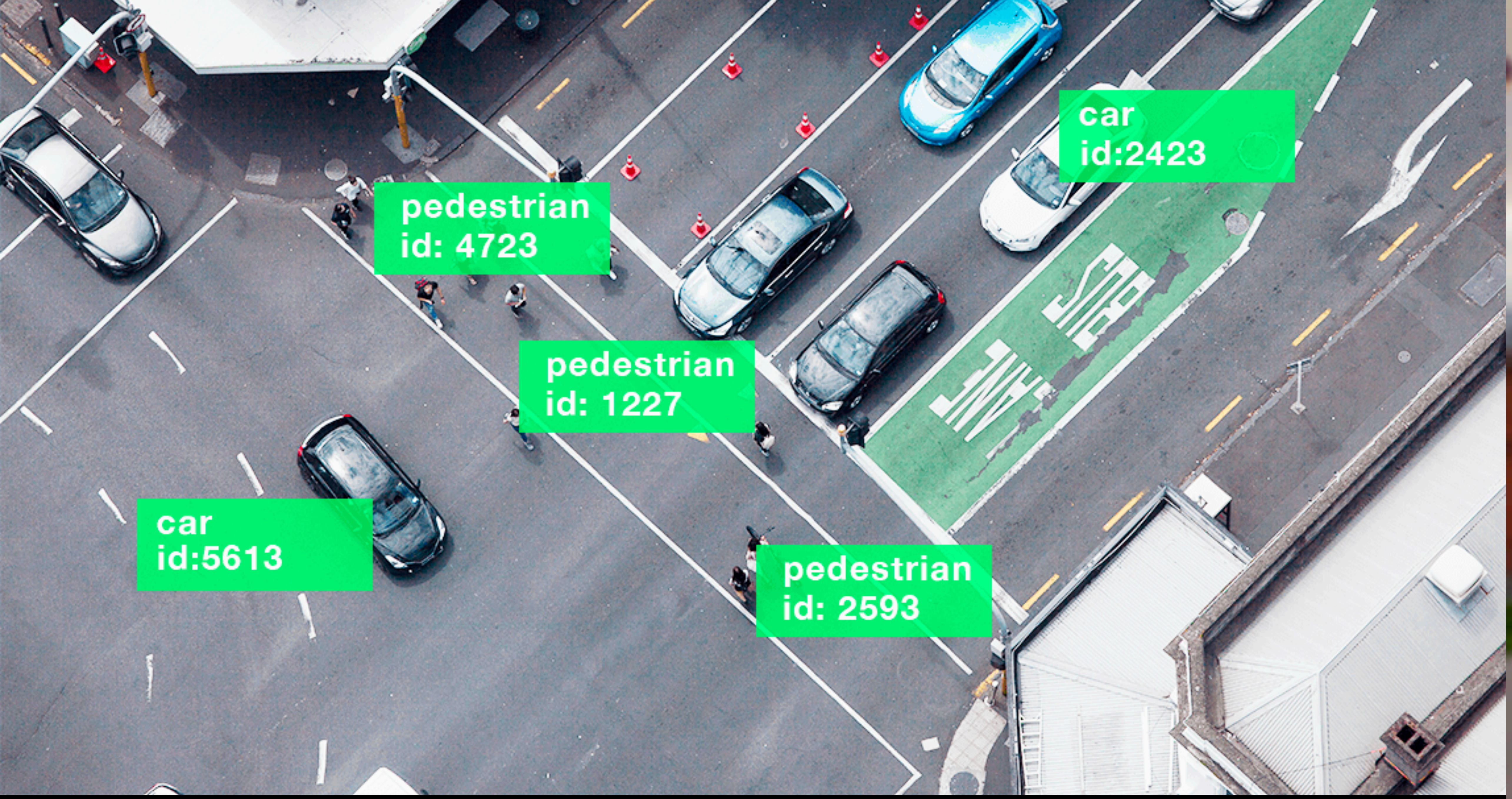
Gemel: Model Merging for Memory-Efficient, Real-Time Video Analytics at the Edge

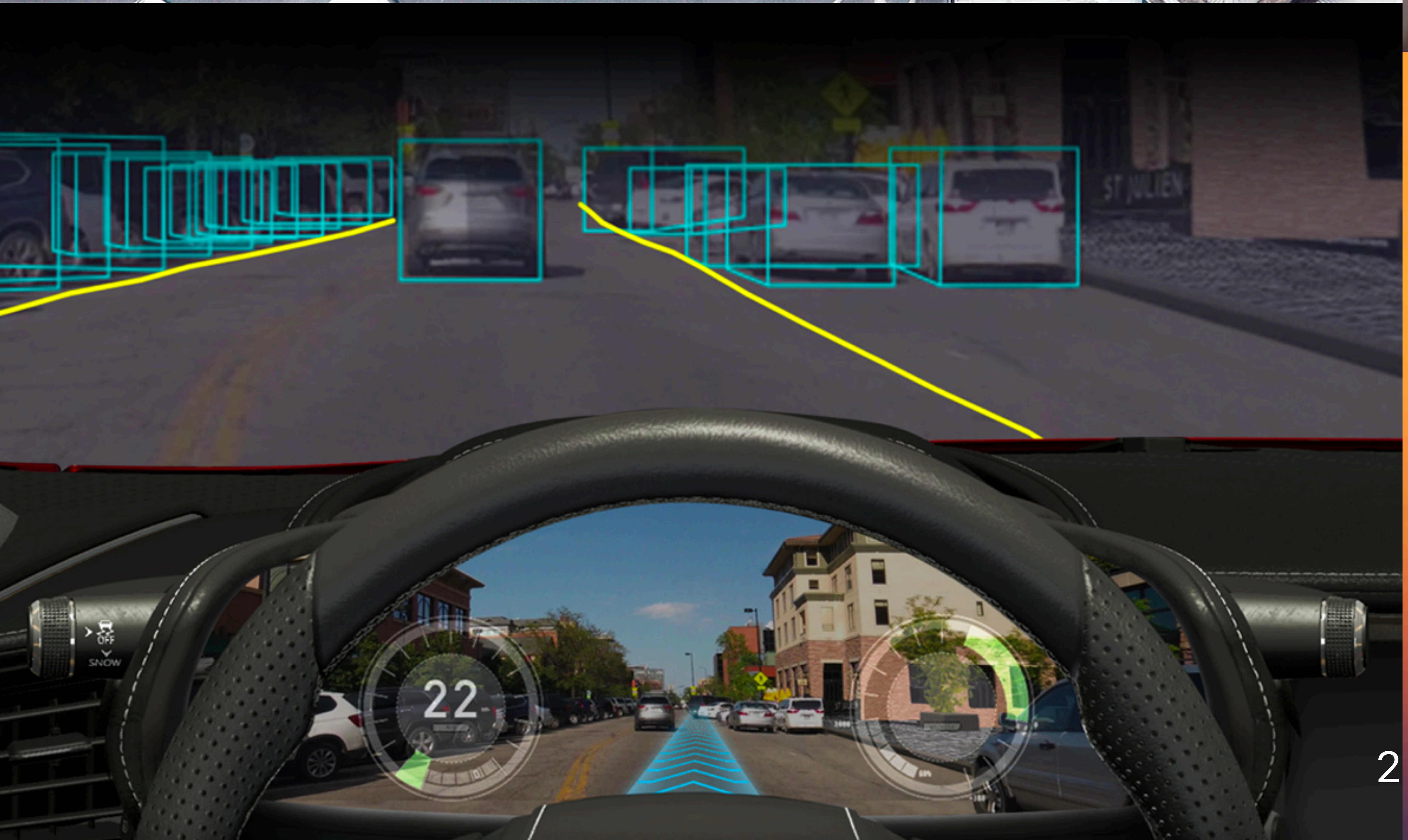
Neil Agarwal*, Arthi Padmanabhan*, Anand Iyer, Ganesh Ananthanarayanan, Yuanchao Shu, Nikolaos Karianakis, Harry Xu, Ravi Netravali

PRINCETON UNIVERSITY UCLA 浙江大學 ZHEJIANG UNIVERSITY Microsoft

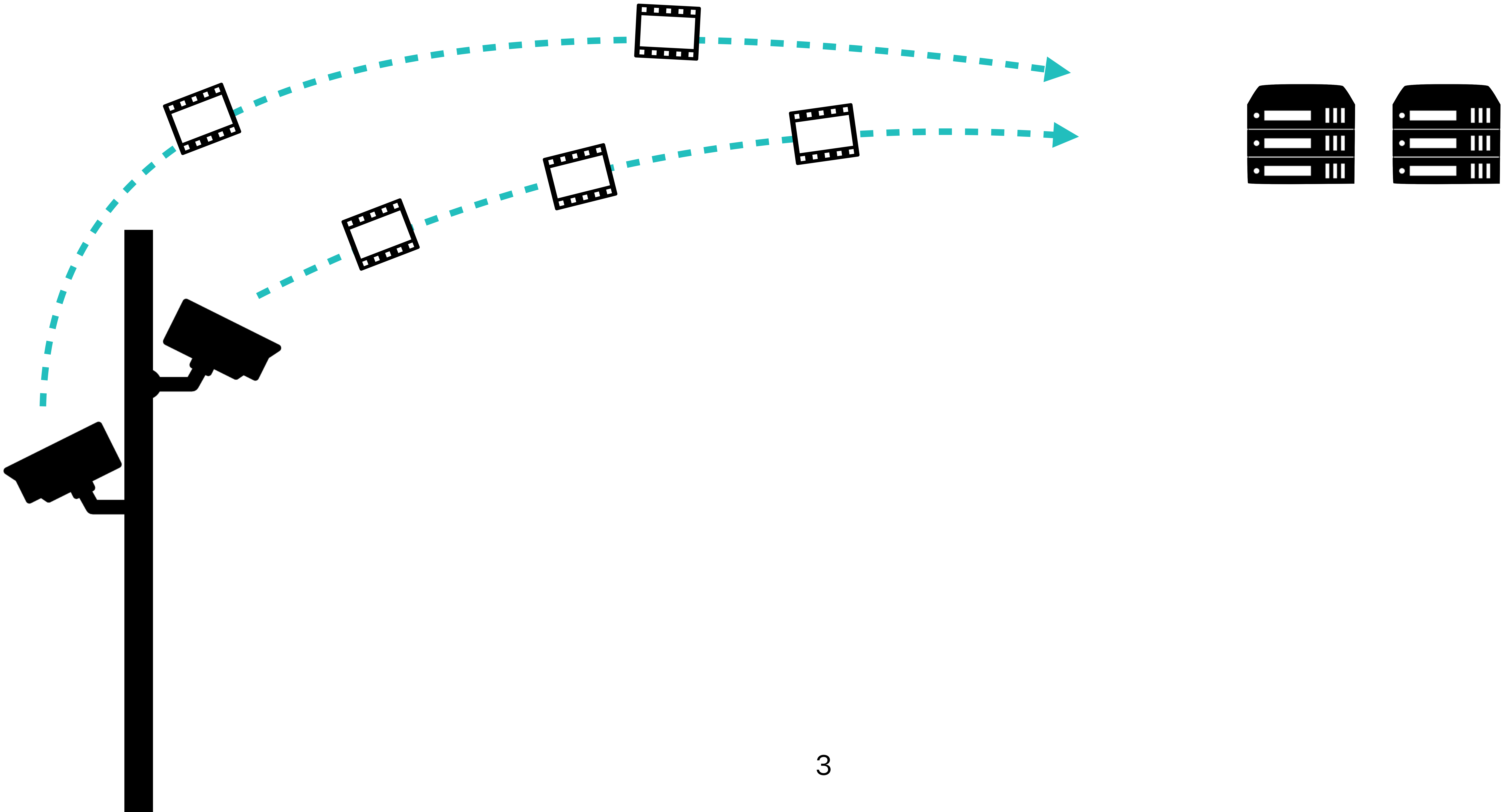
April 18, 2023

NSDI 2023

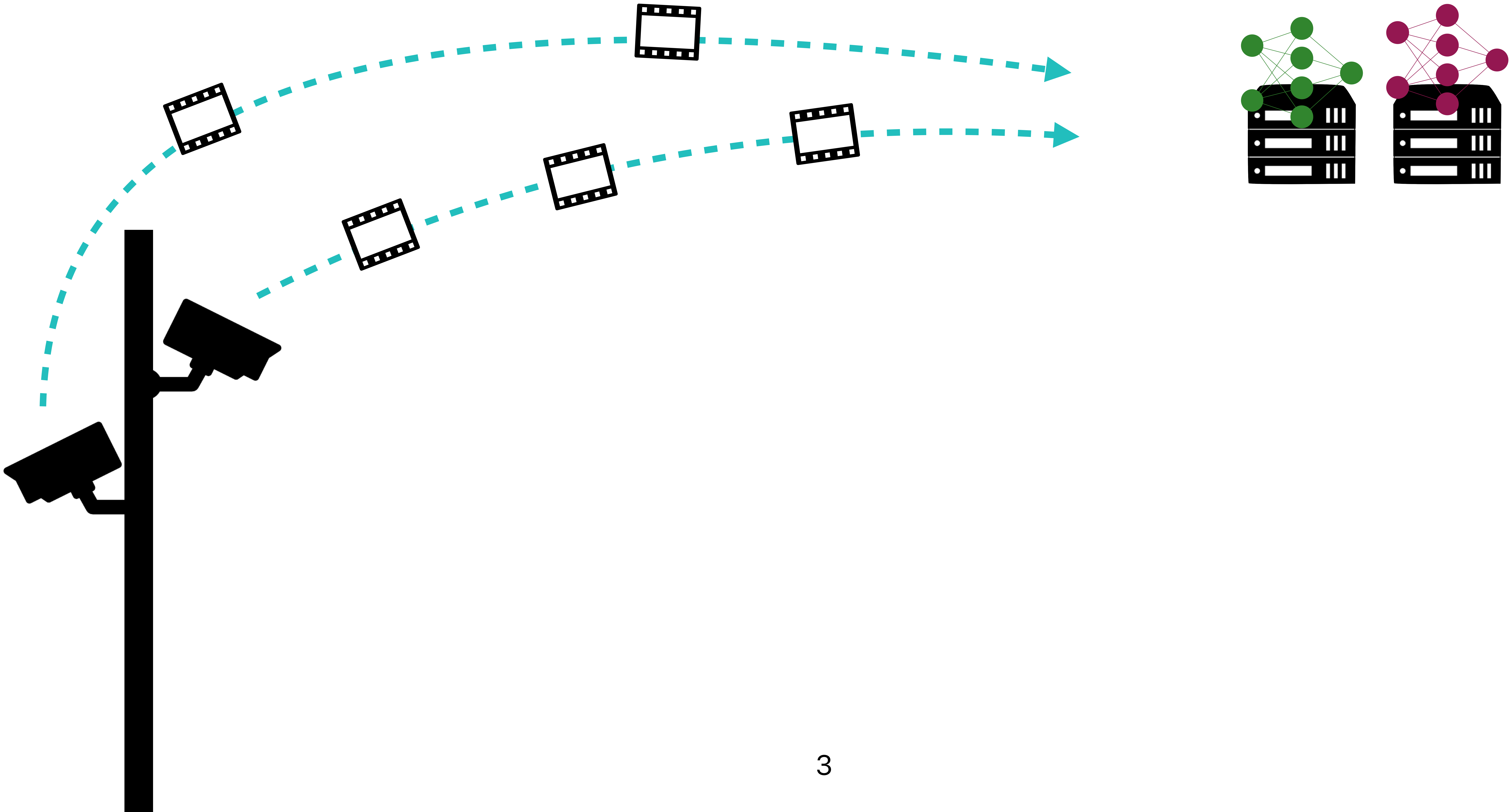

car
id:2423
pedestrian
id: 4723
pedestrian
id: 1227
car
id:5613
pedestrian
id: 2593


