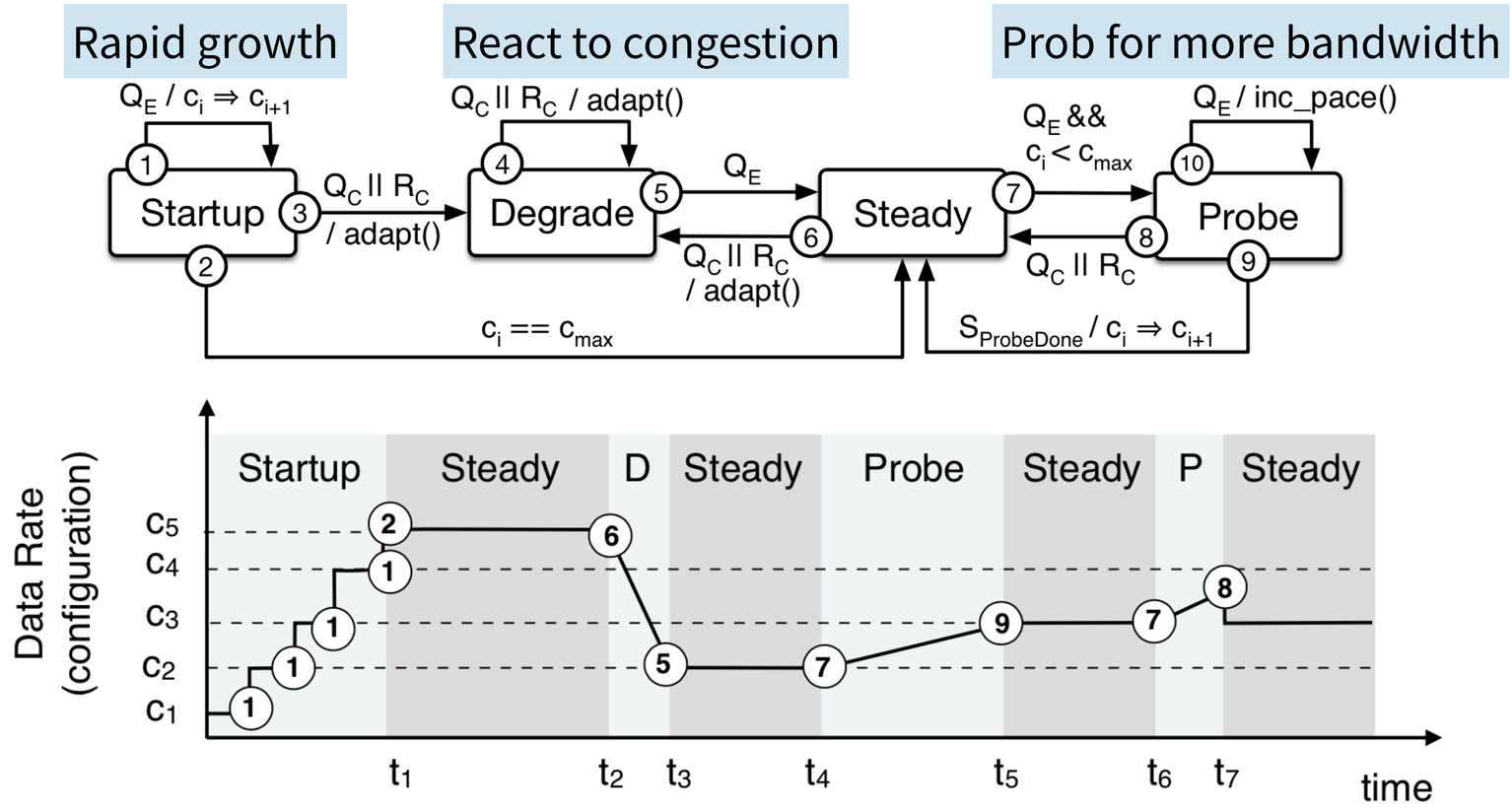
Pareto-optimality: You cannot further improve one of the goals without sacrificing the other

https://en.wikipedia.org/wiki/Pareto_efficiency

AWStream runtime adaptation



AWStream runtime adaptation



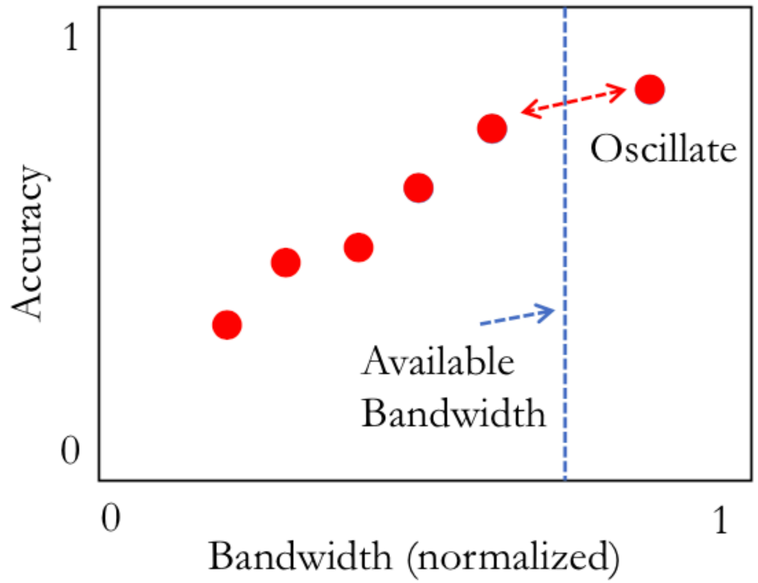
The system keeps probing for more bandwidth and maintains a steady state otherwise.

