Case Study: Systems & Machine Learning

COS 316: Principles of Computer System Design
Lecture 23

Neil Agarwal
December 5, 2023

This class: Designing Computer Systems

- Networked systems, operating systems, distributed systems, database systems, etc.
- Design aspects: naming, layering, concurrency, security, caching, etc.

Systems <-> Machine Learning

- Two main flavors:
  - Machine learning for Systems
  - Systems for Machine Learning

Lecture Outline

- Intro to ML for Systems
- Intro to Systems for ML
- ML for Systems Case Studies
  - *Learning Relaxed Belady for Content Distributional Network Caching*
  - *Neural Adaptive Video Streaming with Pensive*
- Systems for ML Case Studies
  - *Pipedream: Generalized Pipeline Parallelism for DNN Training*
  - *Gemel: Model Merging for Memory-Efficient, Real-Time Video Analytics at the Edge*

Machine Learning for Systems

- Systems rely on many *heuristic* design decisions
  - Congestion control (how many bits to send over the network without causing network congestion?)
  - Caching policies (which object to evict from the cache?)
  - Load balancing (which server to direct traffic to?)
  - …
- ML for Systems: replace heuristics with data-driven approaches (ML)

Benefits of Using ML in Systems

- Tailor design for a specific environment
  - Data, workload, and operating conditions
- Handle hard-to-model system dynamics
  - E.g., interference between workloads on shared resources like CPU caches
- Optimize for high-level system objectives directly
  - E.g., job completion time, rather than low level-metrics like server utilization
- Learn data-driven heuristics for hard algorithmic problems
  - E.g., scheduling often involves combinatorial optimization problems with no general efficient algorithms

Slide Credits: Mohammad Alizadeh

Machine Learning for Systems Case Studies

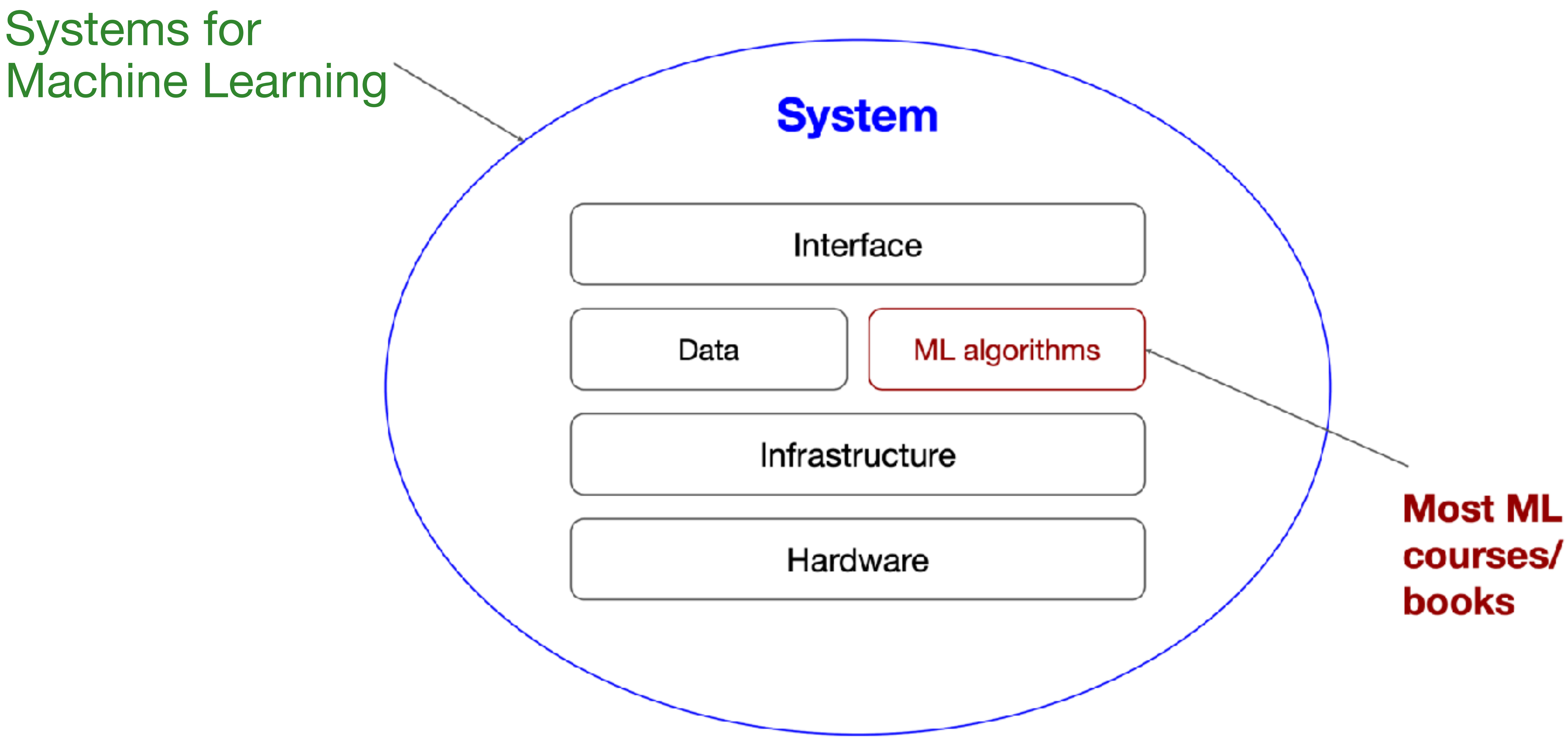
- *Learning Relaxed Belady for Content Distributional Network Caching*
  - Using ML to predict which object to evict from the cache
- *Neural Adaptive Video Streaming with Pensive*
  - Using ML to predict the rate at which you should stream video data

Lecture Outline

- Intro to ML for Systems
- Intro to Systems for ML
- ML for Systems Case Studies
  - *Learning Relaxed Belady for Content Distributional Network Caching*
  - *Neural Adaptive Video Streaming with Pensive*
- Systems for ML Case Studies
  - *Pipedream: Generalized Pipeline Parallelism for DNN Training*
  - *Gemel: Model Merging for Memory-Efficient, Real-Time Video Analytics at the Edge*

Systems for Machine Learning

How to make ML algorithms work with other parts to solve real world problems



Slide Credits: Pooyan Jamshidi

Systems for Machine Learning

How to make ML algorithms work with other parts to solve real world problems

- Defining interfaces, algorithms, data, infrastructure, and hardware
- With the goal of satisfying specified requirements (reliability, scalability, maintainability, adaptability, efficiency)
- Example questions
  - How can we run models more efficiently on this heterogeneous cluster of hardware?
  - How can we reduce network or memory bottlenecks?
  - How can we scale to more data?

Slide Credits: Pooyan Jamshidi

Systems for Machine Learning Case Studies

- *Pipedream: Generalized Pipeline Parallelism for DNN Training*
  - Better scheduling techniques for large-scale machine learning training jobs
- *Gemel: Model Merging for Memory-Efficient, Real-Time Video Analytics at the Edge*
  - Reducing GPU memory bottlenecks for video analytics inference on edge servers

Systems <-> Machine Learning

- Two main flavors:
  - Machine learning for Systems
    - Replacing system heuristics/control with ML algorithms
  - Systems for Machine Learning
    - Optimizing system level aspects to improve the machine learning pipeline (e.g., training, inference)

Lecture Outline

- Intro to ML for Systems
- Intro to Systems for ML
- ML for Systems Case Studies
    - *Learning Relaxed Belady for Content Distributional Network Caching*
    - *Neural Adaptive Video Streaming with Pensive*
- Systems for ML Case Studies
    - *Pipedream: Generalized Pipeline Parallelism for DNN Training*
    - *Gemel: Model Merging for Memory-Efficient, Real-Time Video Analytics at the Edge*

ML For Systems Case Study #1: Caching

Learning Relaxed Belady for Content Distribution Network Caching

Zhenyu Song
Princeton University

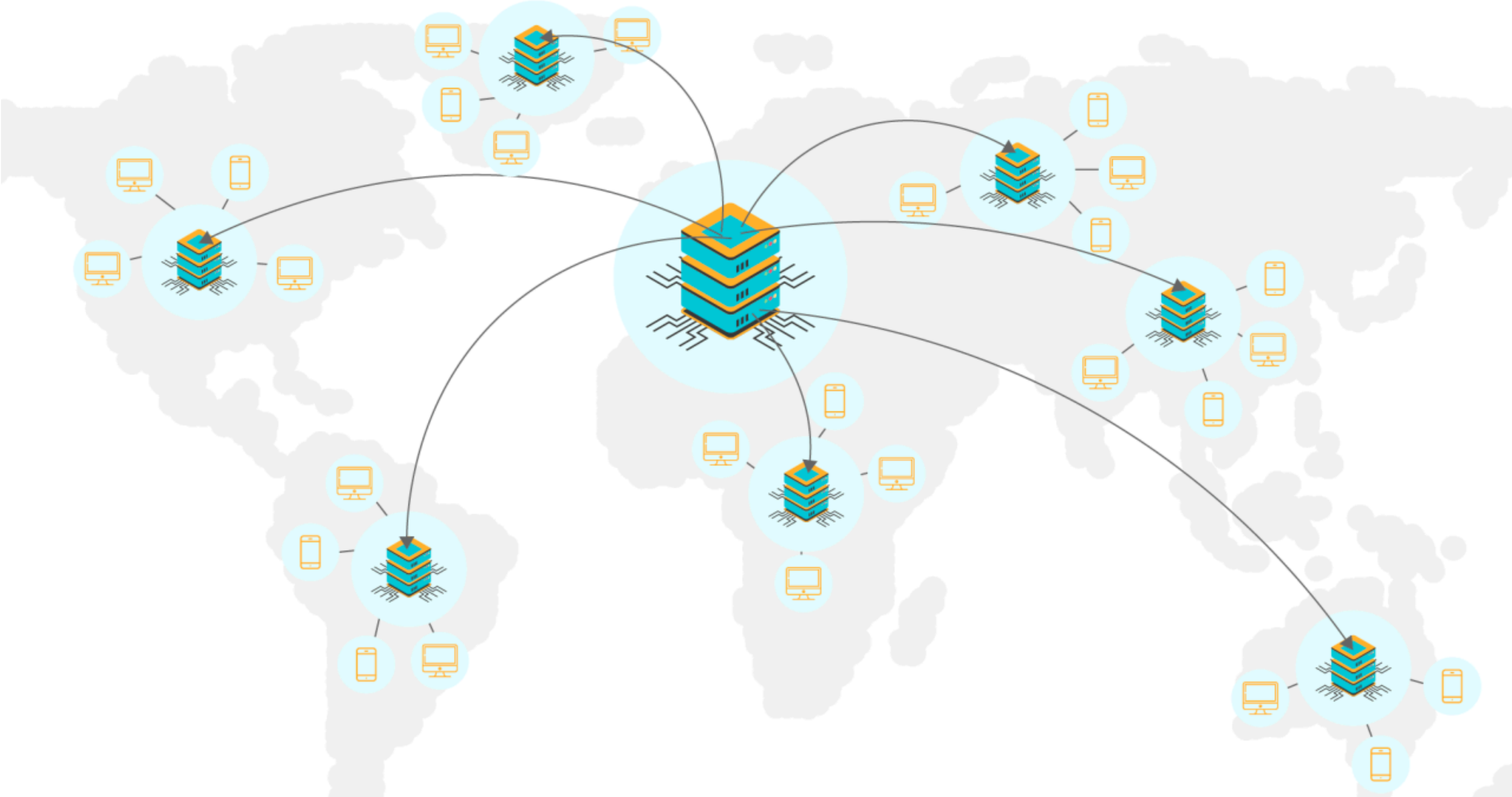
Daniel S. Berger
Microsoft Research & CMU

Kai Li
Princeton University

Wyatt Lloyd
Princeton University

Content Delivery Network (CDN)

CDNs store cached content on edge servers in point-of-presence (POP locations that are close to users, to minimize latency and bandwidth costs.



CDN Caching Goal: Minimize Cache Misses

Goal: reduce cache misses (and requests to cloud server)!

Key question: when cache is full, which object should the cache evict?

Cache Heuristic Algorithms

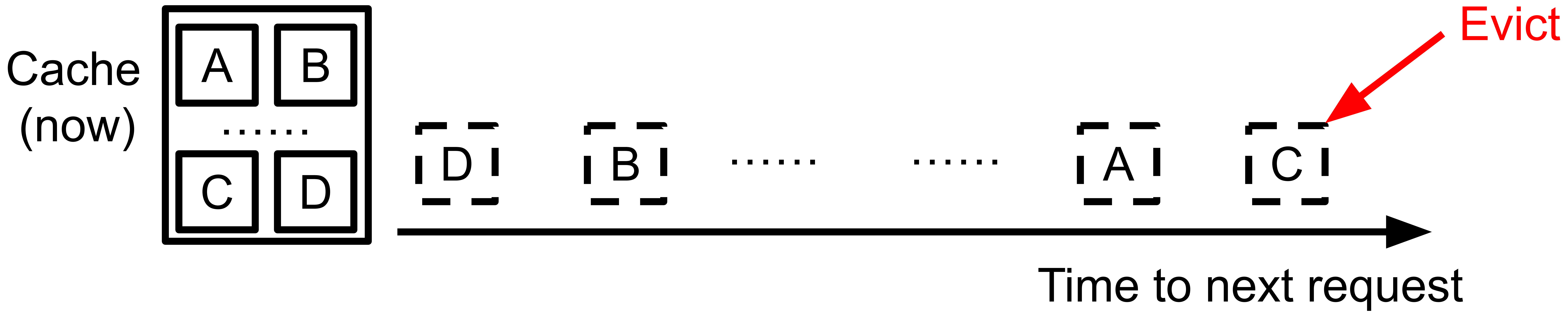
- Which object to evict?
  - First in first out (FIFO)
  - Least Recently Used (LRU)
  - Least Frequently Used (LFU)
  - ...

Oracle Algorithm: Belady's MIN Algorithm

- Oracle algorithm: optimal algorithm if you knew all future requests
  - In reality, this is not possible…
- Belady's MIN Algorithm:
  - Evict object in cache with the furthest next request
- Goal of this paper:
  - Approximate Belady's MIN algorithm

Challenge: Hard to Mimic Belady (Oracle) Algorithm

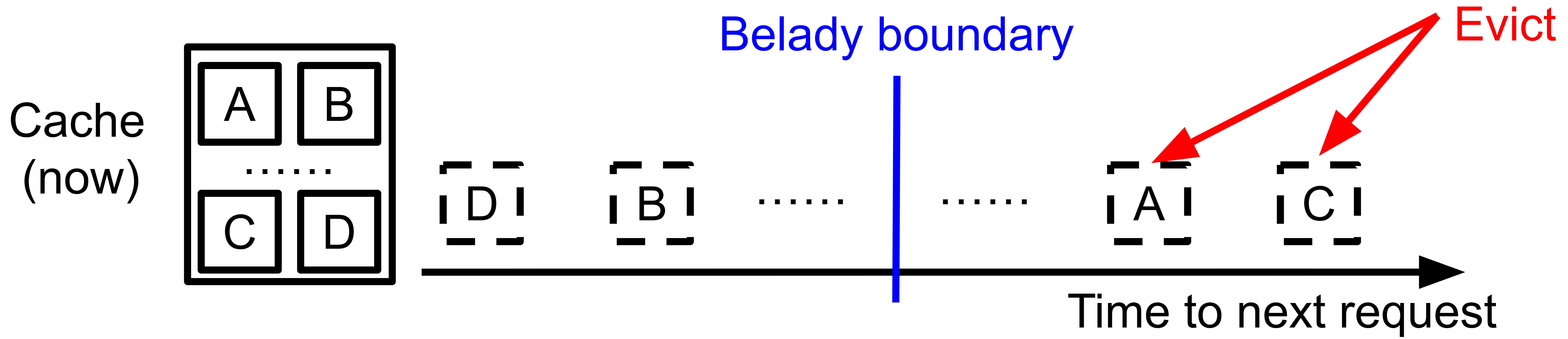
Belady: evict object with next access farthest in the future



Mimicking exact Belady is impractical

- Need predictions for all objects → prohibitive computational cost
- Need exact prediction of next access → further prediction are harder

Introducing the Relaxed Belady Algorithm



Observation: many objects are good candidates for eviction

Relaxed Belady evicts an objects beyond boundary

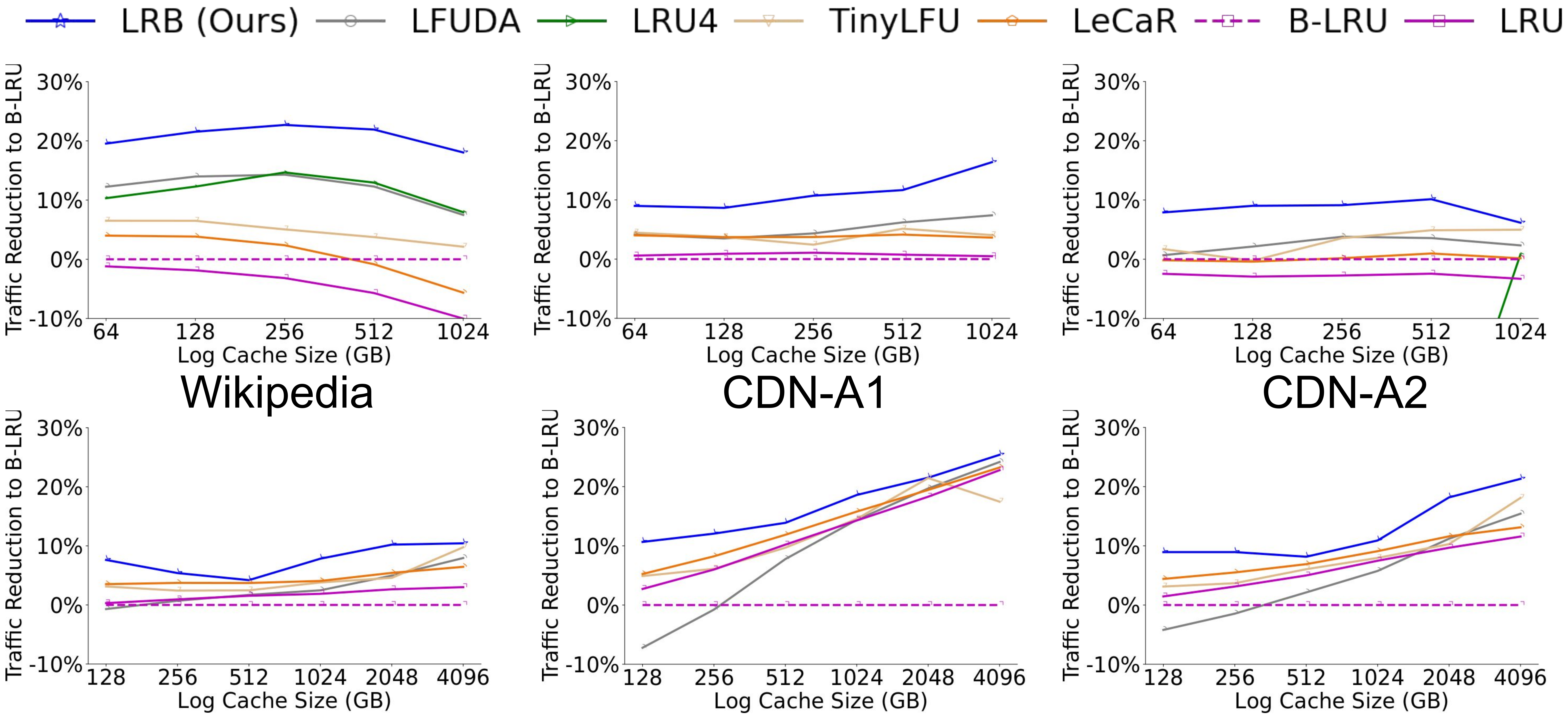
- Do not need predictions for all objects → reasonable computation
- No need to differentiate beyond boundary → simplifies the prediction

Slide Credits: Zhenyu Song

Learning a Relaxed Belady Algorithm

- ML prediction problem: for a subsample of objects in the cache, predict whether their future request time is beyond the “belady boundary”; if so, evict!
- Features:
  - Object size
  - Object type
  - Inter-request distances (recency)
  - Exponential Decay Counters (long term frequencies)
- Model Architecture: gradient boosting decision trees