22

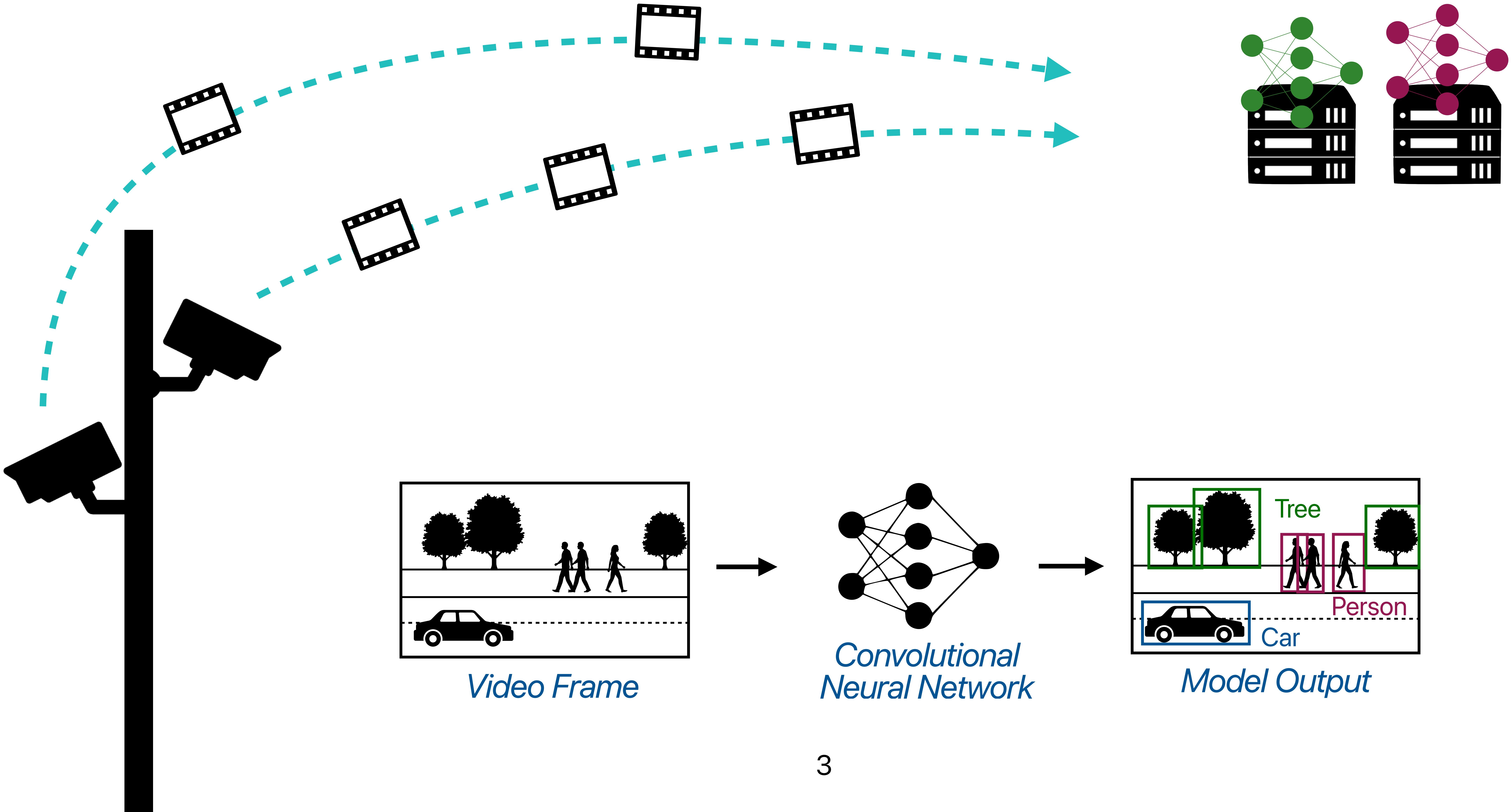

Live Video Analytics Pipeline



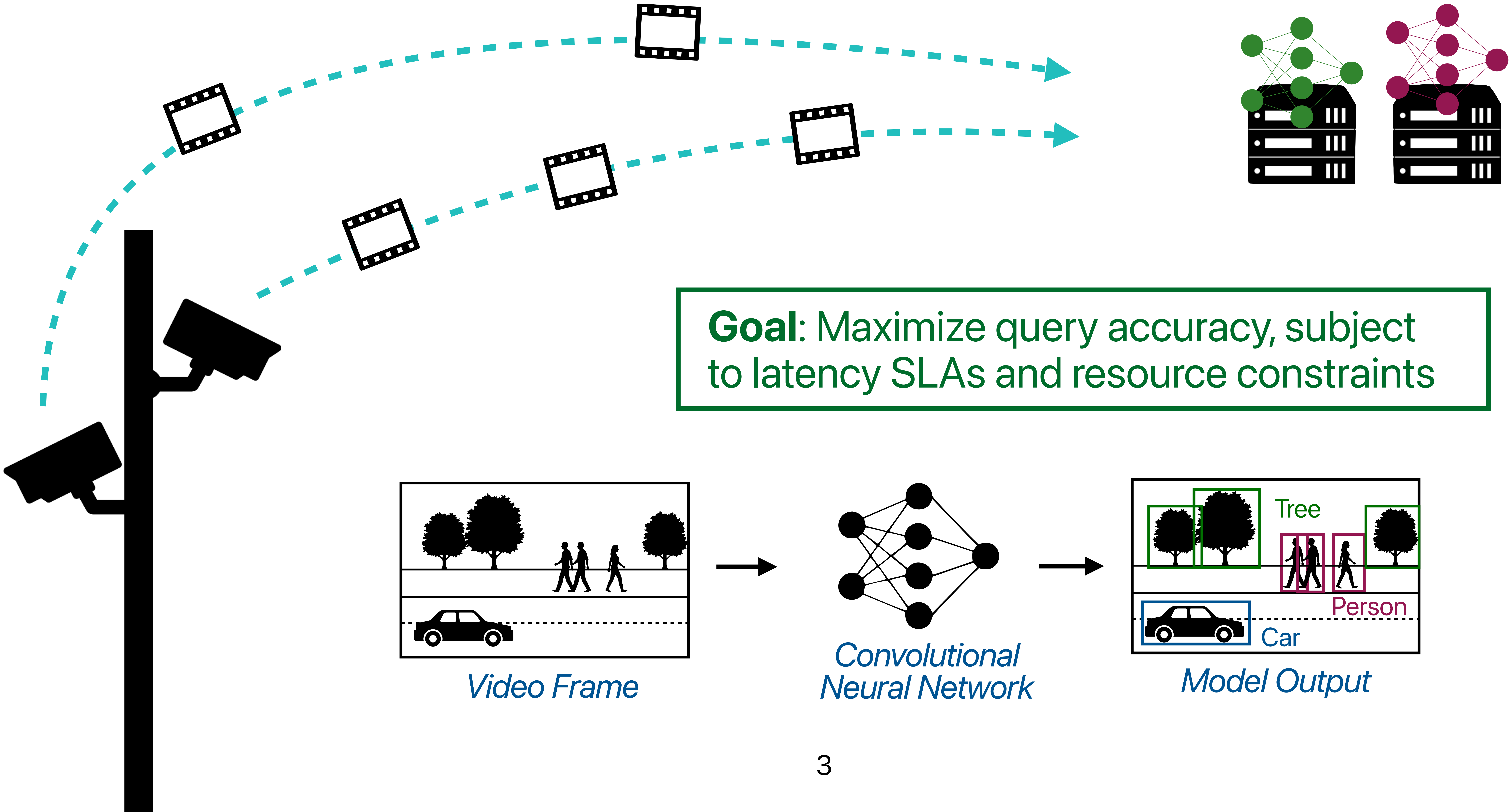
Live Video Analytics Pipeline



Live Video Analytics Pipeline

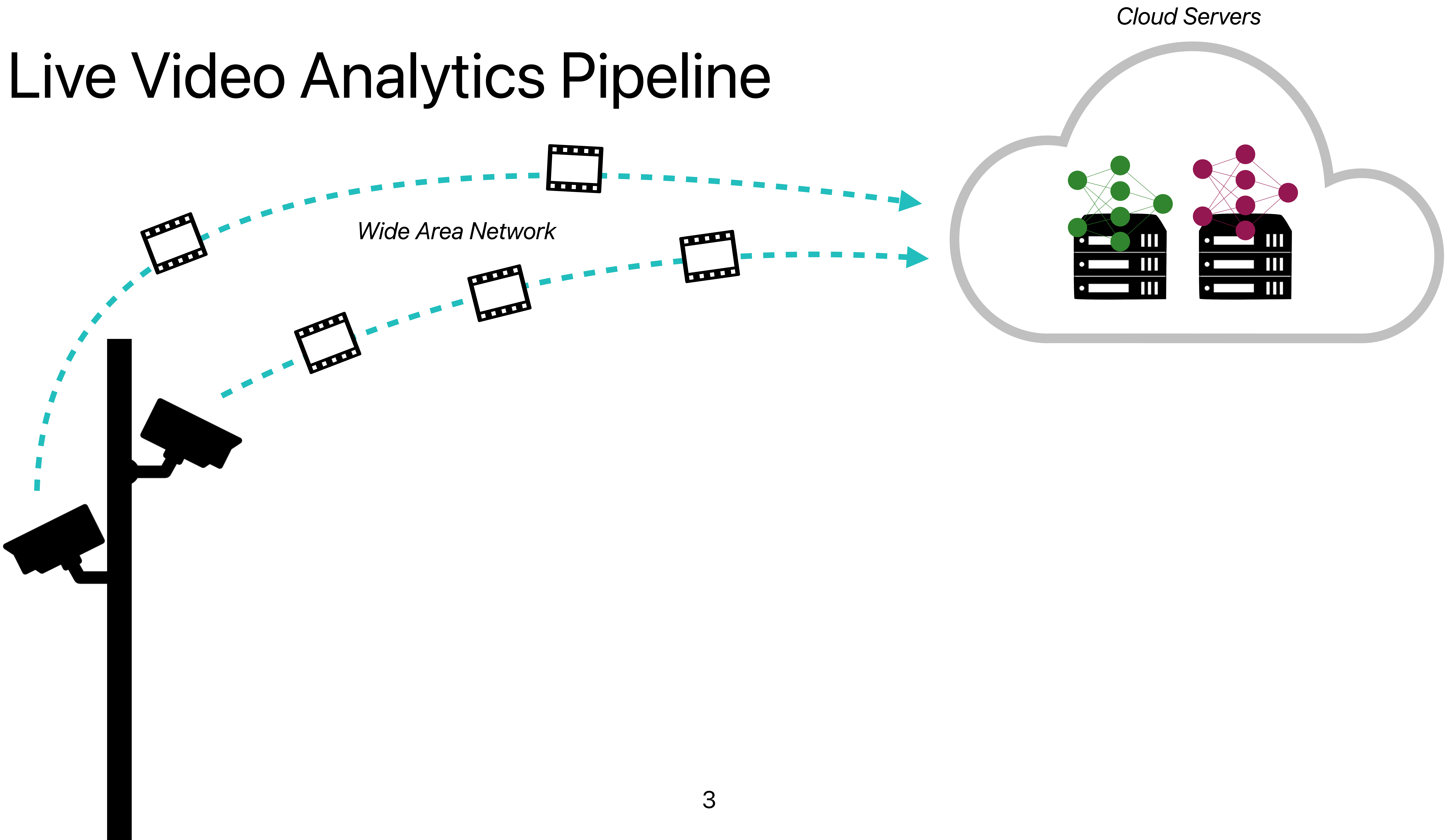


Live Video Analytics Pipeline

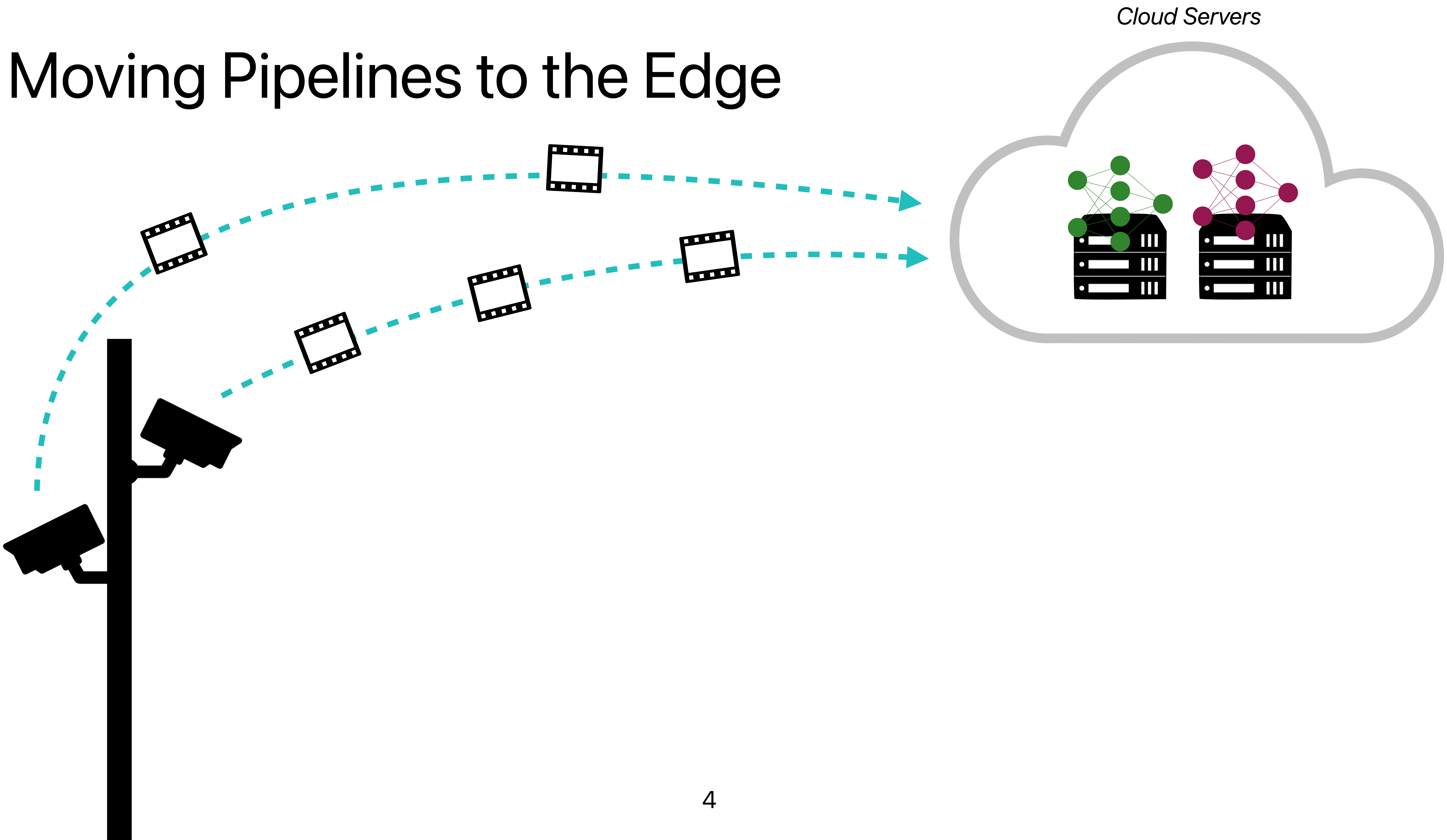


Goal: Maximize query accuracy, subject to latency SLAs and resource constraints

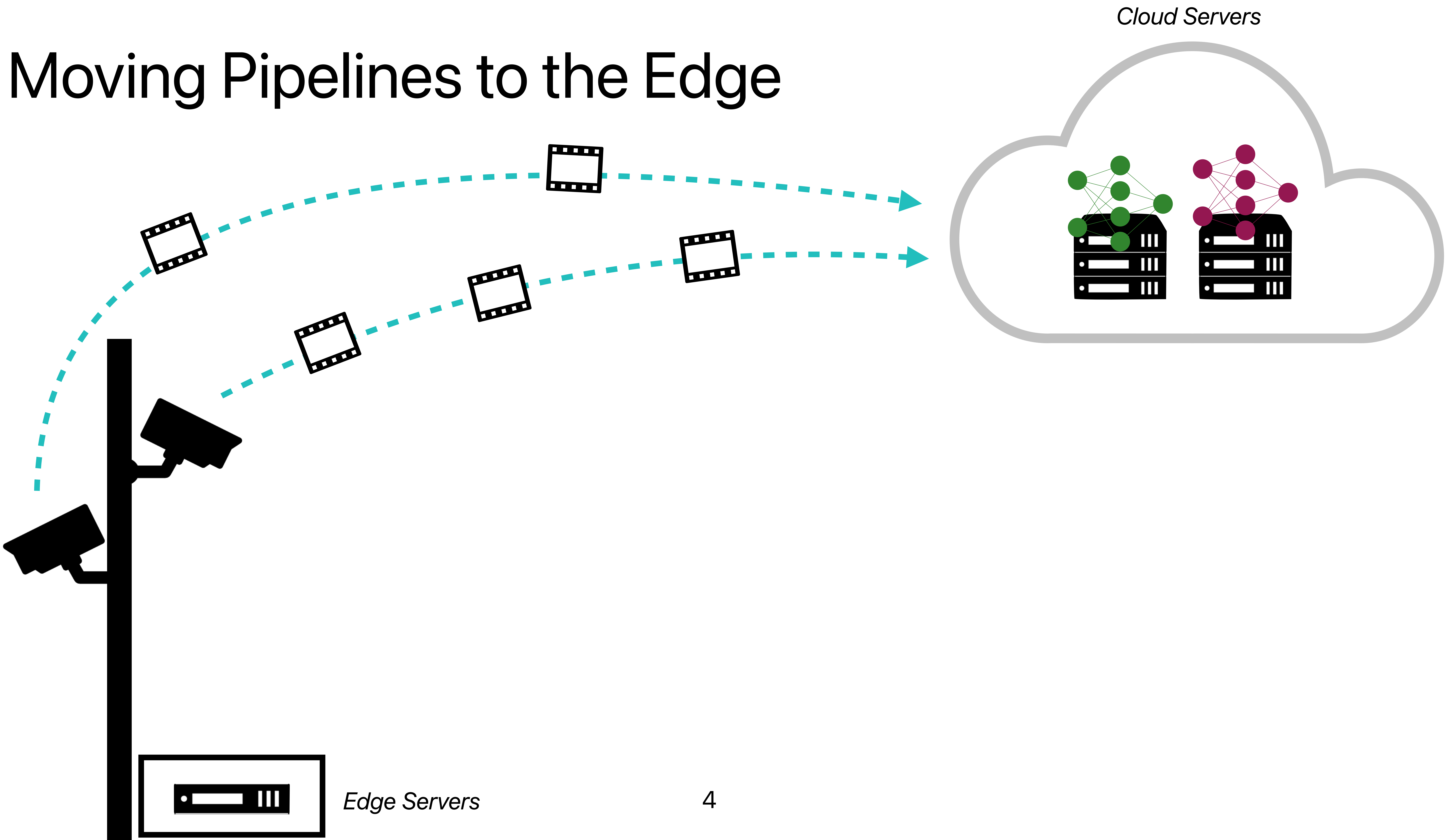
Live Video Analytics Pipeline



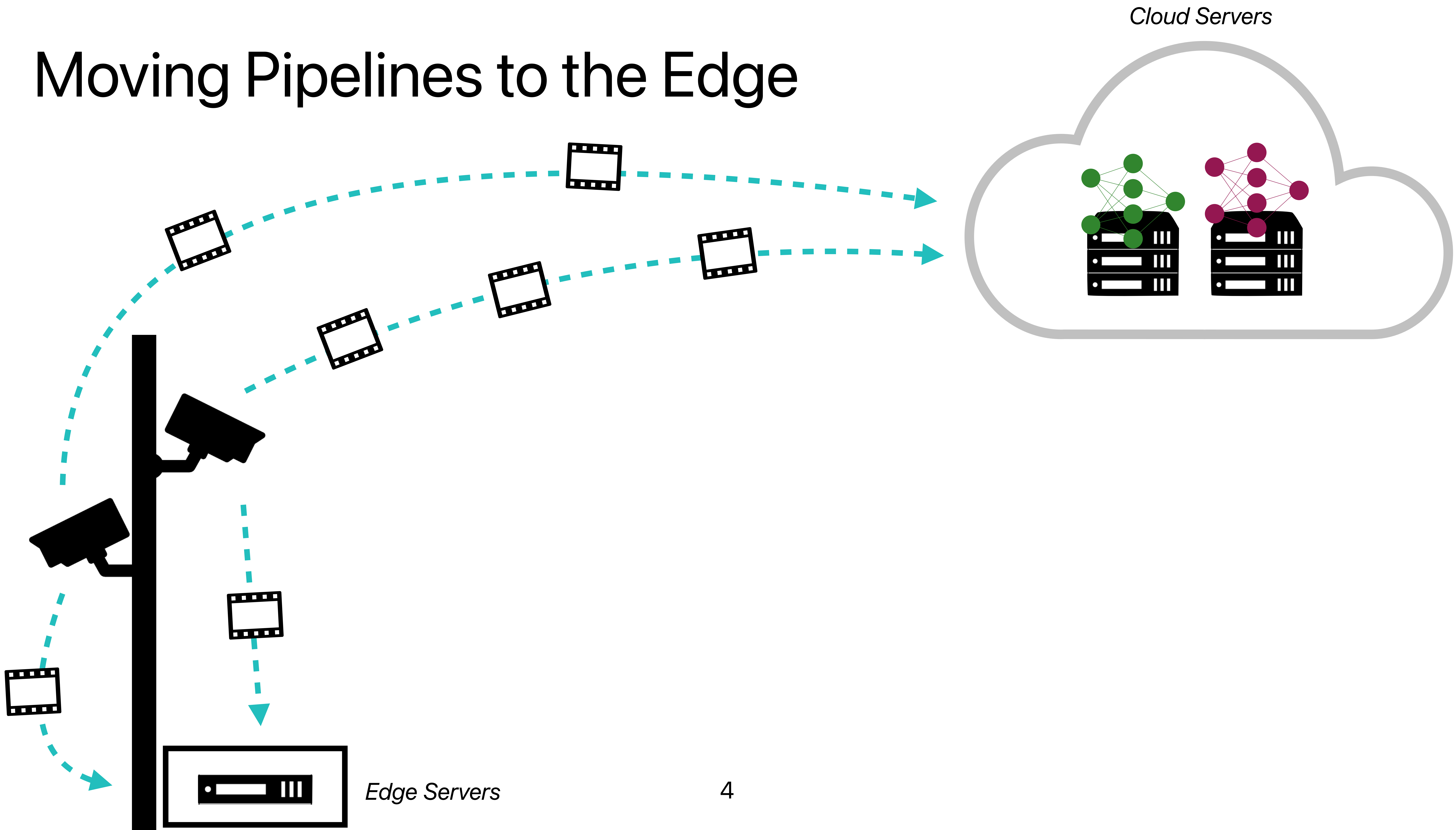
Moving Pipelines to the Edge



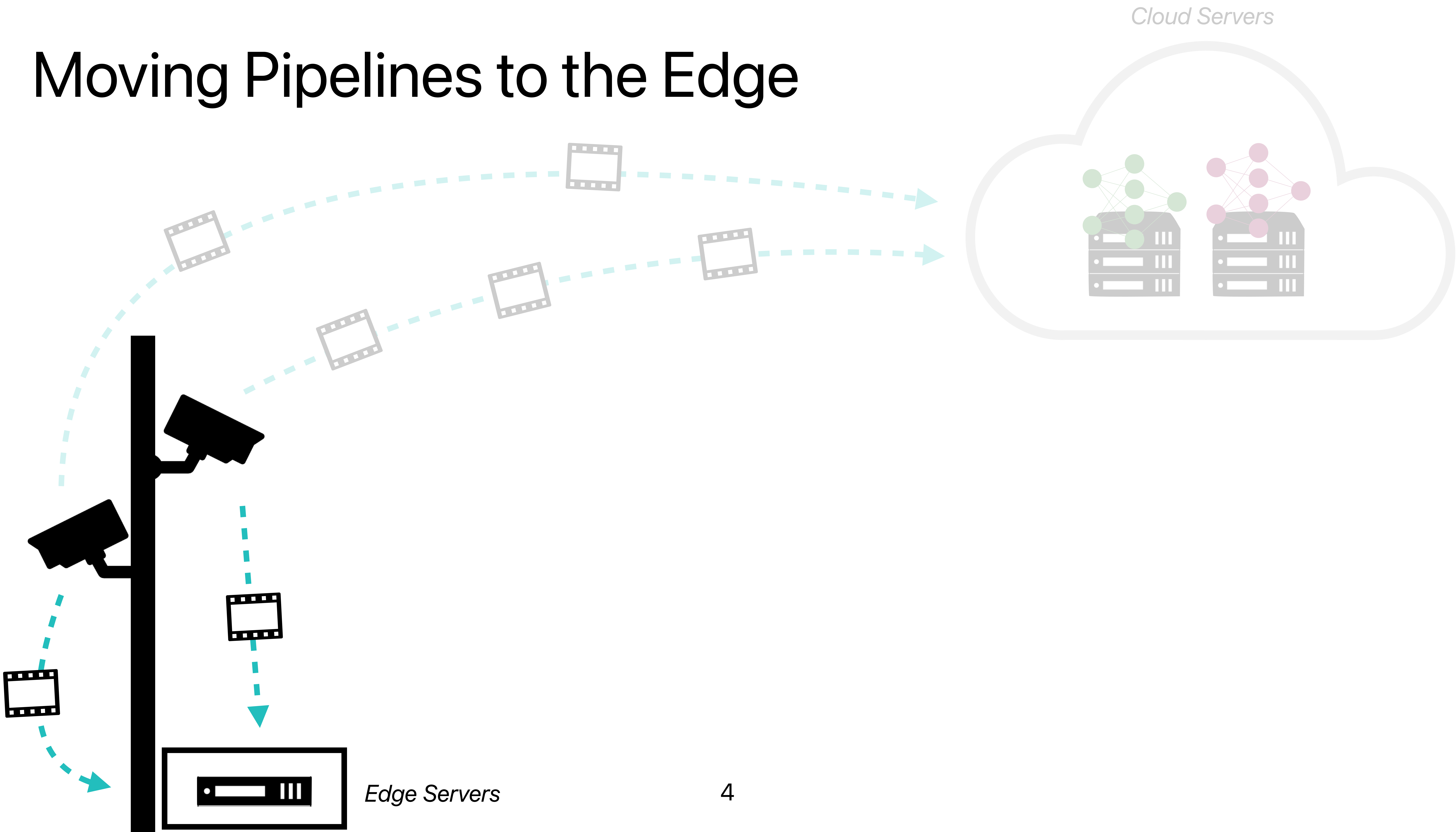
Moving Pipelines to the Edge



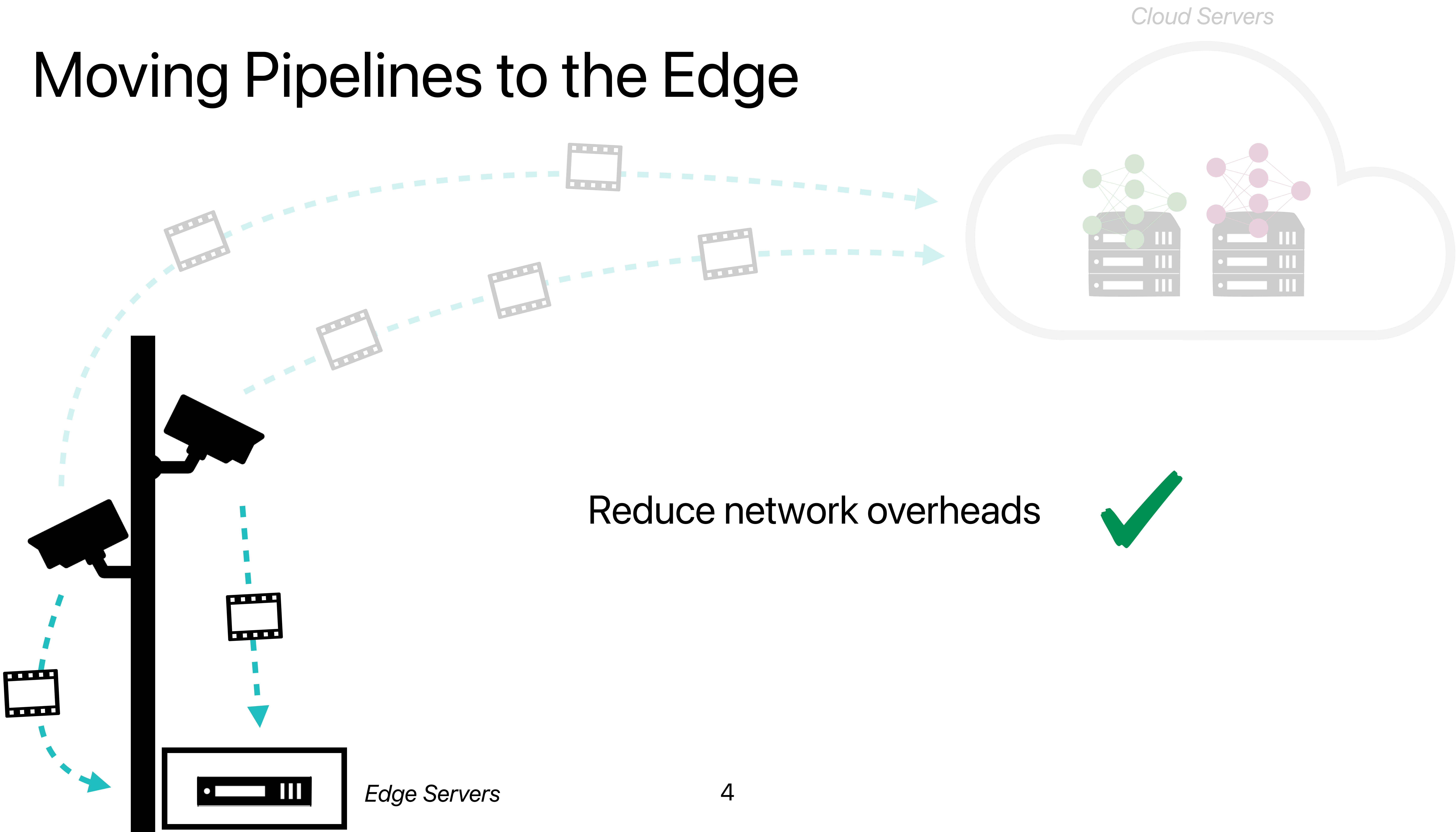
Moving Pipelines to the Edge



Moving Pipelines to the Edge

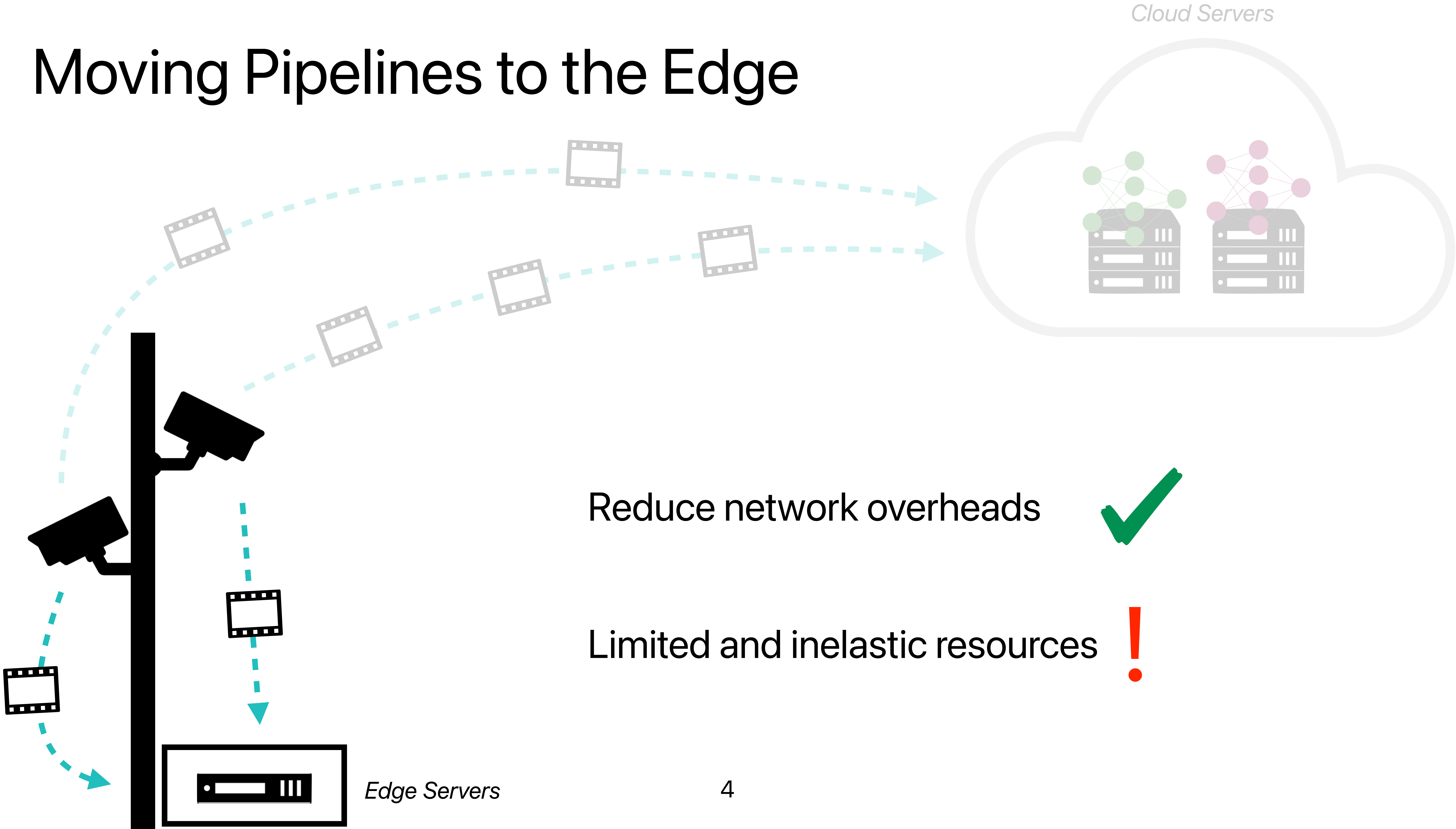