AWStream probing at runtime

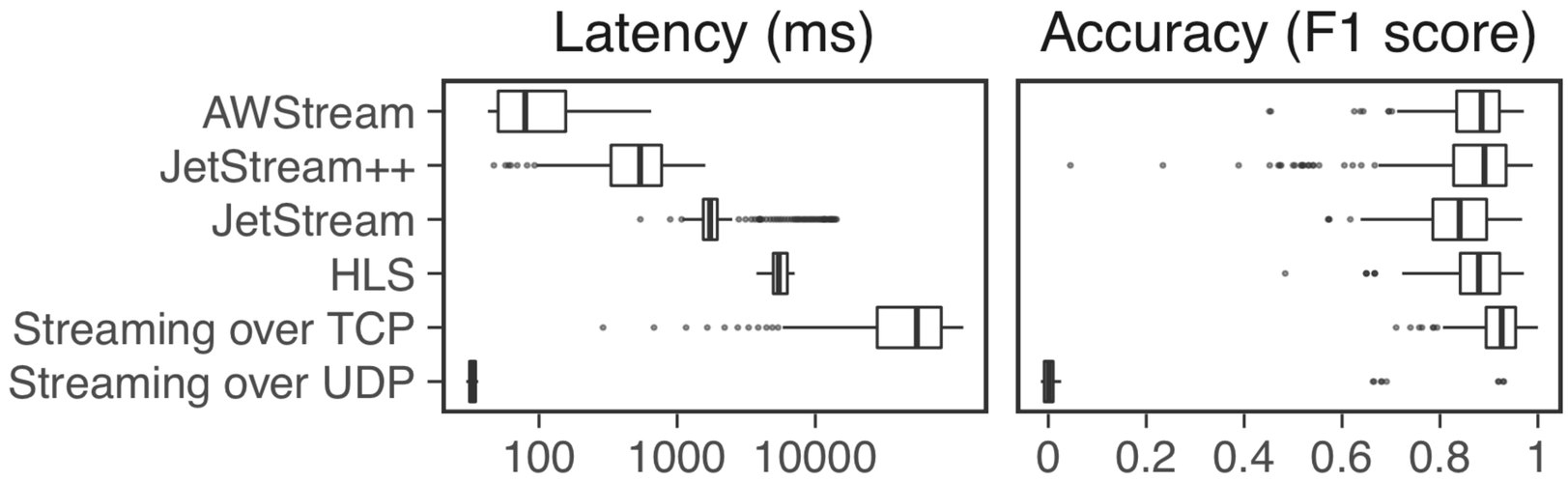
Configurations are discrete

Without probing, applications can jump to the next configuration that demands too much bandwidth and end up **oscillating** between configurations.

Probing (with dummy traffic) stabilizes adaptation



AWStream results

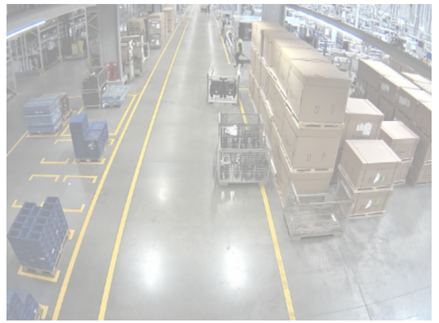


AWStream achieves good tradeoff between latency and accuracy.

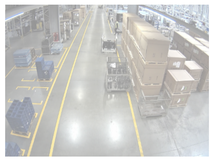
Opportunities

Video compression

- Reduce the video frame resolution or frame rate
- Use different codecs to achieve higher compression ratio



Compression



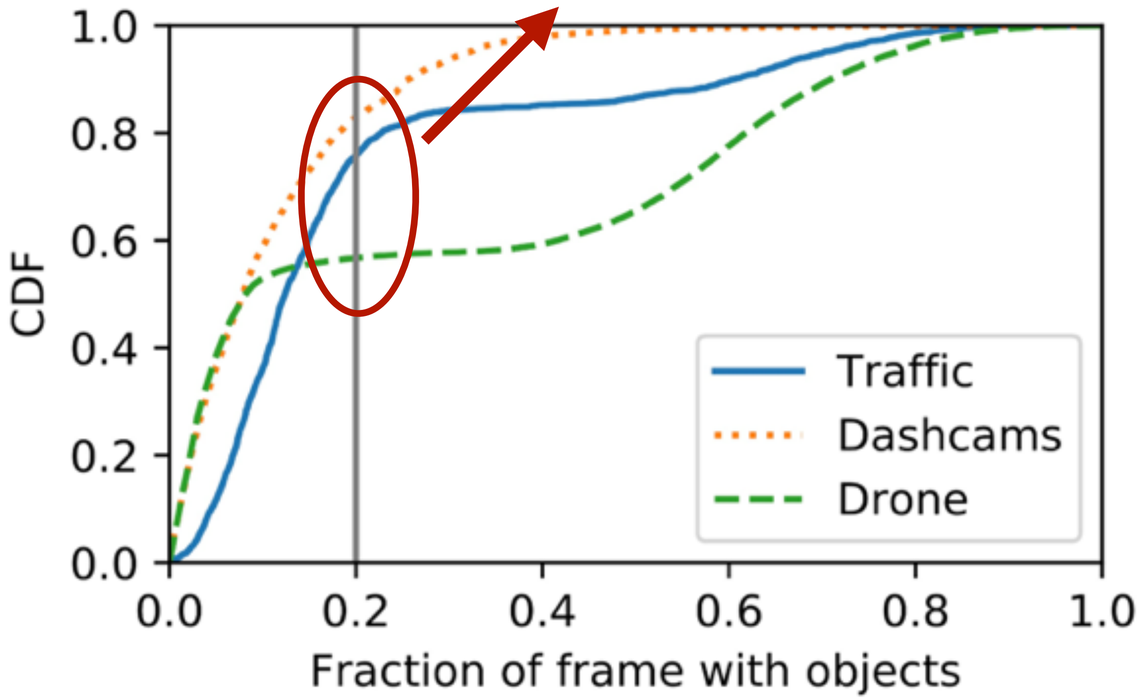
Content-based filtering

- **Early discard:** filter out irrelevant frames and drop them on device
- **Object filtering:** filter out the relevant objects and send only the part with objects

Filtering

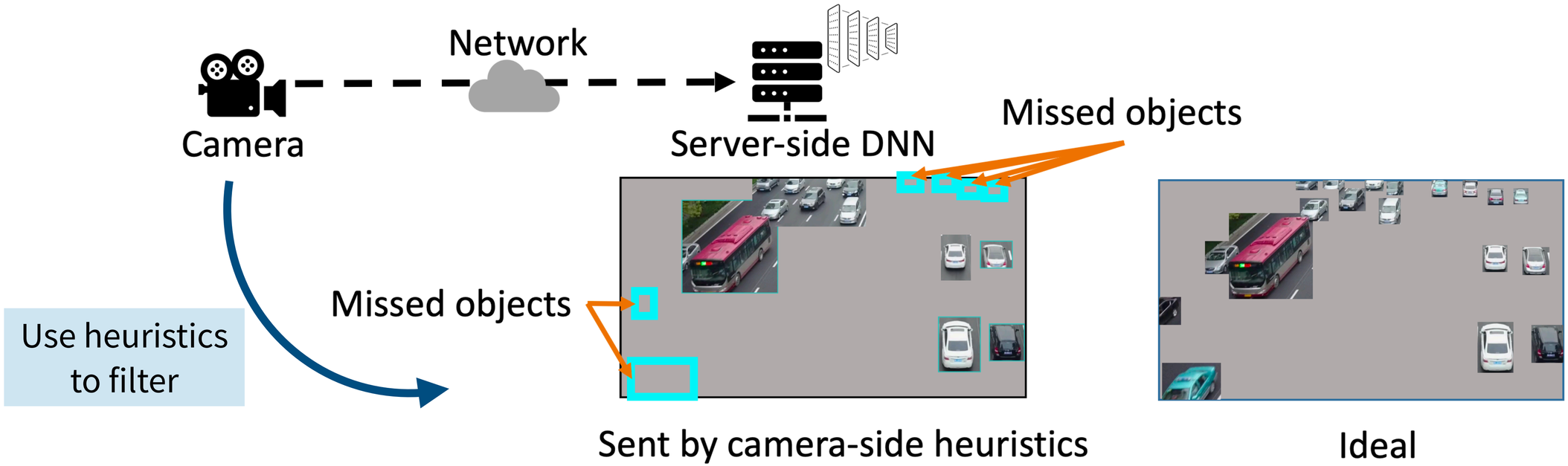
Filtering-based approaches

Objects only occupy less than 20% area of the whole frame in over 50% to 80% cases.