LRB Consistently Improves on the State of the Art



Wikipedia CDN-A1 CDN-A2

CDN-B1 CDN-B2 CDN-B3

Slide Credits: Zhenyu Song

ML For Systems Case Study #1: Caching

Learning Relaxed Belady for Content Distribution Network Caching

Zhenyu Song, *Princeton University*
Daniel S. Berger, *Microsoft Research & CMU*
Kai Li, *Princeton University*
Wyatt Lloyd, *Princeton University*

- Key insight: use machine learning to approximate an oracle caching algorithm
- Paper & presentation available @ https://www.usenix.org/conference/nsdi20/presentation/song

Lecture Outline

- Intro to ML for Systems
- Intro to Systems for ML
- ML for Systems Case Studies
    - *Learning Relaxed Belady for Content Distributional Network Caching*
    - *Neural Adaptive Video Streaming with Pensive*
- Systems for ML Case Studies
    - *Pipedream: Generalized Pipeline Parallelism for DNN Training*
    - *Gemel: Model Merging for Memory-Efficient, Real-Time Video Analytics at the Edge*

ML For Systems Case Study #2: Video Streaming

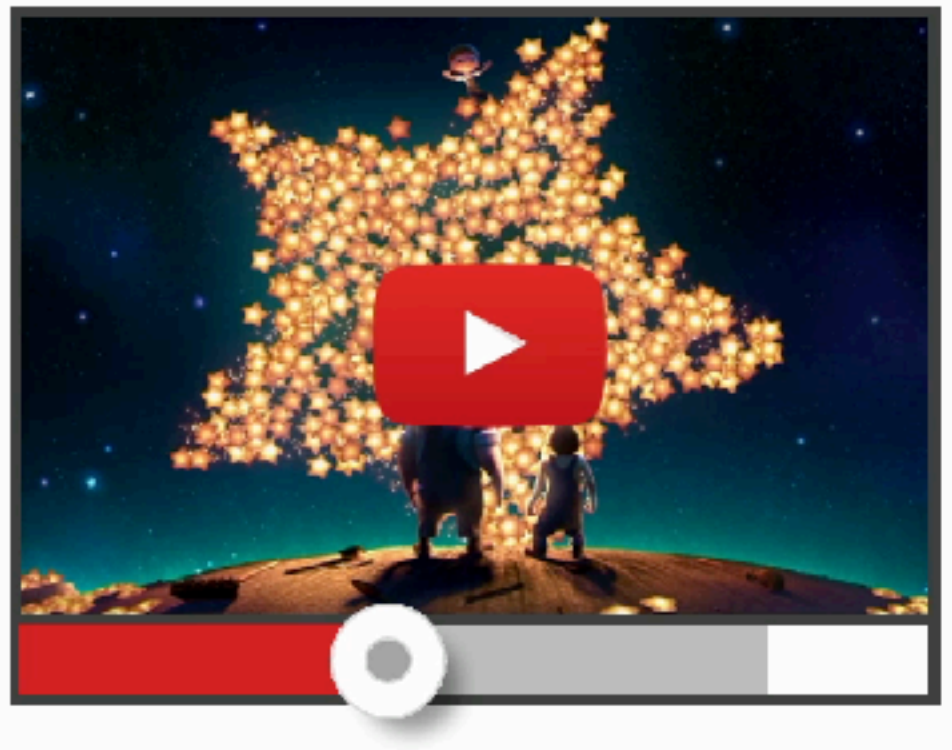
Neural Adaptive Video Streaming with Pensieve

Hongzi Mao, Ravi Netravali, Mohammad Alizadeh
MIT Computer Science and Artificial Intelligence Laboratory
{hongzi,ravinet,alizadeh}@mit.edu

Users start leaving if video doesn’t play in 2 seconds

Slide Credits: Hongzi Mao

Dynamic Streaming over HTTP (DASH)

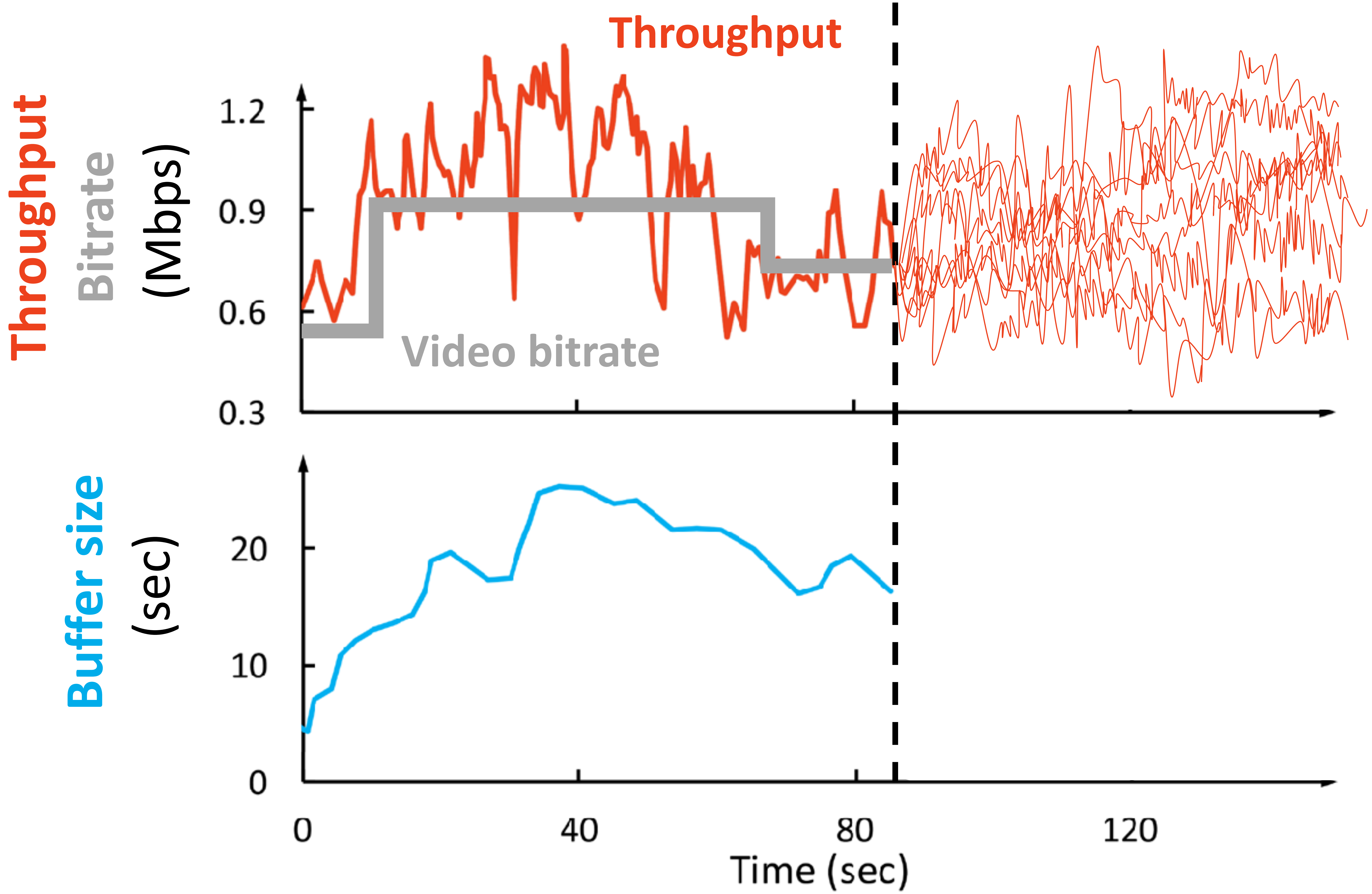


Video Client

Video Server

Slide Credits: Hongzi Mao

Why is ABR Challenging?



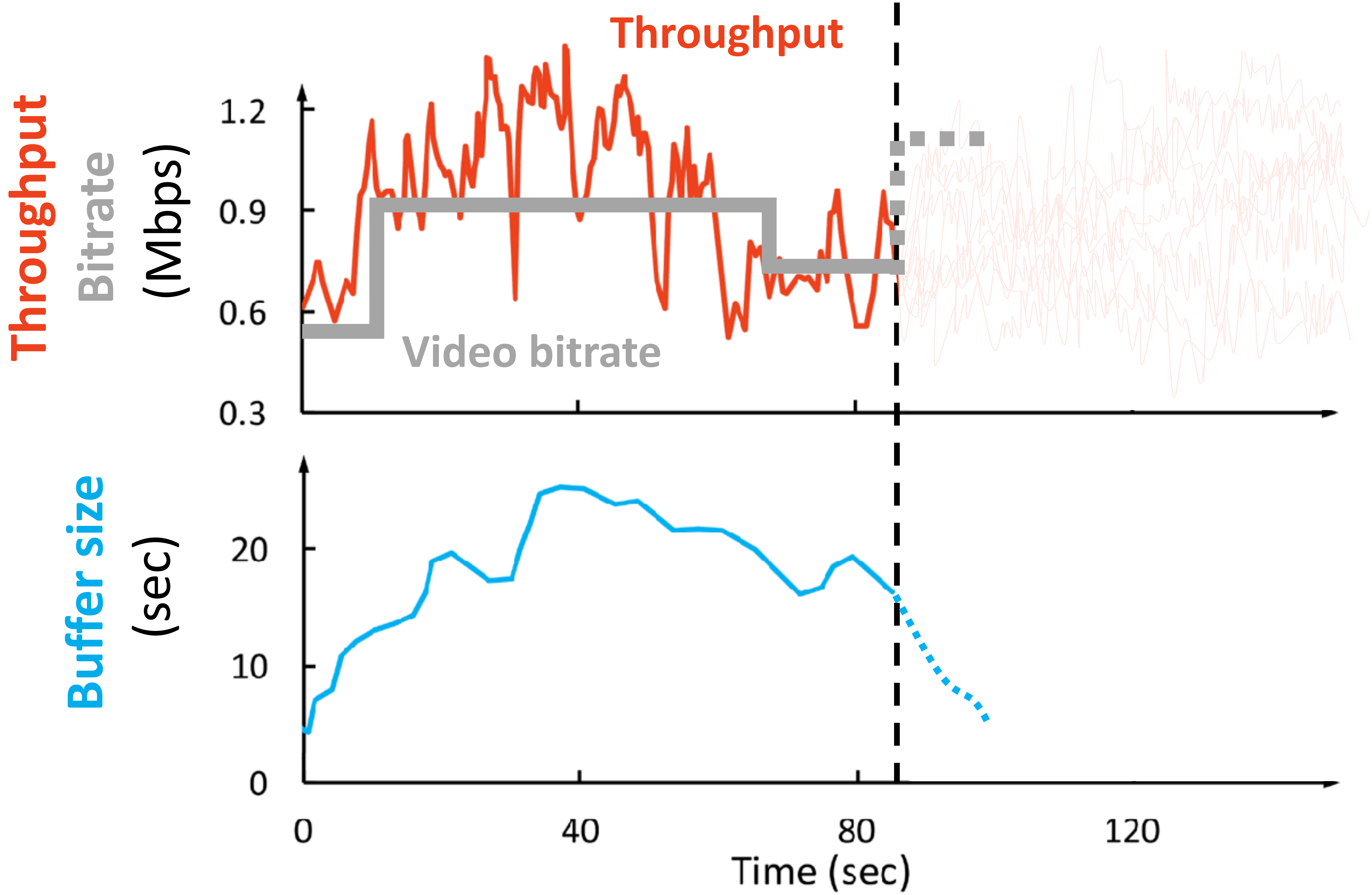
Network throughput is variable & uncertain

Conflicting QoE goals

- Bitrate
- Rebuffering time
- Smoothness

Slide Credits: Hongzi Mao

Why is ABR Challenging?



Network throughput is variable & uncertain

Conflicting QoE goals

- Bitrate
- Rebuffering time
- Smoothness

Cascading effects of decisions

Slide Credits: Hongzi Mao

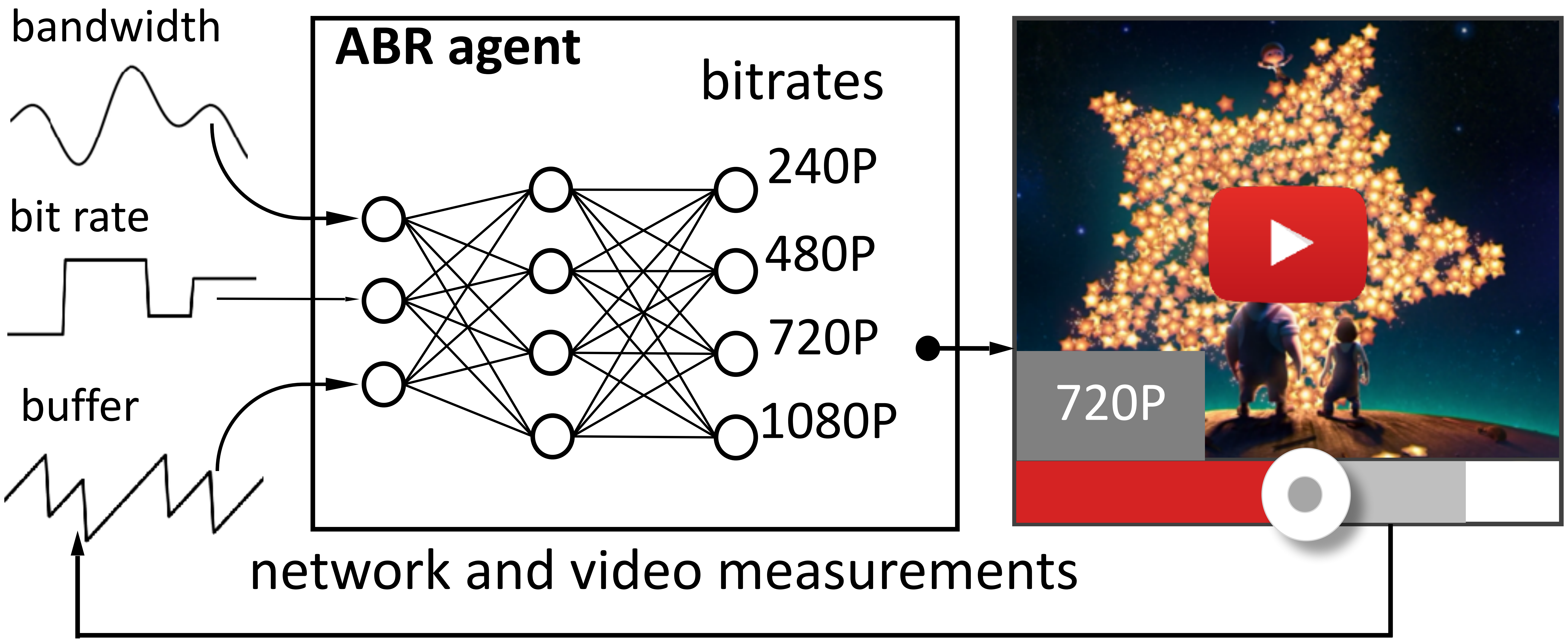
Previous Fixed ABR Algorithms

- Rate-based: pick bitrate based on predicted throughput
  - FESTIVE [CoNEXT'12], PANDA [JSAC'14], CS2P [SIGCOMM'16]
- Buffer-based: pick bitrate based on buffer occupancy
  - BBA [SIGCOMM'14], BOLA [INFOCOM'16]
- Hybrid: use both throughput prediction & buffer occupancy
  - PBA [HotMobile'15], MPC [SIGCOMM'15]

Simplified inaccurate model leads to suboptimal performance

Slide Credits: Hongzi Mao

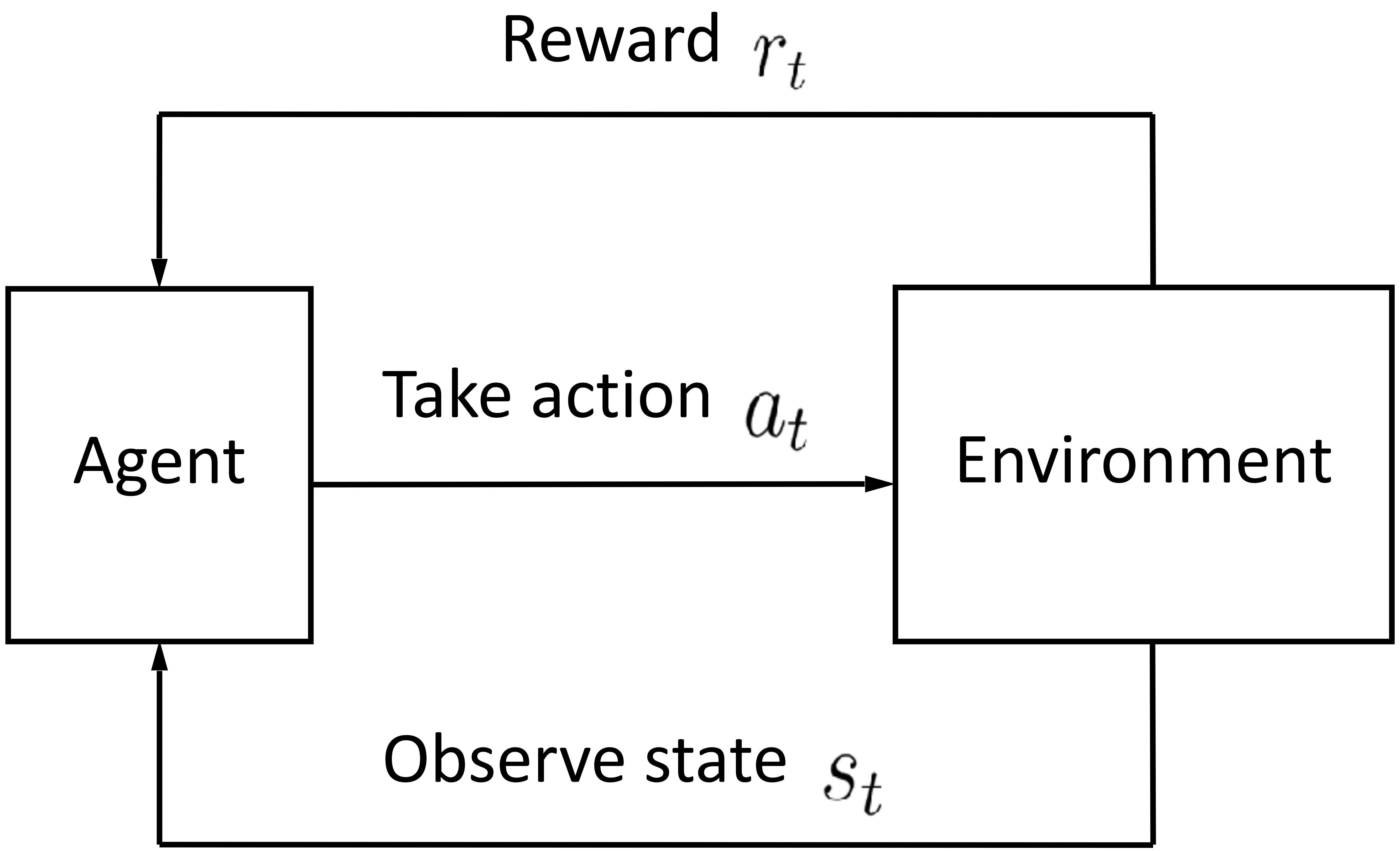
Our Contribution: Pensieve



Pensieve learns ABR algorithm automatically through experience

Slide Credits: Hongzi Mao

Reinforcement Learning



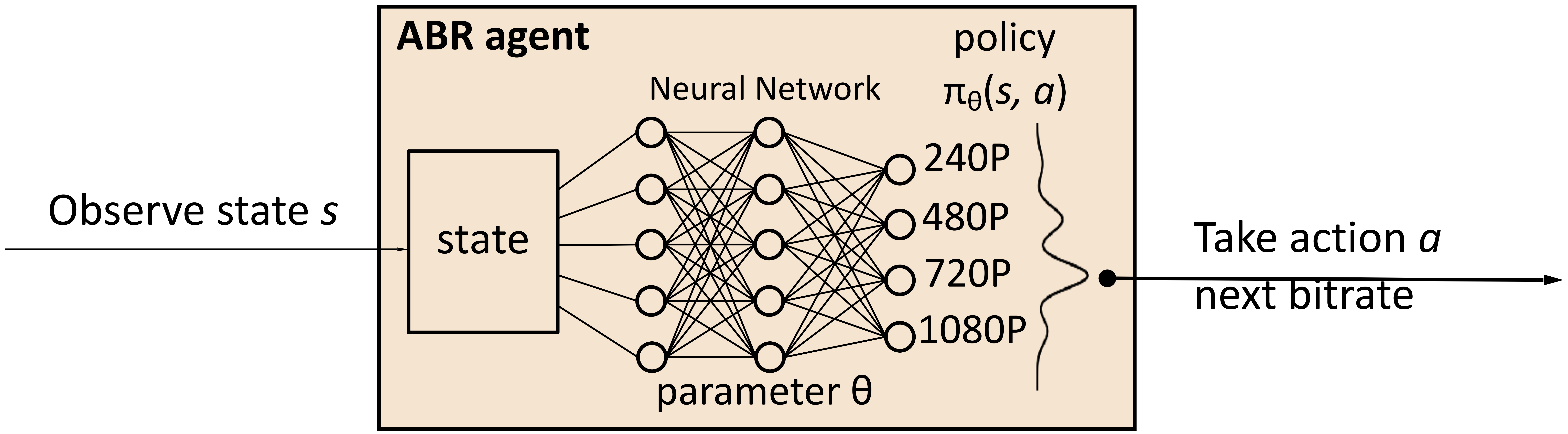
Goal: maximize the cumulative reward $\sum_t r_t$