Moving Pipelines to the Edge



Reduce network overheads ✓

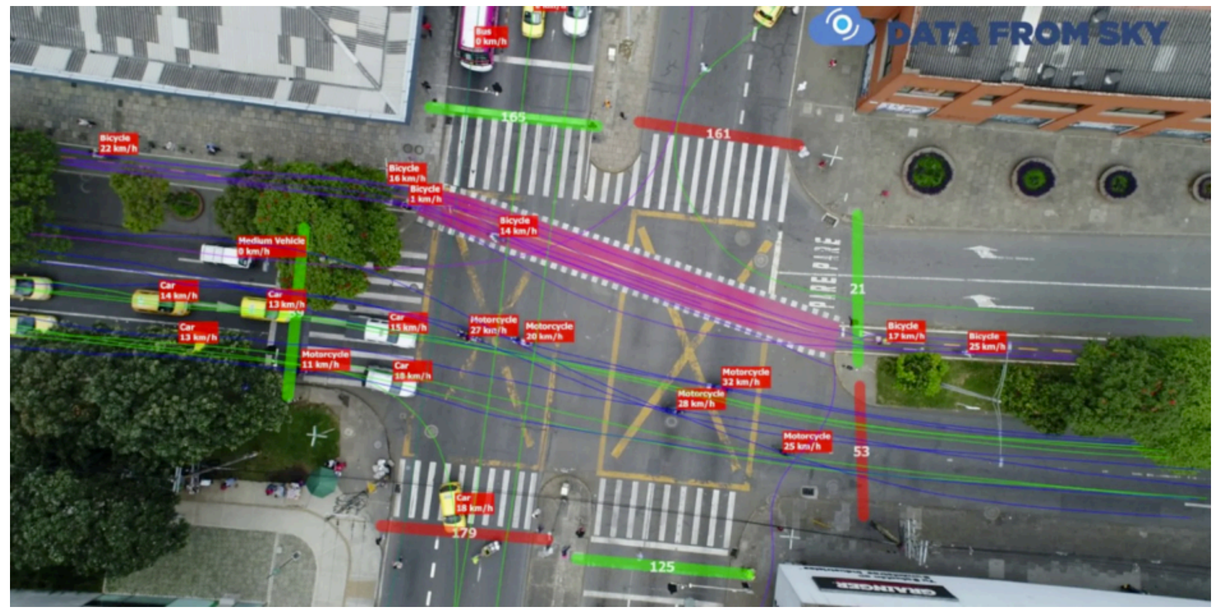
Moving Pipelines to the Edge



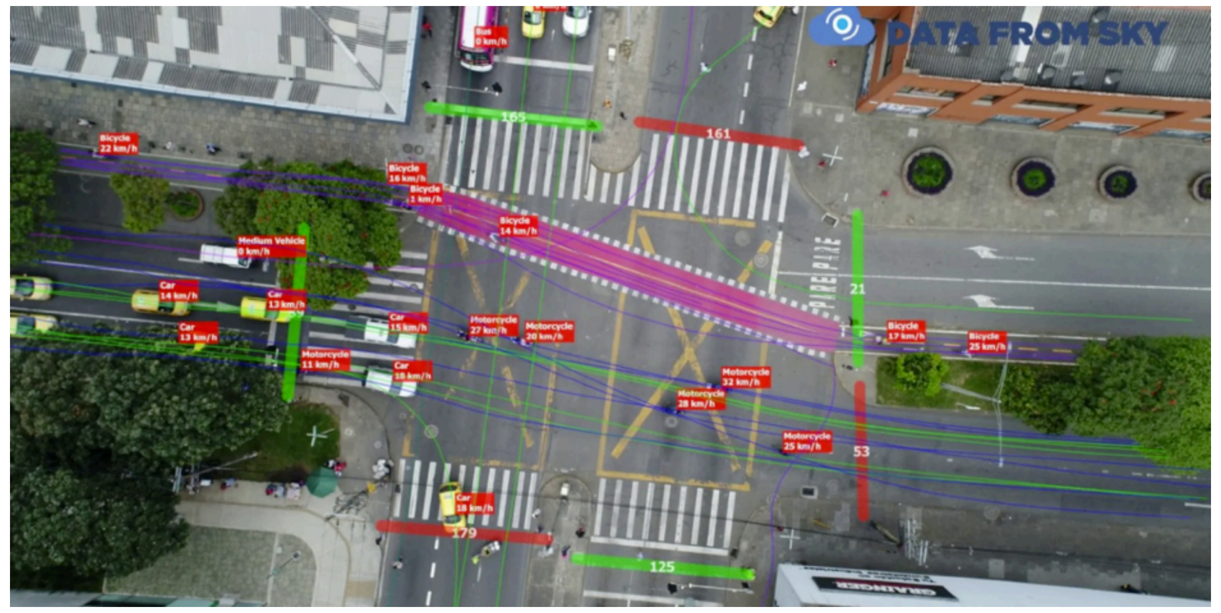
Reduce network overheads

Limited and inelastic resources

Edge Workloads in the Wild

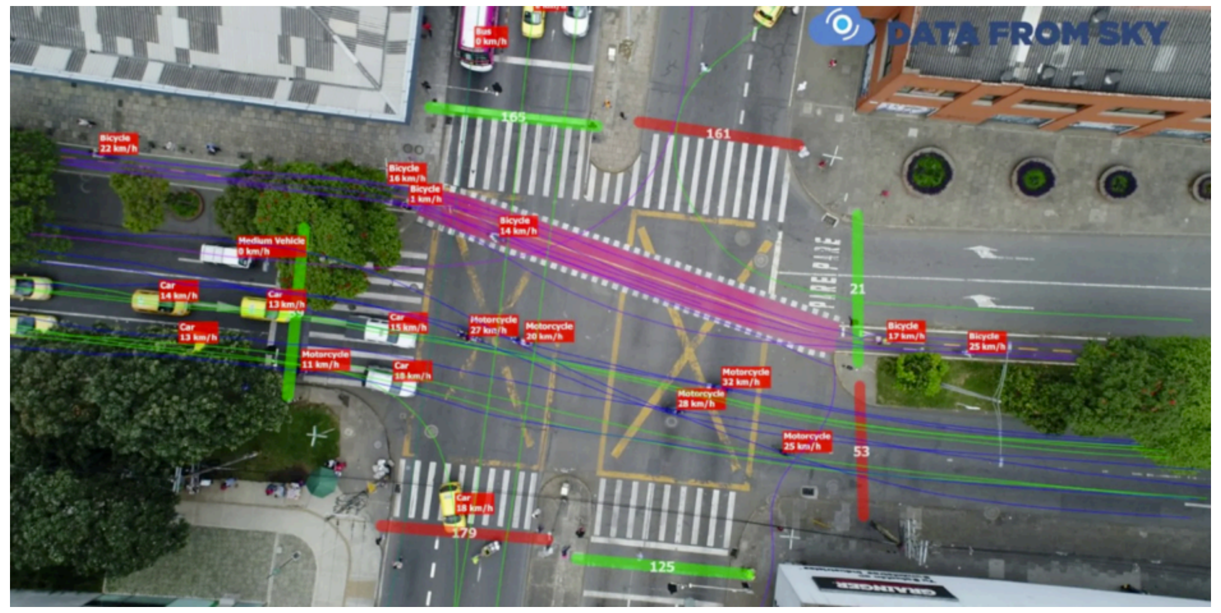


Edge Workloads in the Wild



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

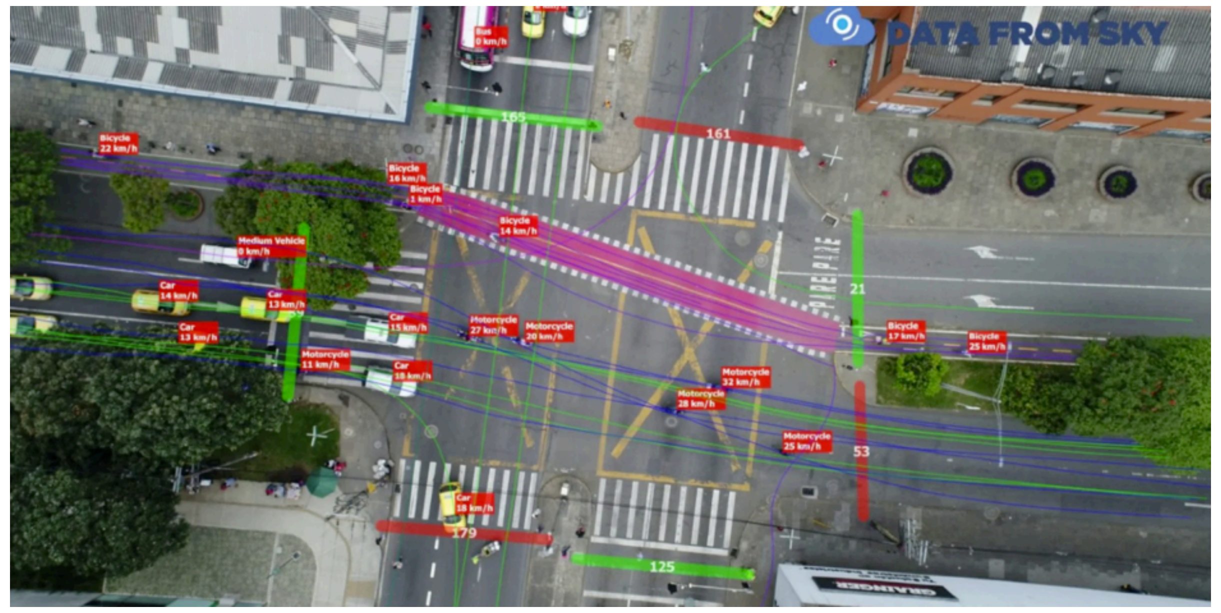
Edge Workloads in the Wild



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

Query: <camera feed, model, task>

Edge Workloads in the Wild



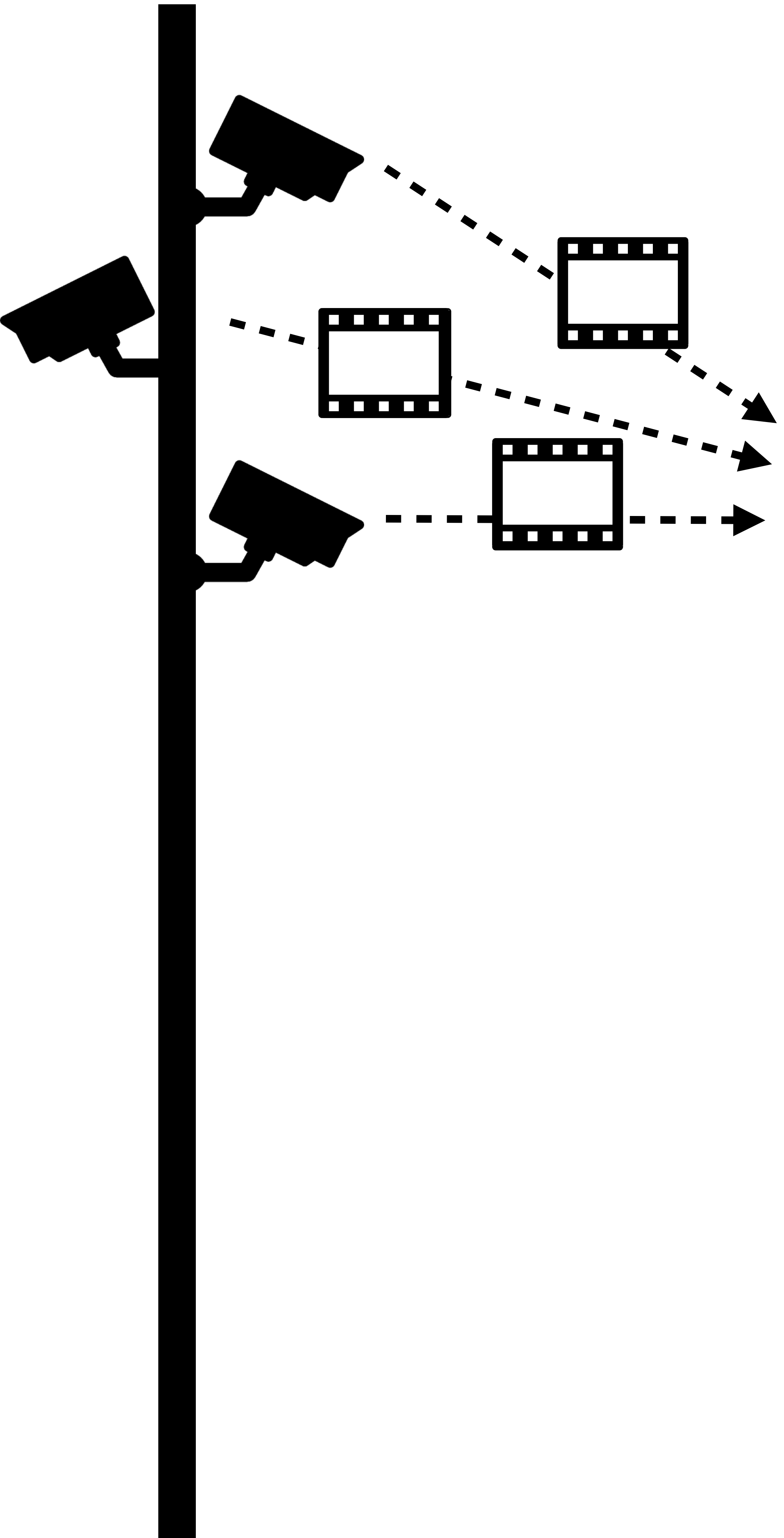
Pilot video analytics deployment across 2 major US cities, targeted at road traffic monitoring