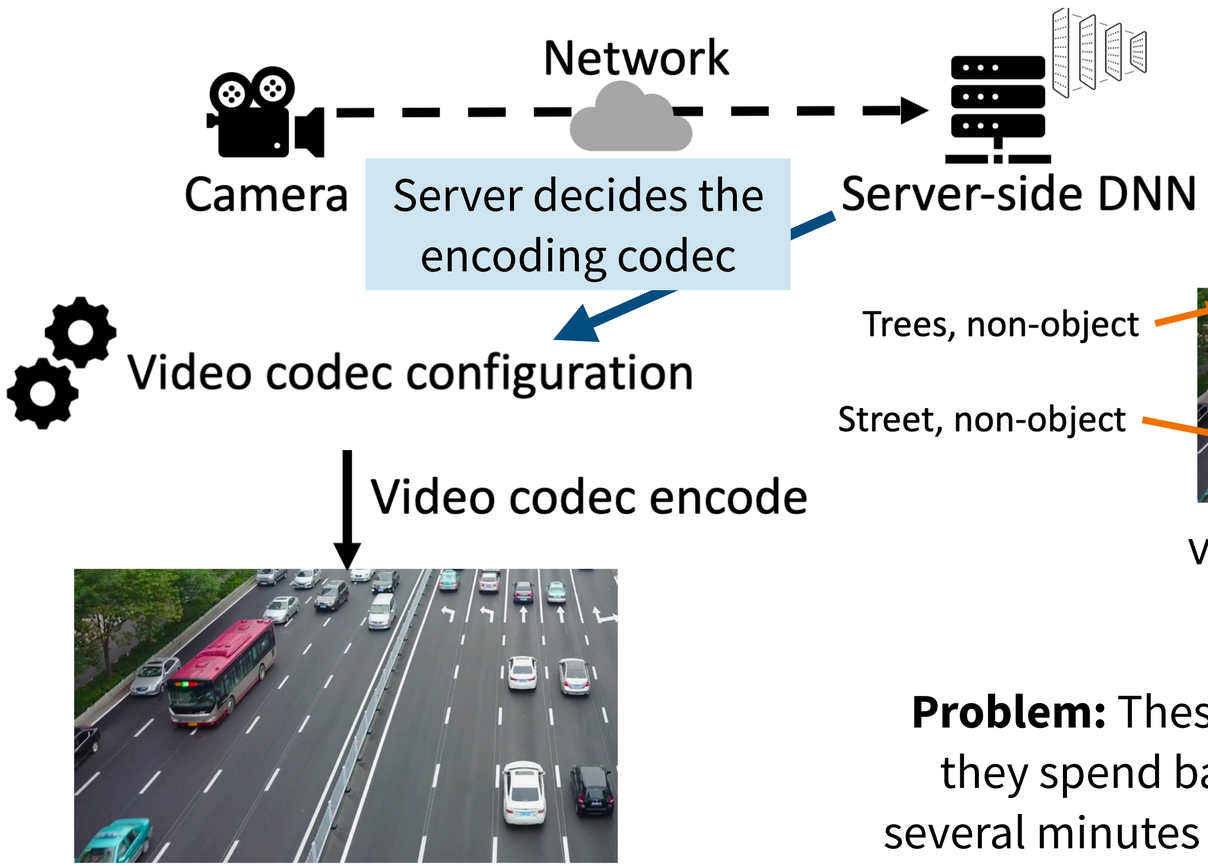
Motivation: Around 80% of the pixels in frames can be compressed without hurting accuracy!

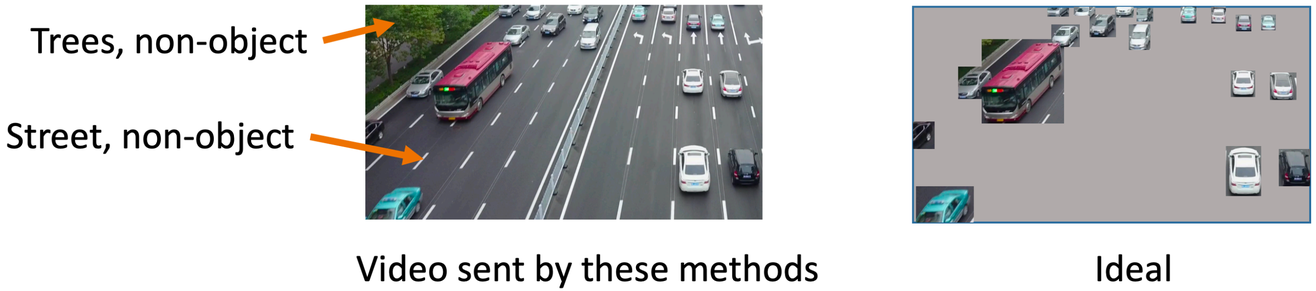
Client-side heuristics for filtering



Problem: Client-side heuristics are not accuracy and thus may miss many objects that will not be seen by the server-side DNNs.

Server-informed encoding for filtering





Video sent by these methods

Ideal

Problem: These approach can stream all objects, but they spend bandwidth on non-objects and require several minutes to new content (no real-time capability)

Motivation for new designs

The video streaming protocol should

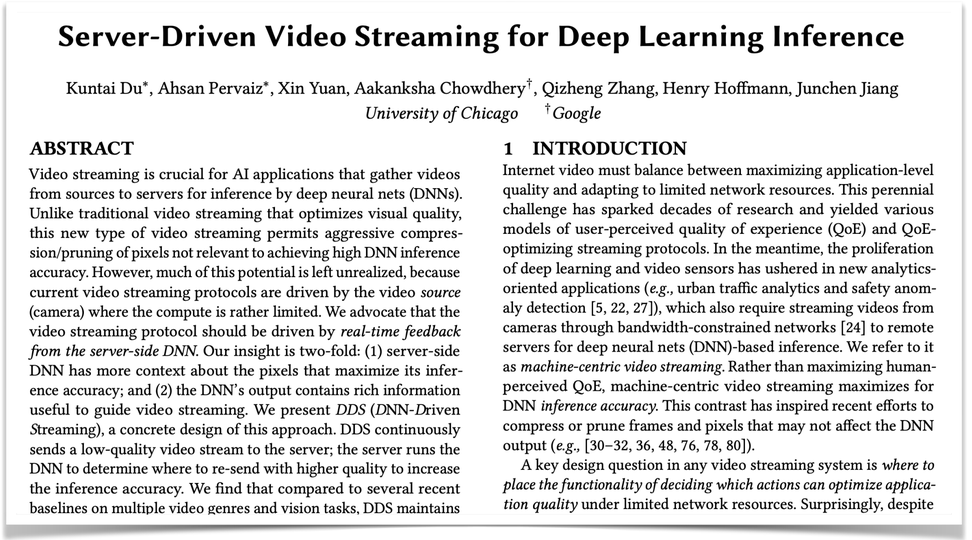
- Be completely driven by the server-side DNN feedback
- React to real-time video content