Slide Credits: Hongzi Mao

Pensieve Design

Slide Credits: Hongzi Mao

How to Train the ABR Agent



Collect experience data: trajectory of [state, action, reward]

Training:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \mathbb{E}_{\pi_\theta}\left[\sum_t r_t\right]$$

estimate from empirical data

Slide Credits: Hongzi Mao

What Pensieve is good at

- Learn the dynamics **directly from experience**
- Optimize the high level QoE objective **end-to-end**
- Extract control rules from **raw high-dimensional** signals

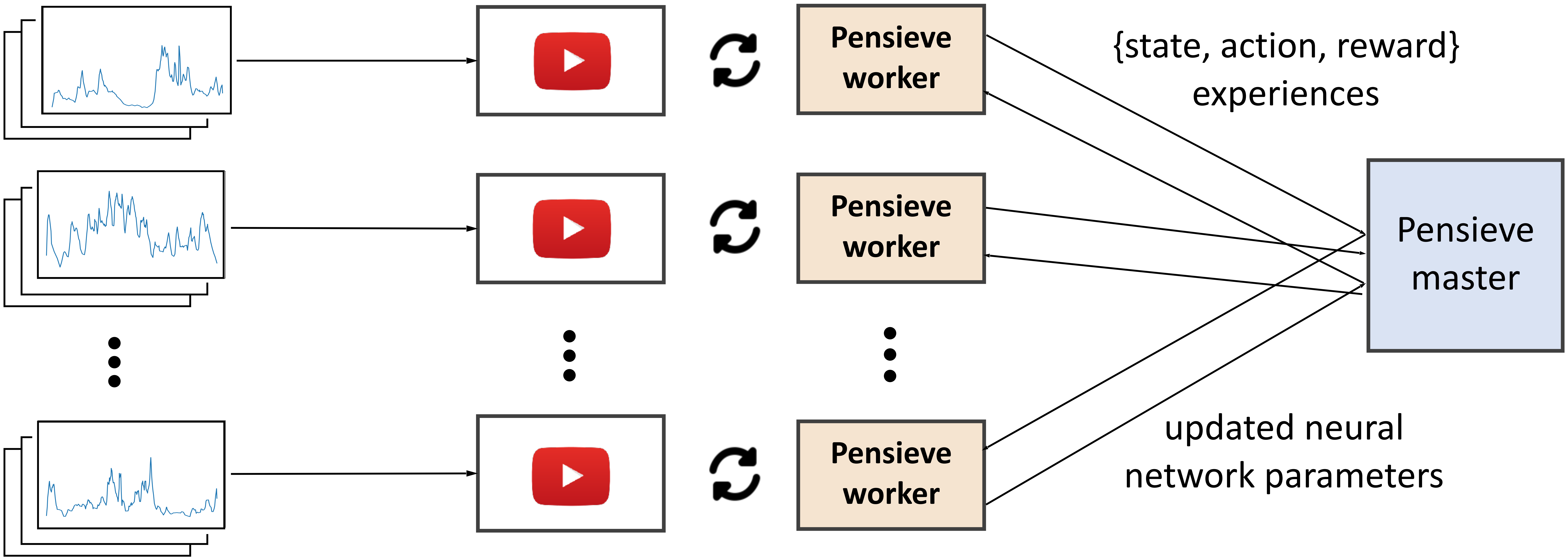
Slide Credits: Hongzi Mao

Pensieve Training System

Large corpus of network traces
cellular, broadband, synthetic

Video playback
Fast chunk-level simulator

Model update
TensorFlow

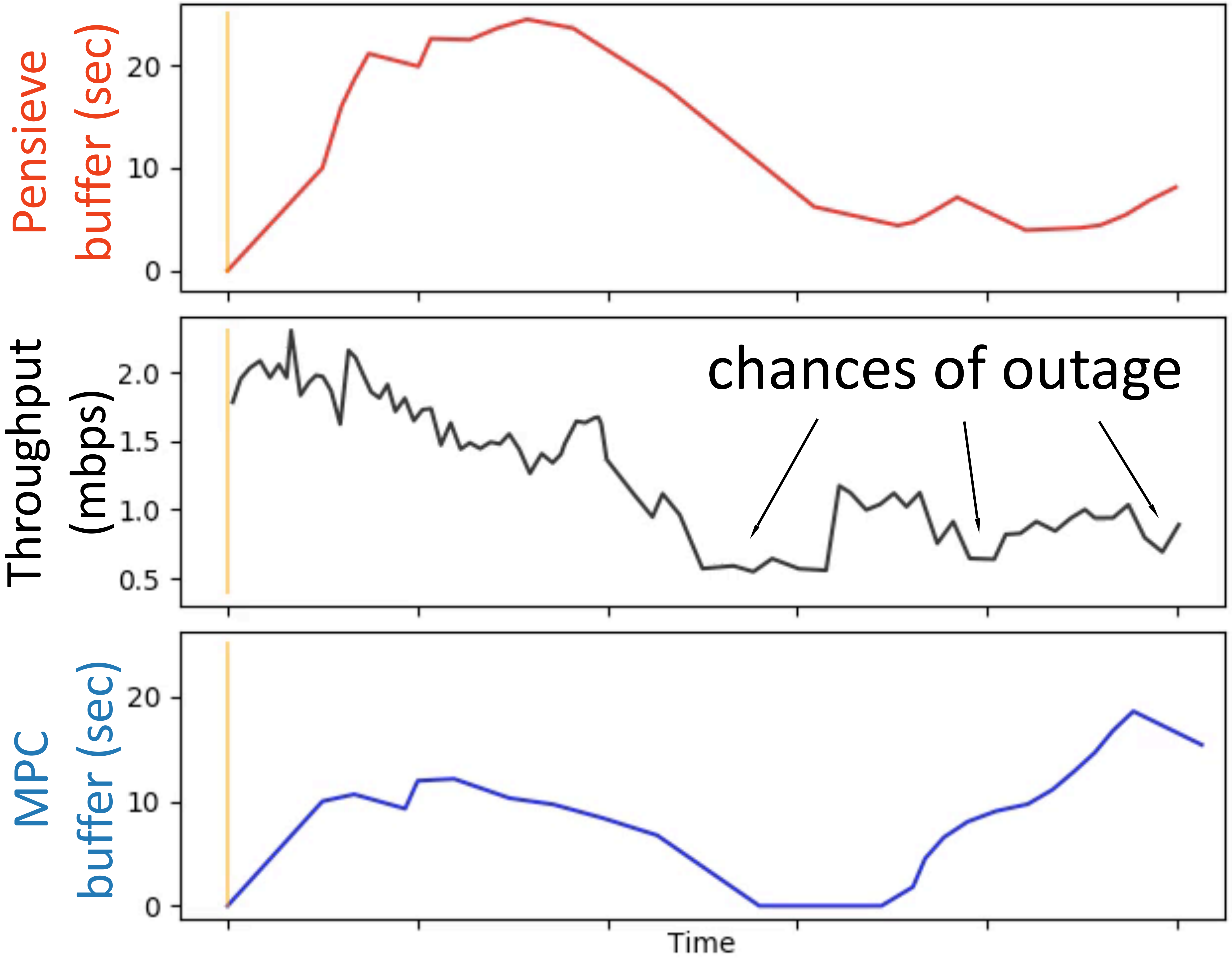


Slide Credits: Hongzi Mao

Demo

Pensieve

MPC



Slide Credits: Hongzi Mao

ML For Systems Case Study #2: Video Streaming

Neural Adaptive Video Streaming with Pensieve

Hongzi Mao, Ravi Netravali, Mohammad Alizadeh
MIT Computer Science and Artificial Intelligence Laboratory
{hongzi,ravinet,alizadeh}@mit.edu

- Key insight: use machine learning to learn an adaptive bitrate control algorithm!
- Paper & presentation available @ https://web.mit.edu/pensieve/

Lecture Outline

- Intro to ML for Systems
- Intro to Systems for ML
- ML for Systems Case Studies
  - *Learning Relaxed Belady for Content Distributional Network Caching*
  - *Neural Adaptive Video Streaming with Pensive*
- Systems for ML Case Studies
  - *Pipedream: Generalized Pipeline Parallelism for DNN Training*
  - *Gemel: Model Merging for Memory-Efficient, Real-Time Video Analytics at the Edge*

Systems for ML Case Study #1: ML Training

PipeDream: Generalized Pipeline Parallelism for DNN Training

Deepak Narayanan‡*, Aaron Harlap†*, Amar Phanishayee⋆,
Vivek Seshadri⋆, Nikhil R. Devanur⋆, Gregory R. Ganger†, Phillip B. Gibbons†, Matei Zaharia‡
⋆*Microsoft Research* †*Carnegie Mellon University* ‡*Stanford University*

Deep Neural Networks have empowered state of the art results across a range of applications…

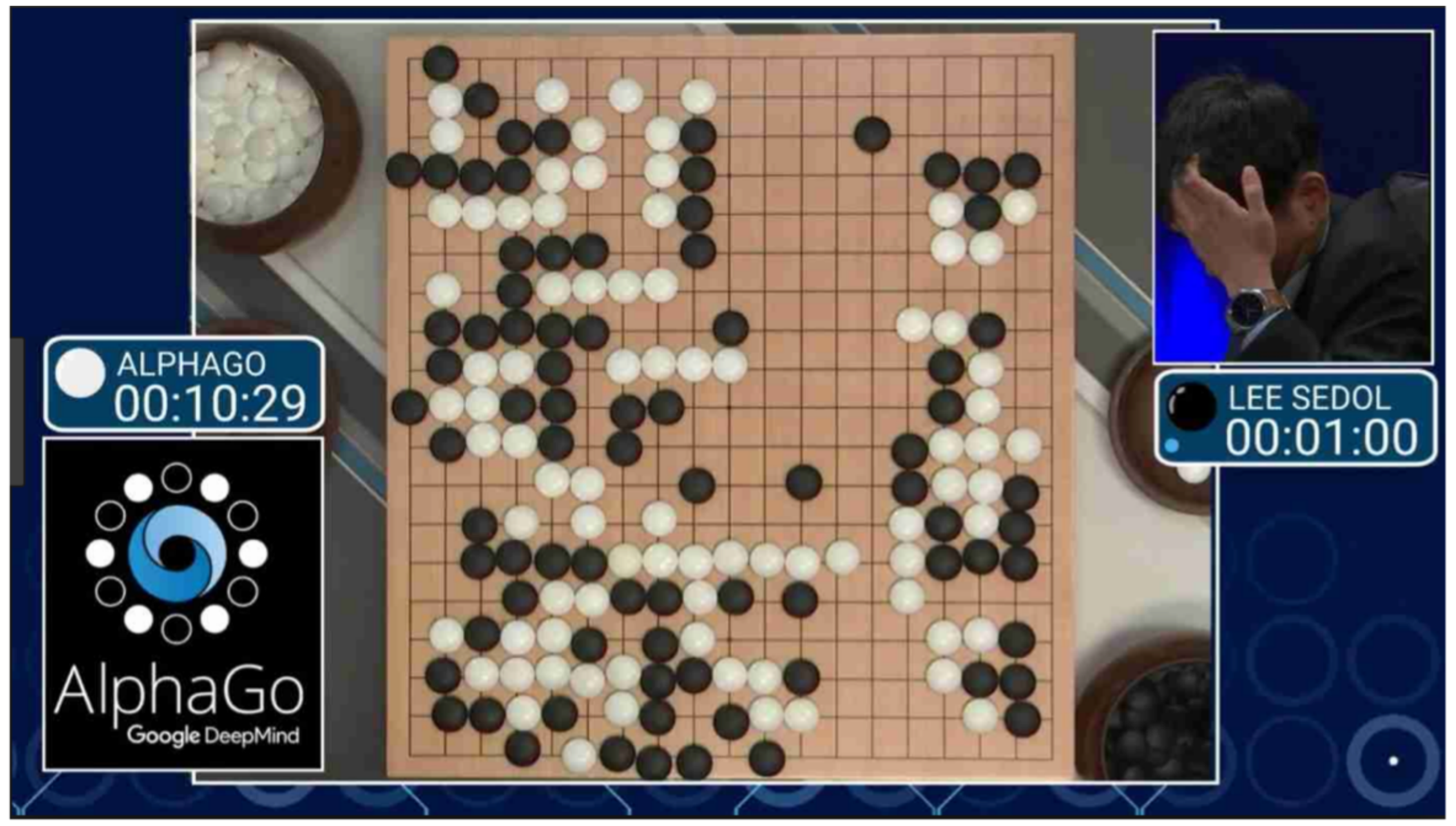
Image Classification

வணக்கம் என் பெயர் தீபக்

Hello, my name is Deepak

Machine Translation

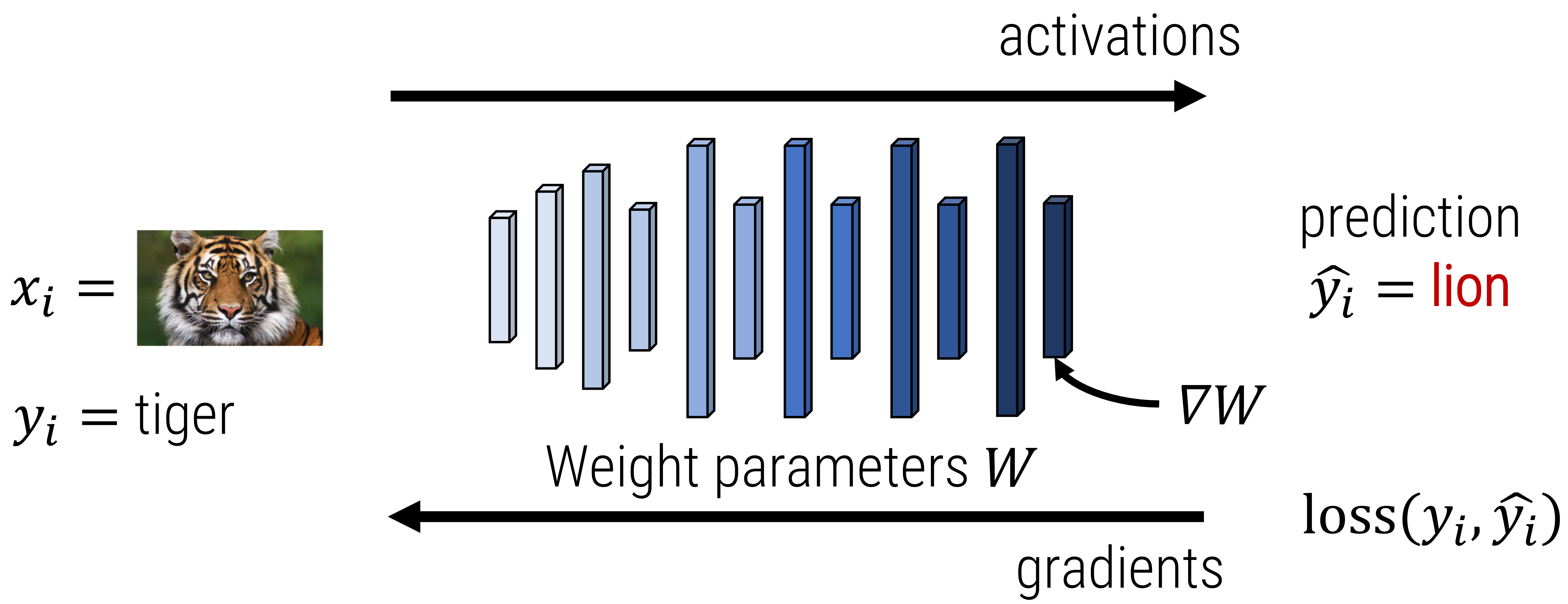
Speech-to-Text



Game Playing

Slide Credits:
Deepak Narayanan

…but first need to be trained!



W optimized using standard iterative optimization procedures

$$W = W - \eta \cdot \nabla W$$

Slide Credits:
Deepak Narayanan

Background: DNN Training

Model training time- and compute- intensive!