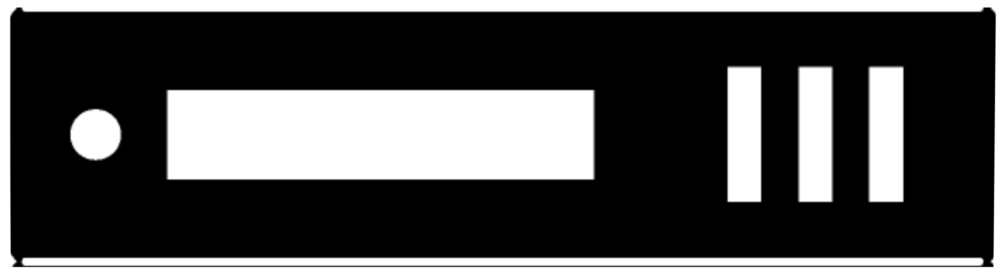
Query: <camera feed, model, task>

Sample Workload

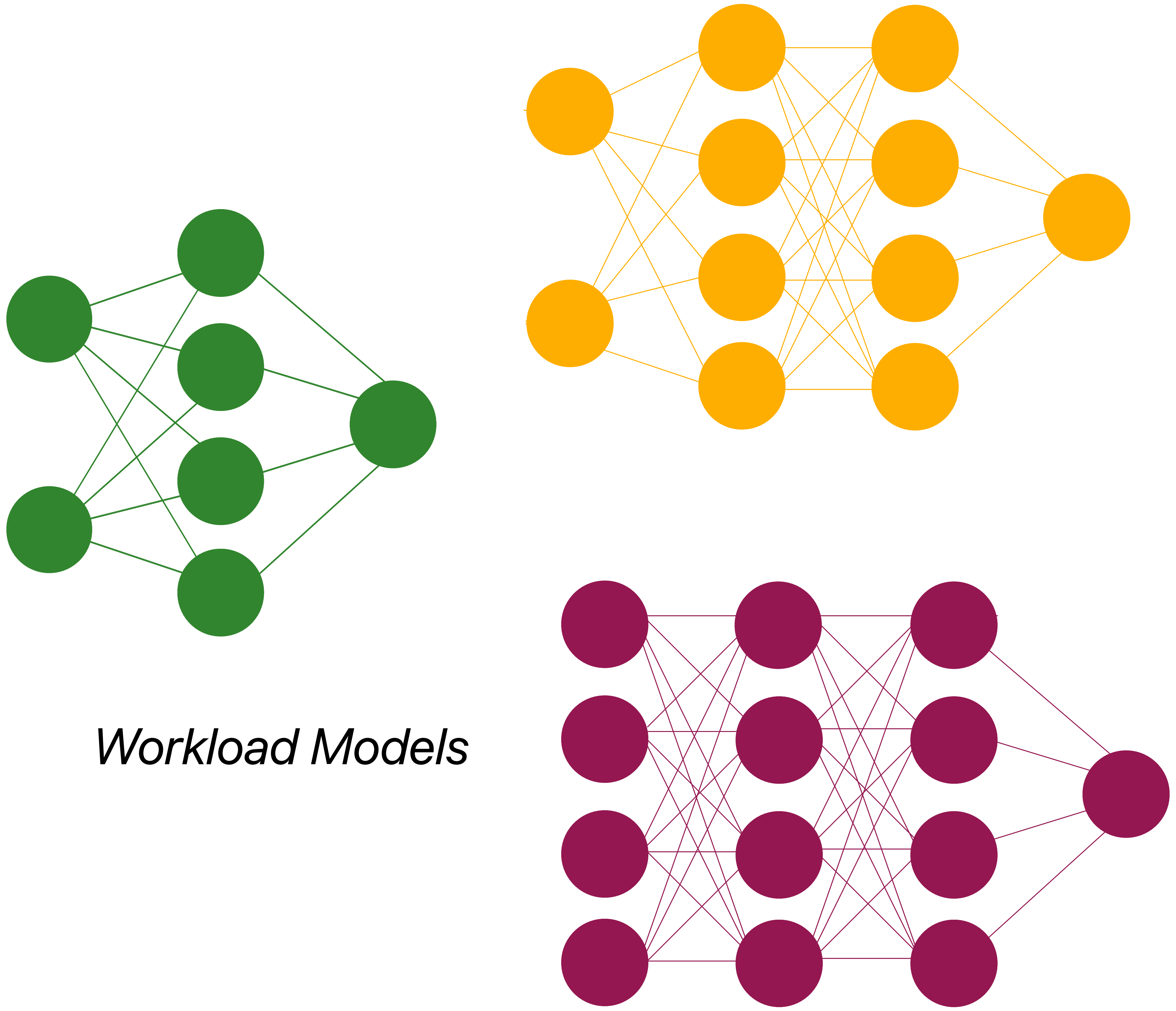
Query #	Camera Feed	Model Architecture	Task Description
1	3	FRCNN-R50	Object detection of cars
2	1	YOLOv3	Object detection of people
3	1	Inception	Binary Classification of people, vehicles
4	6	ResNet50	Binary Classification of cars, buses, trucks
5	3	Tiny-YOLOv3	Object Detection of people
…	…	…	…