A chicken-egg problem

- Cameras need the **server-side feedback** of the current video to encode the current video

DDS: DNN-driven streaming

- Quickly see the video first and iterate on how to encode the video



Server-Driven Video Streaming for Deep Learning Inference

Kuntai Du*, Ahsan Pervaiz*, Xin Yuan, Aakanksha Chowdhery†, Qizheng Zhang, Henry Hoffmann, Junchen Jiang
University of Chicago †*Google*

ABSTRACT

Video streaming is crucial for AI applications that gather videos from sources to servers for inference by deep neural nets (DNNs). Unlike traditional video streaming that optimizes visual quality, this new type of video streaming permits aggressive compression/pruning of pixels not relevant to achieving high DNN inference accuracy. However, much of this potential is left unrealized, because current video streaming protocols are driven by the video *source* (camera) where the compute is rather limited. We advocate that the video streaming protocol should be driven by *real-time feedback from the server-side DNN*. Our insight is two-fold: (1) server-side DNN has more context about the pixels that maximize its inference accuracy; and (2) the DNN's output contains rich information useful to guide video streaming. We present *DDS* (*D*NN-*D*riven *S*treaming), a concrete design of this approach. DDS continuously sends a low-quality video stream to the server; the server runs the DNN to determine where to re-send with higher quality to increase the inference accuracy. We find that compared to several recent baselines on multiple video genres and vision tasks, DDS maintains

1 INTRODUCTION

Internet video must balance between maximizing application-level quality and adapting to limited network resources. This perennial challenge has sparked decades of research and yielded various models of user-perceived quality of experience (QoE) and QoE-optimizing streaming protocols. In the meantime, the proliferation of deep learning and video sensors has ushered in new analytics-oriented applications (*e.g.*, urban traffic analytics and safety anomaly detection [5, 22, 27]), which also require streaming videos from cameras through bandwidth-constrained networks [24] to remote servers for deep neural nets (DNN)-based inference. We refer to it as *machine-centric video streaming*. Rather than maximizing human-perceived QoE, machine-centric video streaming maximizes for DNN *inference accuracy*. This contrast has inspired recent efforts to compress or prune frames and pixels that may not affect the DNN output (*e.g.*, [30–32, 36, 48, 76, 78, 80]).

A key design question in any video streaming system is *where to place the functionality of deciding which actions can optimize application quality* under limited network resources. Surprisingly, despite

ACM SIGCOMM 2020

Feedback regions depend on the analytics results

Camera field-of-view

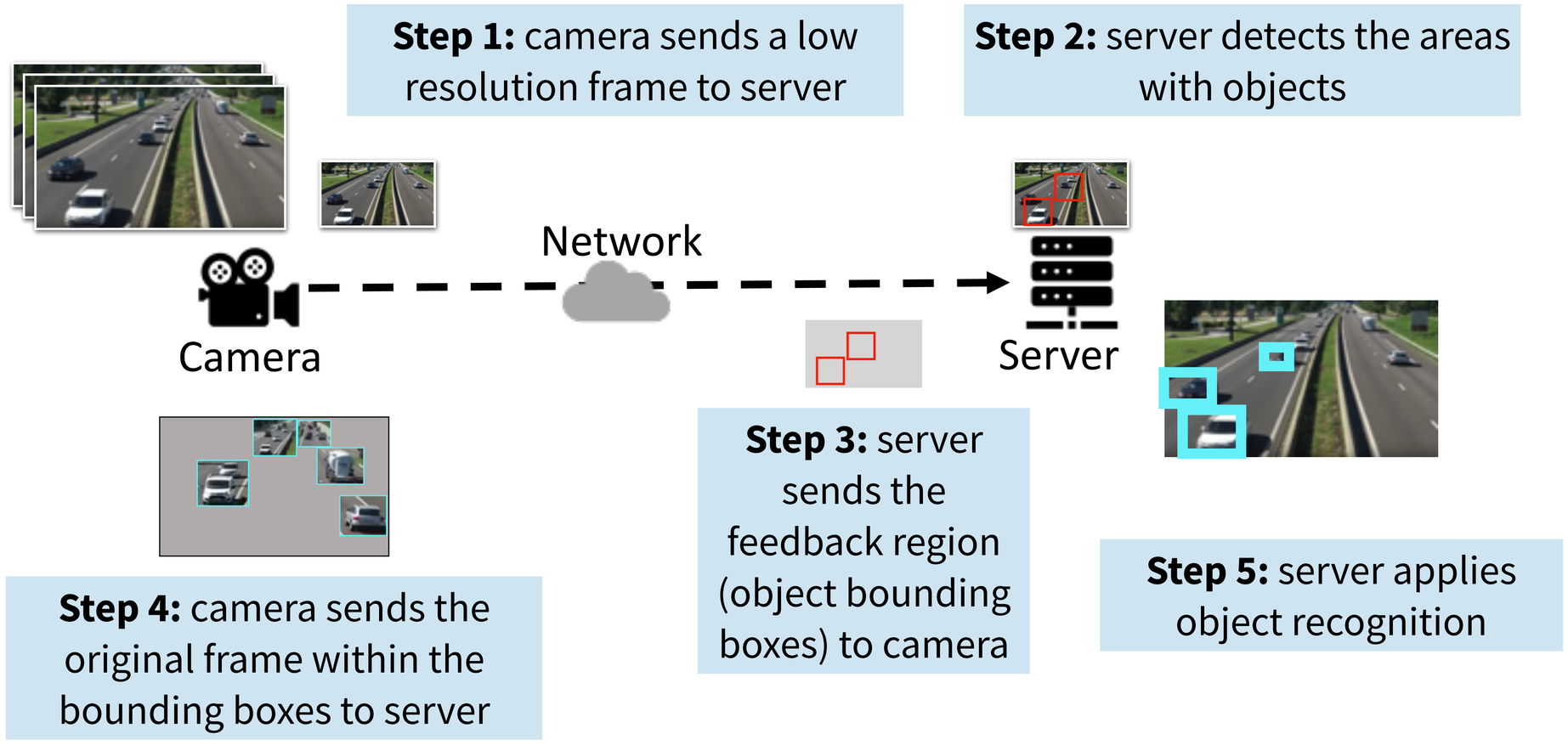
When a car just enters the field-of-view of the camera, it is too small to be detected on low-quality frame. → **Need to be encoded with high quality**

Camera field-of-view

The car become large enough to be detected when it reaches the center of the camera field-of-view. → **No need to be encoded with high quality**

DDS: DNN-driven streaming

An iterative approach for server-based video stream analytics



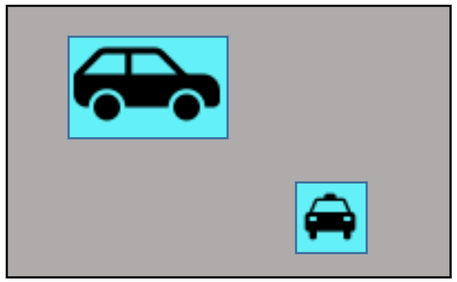
Challenges in DDS

How to generate feedback regions?