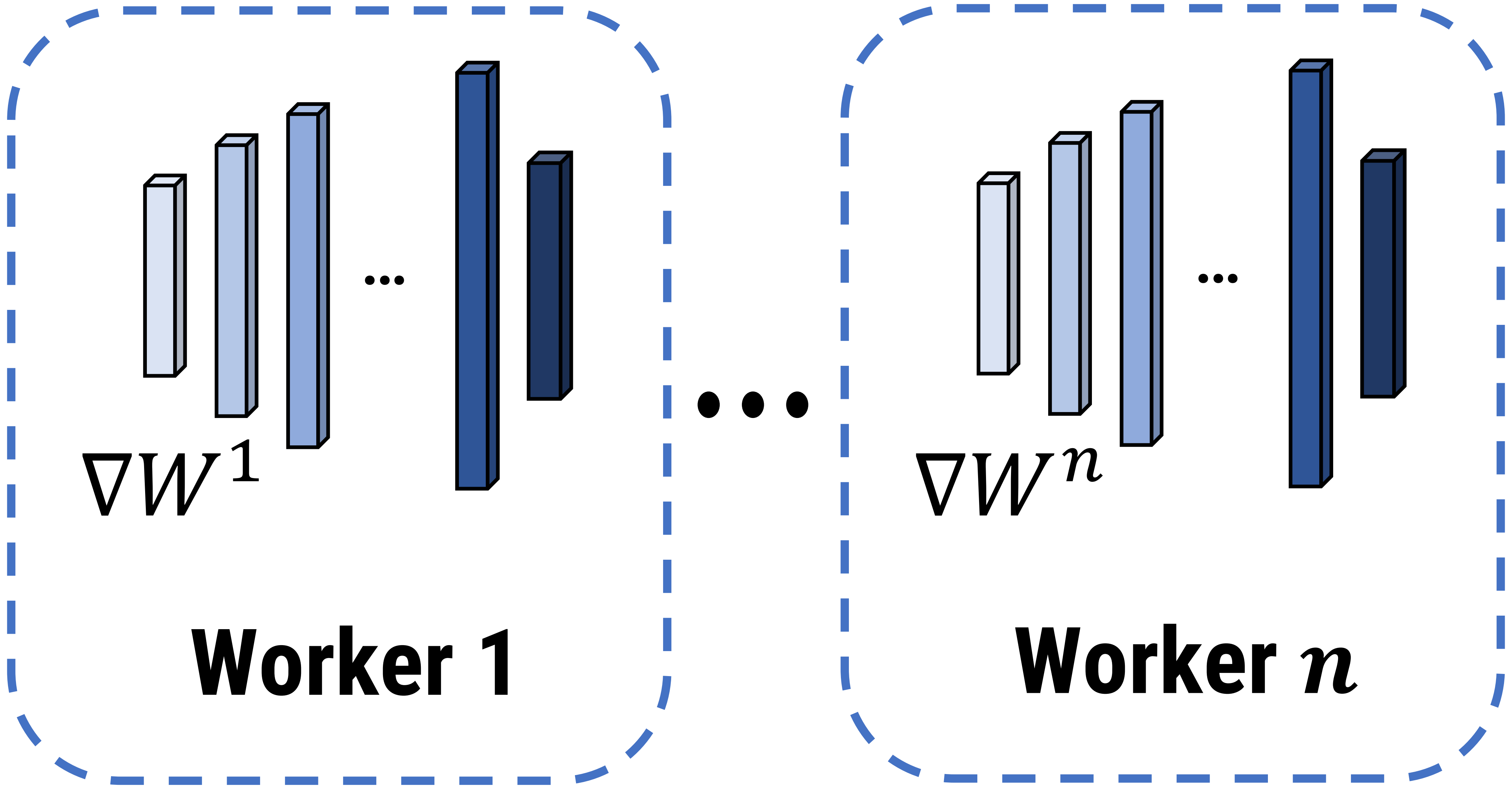
W optimized using standard iterative optimization procedures

$$W = W - \eta \cdot \nabla W$$

Slide Credits:
Deepak Narayanan

Parallelizing DNN Training: Data Parallelism

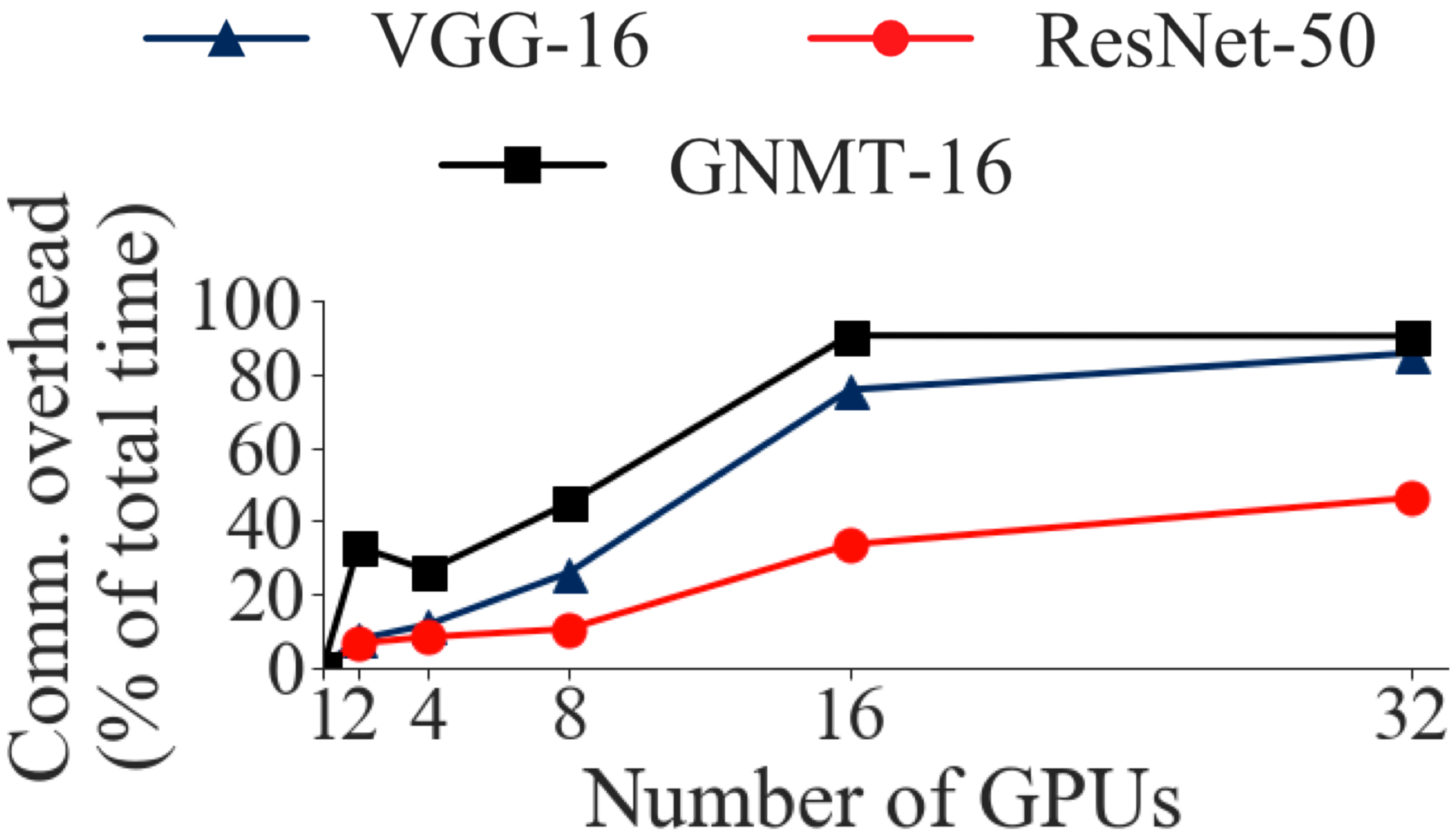
n copies of the same model



$$\nabla W = \nabla W^1 + \nabla W^2 + \cdots + \nabla W^n$$

Gradient aggregation using AllReduce

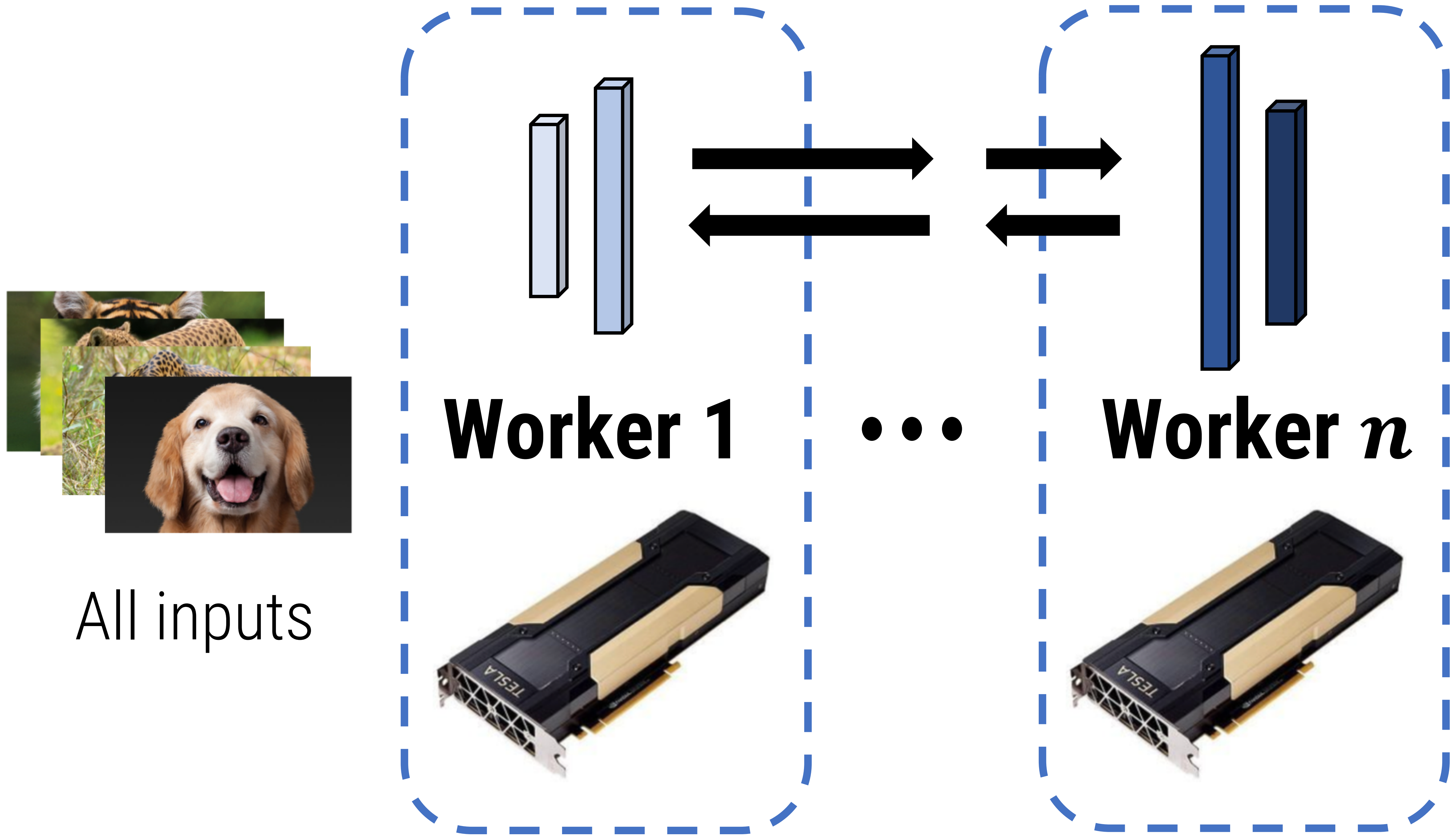
Despite many performance optimizations, communication overhead high!



8xV100s with NVLink (AWS)
PyTorch + NCCL 2.4

Slide Credits: Deepak Narayanan

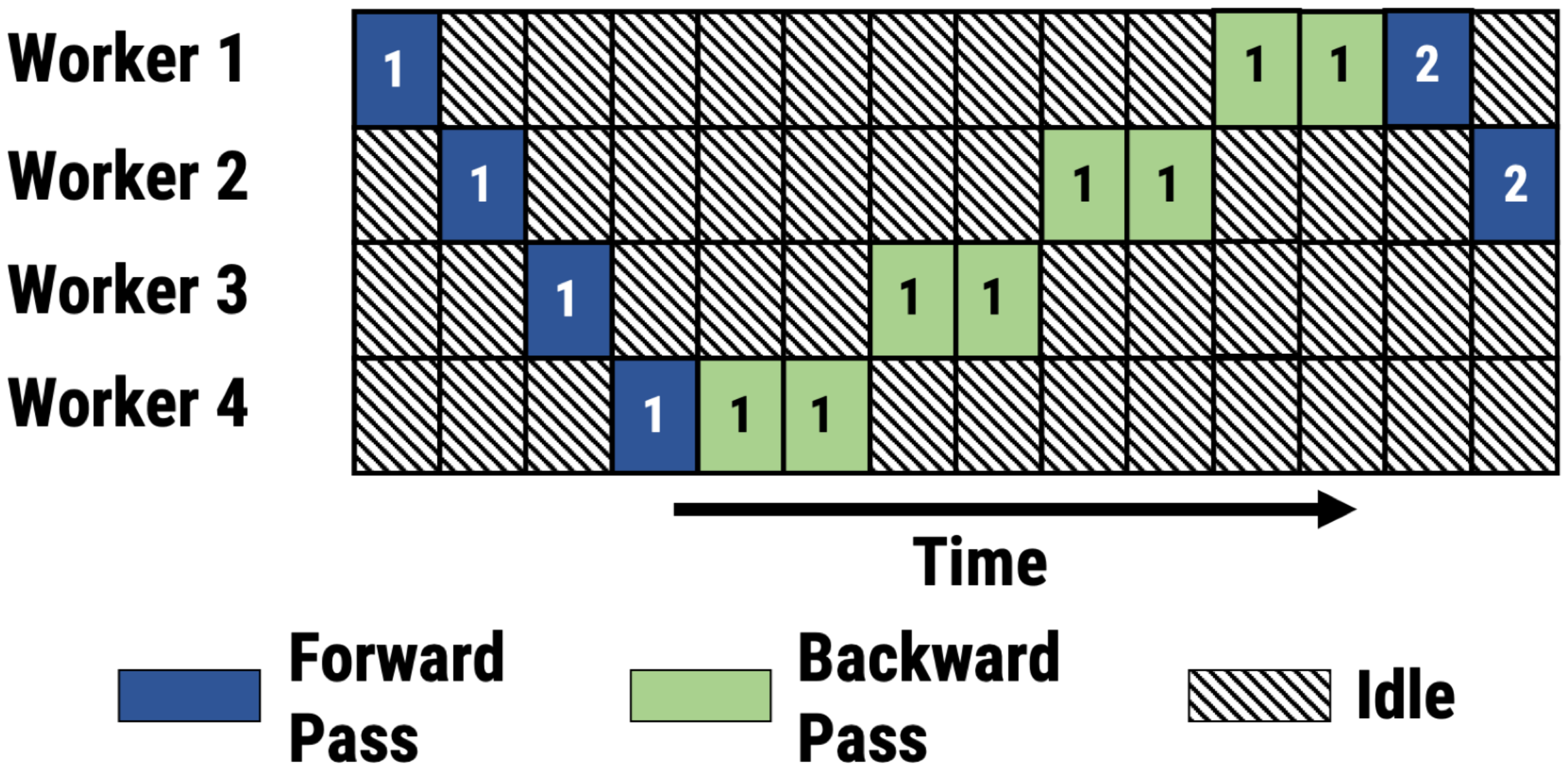
Parallelizing DNN training: Model Parallelism



Single version of weights split over workers

Activations and gradients sent between workers using peer-to-peer communication

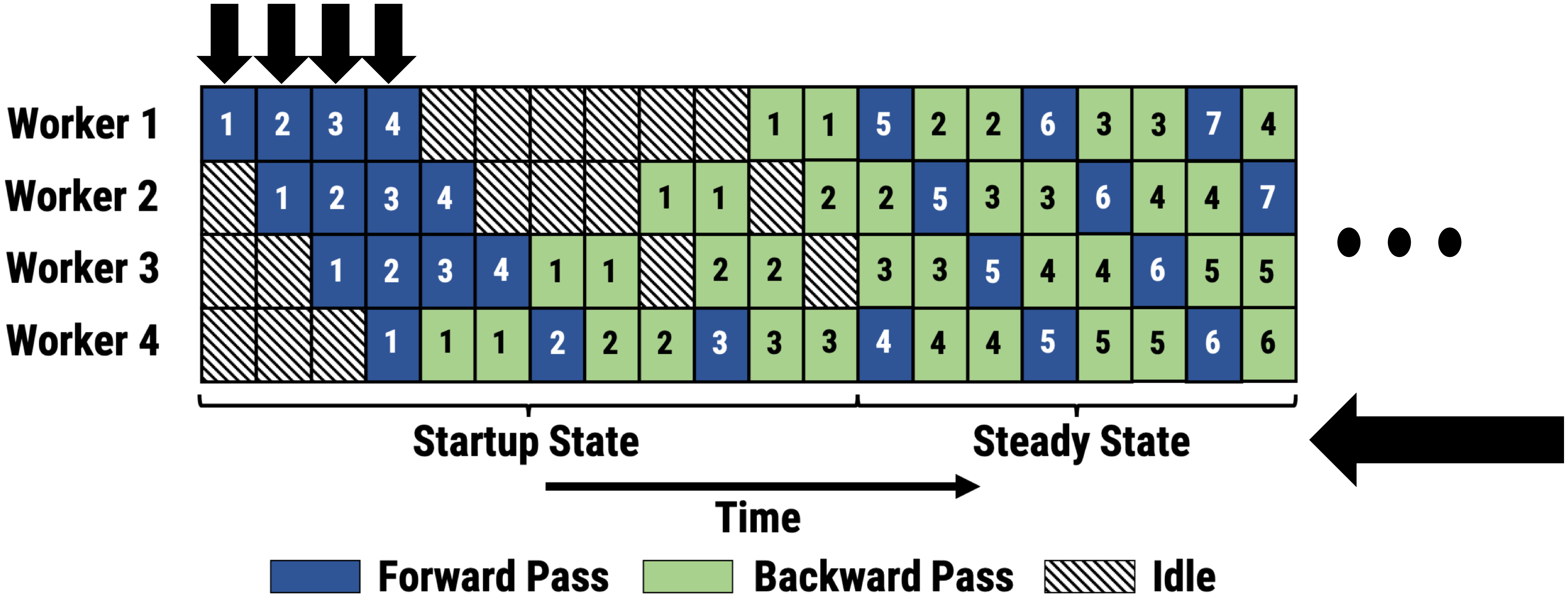
Low hardware efficiency



Slide Credits:
Deepak Narayanan

PipeDream: Pipeline-Parallel Training

We propose **pipeline parallelism**, a combination of data and model parallelism with pipelining



Pipeline-parallel training up to **5.3x faster** than data parallelism without sacrificing on final accuracy of the model

Slide Credits: Deepak Narayanan

Pipelining in DNN Training != Traditional Pipelining

- How should the operators in a DNN model be partitioned into pipeline stages?
  - Each operator has a **different computation time**
  - Activations and gradients need to be **communicated** across stages

- How should forward and backward passes of different inputs be scheduled?
  - Training is **bidirectional**
  - Forward pass followed by backward pass to compute gradients

- How should weight and activation versions be managed?
  - Backward pass operators depend on **internal state** (W, activations)

Slide Credits:
Deepak Narayanan

PipeDream Profiler and Optimizer

Input DNN $\xrightarrow{\text{Profiler}}$ Computational graph with profile

↓

Optimizer

↑

Deployment constraints such as number of accelerators, memory and interconnect characteristics

Determines a partitioning of operators amongst workers, while also deciding replication factors

Generalizes along many axes

- Hardware topologies
- Model structures
- Memory capacities of workers

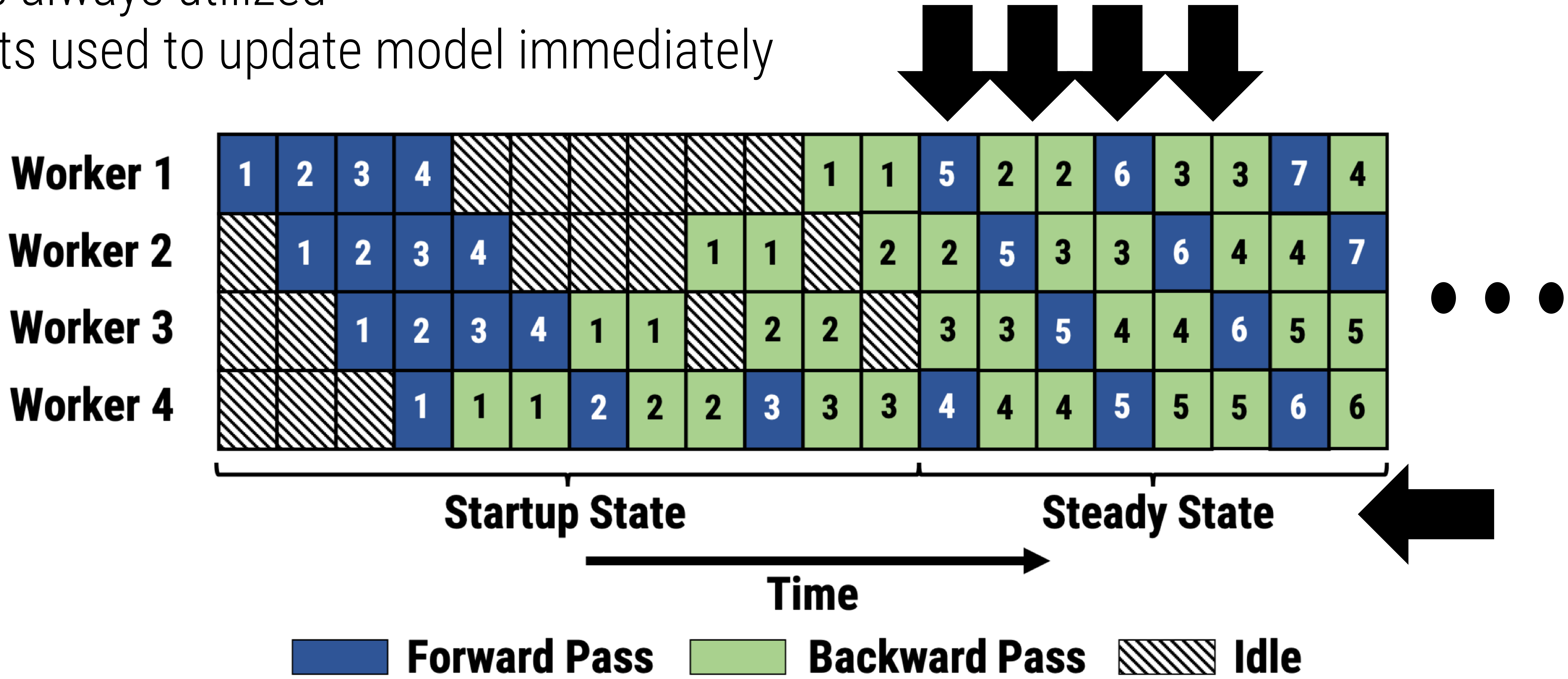
See paper for details of algorithm!