Executing Edge Workloads



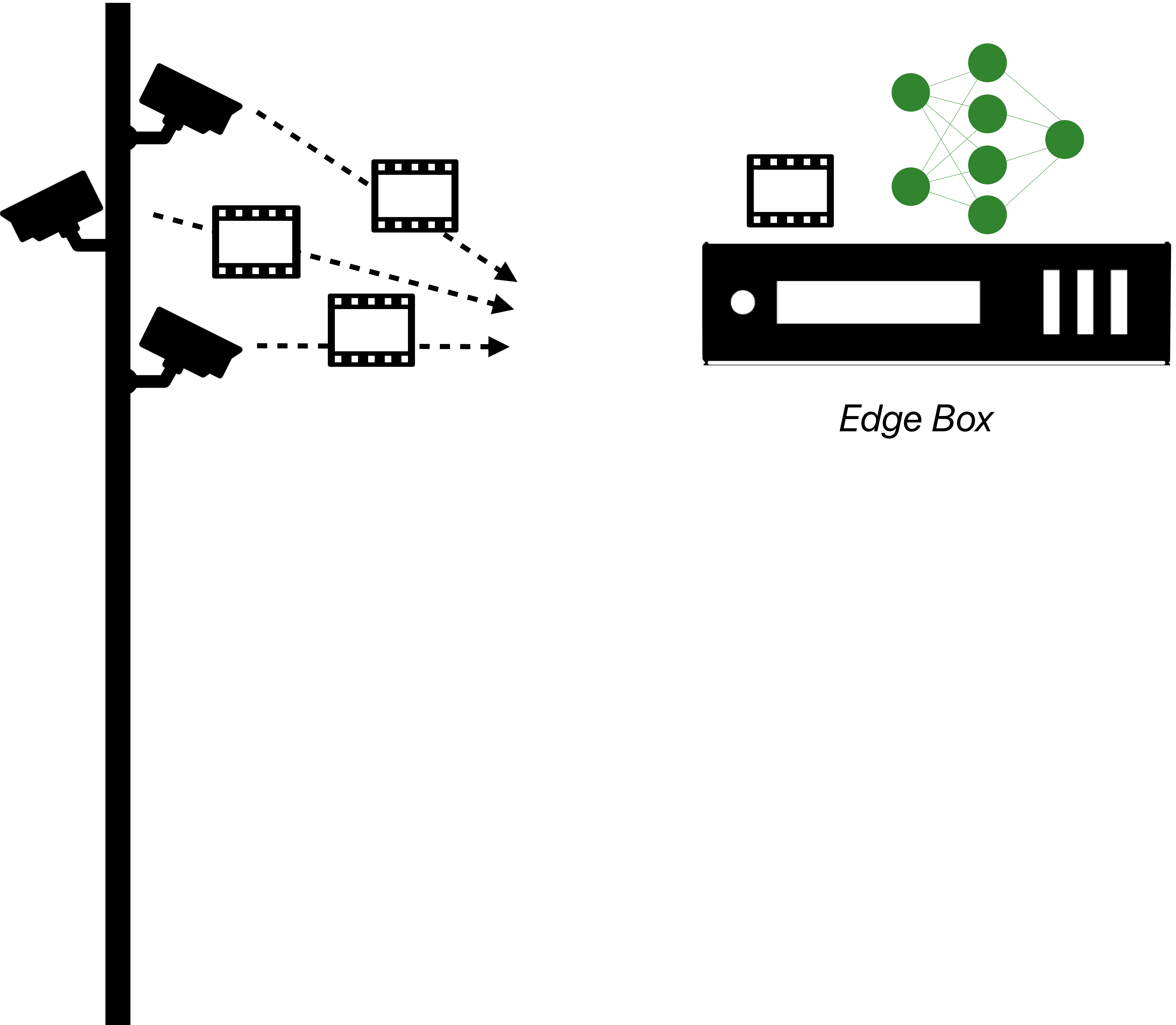


Edge Box

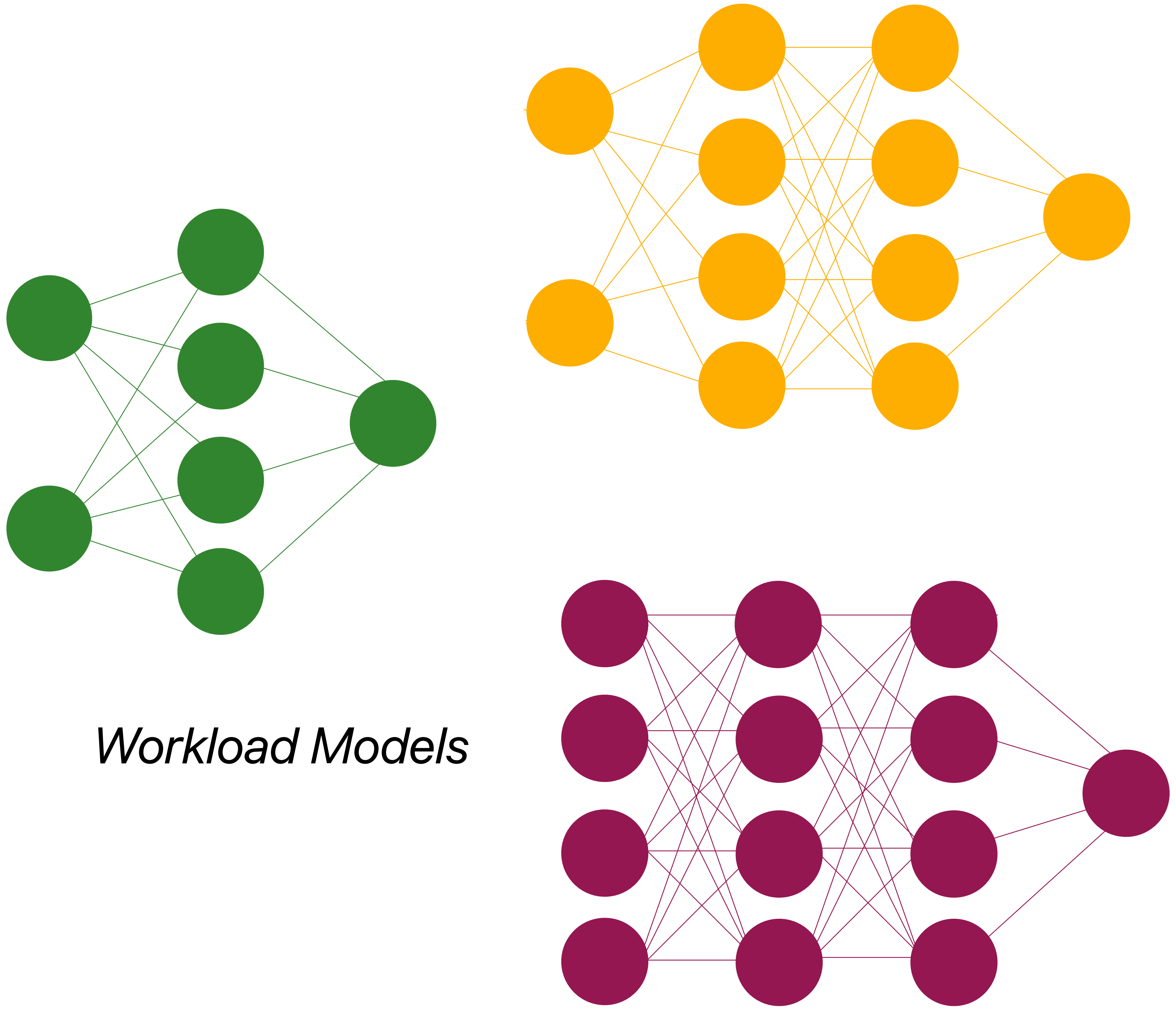


Workload Models

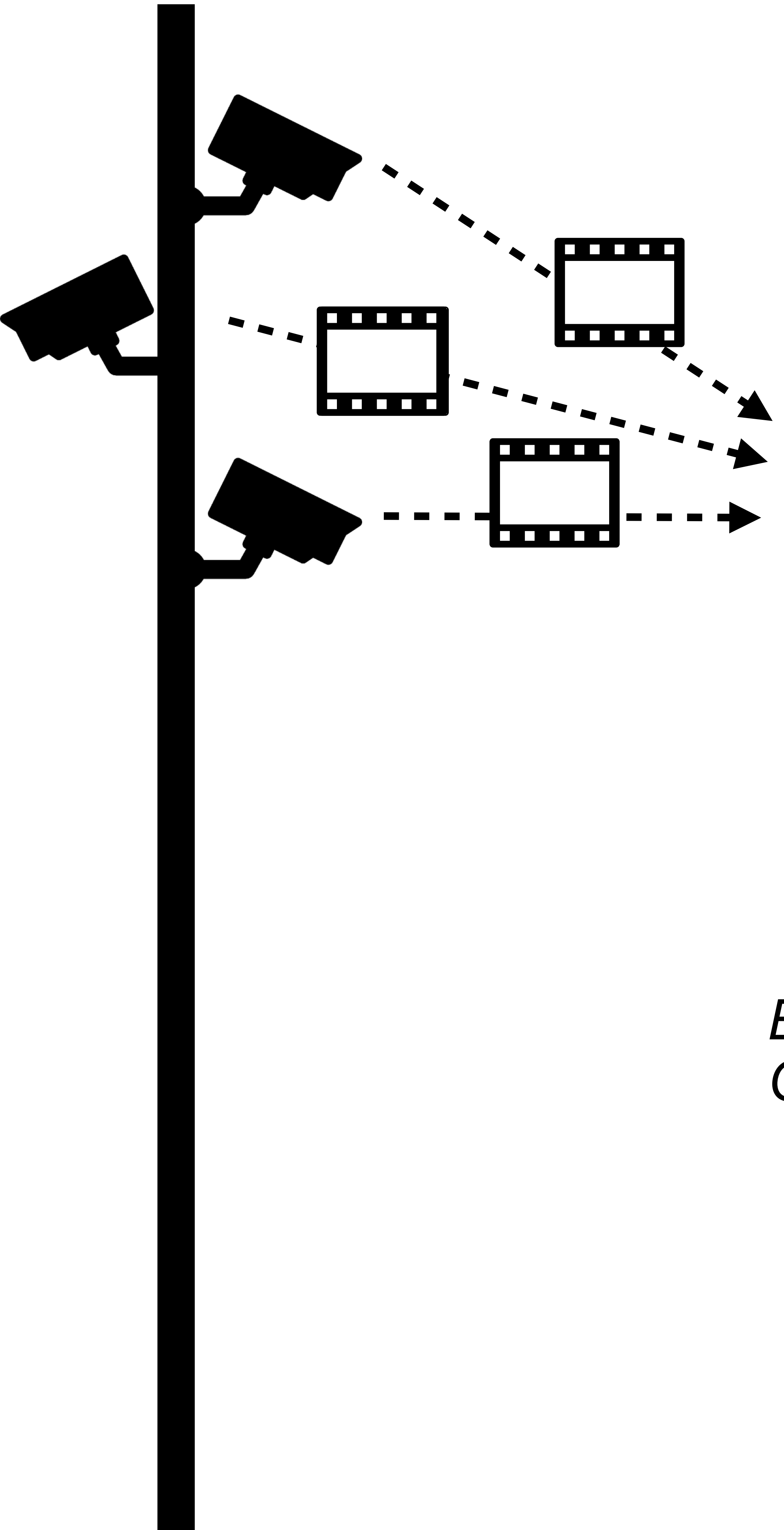
Executing Edge Workloads

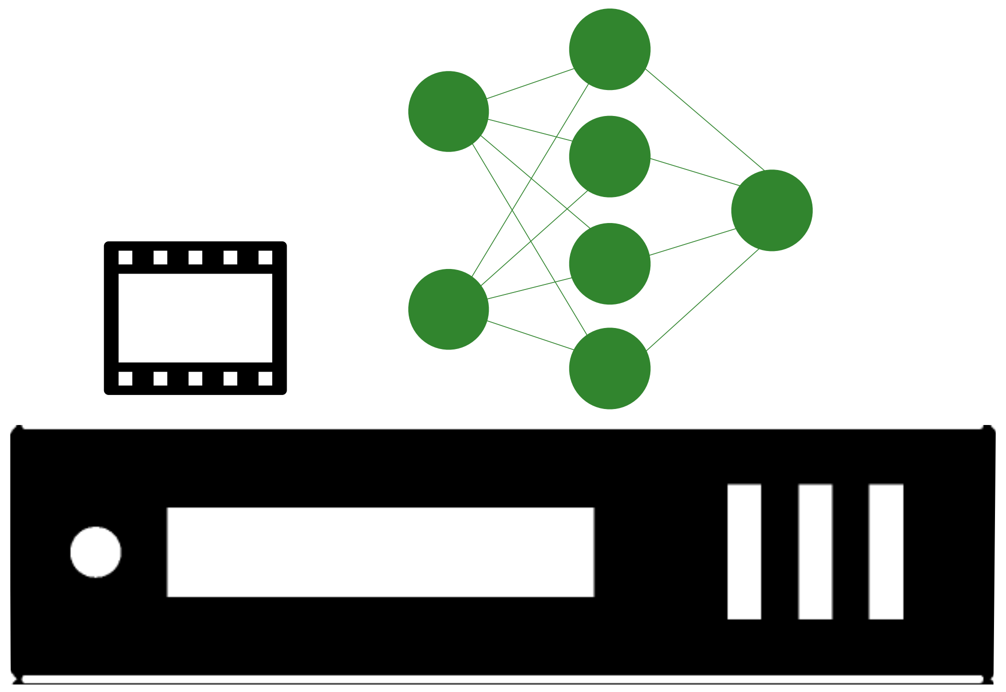


Edge Box

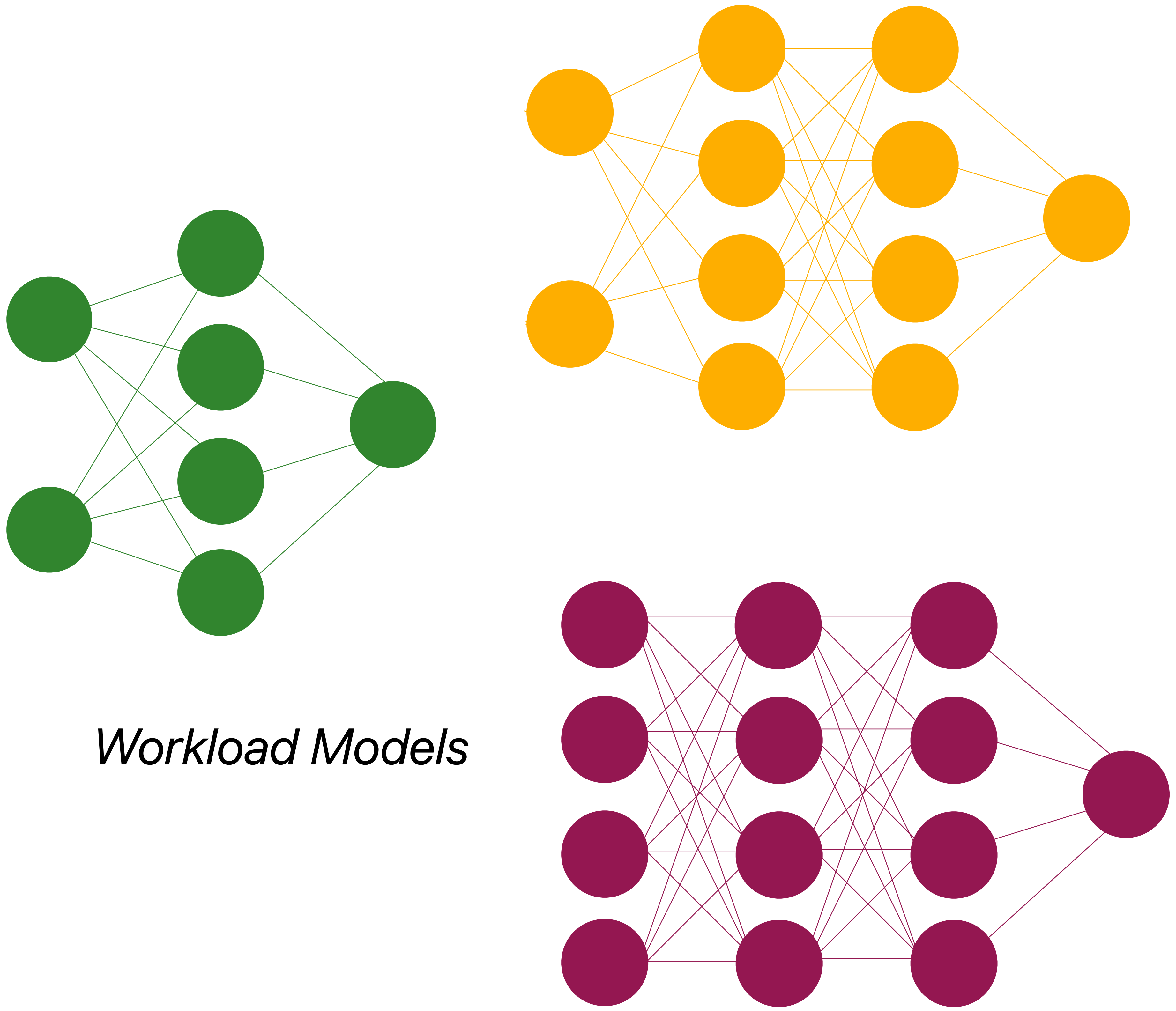


Executing Edge Workloads

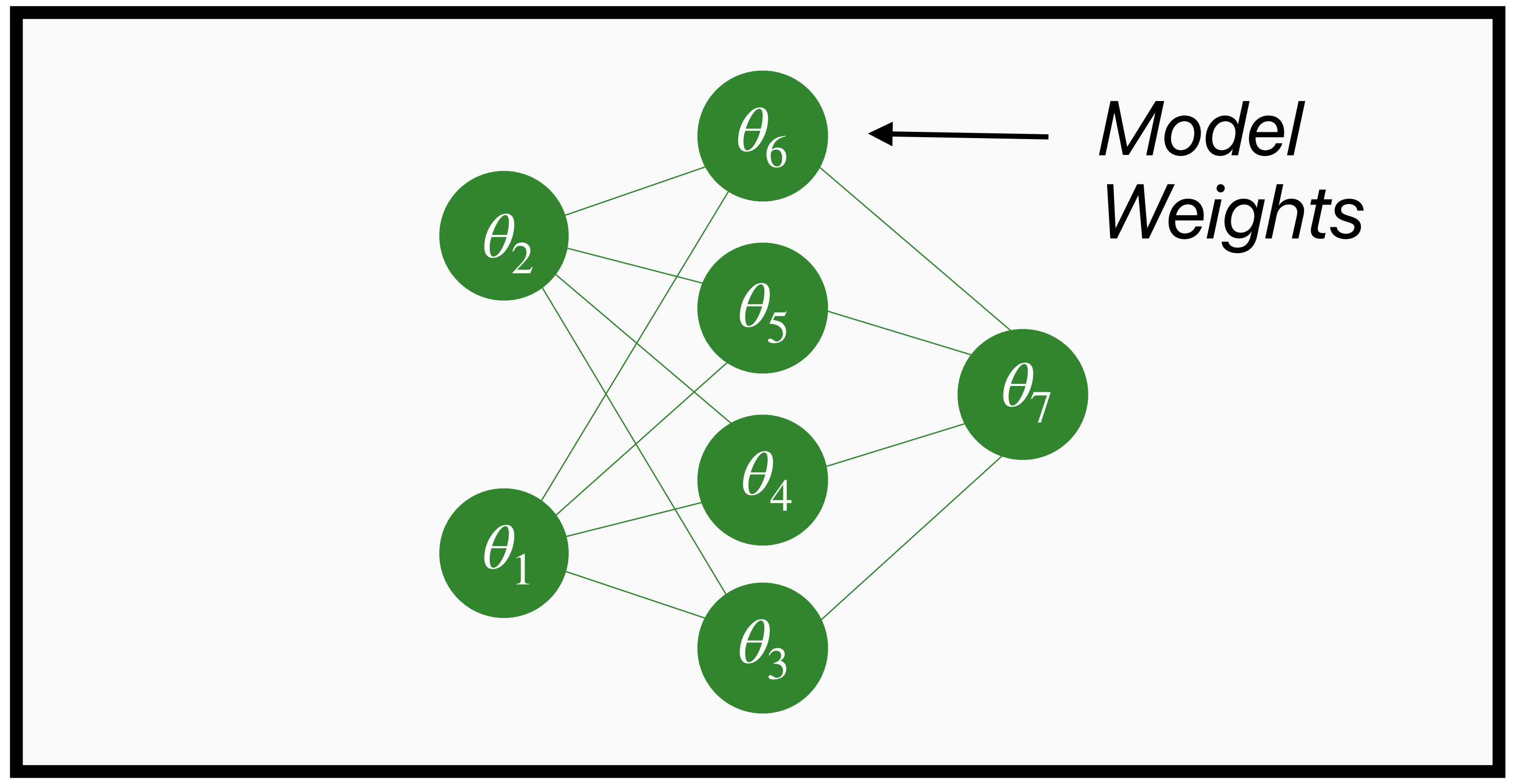




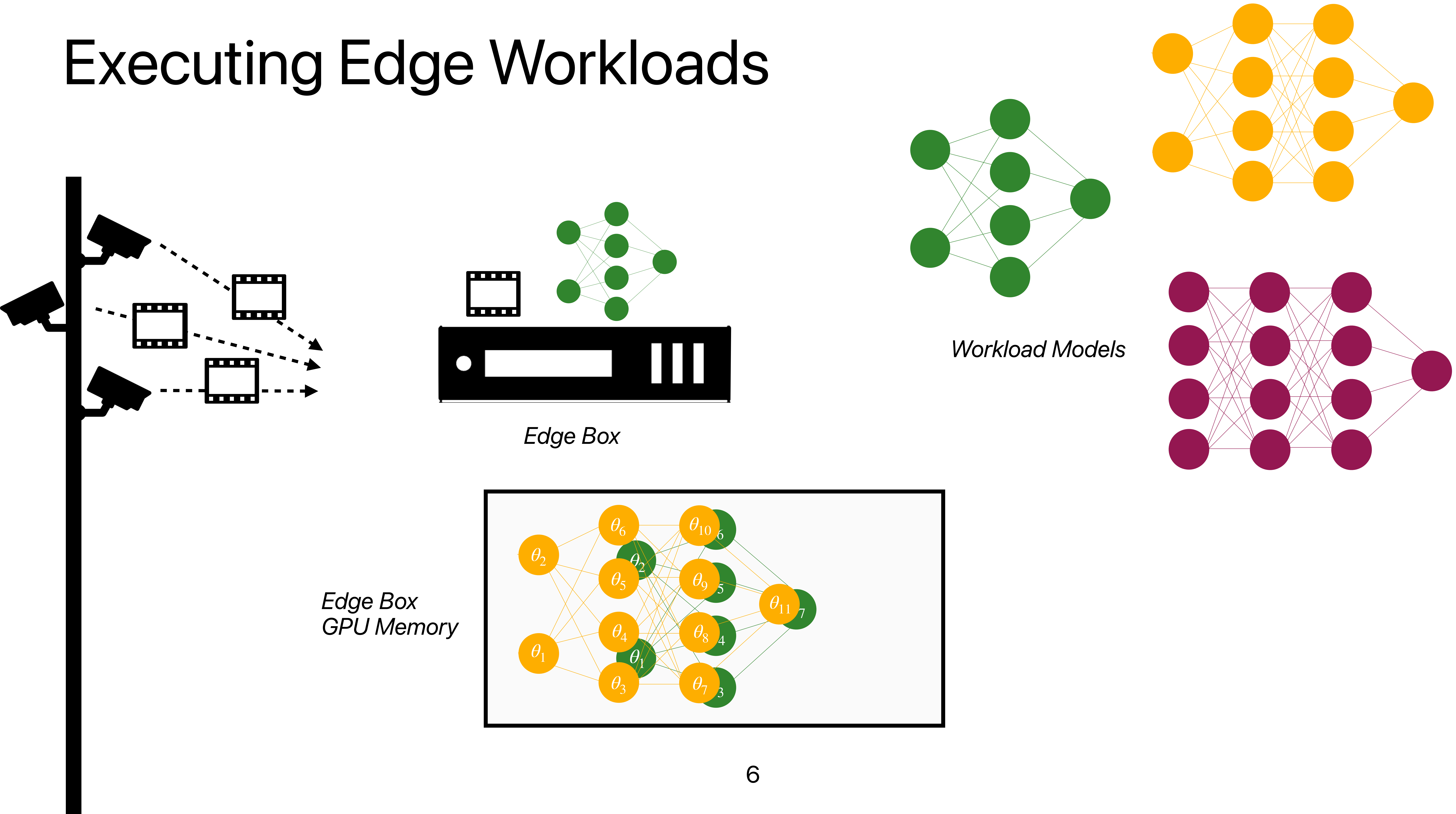
Edge Box



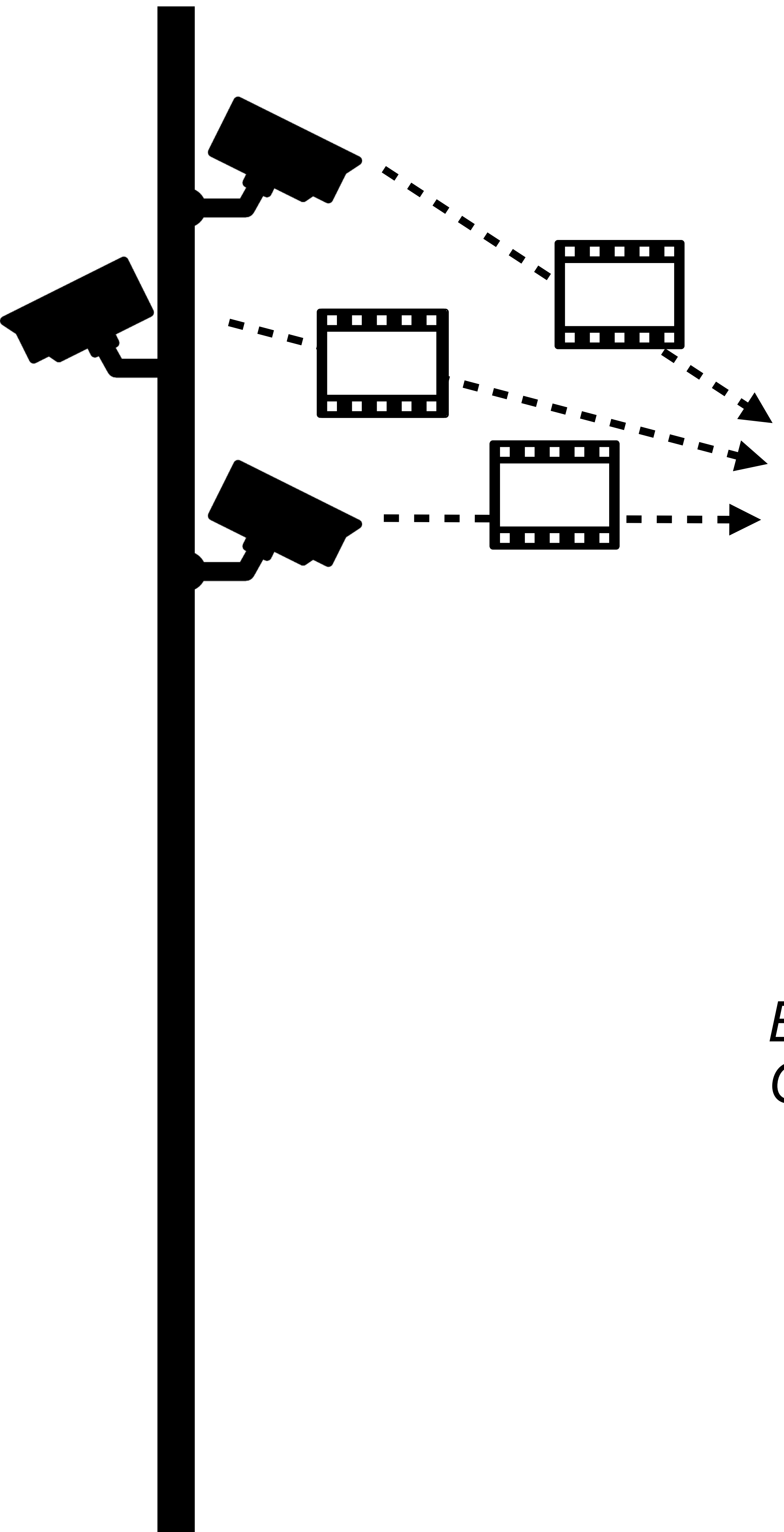
Edge Box
GPU Memory