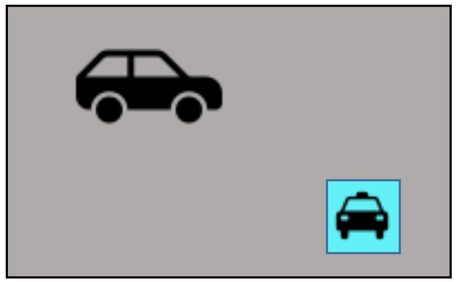
How to reduce end-to-end analytics latency?

General guidelines for feedback region generation

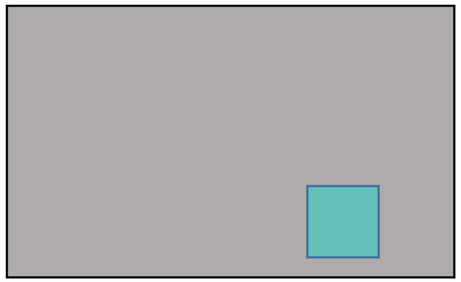
Generate regions that may contain objects



Eliminate regions that overlap with high-confidence inference results



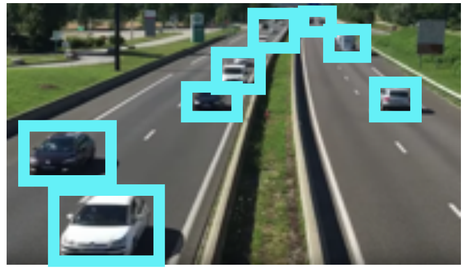
Encode remaining regions with codec-friendly bounding boxes



General idea: DNN finds the region which it is not confident about!

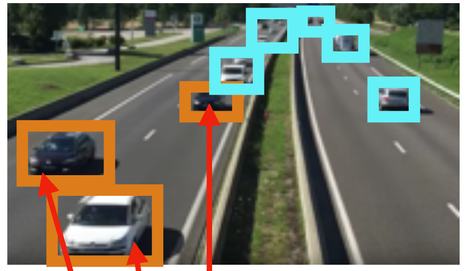
Example: object detection

Generate regions that may contain objects



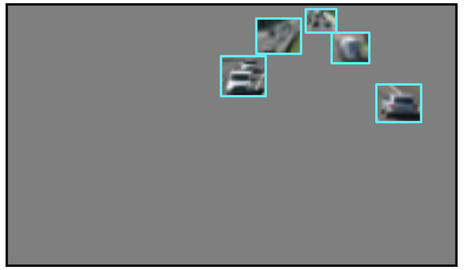
Generate from the intermediate output of the DNN (e.g., Faster-RCNN, YoLo)

Eliminate regions that overlap with high-confidence inference results



DNNs are already confident so we can eliminate them

Encode remaining regions with codec-friendly bounding boxes



Example: semantic segmentation

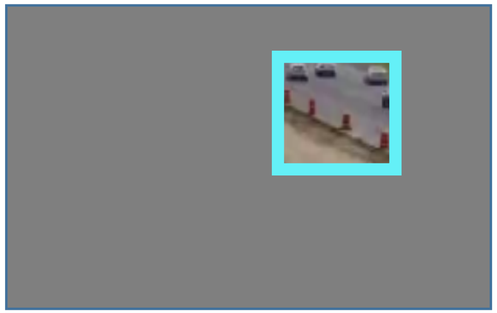
Generate regions that may contain objects

Label each pixel with the sum of the probability of all non-background classes. (Brighter the pixel, higher the probability.)

Eliminate regions that overlap with high-confidence inference results

DNNs are already confident so we can eliminate them

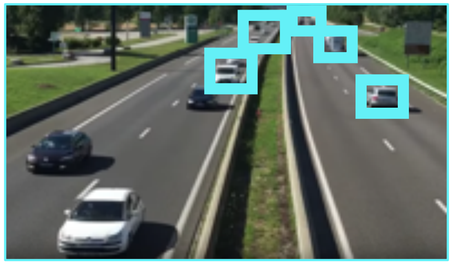
Encode remaining regions with codec-friendly bounding boxes



Comparison to other region-based streaming

High-quality regions sent to the server

DDS

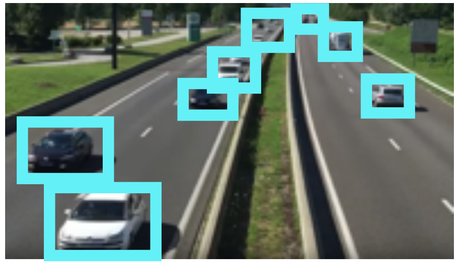


Only sends the regions that the DNN is not confident with

Vigil (MobiCom'15)

Sends the regions that the DNN is confident with, but misses the other regions

EAAR (MobiCom'19)



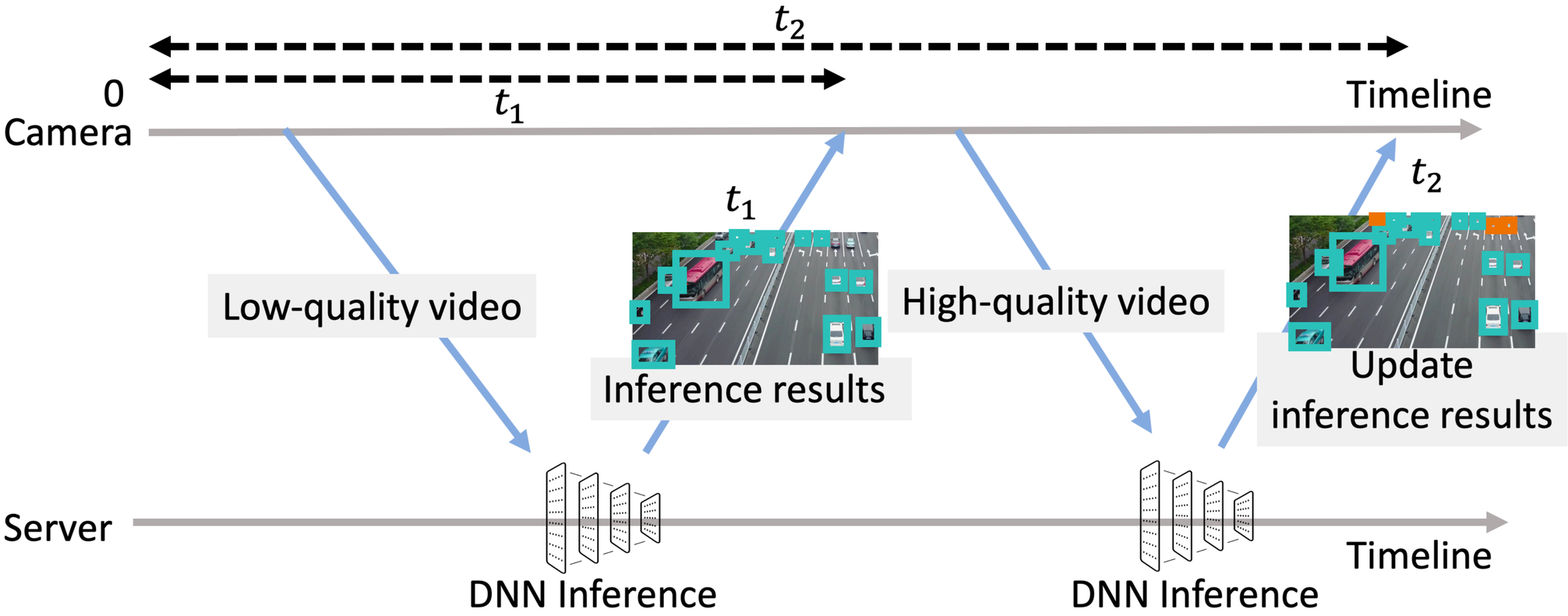
Sends all regions regardless of the confidence level of the DNN

Challenges in DDS

How to generate feedback regions?