Slide Credits:
Deepak Narayanan

1F1B Scheduling

Workers **alternate** between forward and backward passes

- Workers always utilized
- Gradients used to update model immediately



To support stage replication, need to modify this mechanism slightly – see paper for details!

Slide Credits: Deepak Narayanan

Naïve pipelining leads to weight version mismatches

Naïve pipelining leads to **mismatch in weight versions**

$x_n \longrightarrow W_n \longrightarrow y_n$ Forward pass

W_{n+1}

$\vdots$

$\nabla x_n \longleftarrow W_{n+p} \longleftarrow \nabla y_n$ Backward pass

Input n sees updates in backward pass not seen in the forward pass, leading to incorrect gradients

Slide Credits:
Deepak Narayanan

1F1B Scheduling + Weight Stashing

Naïve pipelining leads to **mismatch in weight versions**

Store **multiple <weight, activation> versions**

- Ensures same weight versions used in both forward and backward pass

$x_n \longrightarrow W_n \longrightarrow y_n$ Forward pass

W_{n+1}

$\vdots$

$\boxed{W_n}\boxed{W_{n+1}}\boxed{W_{n+2}} \cdots$

Stashed weights

$\nabla x_n \longleftarrow \mathbf{W_n} \longleftarrow \nabla y_n$ Backward pass

- Worst case memory footprint similar to data parallelism ($= n \cdot {}^{(|W|+|A|)}/_{n}$)

Slide Credits: Deepak Narayanan

Systems for ML Case Study #1: ML Training

PipeDream: Generalized Pipeline Parallelism for DNN Training

Deepak Narayanan[‡*], Aaron Harlap[†*], Amar Phanishayee[⋆],
Vivek Seshadri[⋆], Nikhil R. Devanur[⋆], Gregory R. Ganger[†], Phillip B. Gibbons[†], Matei Zaharia[‡]
[⋆]*Microsoft Research* [†]*Carnegie Mellon University* [‡]*Stanford University*

- Key insight: use pipelining of mini batches of data to improve parallel training throughput
- Paper available @ https://www.microsoft.com/en-us/research/uploads/prod/2019/08/pipedream.pdf
- Presentation available @ https://sosp19.rcs.uwaterloo.ca/videos/D1-S1-P1.mp4

Lecture Outline

- Intro to ML for Systems
- Intro to Systems for ML
- ML for Systems Case Studies
    - *Learning Relaxed Belady for Content Distributional Network Caching*
    - *Neural Adaptive Video Streaming with Pensive*
- Systems for ML Case Studies
    - *Pipedream: Generalized Pipeline Parallelism for DNN Training*
    - *Gemel: Model Merging for Memory-Efficient, Real-Time Video Analytics at the Edge*

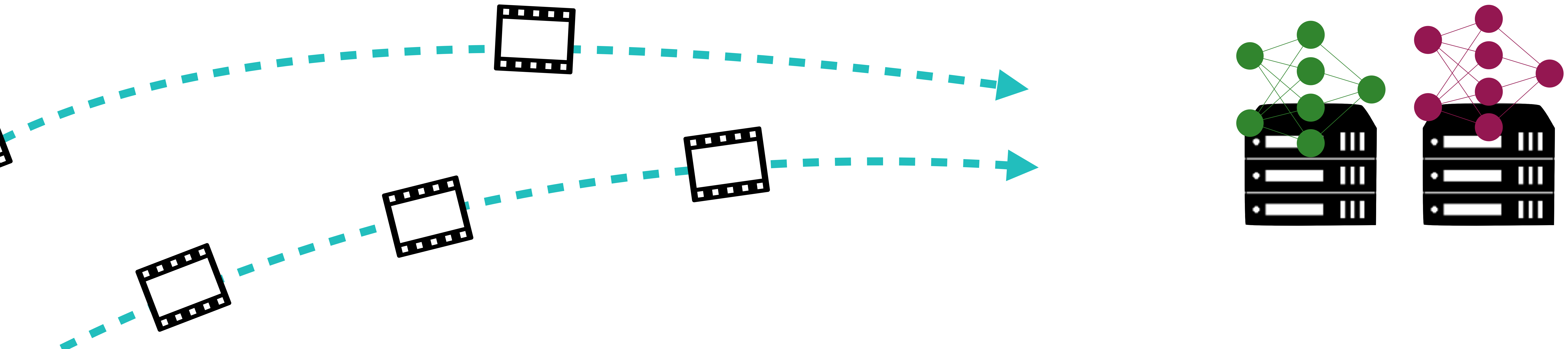
Systems For ML Case Study #2: ML Inference

Gemel: Model Merging for Memory-Efficient, Real-Time Video Analytics at the Edge

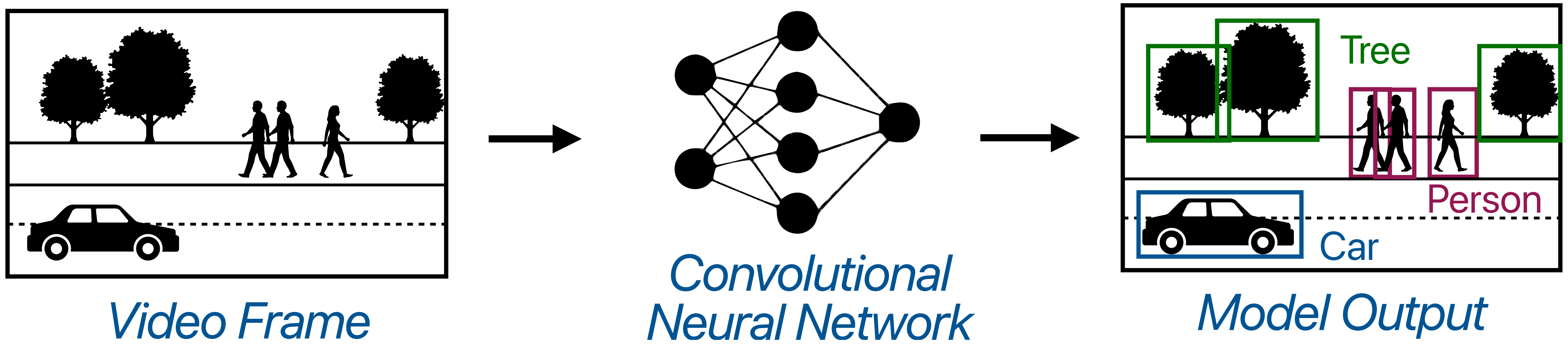
Arthi Padmanabhan[⋆§] Neil Agarwal[⋆¶] Anand Iyer[†] Ganesh Ananthanarayanan[†]
Yuanchao Shu[‡] Nikolaos Karianakis[†] Guoqing Harry Xu[§] Ravi Netravali[¶]

[§]UCLA [†]Microsoft Research [‡]Zhejiang University [¶]Princeton University

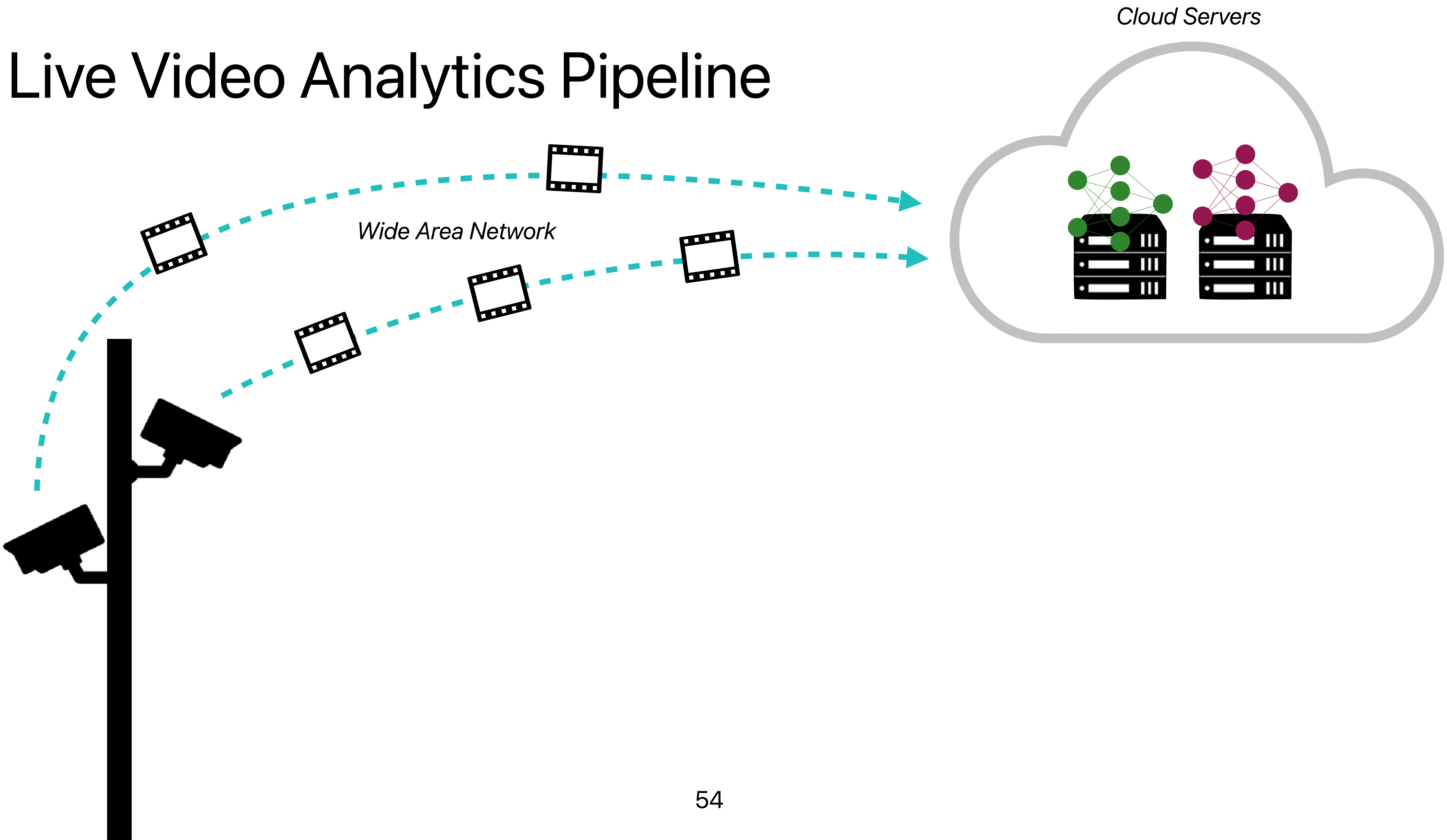
Live Video Analytics Pipeline



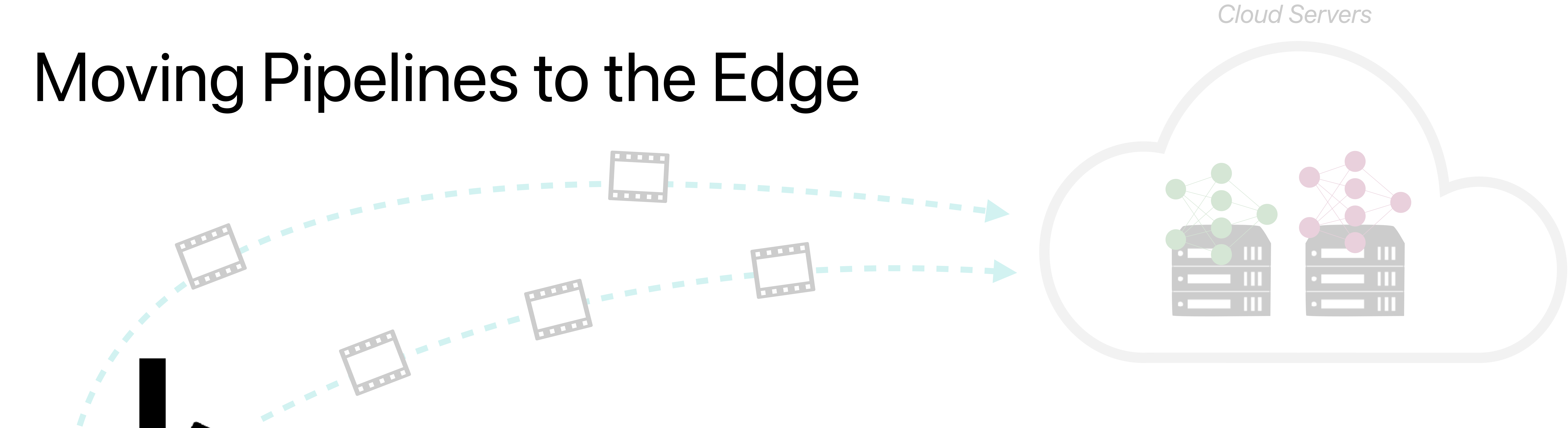
Goal: Maximize query accuracy, subject to latency SLAs and resource constraints

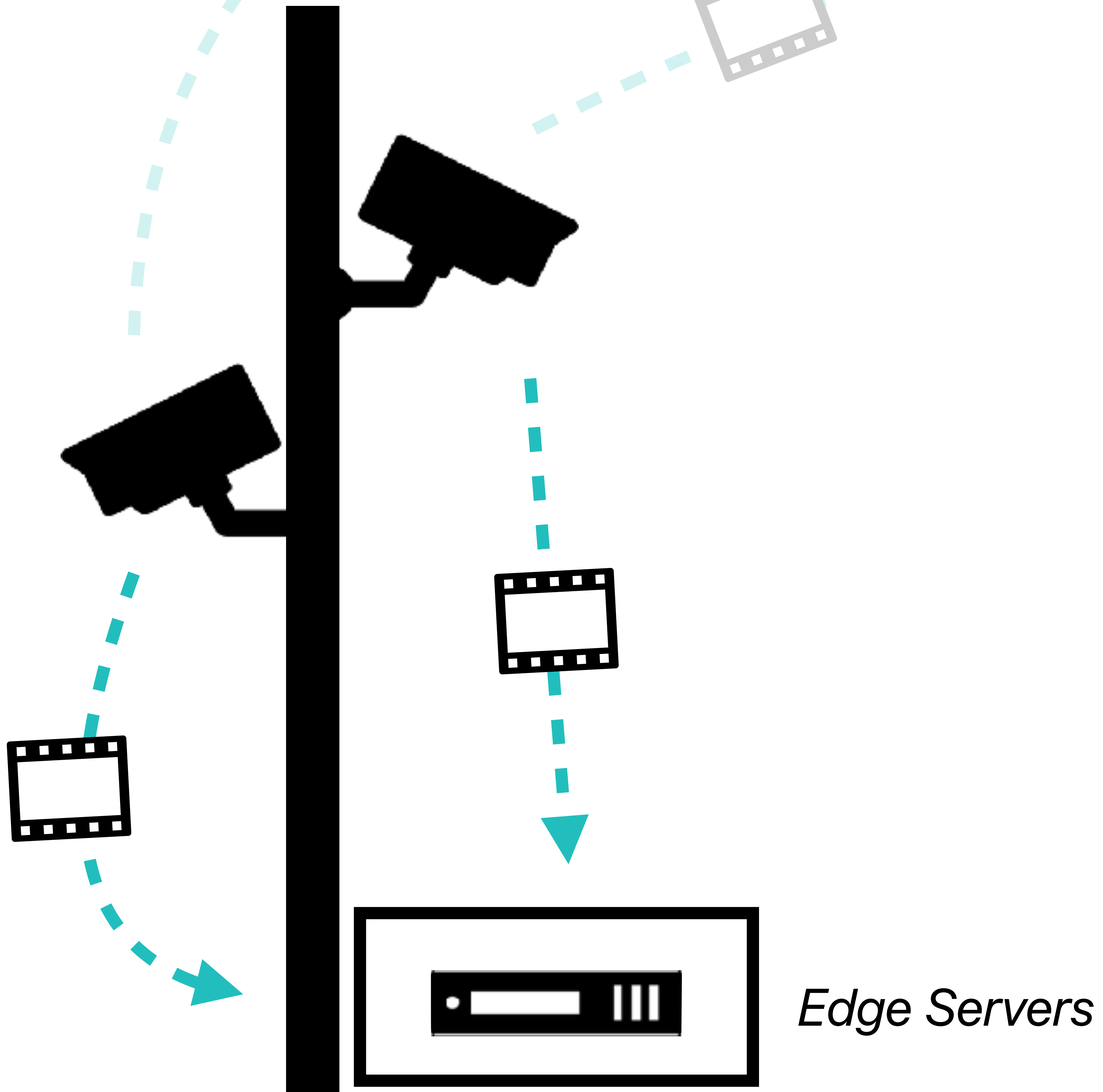


Live Video Analytics Pipeline



Moving Pipelines to the Edge

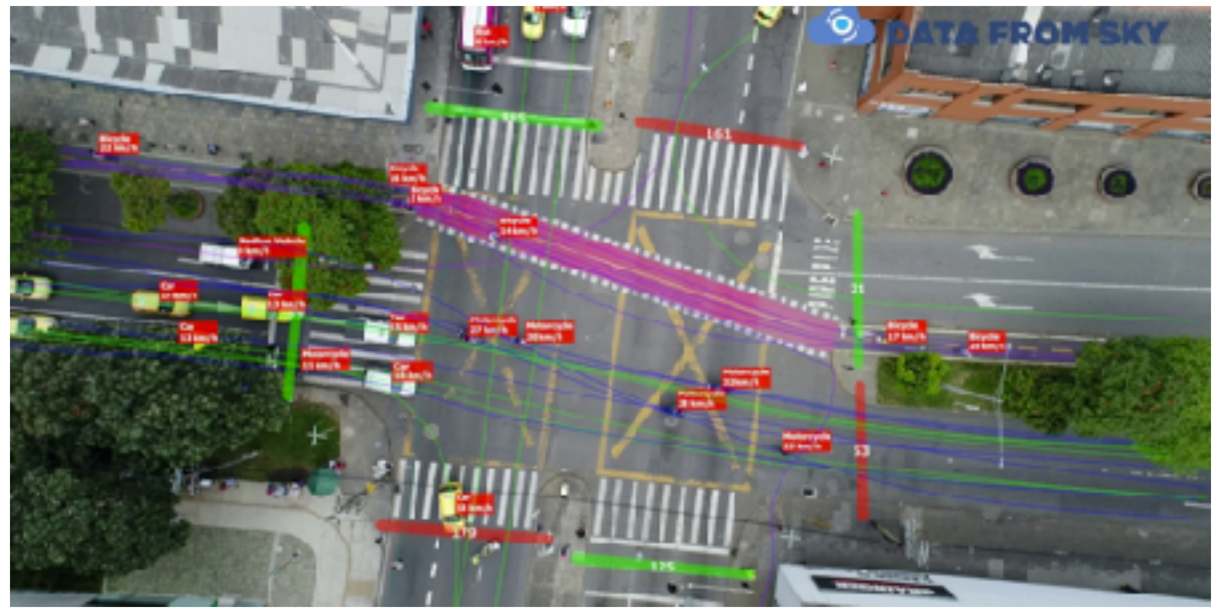




Reduce network overheads ✓

Limited and inelastic resources !