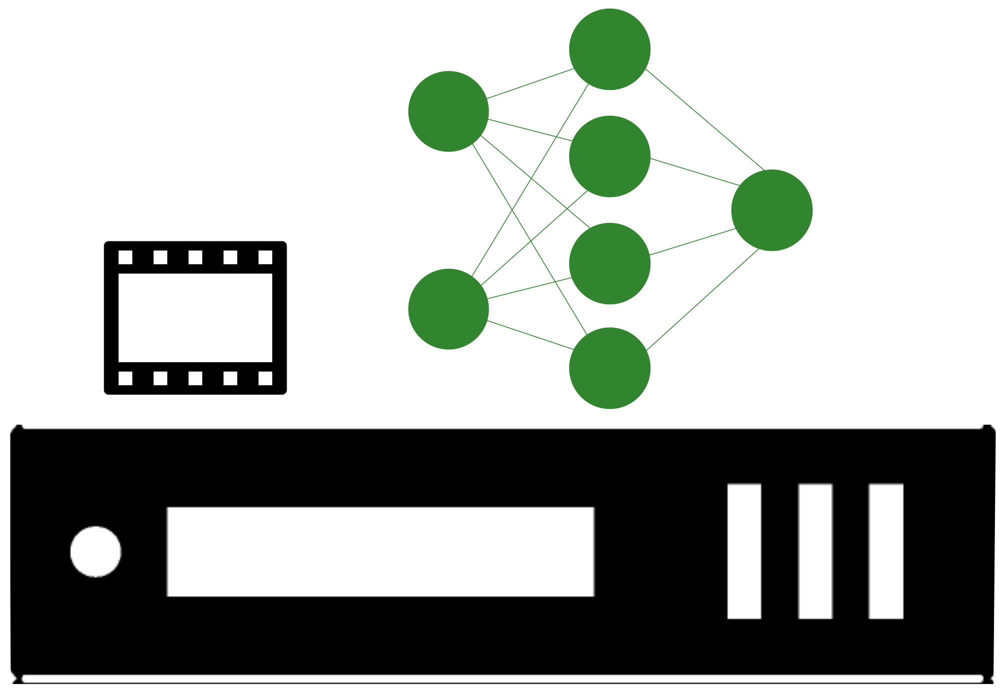


Executing Edge Workloads

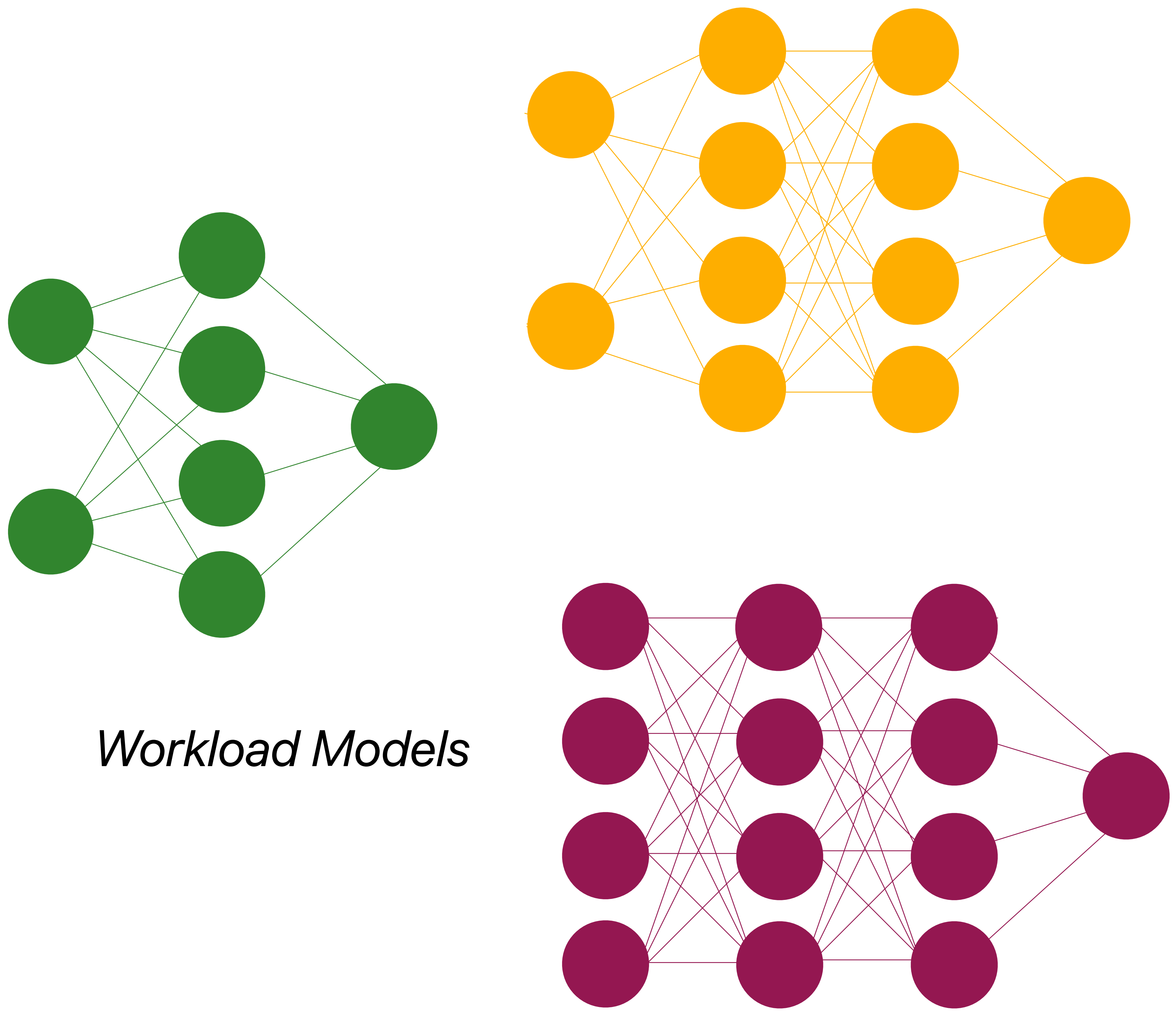


Executing Edge Workloads



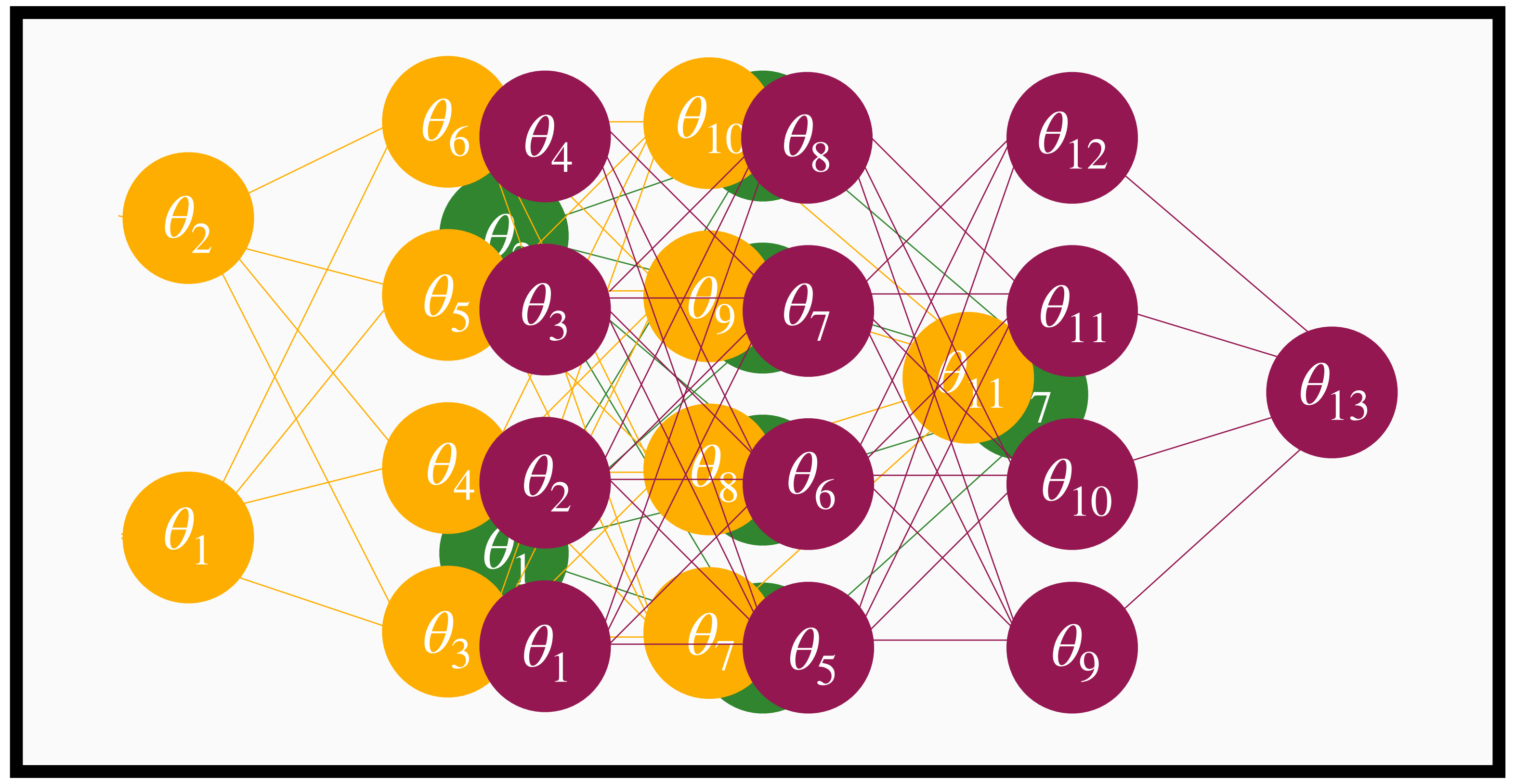


Edge Box

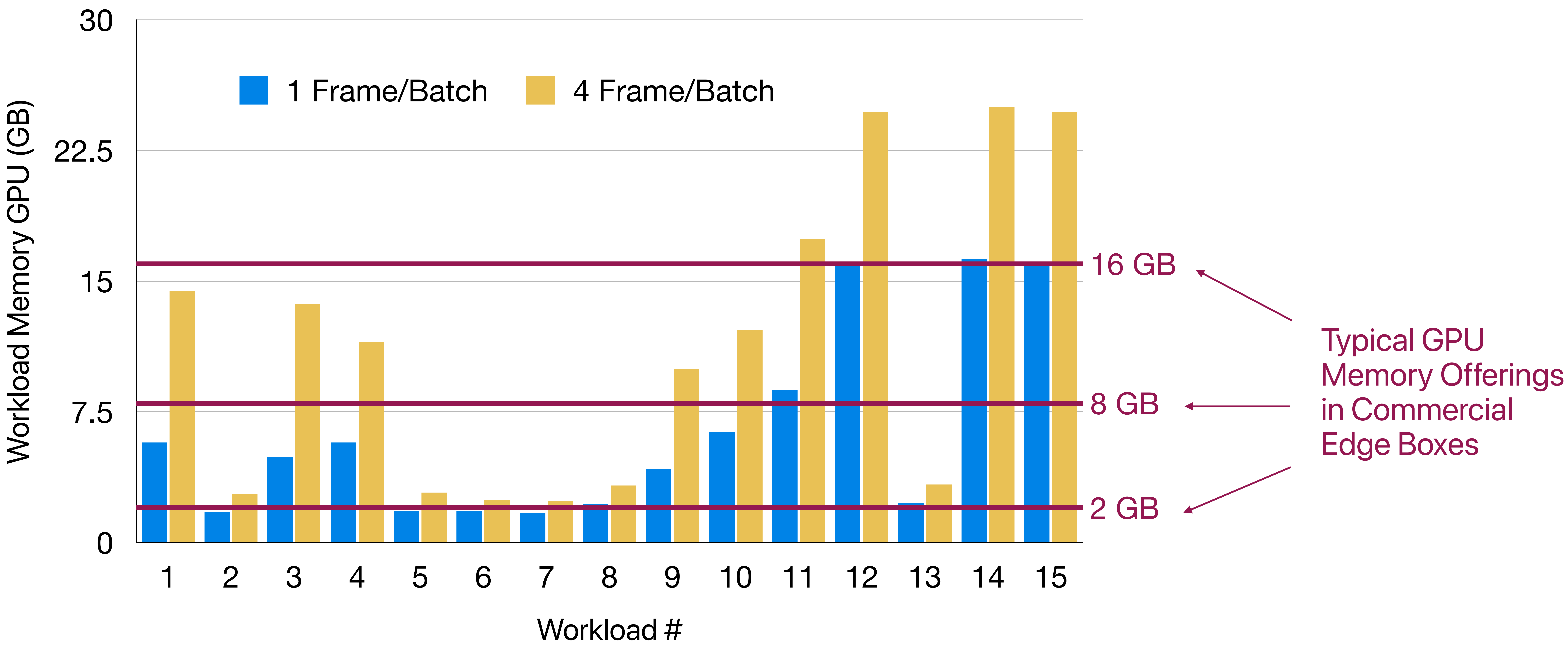


Workload Models

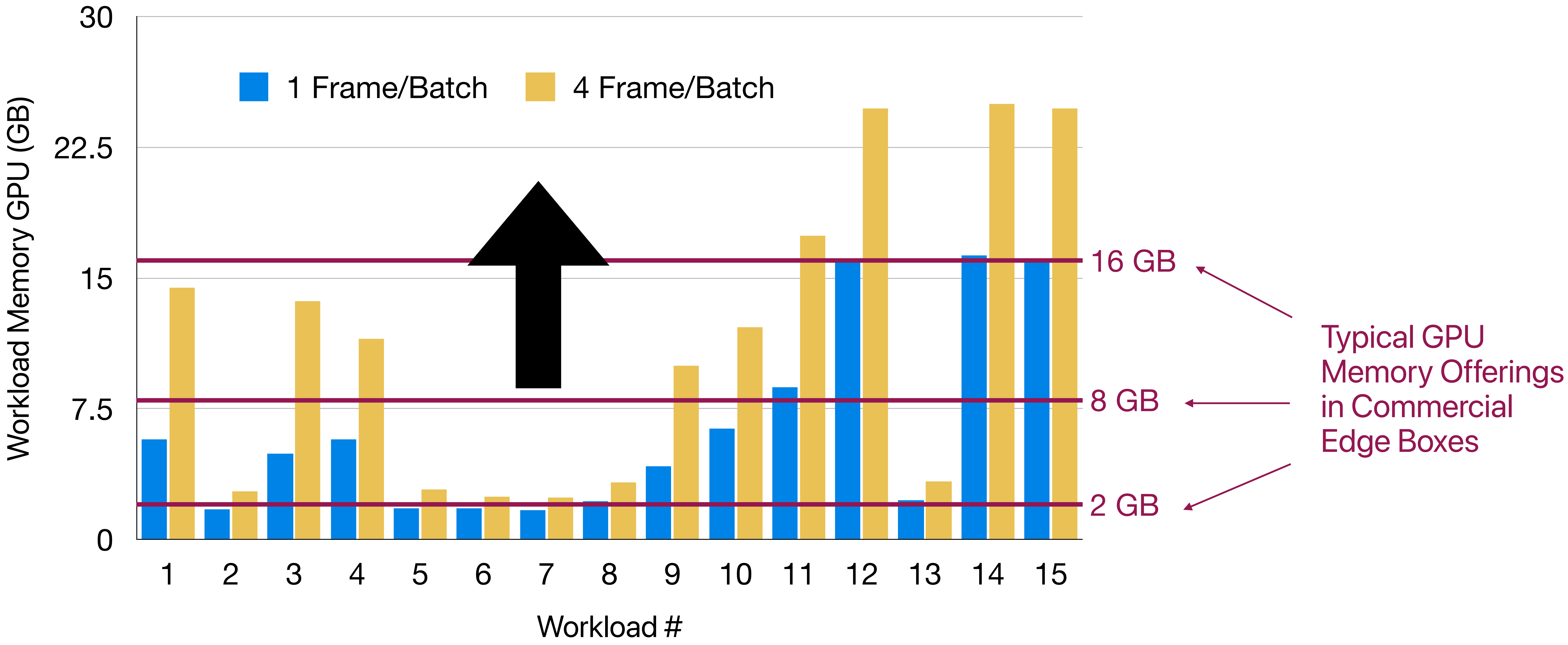
Edge Box
GPU Memory



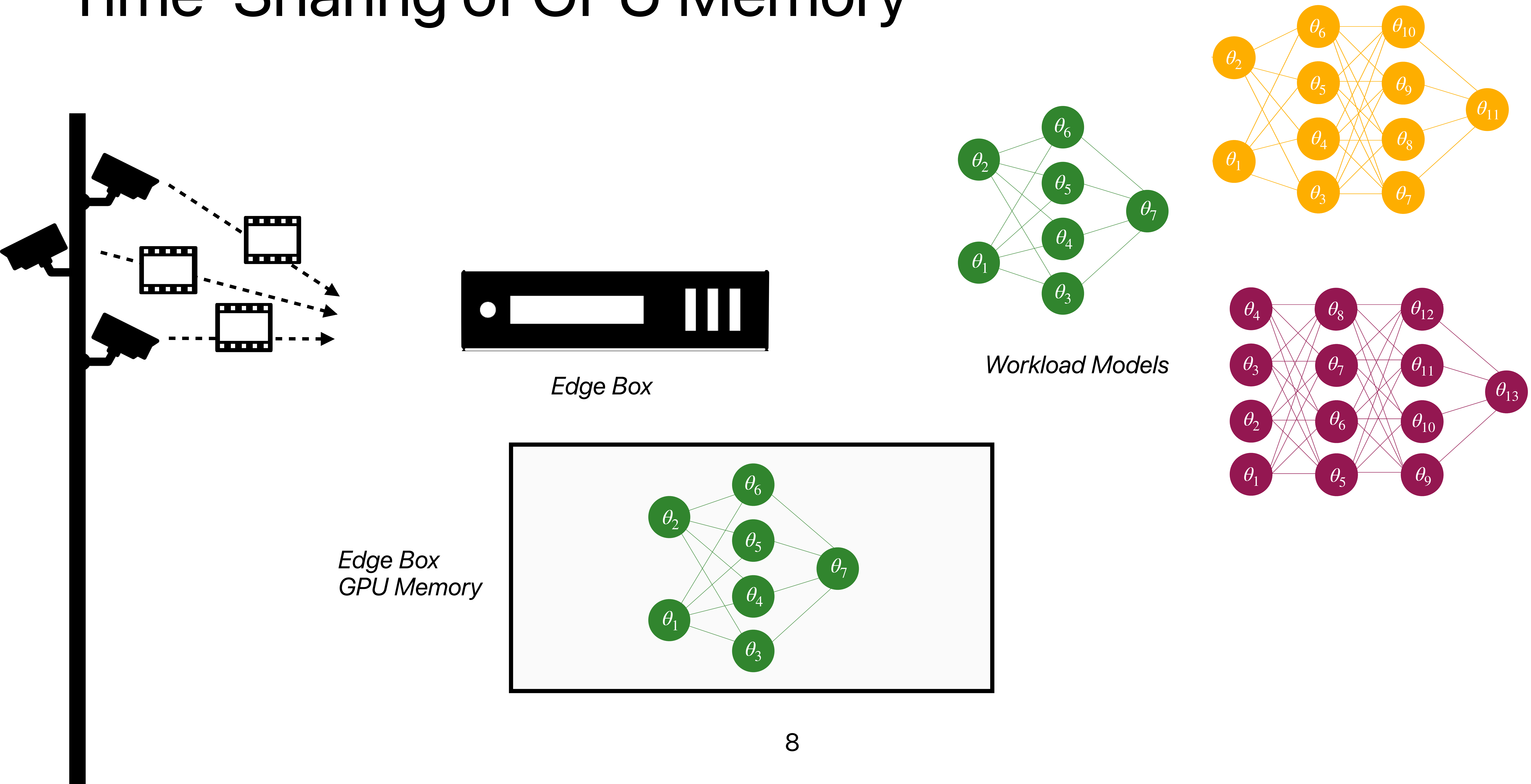
Workloads are Outgrowing Edge GPU Memory



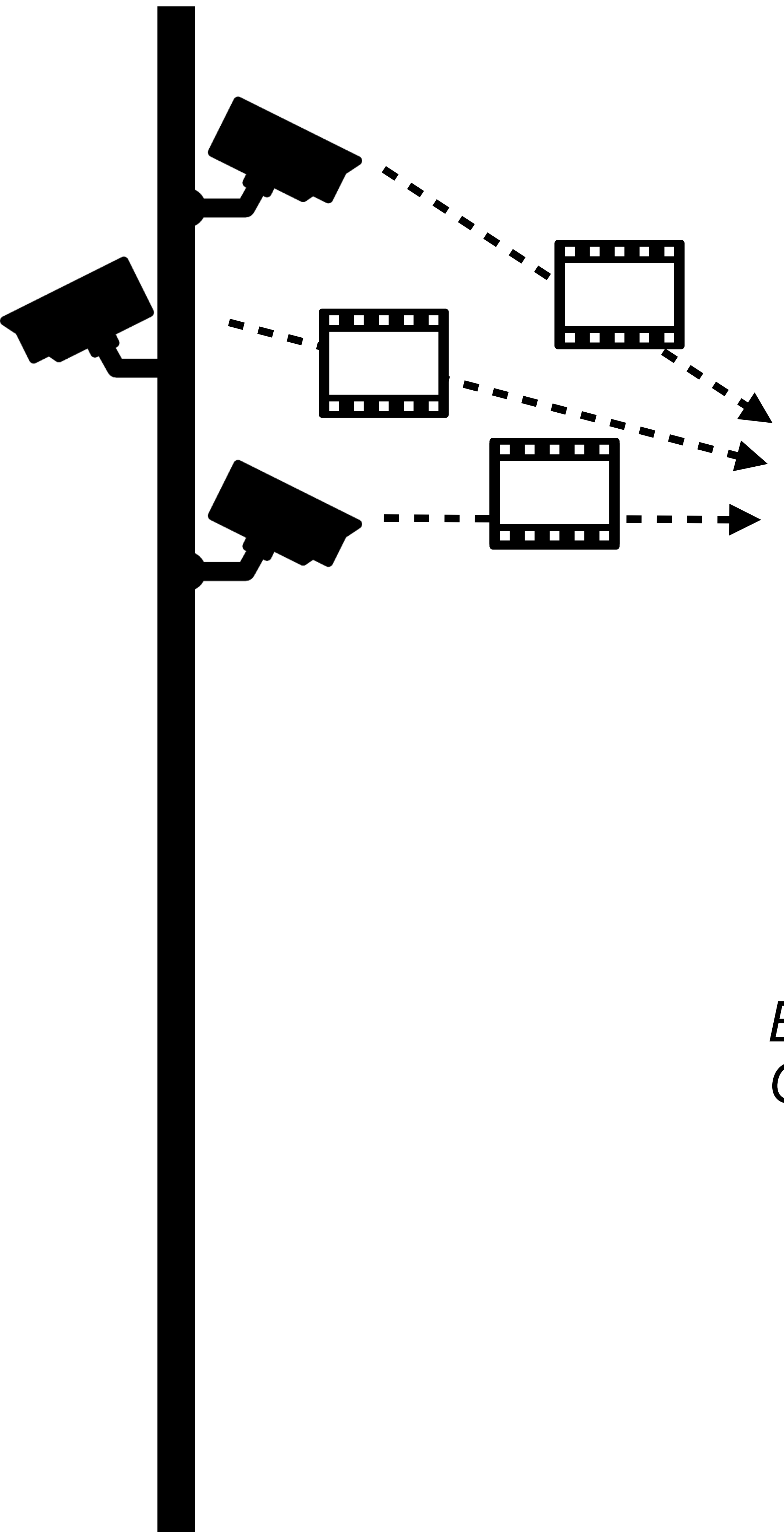
Workloads are Outgrowing Edge GPU Memory