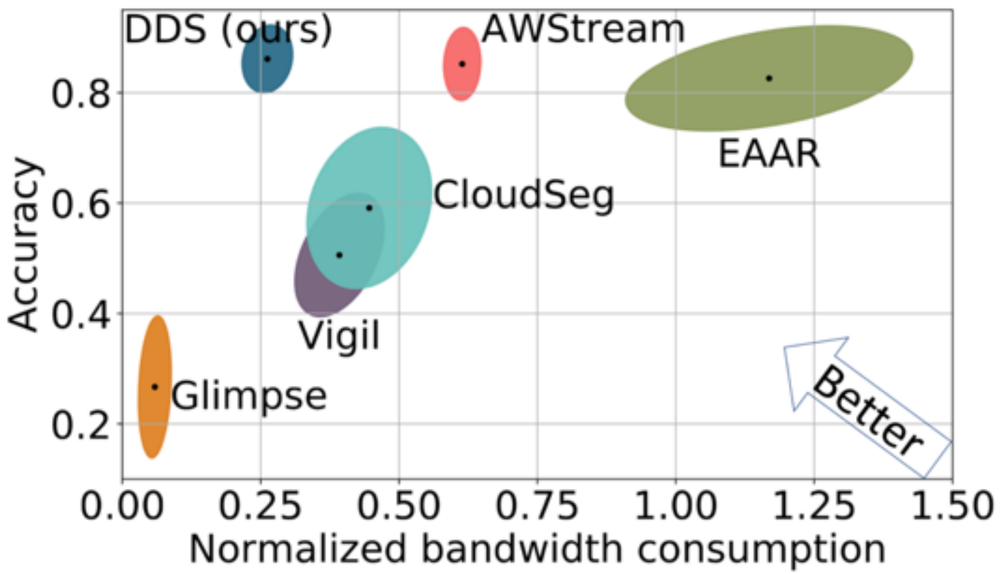
How to reduce end-to-end analytics latency?

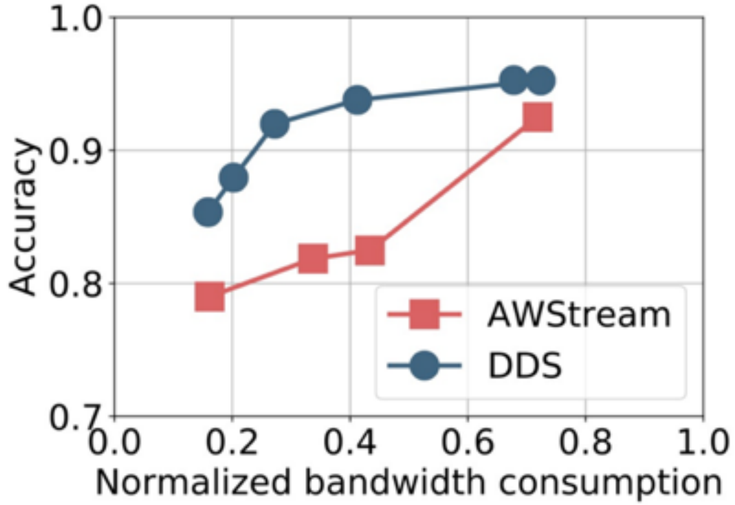
Breakdown of the end-to-end analytics latency



90% of the results are returned in the first inference round and can be returned quickly.

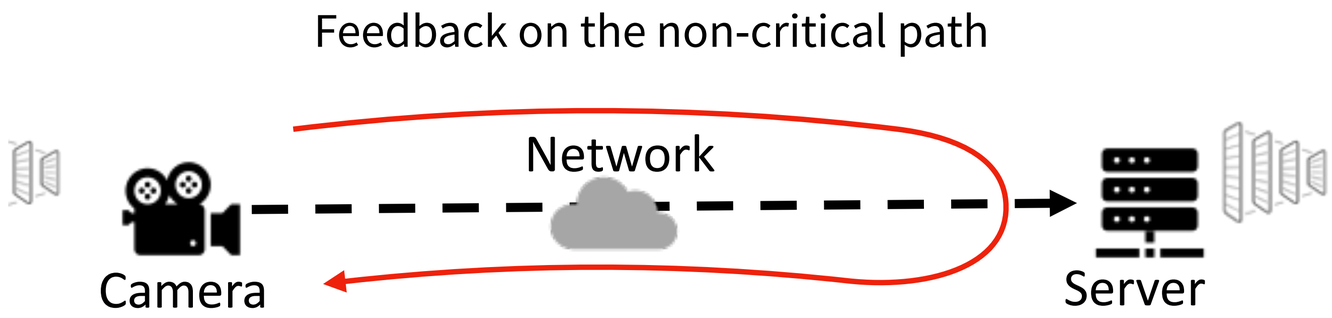
DDS results





DDS is able to achieve the **best tradeoff between bandwidth and accuracy** among all existing solutions, and outperforms AWStream.

Alternative design: remove the server from the critical path



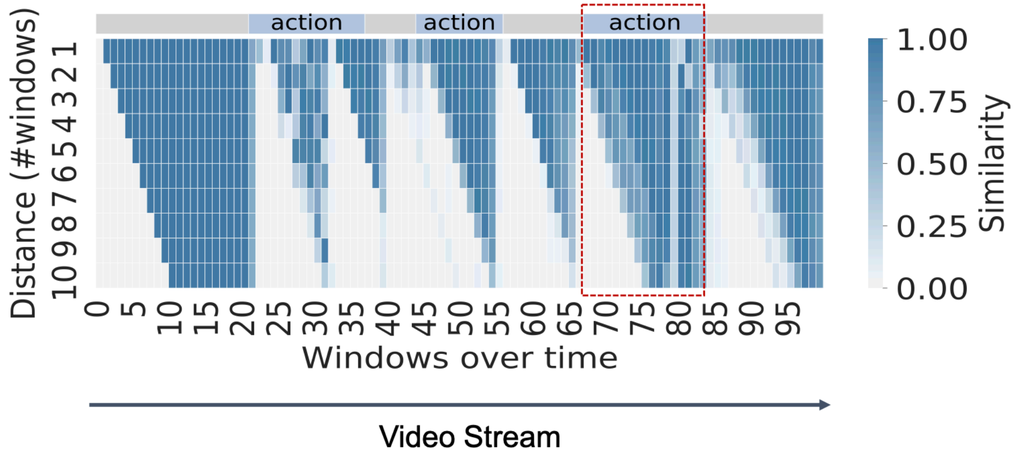
Run a smaller DNN at the camera for fast response on the critical path