Edge Workloads in the Wild



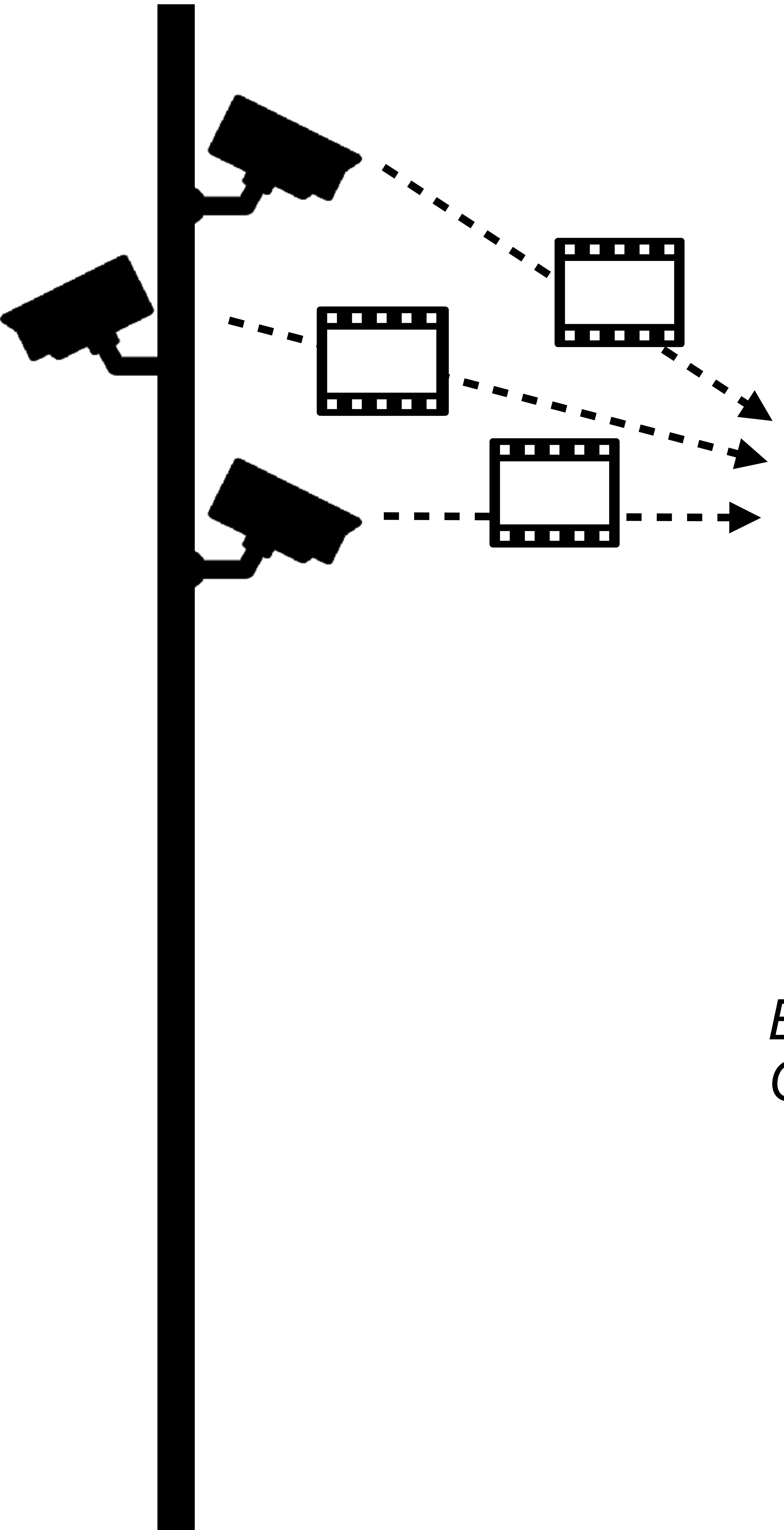
Pilot video analytics deployment across 2 major US cities, targeted at road traffic monitoring

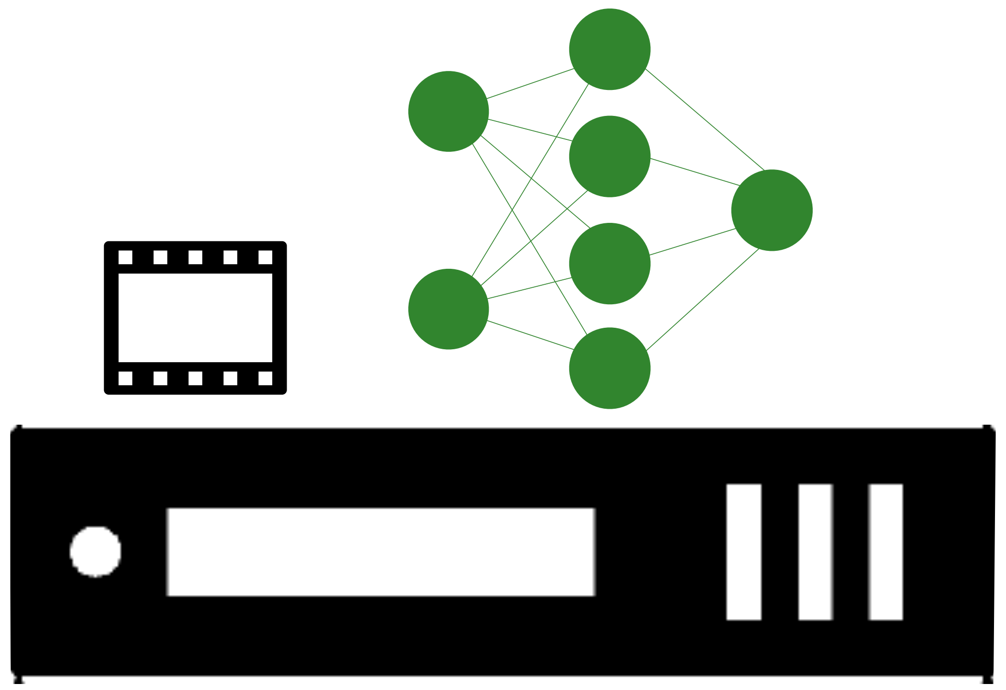
Query: <camera feed, model, task>

Sample Workload

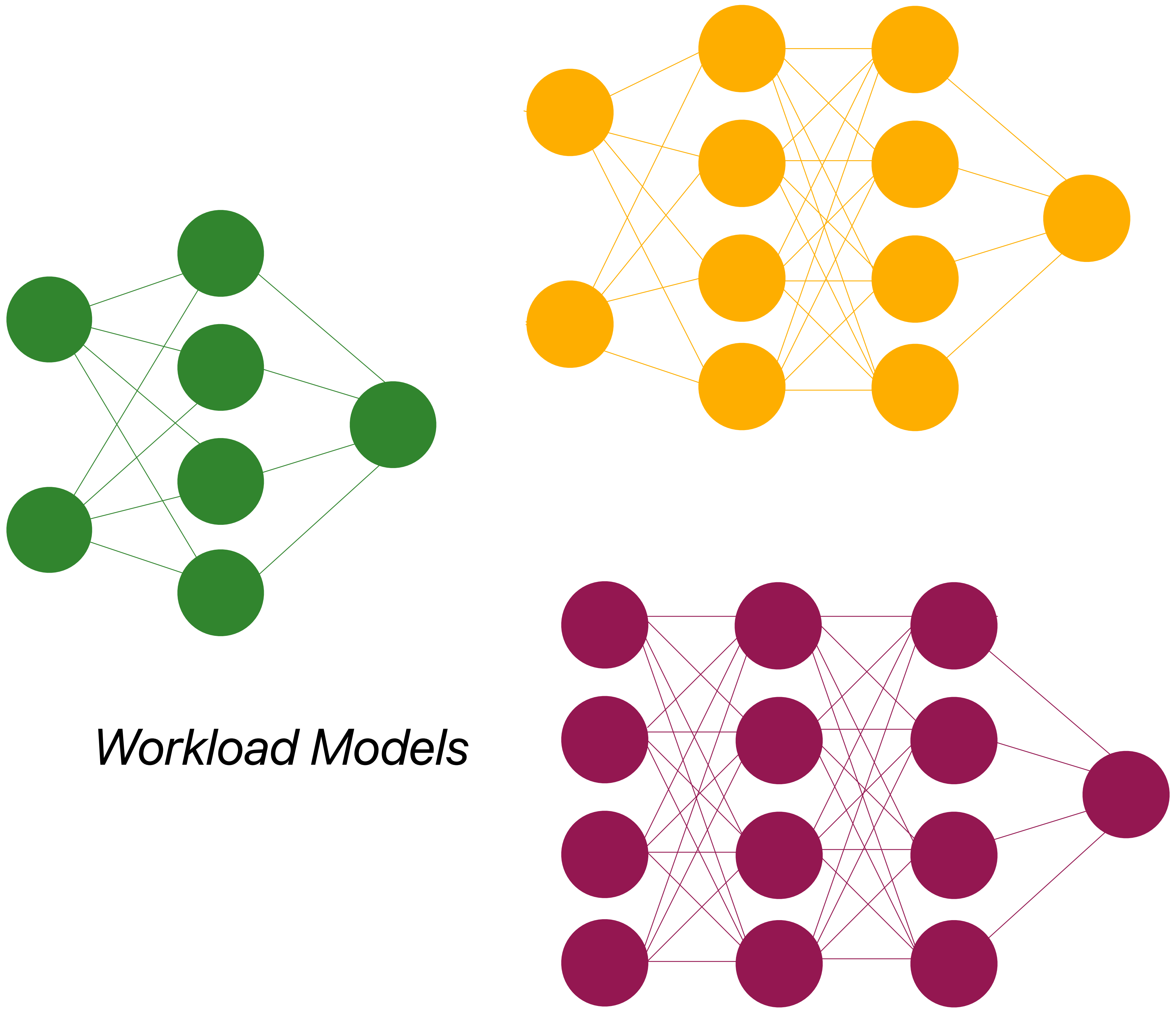
Query #	Camera Feed	Model Architecture	Task Description
1	3	FRCNN-R50	Object detection of cars
2	1	YOLOv3	Object detection of people
3	1	Inception	Binary Classification of people, vehicles
4	6	ResNet50	Binary Classification of cars, buses, trucks
5	3	Tiny-YOLOv3	Object Detection of people
…	…	…	…