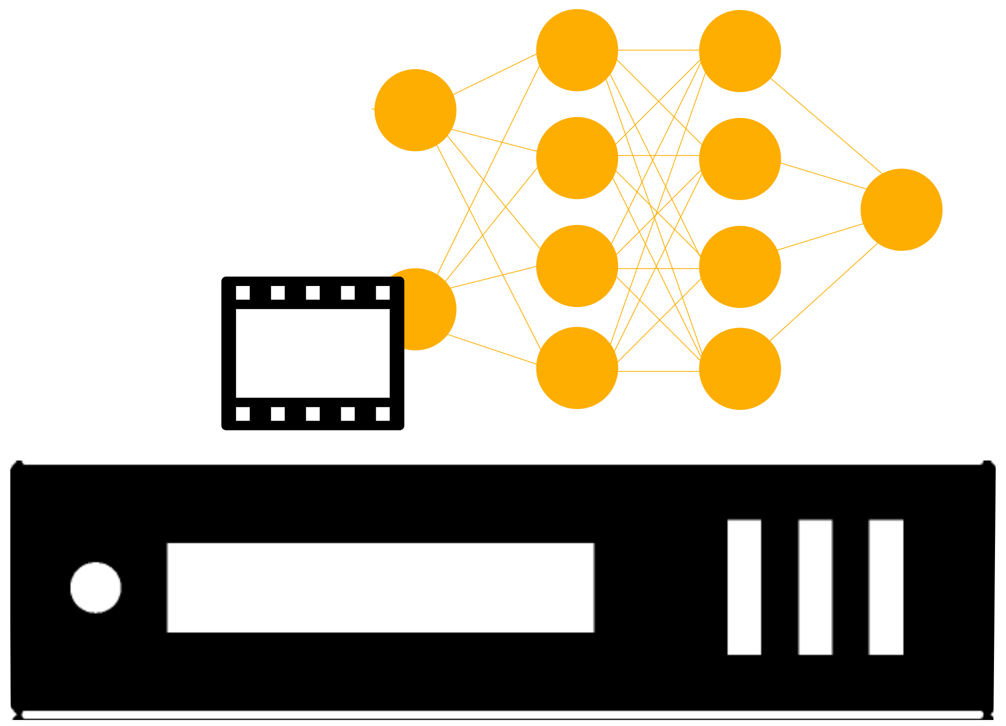


Time-Sharing of GPU Memory

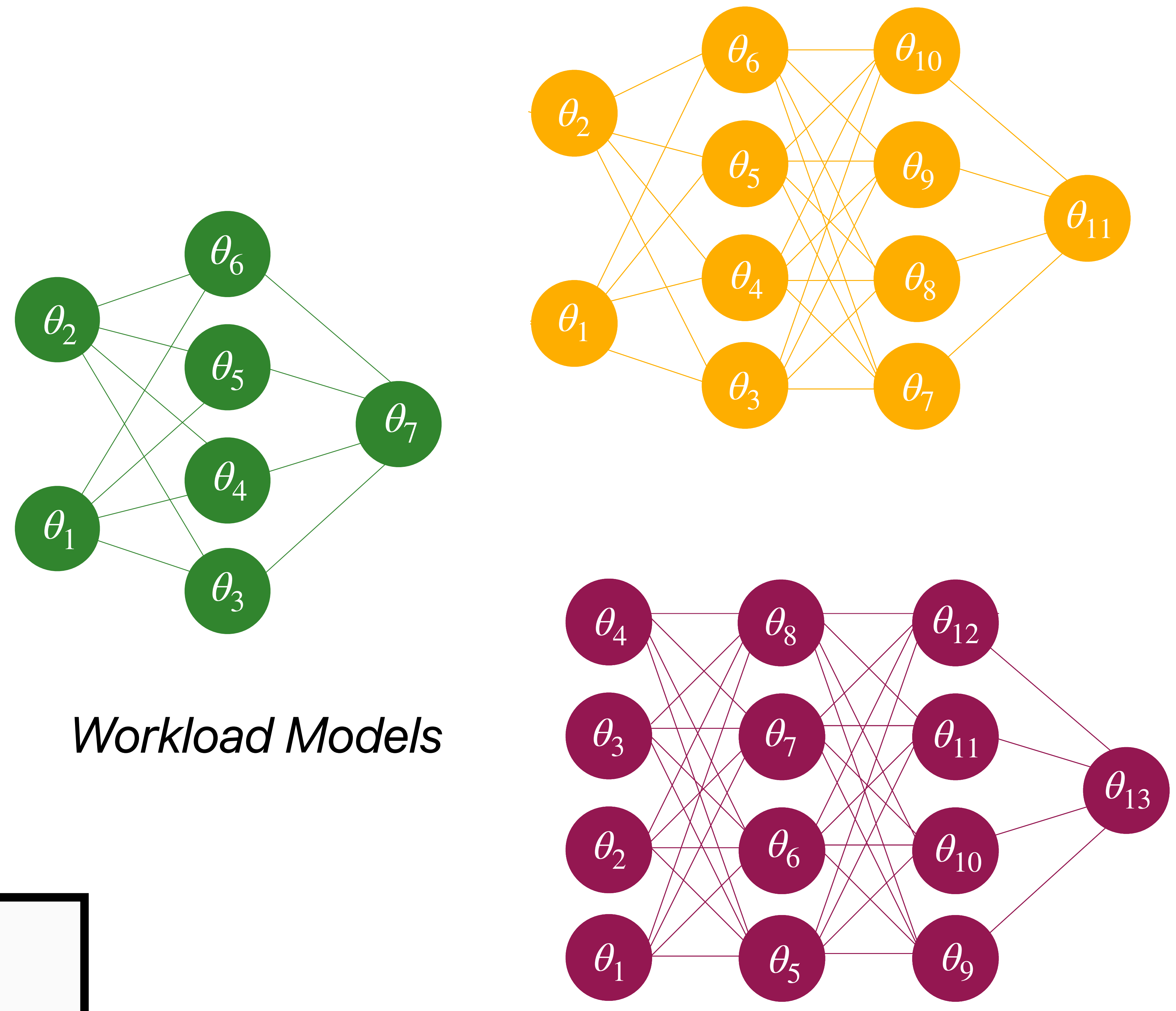


Time-Sharing of GPU Memory



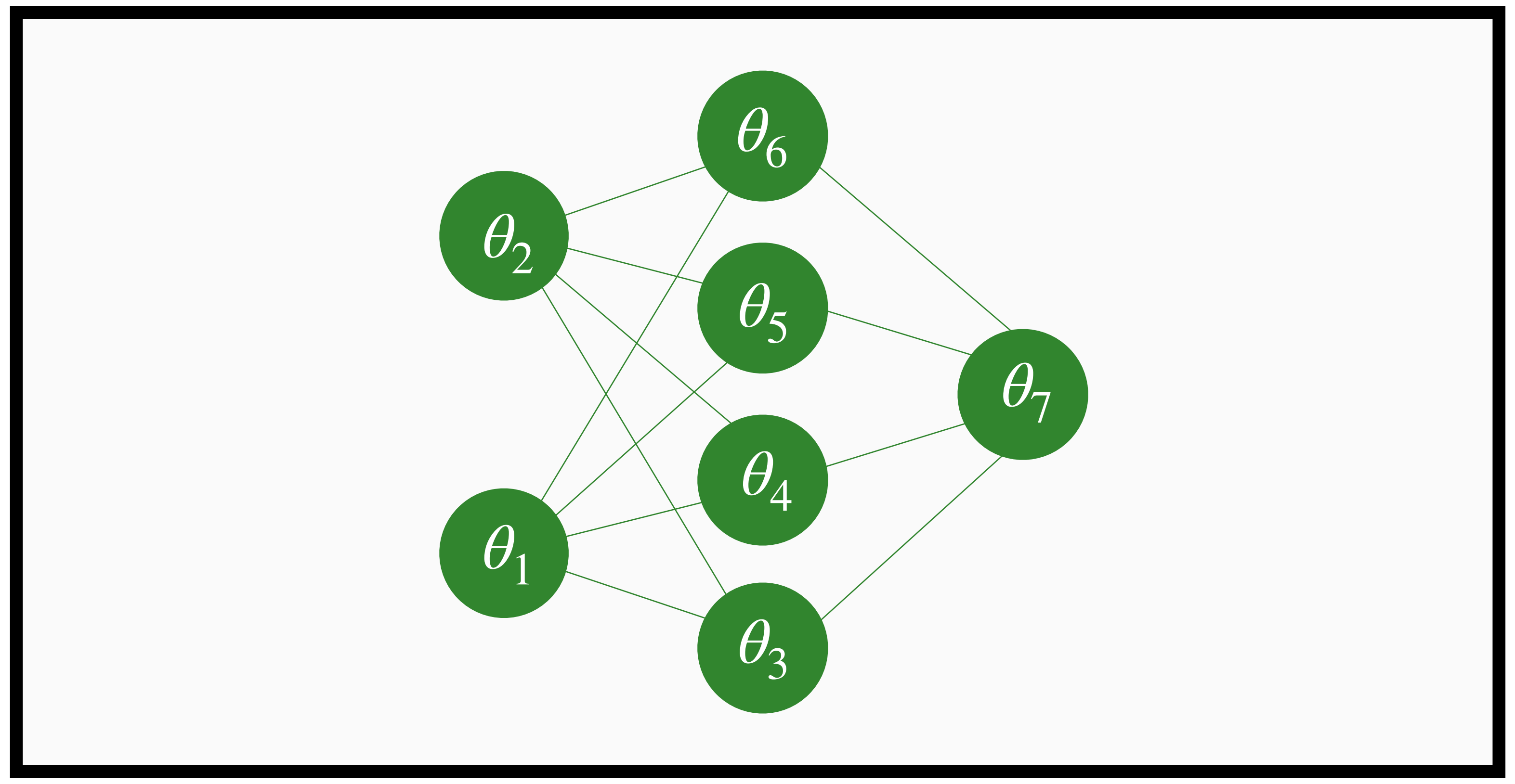


Edge Box

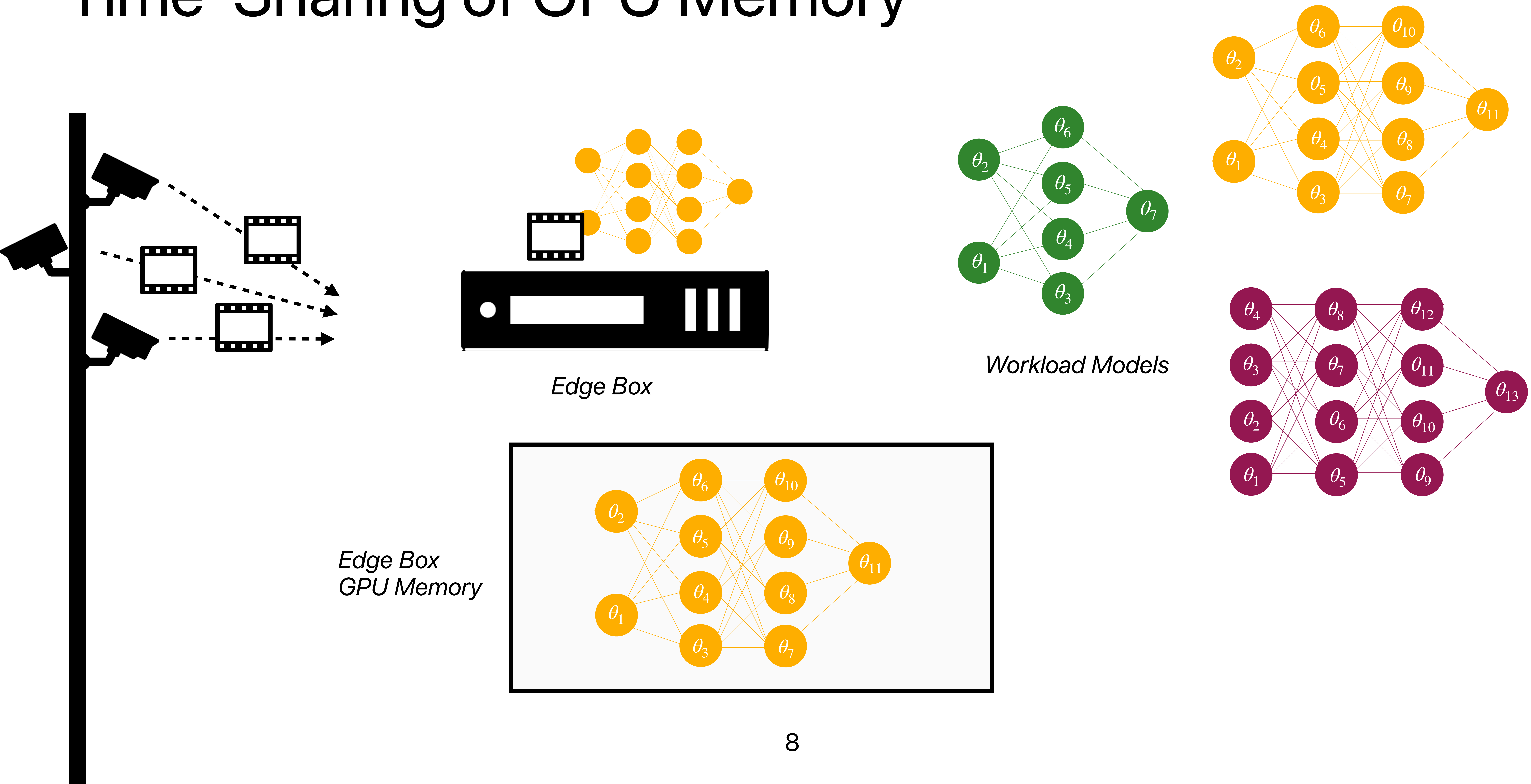


Workload Models

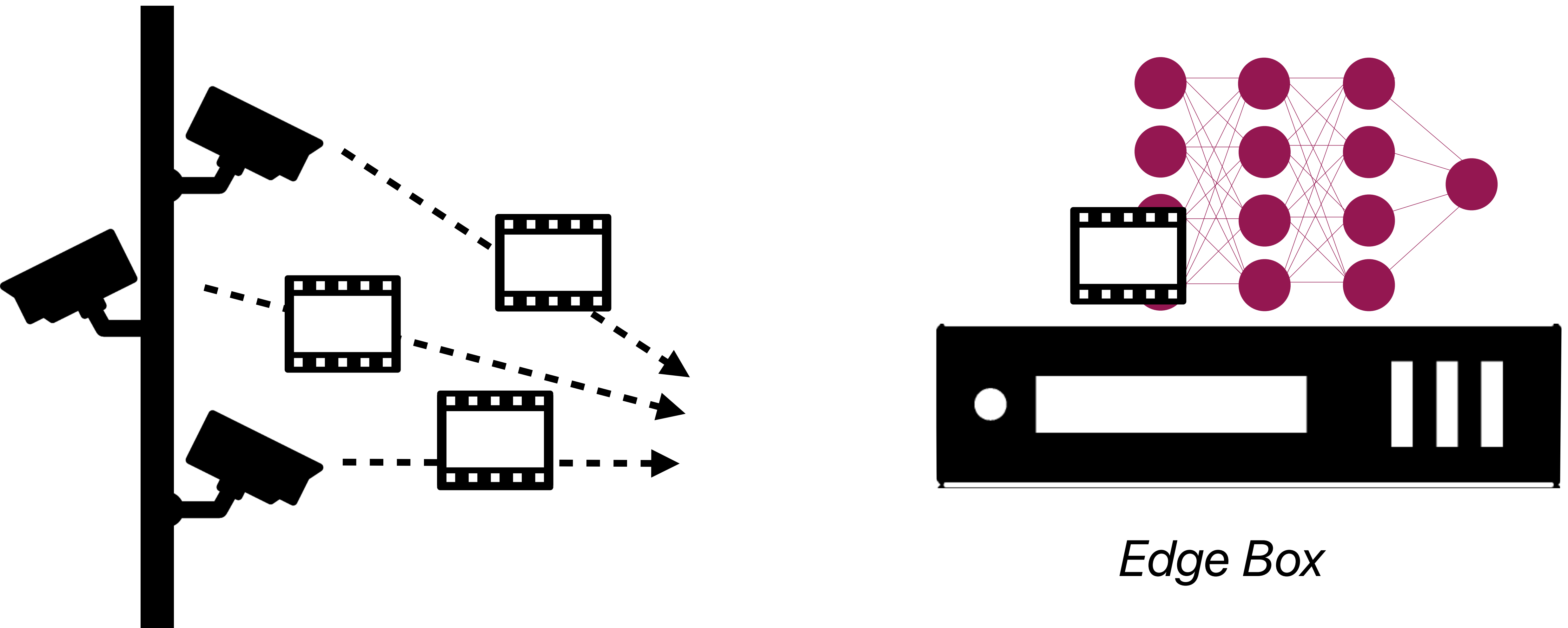
Edge Box
GPU Memory

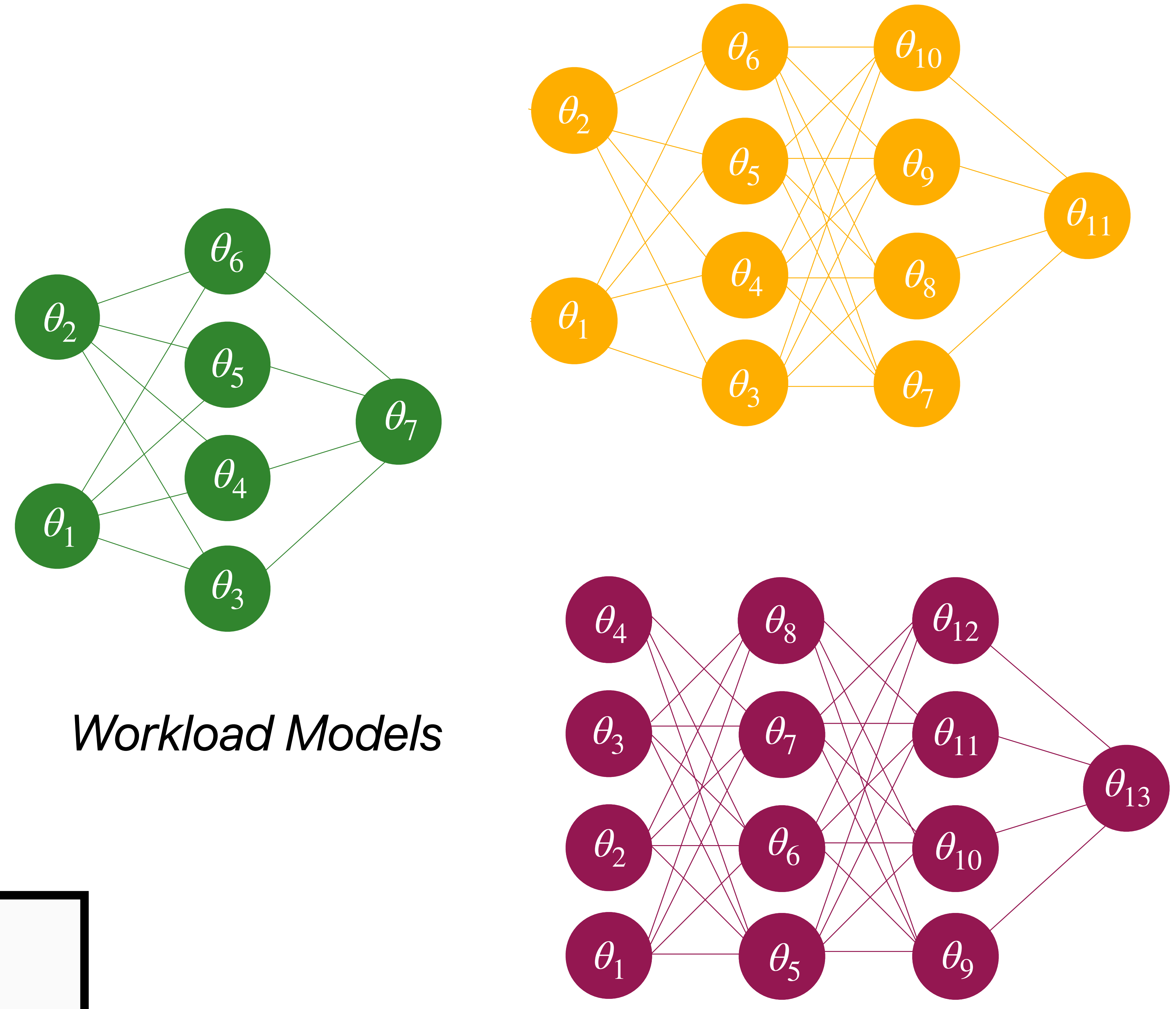


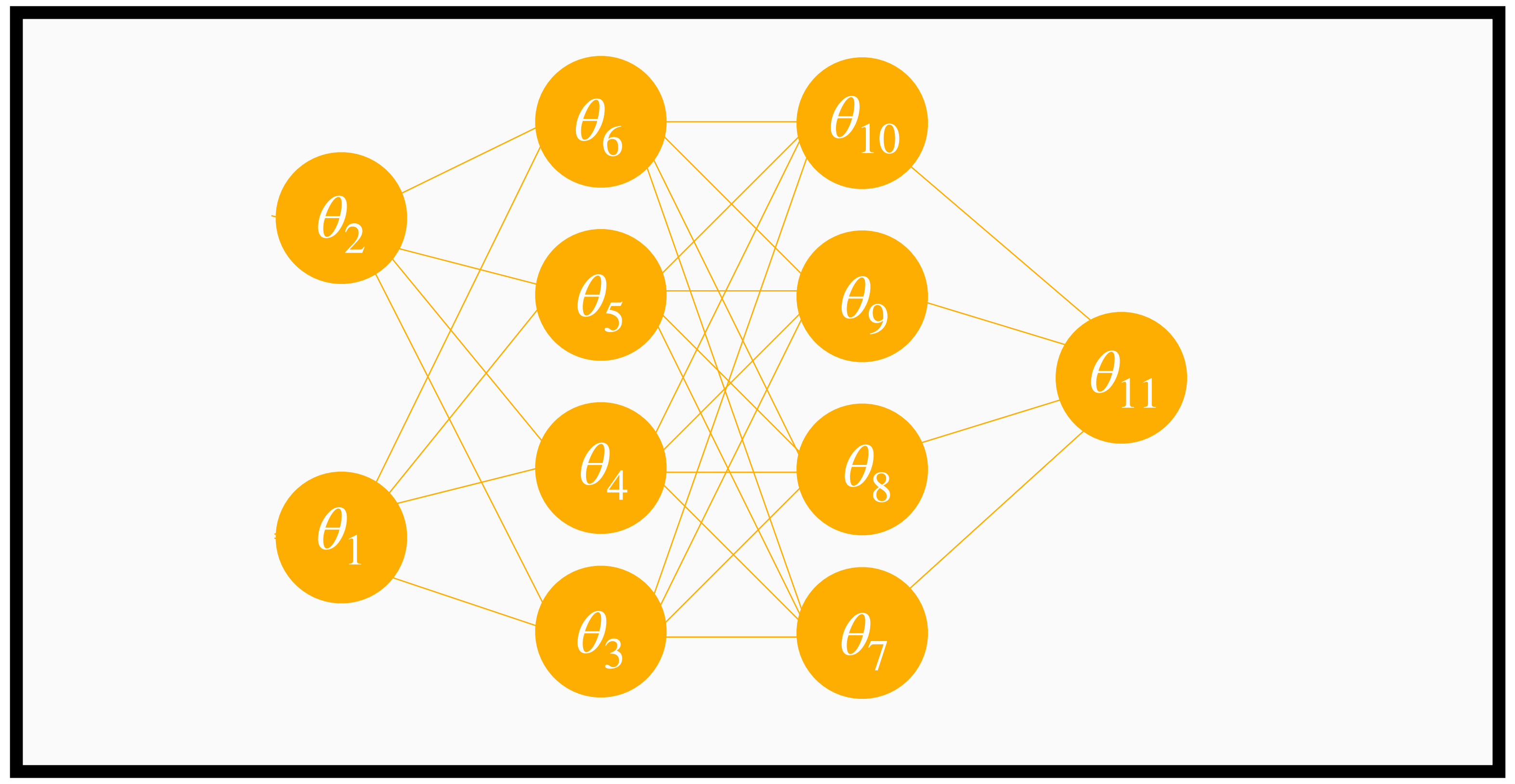
Time-Sharing of GPU Memory



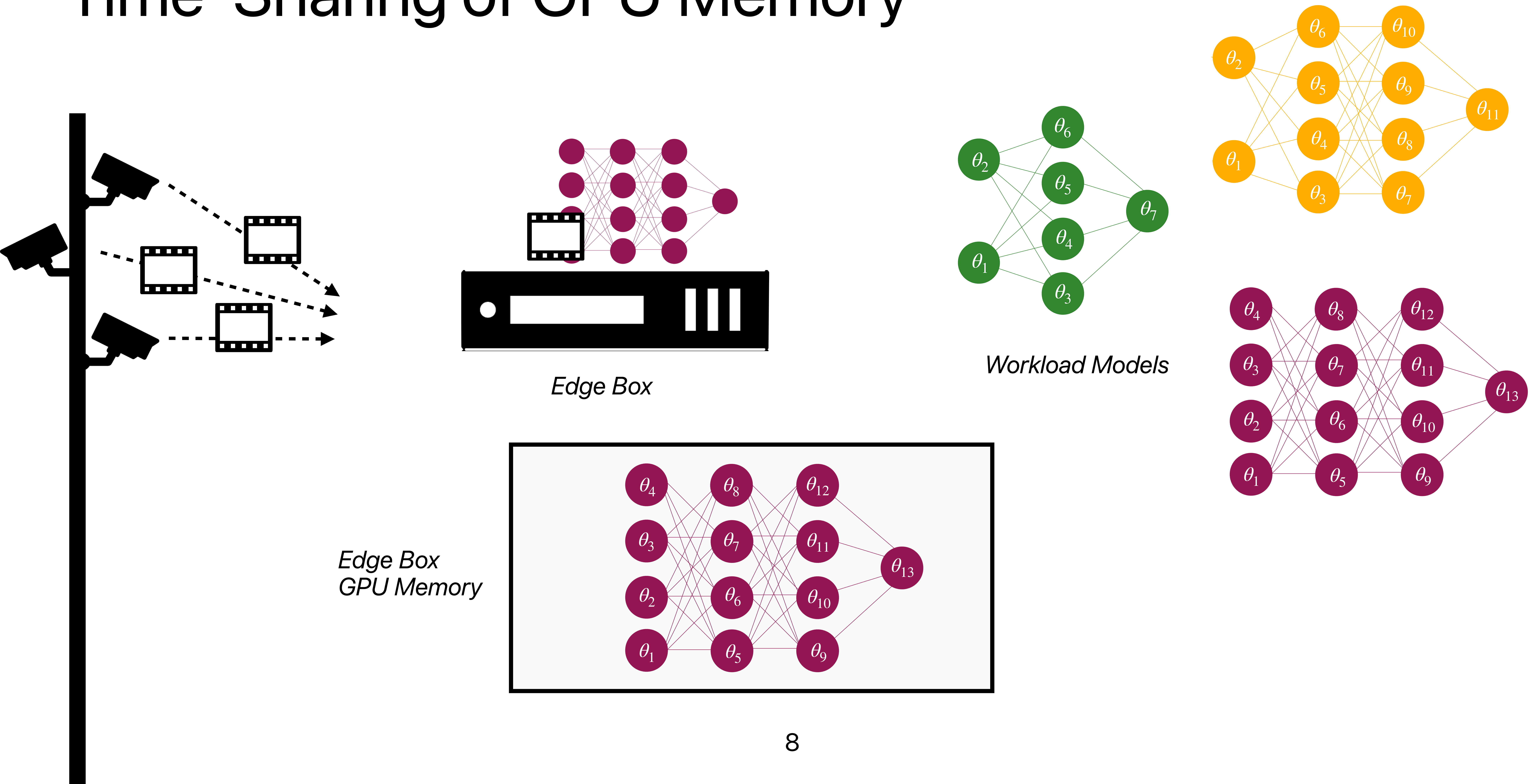
Time-Sharing of GPU Memory



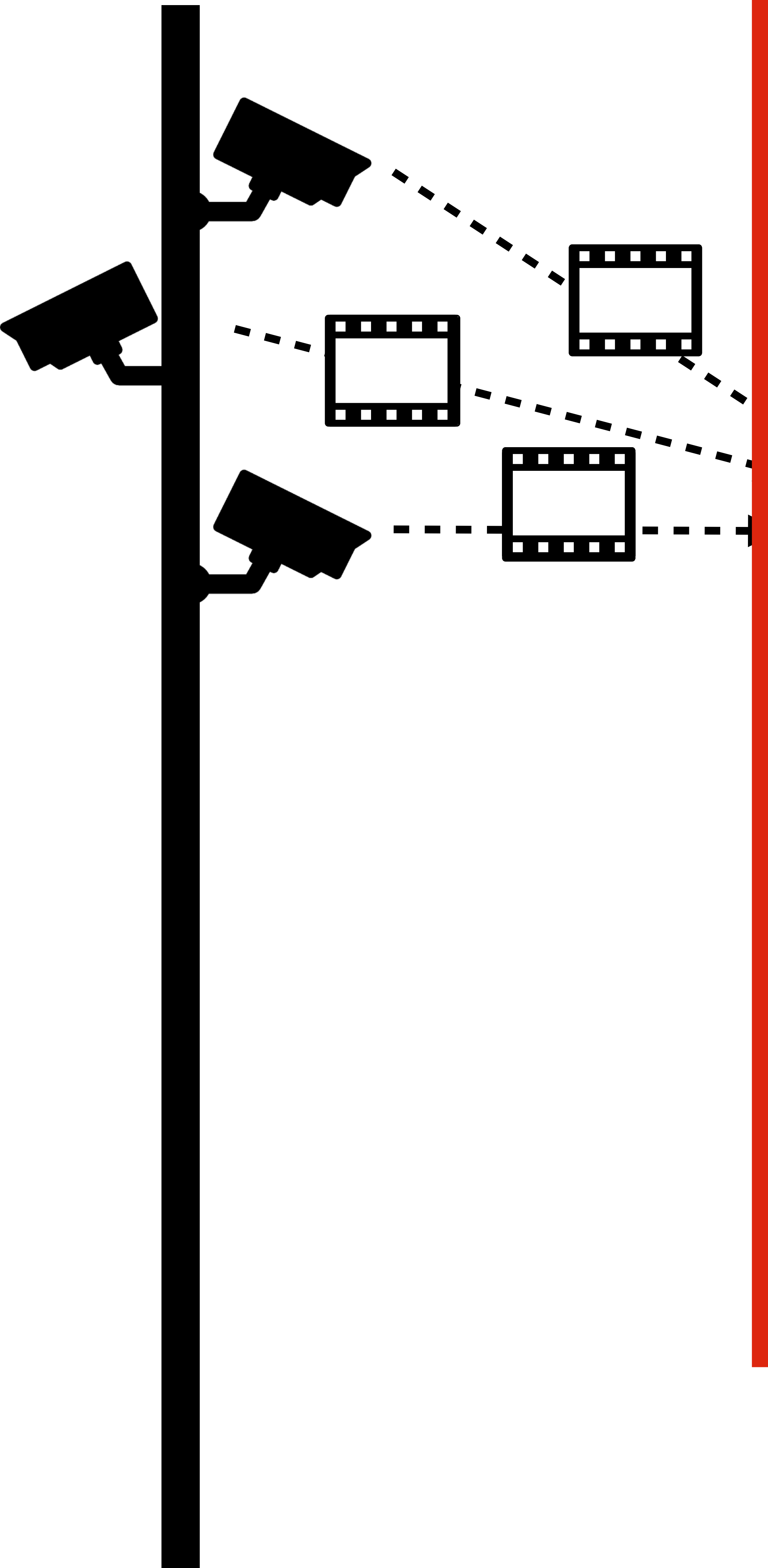




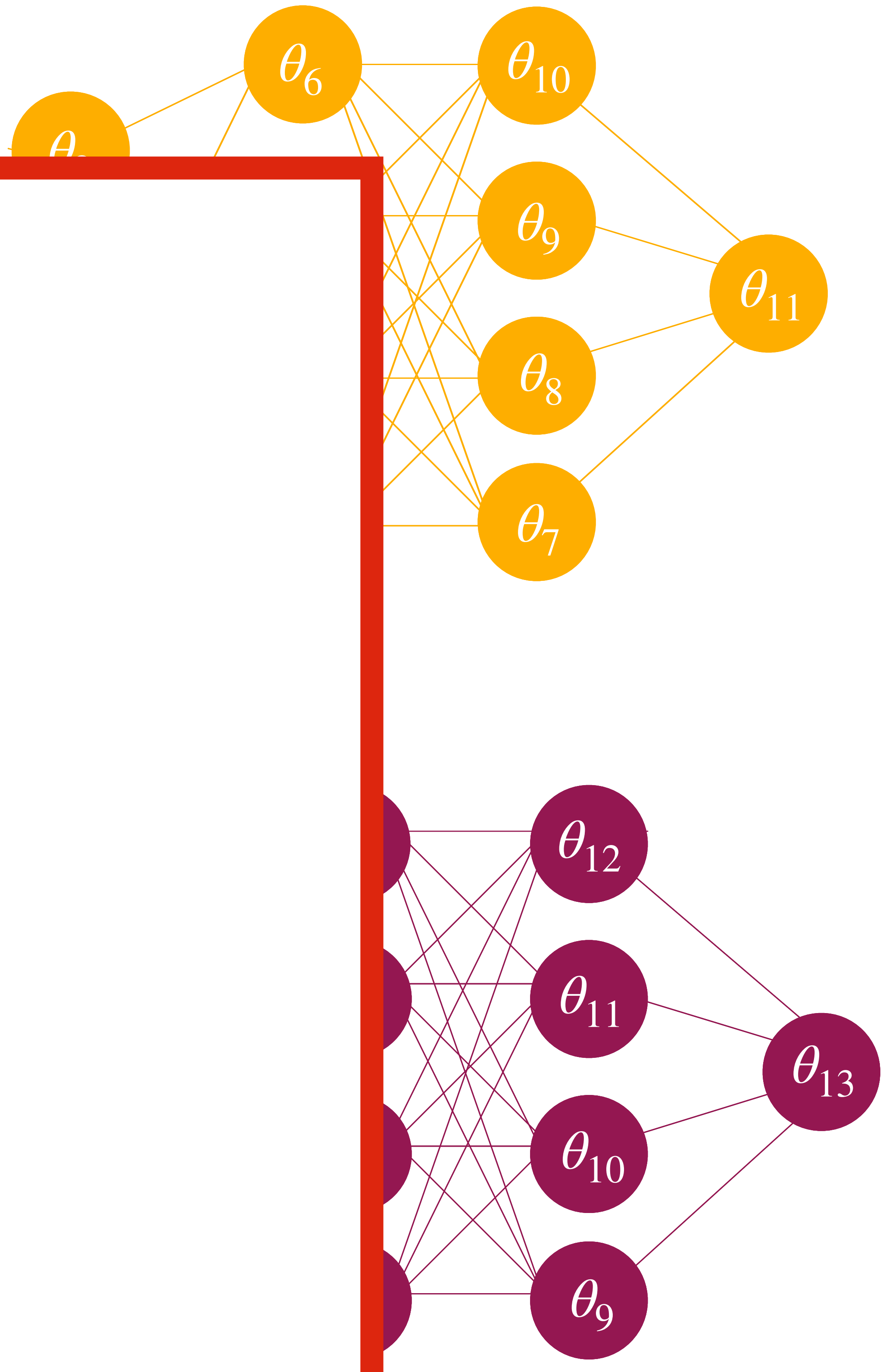
Time-Sharing of GPU Memory



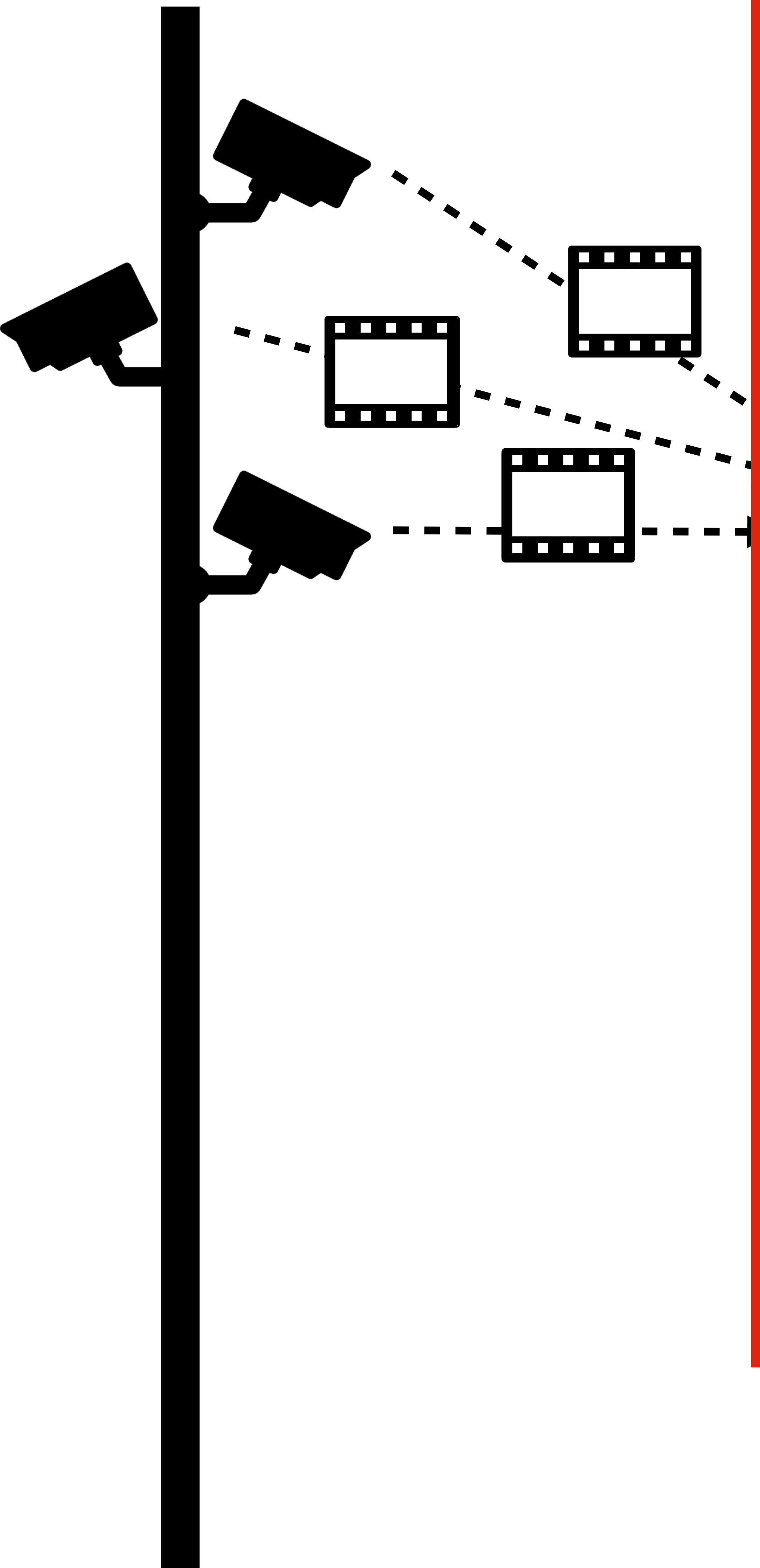
Time-Sharing of GPU Memory

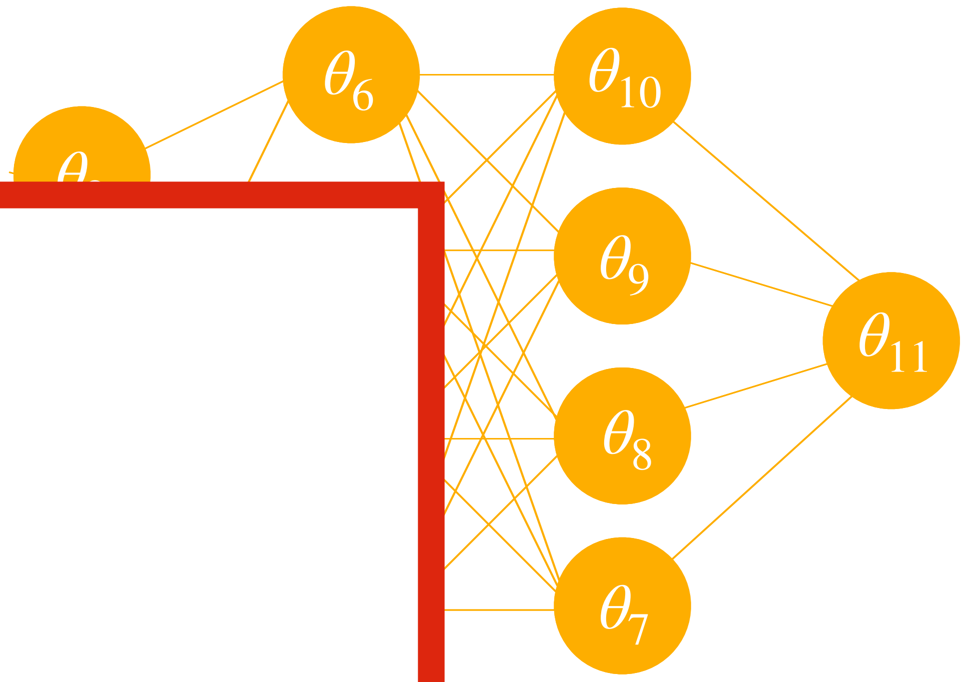