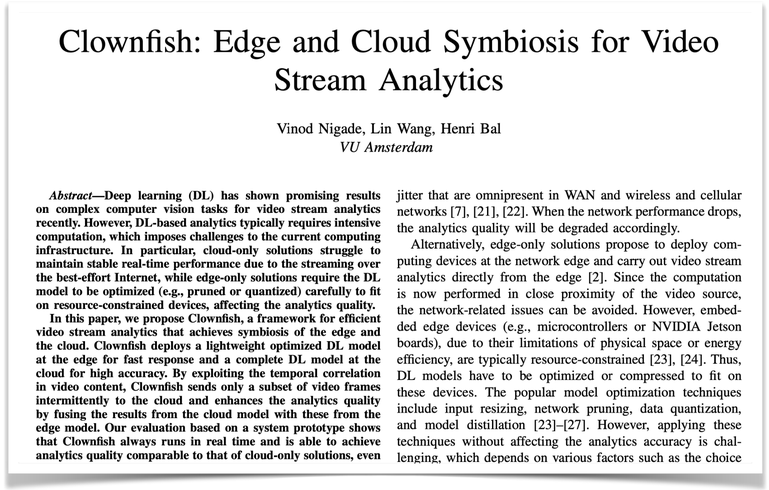
Run a larger DNN at the server to help the camera-local model to achieve high accuracy

Clownfish: hybrid video analytics

Motivation: leverage temporal correlation in video content

- Video has significant temporal correlation across frames
- For window-based inference such as action recognition, windows have overlapping frames





Clownfish: Edge and Cloud Symbiosis for Video Stream Analytics

Vinod Nigade, Lin Wang, Henri Bal
VU Amsterdam

***Abstract*—Deep learning (DL) has shown promising results on complex computer vision tasks for video stream analytics recently. However, DL-based analytics typically requires intensive computation, which imposes challenges to the current computing infrastructure. In particular, cloud-only solutions struggle to maintain stable real-time performance due to the streaming over the best-effort Internet, while edge-only solutions require the DL model to be optimized (e.g., pruned or quantized) carefully to fit on resource-constrained devices, affecting the analytics quality.**

In this paper, we propose Clownfish, a framework for efficient video stream analytics that achieves symbiosis of the edge and the cloud. Clownfish deploys a lightweight optimized DL model at the edge for fast response and a complete DL model at the cloud for high accuracy. By exploiting the temporal correlation in video content, Clownfish sends only a subset of video frames intermittently to the cloud and enhances the analytics quality by fusing the results from the cloud model with these from the edge model. Our evaluation based on a system prototype shows that Clownfish always runs in real time and is able to achieve analytics quality comparable to that of cloud-only solutions, even jitter that are omnipresent in WAN and wireless and cellular networks [7], [21], [22]. When the network performance drops, the analytics quality will be degraded accordingly.

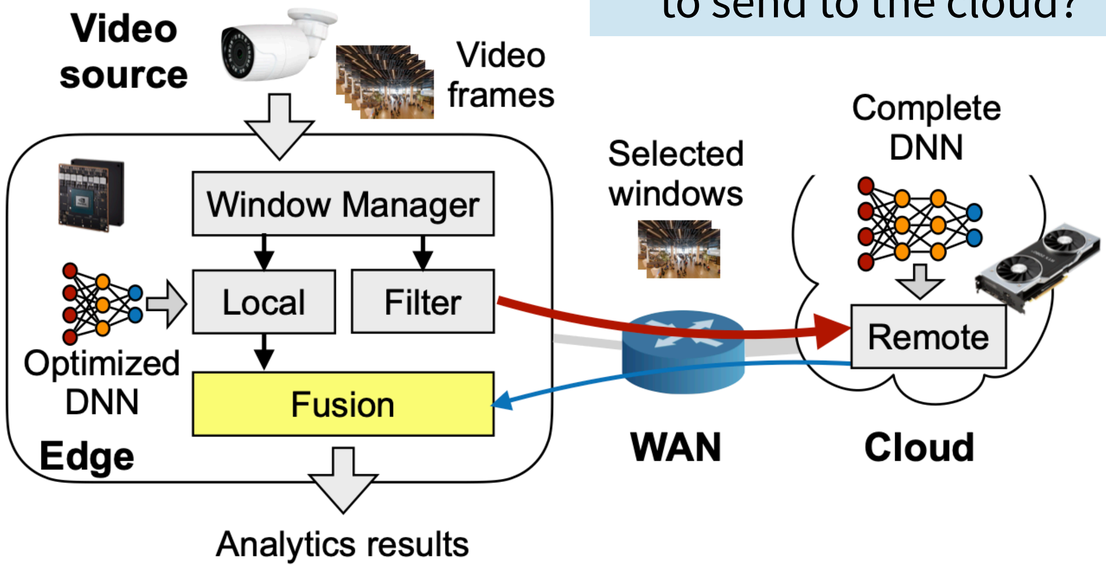
Alternatively, edge-only solutions propose to deploy computing devices at the network edge and carry out video stream analytics directly from the edge [2]. Since the computation is now performed in close proximity of the video source, the network-related issues can be avoided. However, embedded edge devices (e.g., microcontrollers or NVIDIA Jetson boards), due to their limitations of physical space or energy efficiency, are typically resource-constrained [23], [24]. Thus, DL models have to be optimized or compressed to fit on these devices. The popular model optimization techniques include input resizing, network pruning, data quantization, and model distillation [23]–[27]. However, applying these techniques without affecting the analytics accuracy is challenging, which depends on various factors such as the choice

IEEE/ACM SEC 2020

https://linwang.info/docs/sec20-clownfish.pdf

Clownfish design

Challenge 2: What frames to send to the cloud?



Challenge 1: How to fuse the results?

Goal: achieve a symbiosis relationship between the edge and the cloud for real-time video stream analytics