Executing Edge Workloads



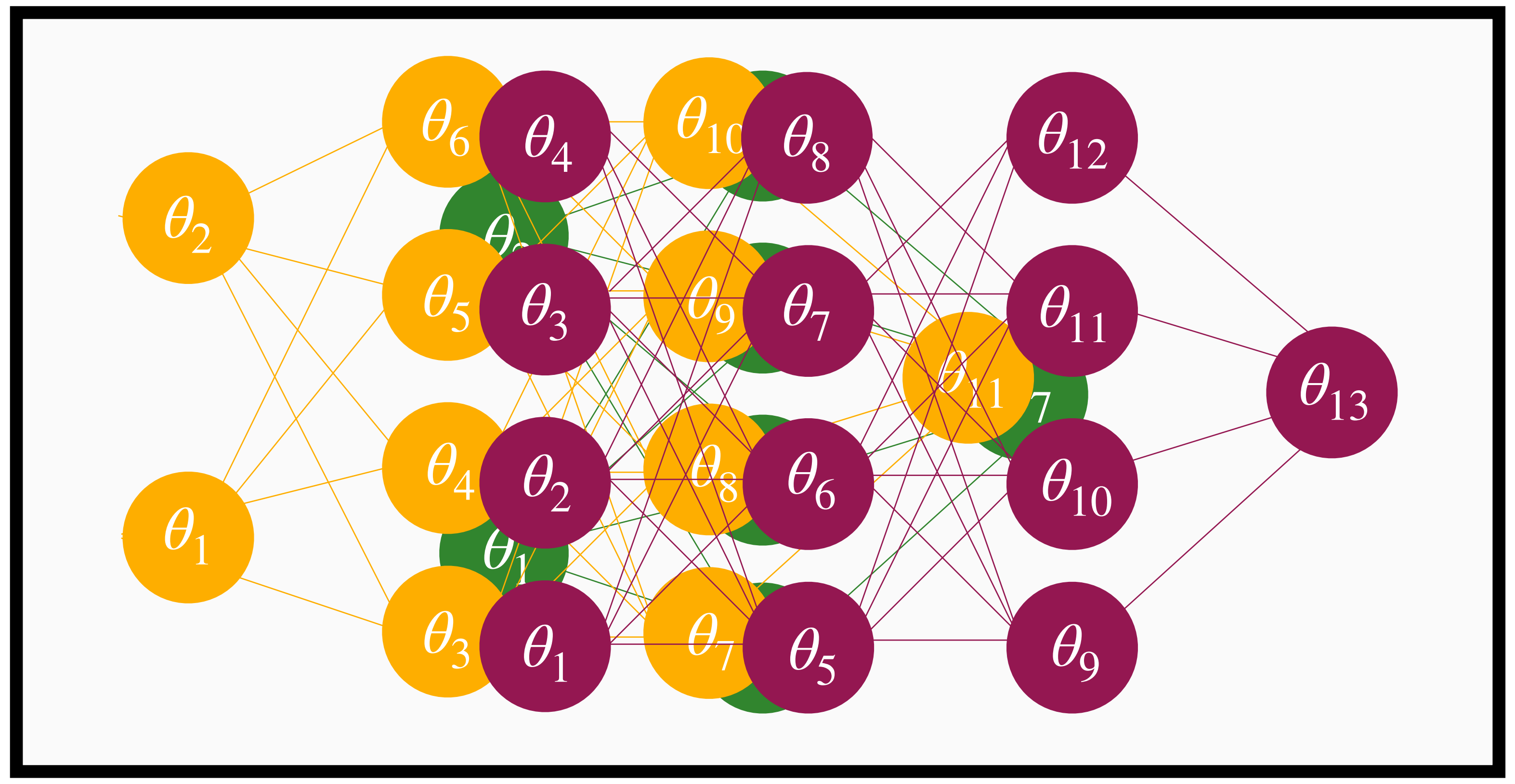


Edge Box

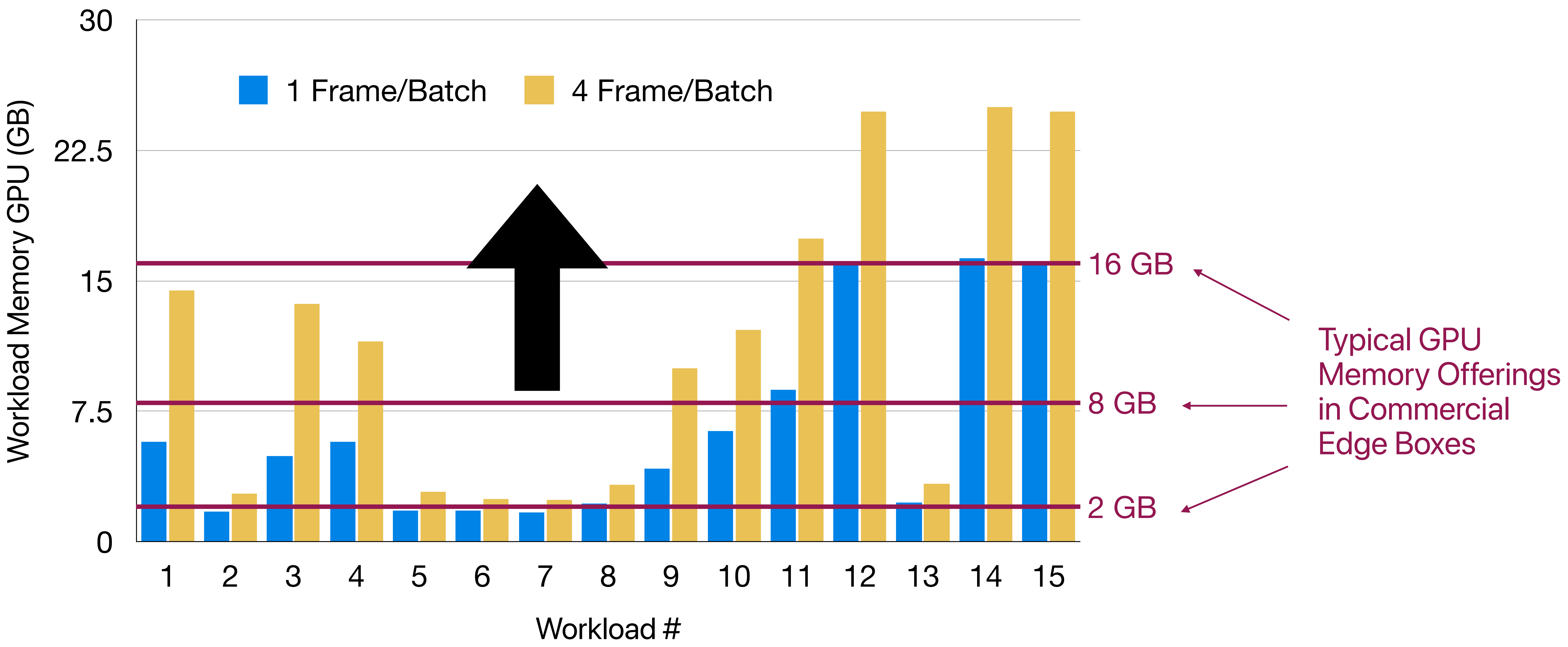


Workload Models

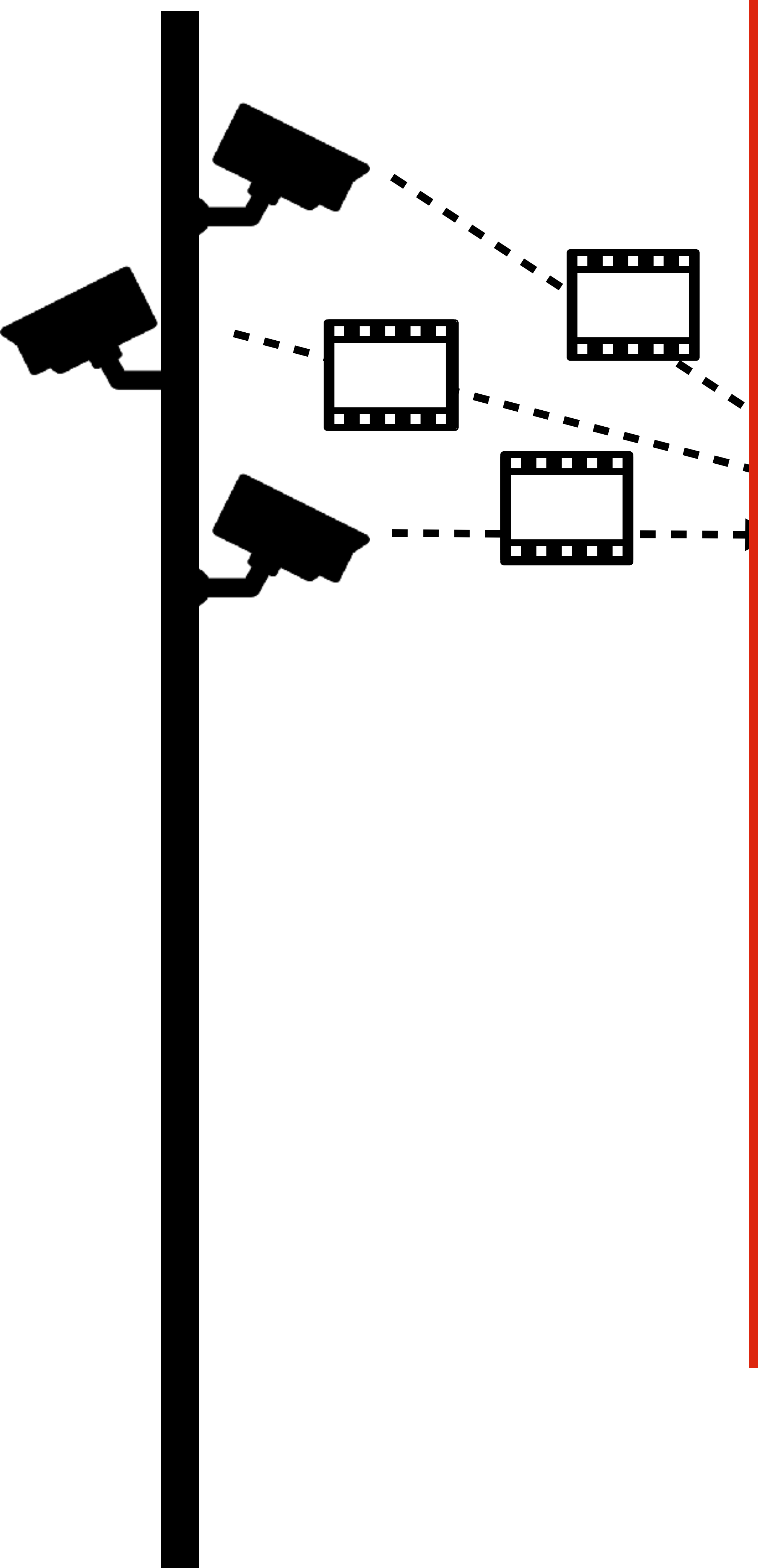
Edge Box
GPU Memory

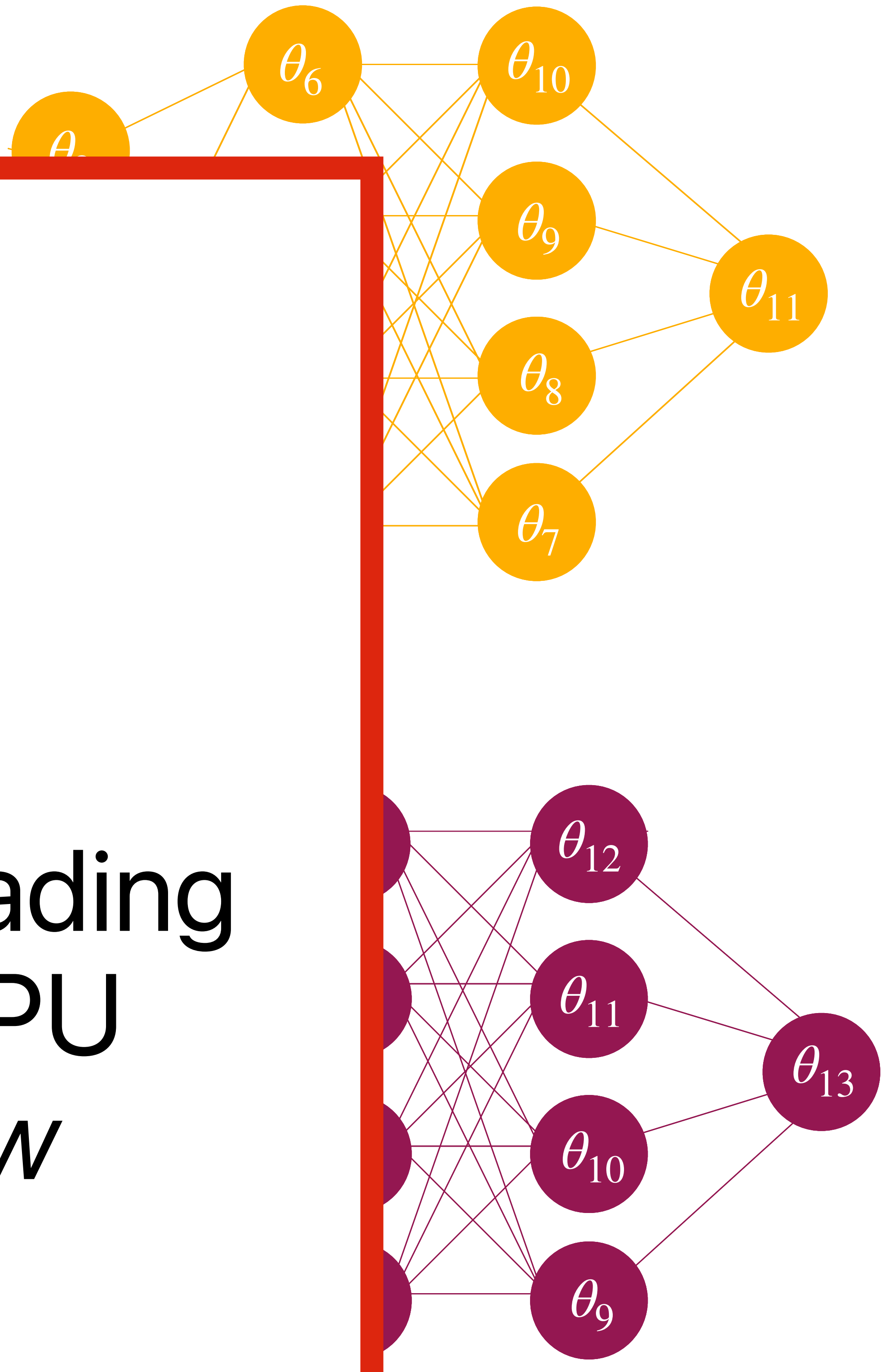


Workloads are Outgrowing Edge GPU Memory



Time-Sharing of GPU Memory





Skipped processing of 19-84% of frames and accuracy drops up to 43%

Model	**Loading Time** (ms)	**Run Time** (ms)
YOLOv3	49.5	17.0
ResNet152	73.3	24.8
ResNet50	27.1	8.4
VGG16	72.2	2.1
Tiny YOLOv3	6.7	3.0

Repeatedly loading models into GPU memory is *slow*

Implication: cannot keep up with frame rate and must drop frames due to SLA violations

Gemel

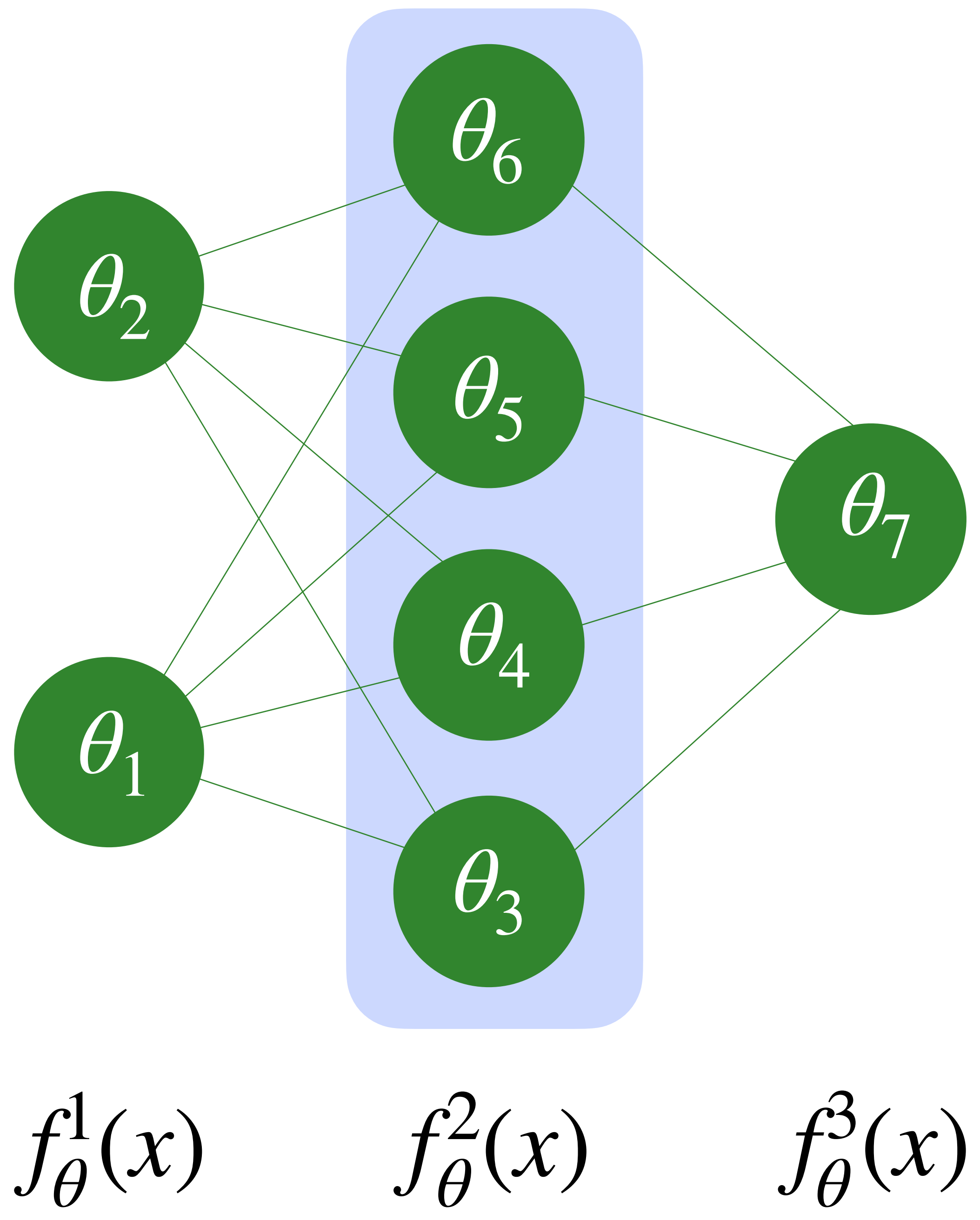
How to reduce GPU memory bottlenecks in edge video analytics?

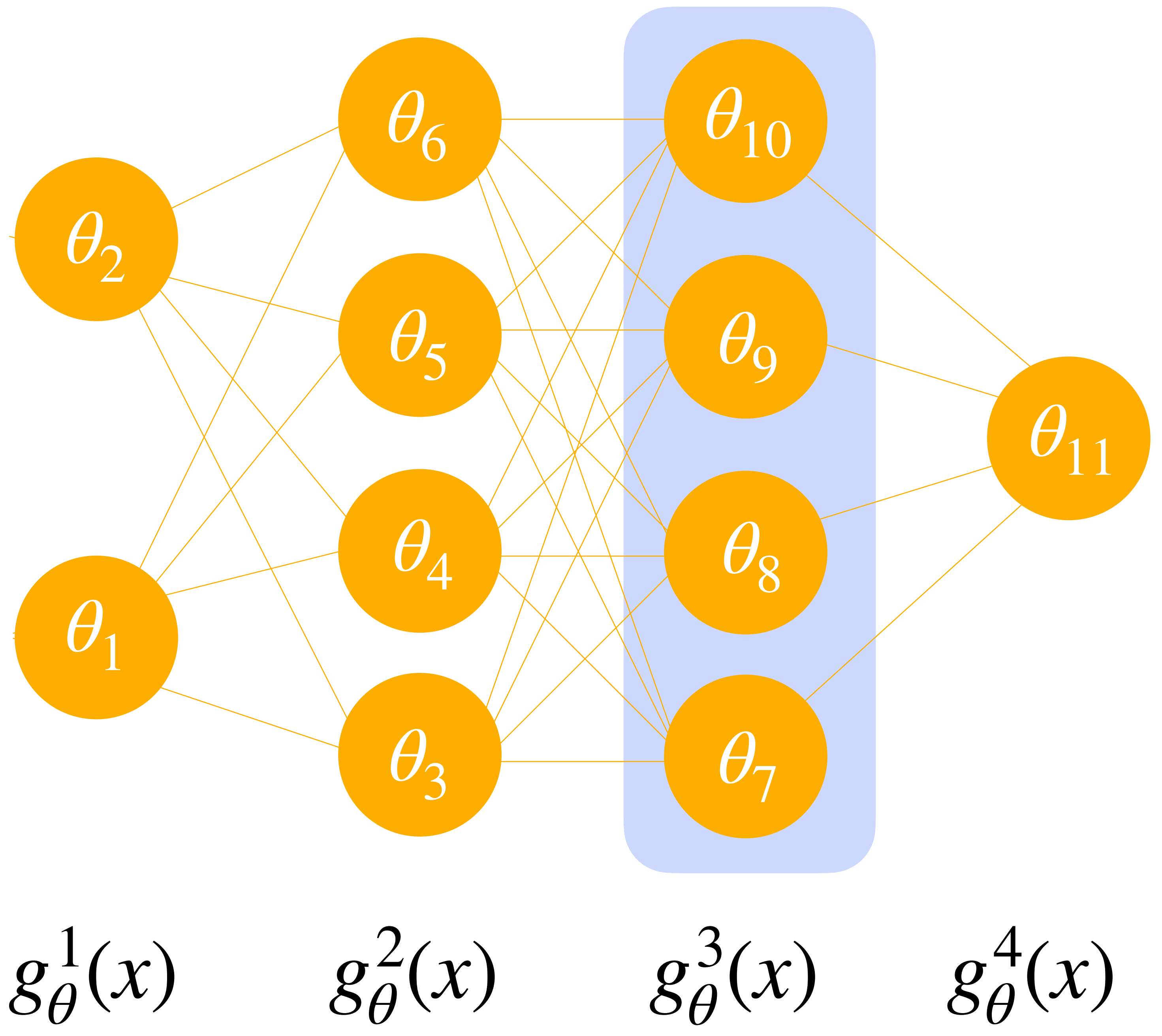
Opportunity: reduce memory overheads by exploiting redundancies *across* models

Observation: despite workload diversity, models often share many *layer definitions*

Shared Layer Definitions Across Models

$$f_\theta^2(x) \equiv g_\theta^3(x)$$

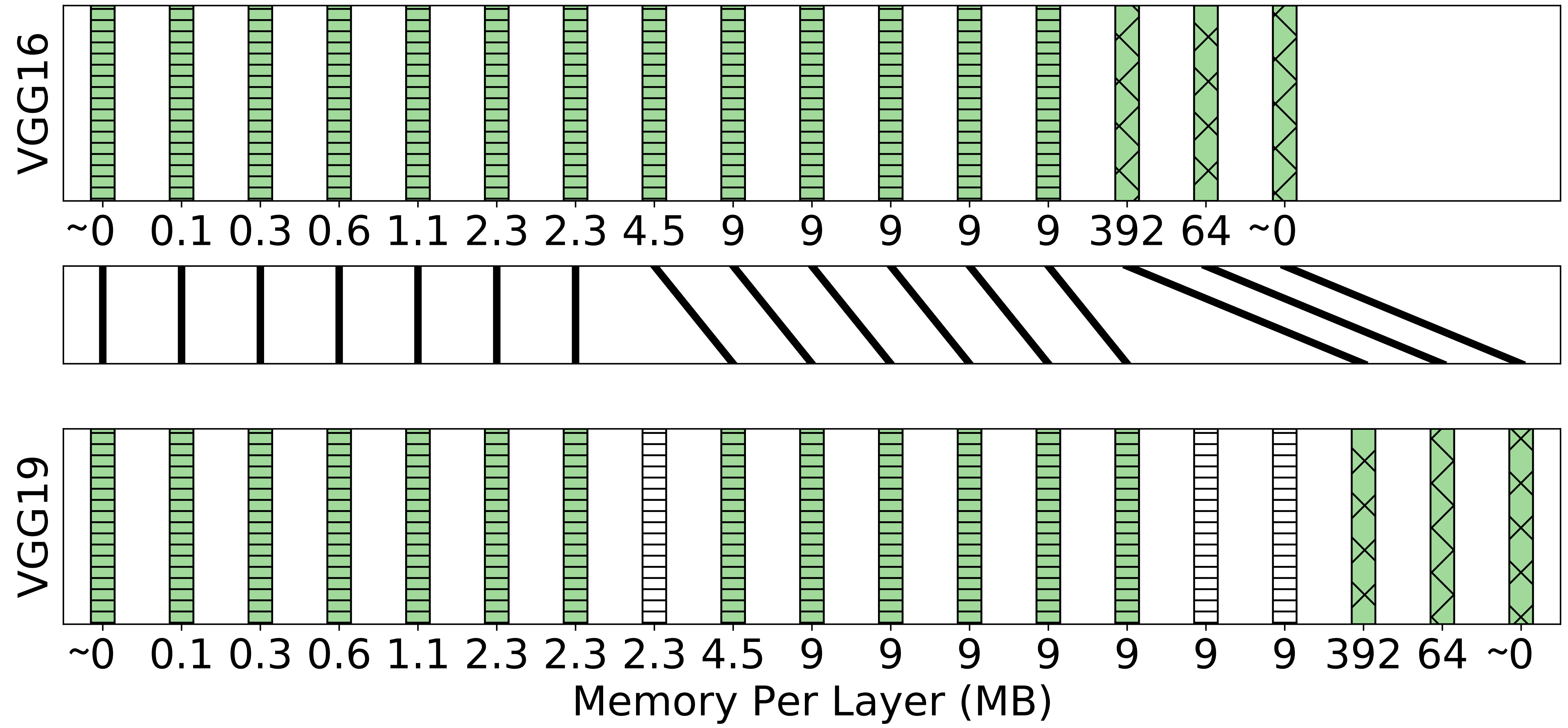




Shared layer definitions appear in...

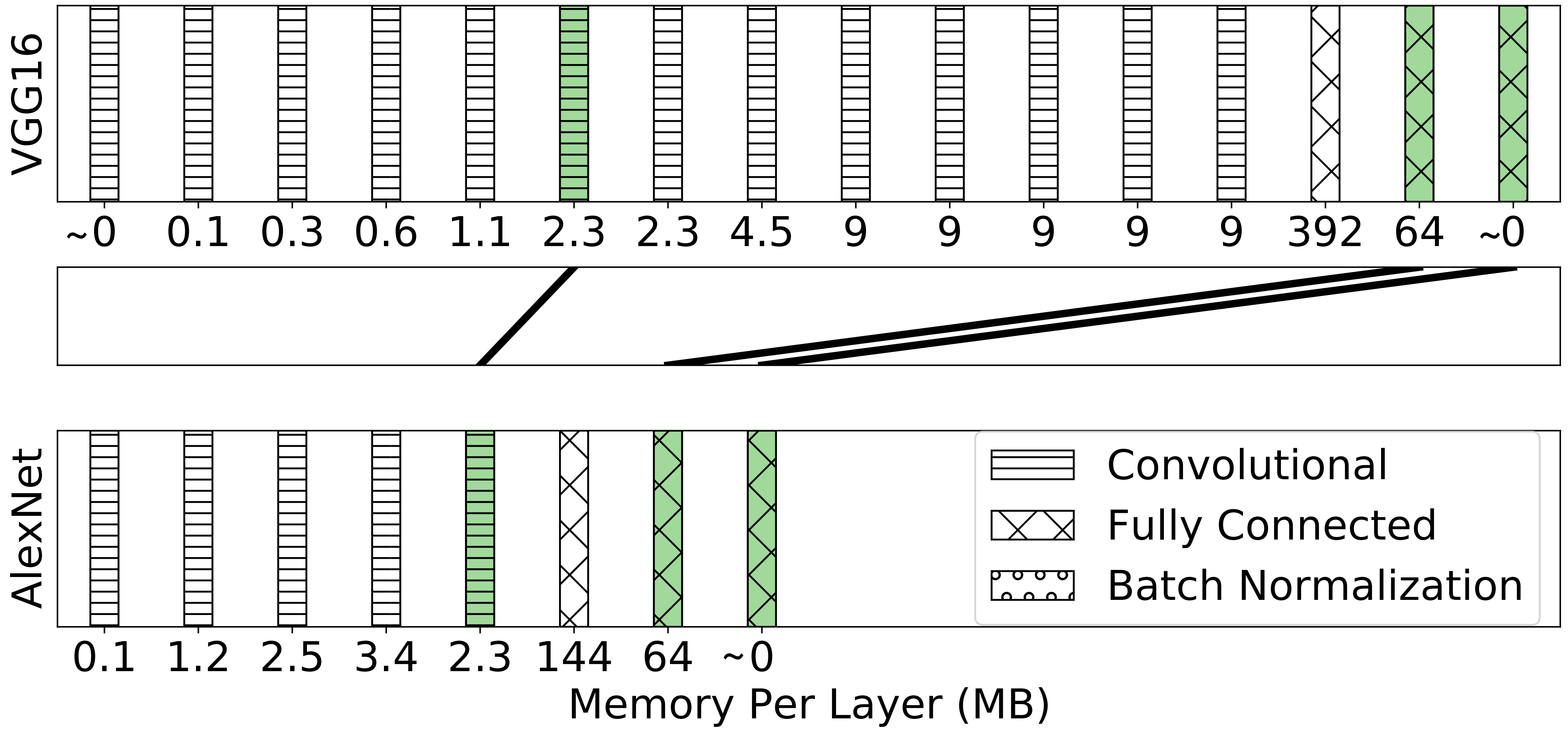
Models from the Same Architecture Family

e.g., VGG16 & VGG19



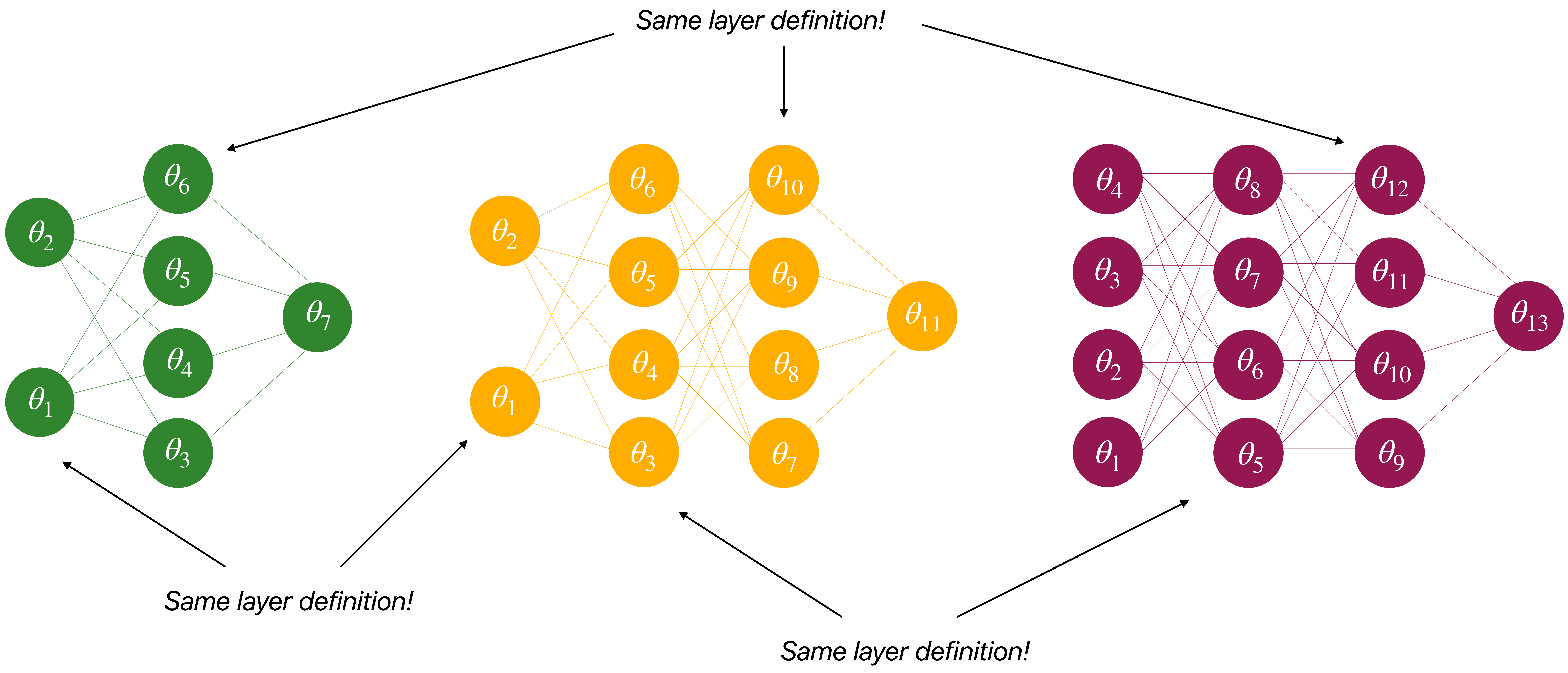
Models from Different Architecture Families

e.g., VGG16 & AlexNet

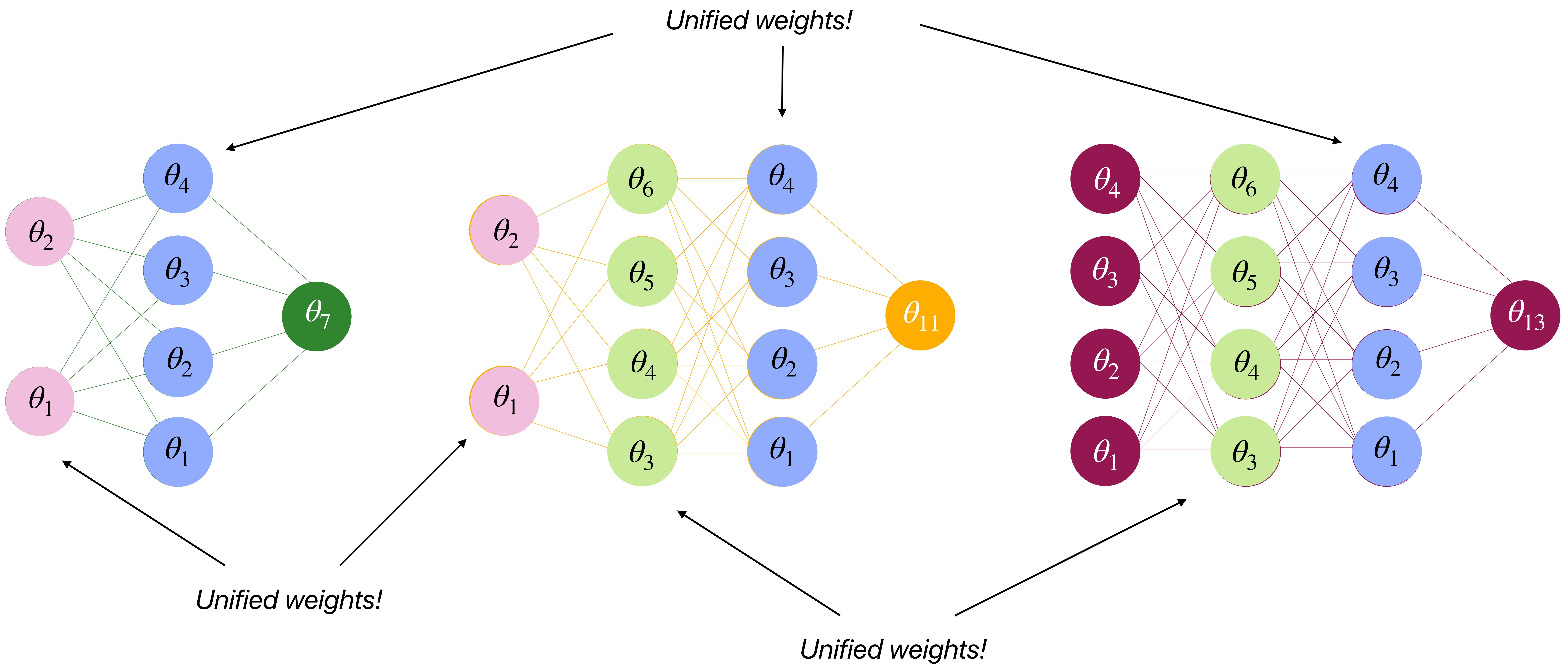


Across 24 different architectures, 43% of all pairs of different models have shared layers

Idea: Find unified weights for shared layers



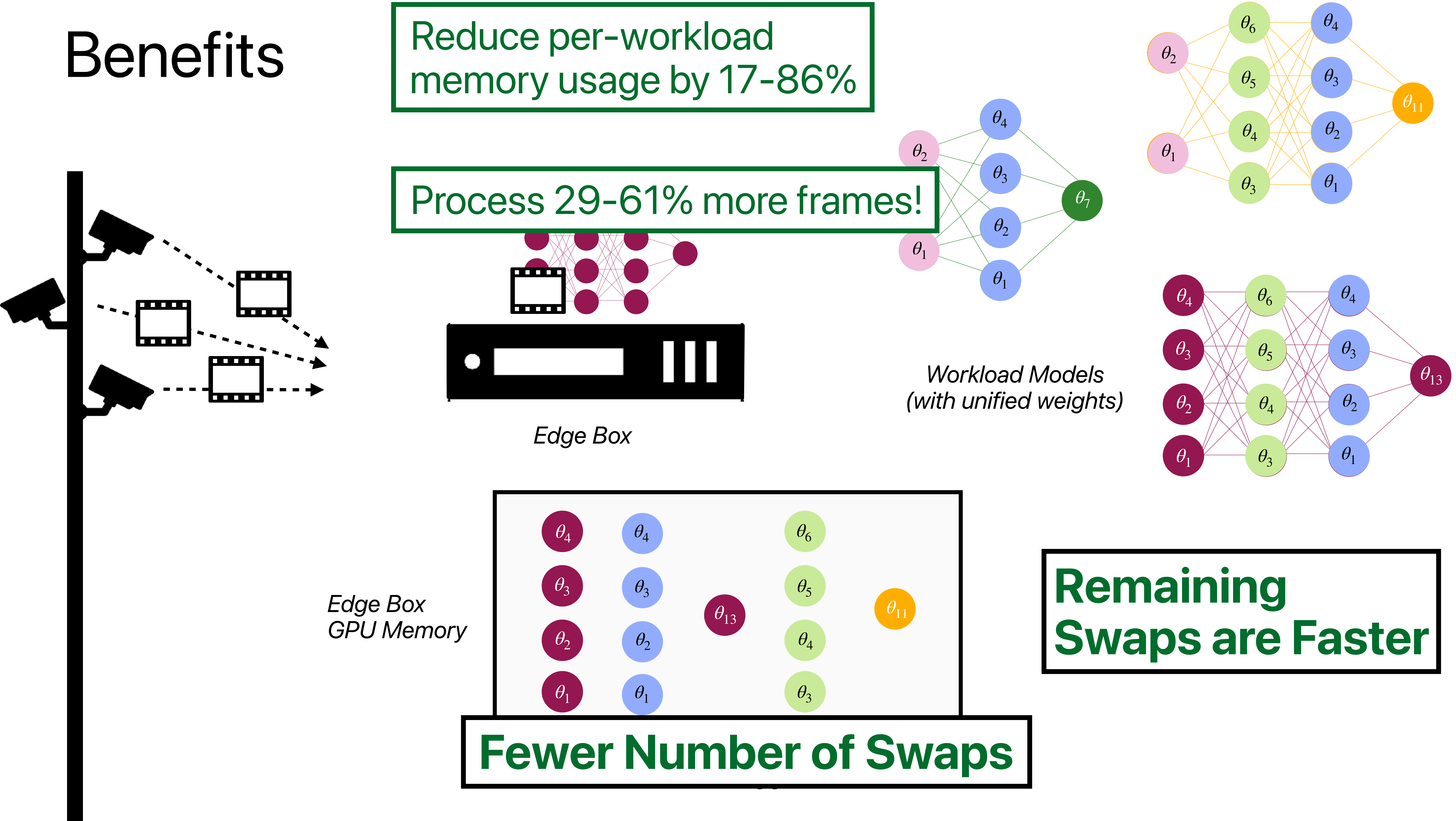
Idea: Find unified weights for shared layers



Benefits

Reduce per-workload memory usage by 17-86%

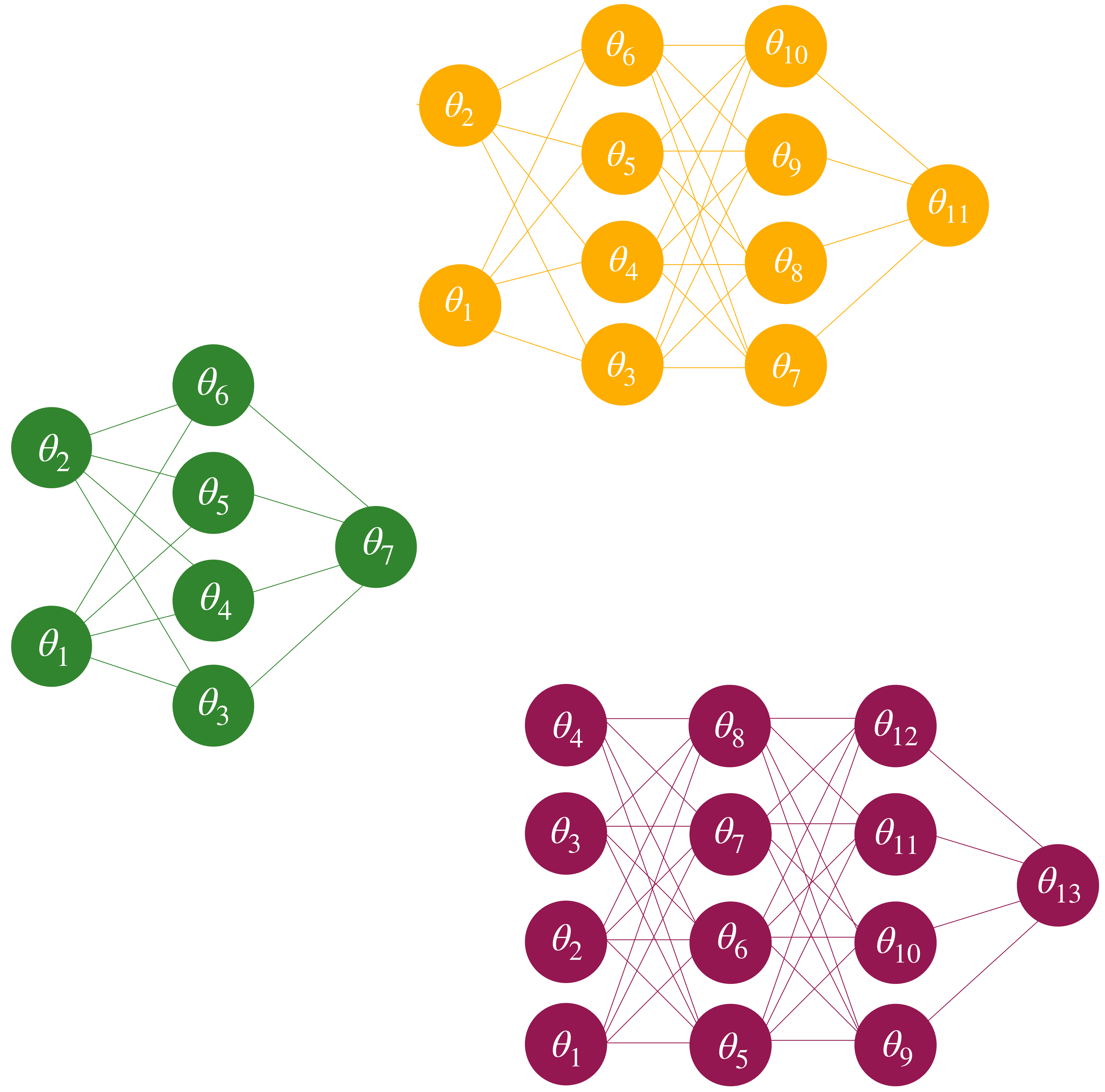
Process 29-61% more frames!



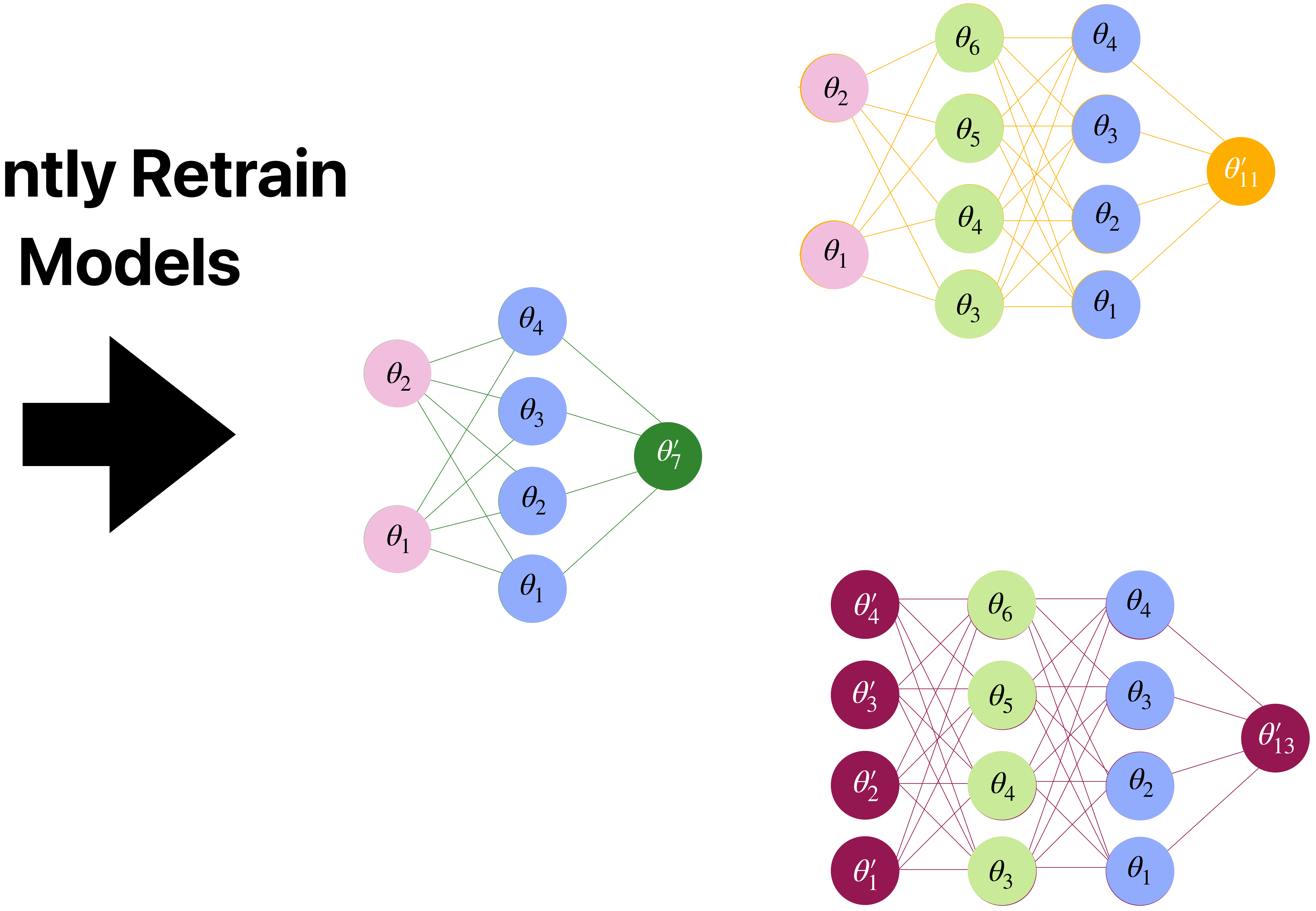
Fewer Number of Swaps

Remaining Swaps are Faster

Model Merging



Jointly Retrain Models



Systems For ML Case Study #2: ML Inference

Gemel: Model Merging for Memory-Efficient, Real-Time Video Analytics at the Edge

Arthi Padmanabhan[*§] Neil Agarwal[*¶] Anand Iyer[†] Ganesh Ananthanarayanan[†]
Yuanchao Shu[‡] Nikolaos Karianakis[†] Guoqing Harry Xu[§] Ravi Netravali[¶]

[§]UCLA [†]Microsoft Research [‡]Zhejiang University [¶]Princeton University

- Key insight: compress layers across models to reduce GPU memory overheads!
- Paper & presentation available @ https://www.usenix.org/conference/nsdi23/presentation/padmanabhan

Systems <-> Machine Learning

- Machine learning for Systems
  - Replacing system heuristics/control with ML algorithms
  - *Examples*: caching eviction policy, ABR algorithm
- Systems for Machine Learning
  - Optimizing system level aspects to improve the machine learning pipeline (e.g., training, inference)
  - *Examples*: use pipeline parallelism to improve resource utilization for large-model training, use inter-model compression to reduce GPU memory overheads for video analytics inference jobs

Systems <-> Machine Learning Resources

- MIT 6.887: Machine Learning for Systems (https://dsg.csail.mit.edu/6.887/assign.php)
- Stanford CS329: Machine Learning Systems Design (https://stanford-cs329s.github.io/)
- UofSC CSCE 585: Machine Learning Systems (https://pooyanjamshidi.github.io/mls/)
- Princeton COS 598D: Systems and Machine Learning (https://www.cs.princeton.edu/courses/archive/spring21/cos598D/general.html)
- Cassie Kozyrkov's Making Friends with Machine Learning (https://www.youtube.com/watch?v=1vkb7BCMQd0)
- Chip Huyen's MLOps Guide (https://huyenchip.com/mlops/)