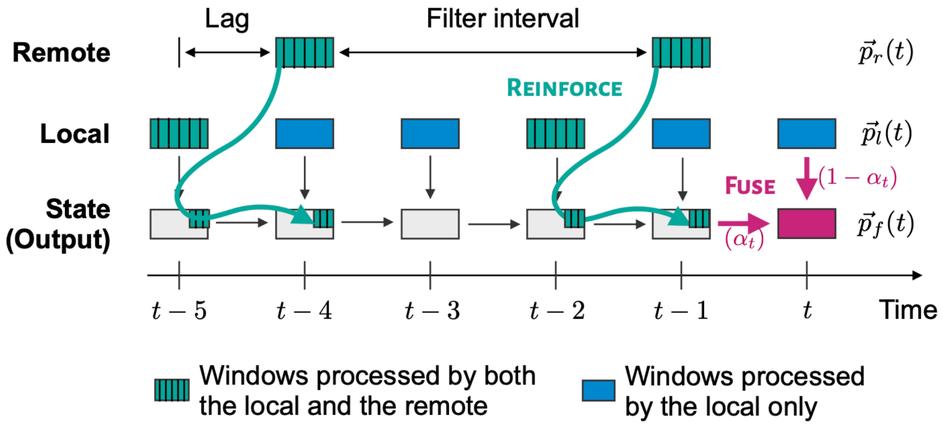
Clownfish components

Fusion

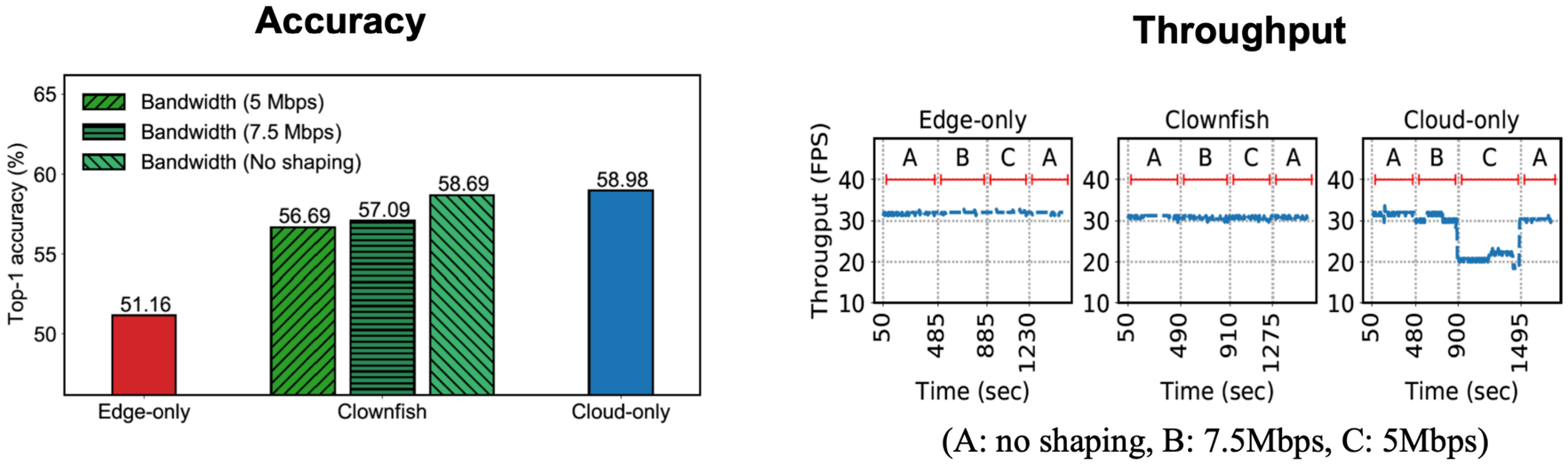
- Use a weighted sum to combine the **edge state** and the current edge analytics results
- The weight is obtained by estimating the **temporal correlation** between the frames
- Use an **exponential smoothing approach** to update the edge state with intermittent cloud feedbacks

Filter

- Send a frame at **context switching** points (use the temporal correlation to decide the context)
- **Periodically** send frames at a target bandwidth consumption



Clownfish results

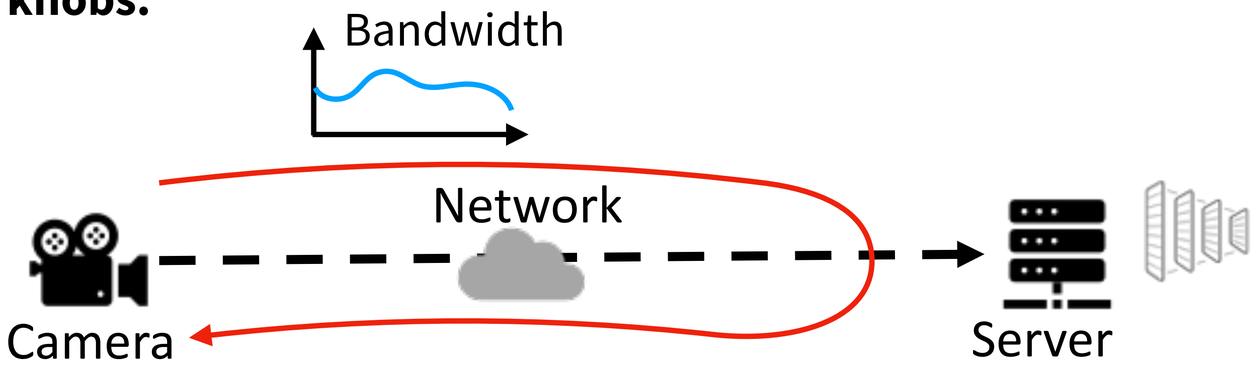


Clownfish achieves **comparable accuracy to the cloud-only solutions** with large DNNs and **real-time throughput (30fps)** as in the edge-only solutions.

Final bonus assignment

Goal: guarantee the end-to-end latency is no more than 33ms

Possible adaptation knobs:
- Frame resolution
- DNN model size



Task: design a control logo at the camera to achieve the goal!

If you have some ideas, come talk to us and we can make a plan!

Summary

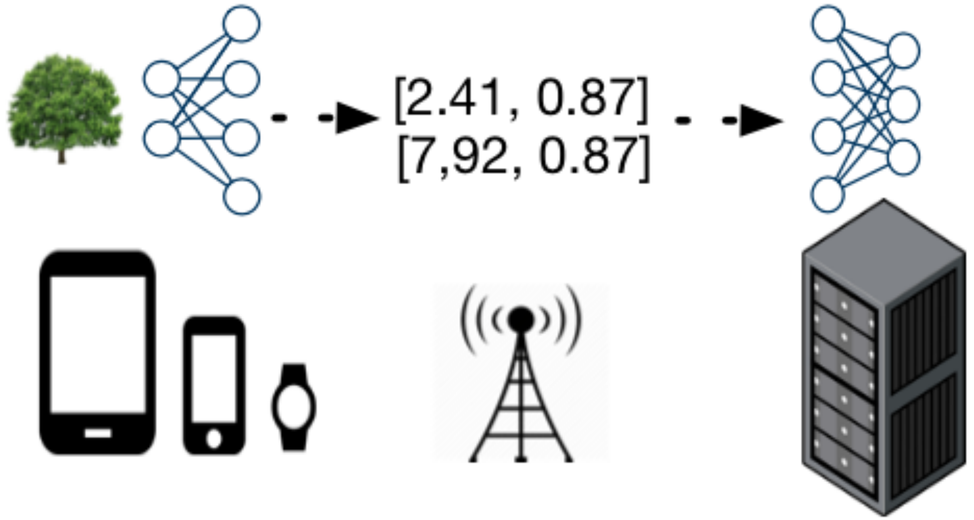
What are the goals of video stream analytics?

- Achieve high analytics accuracy
- Save network bandwidth
- Reduce analytics latency

How to optimize for these goals

- **AWStream:** explore and conduct automatic tradeoff between bandwidth and accuracy
- **DDS:** ask the DNN to give a quick feedback to decide the streaming strategy
- **Clownfish:** remove the server (or the cloud) from the critical path and use it only for accuracy improvements

Next time: networking for ML



How to design networked systems to better support machine learning applications?