Skipped processing of 19-84% of frames and accuracy drops up to 43%



Time-Sharing of GPU Memory

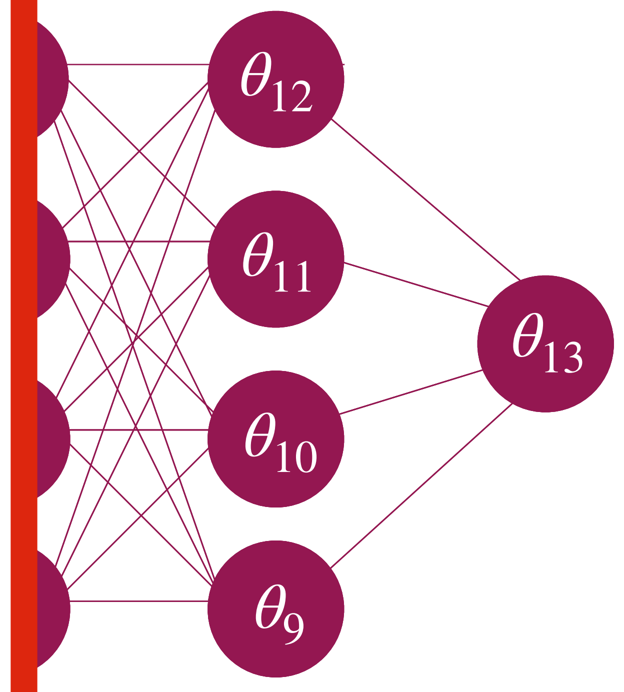




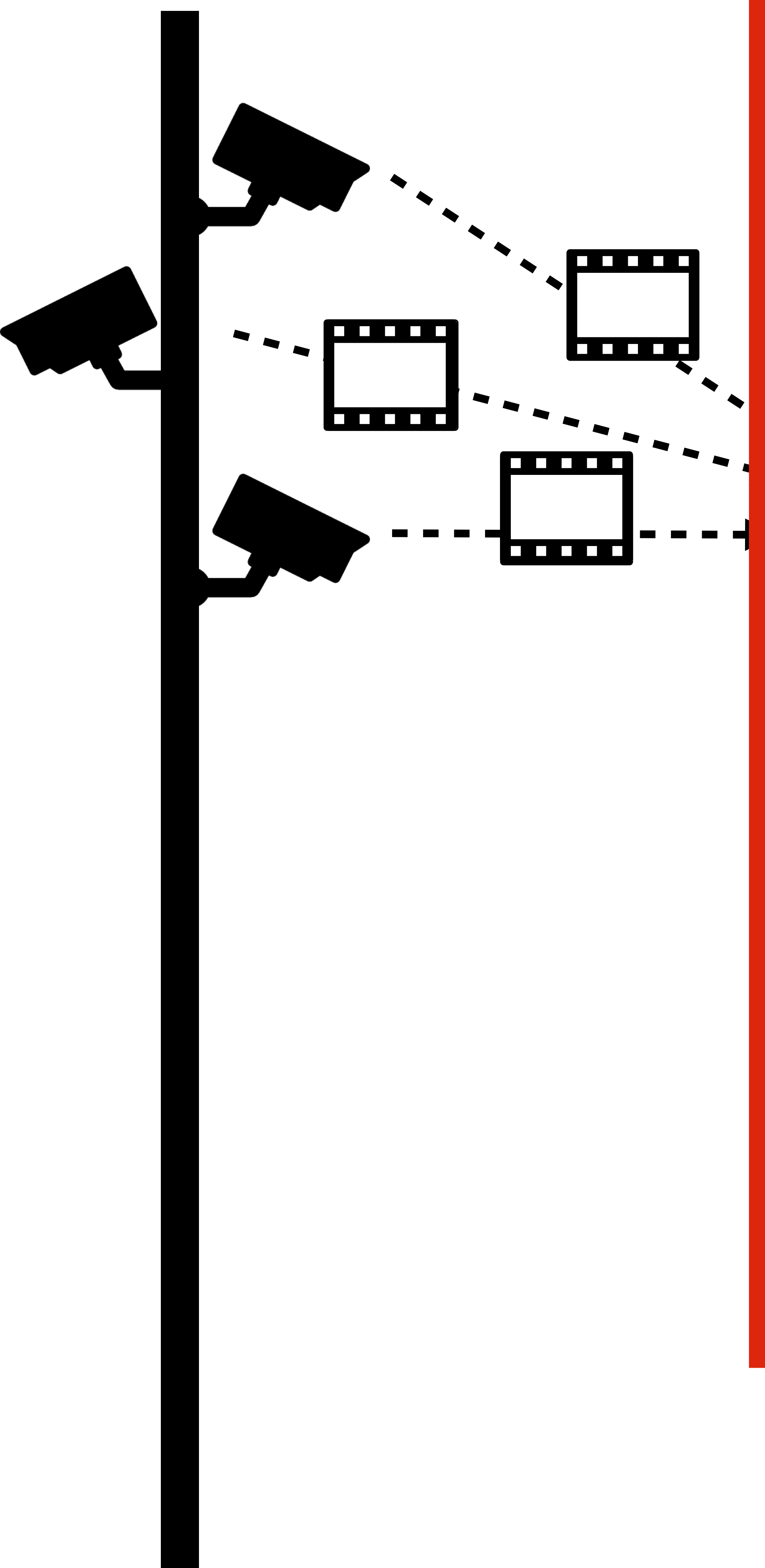
Skipped processing of 19-84% of frames and accuracy drops up to 43%

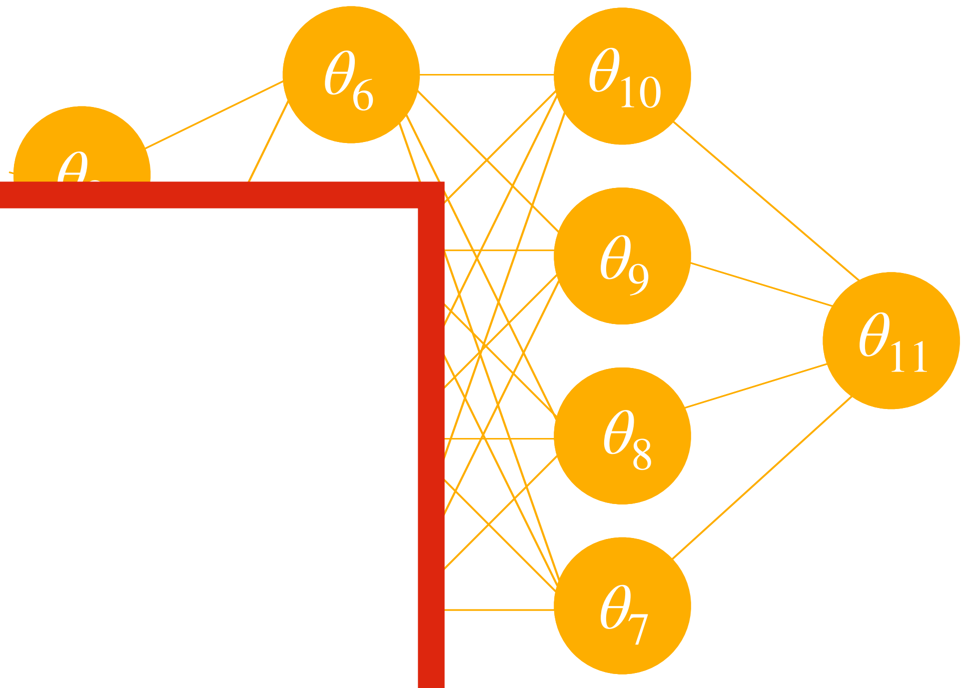
Model	**Loading Time** (ms)	**Run Time** (ms)
YOLOv3	49.5	17.0
ResNet152	73.3	24.8
ResNet50	27.1	8.4
VGG16	72.2	2.1
Tiny YOLOv3	6.7	3.0

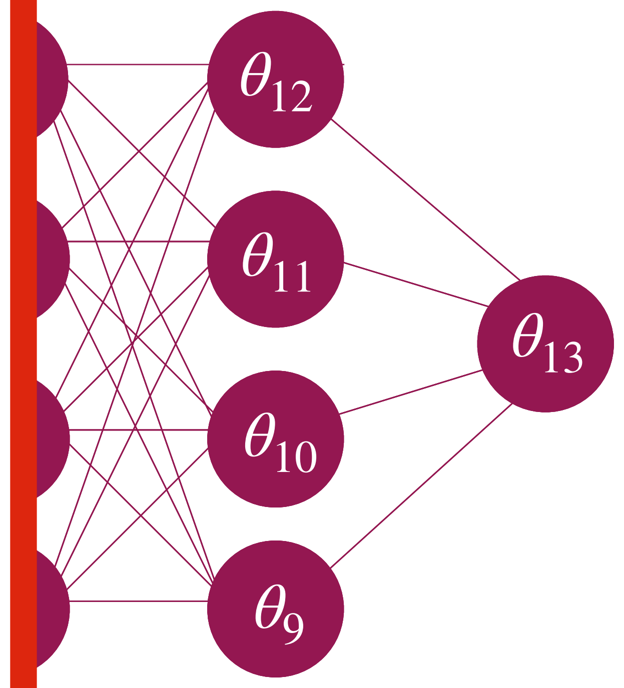
Repeatedly loading models into GPU memory is *slow*



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?

Gemel

How to reduce GPU memory bottlenecks in edge video analytics?

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

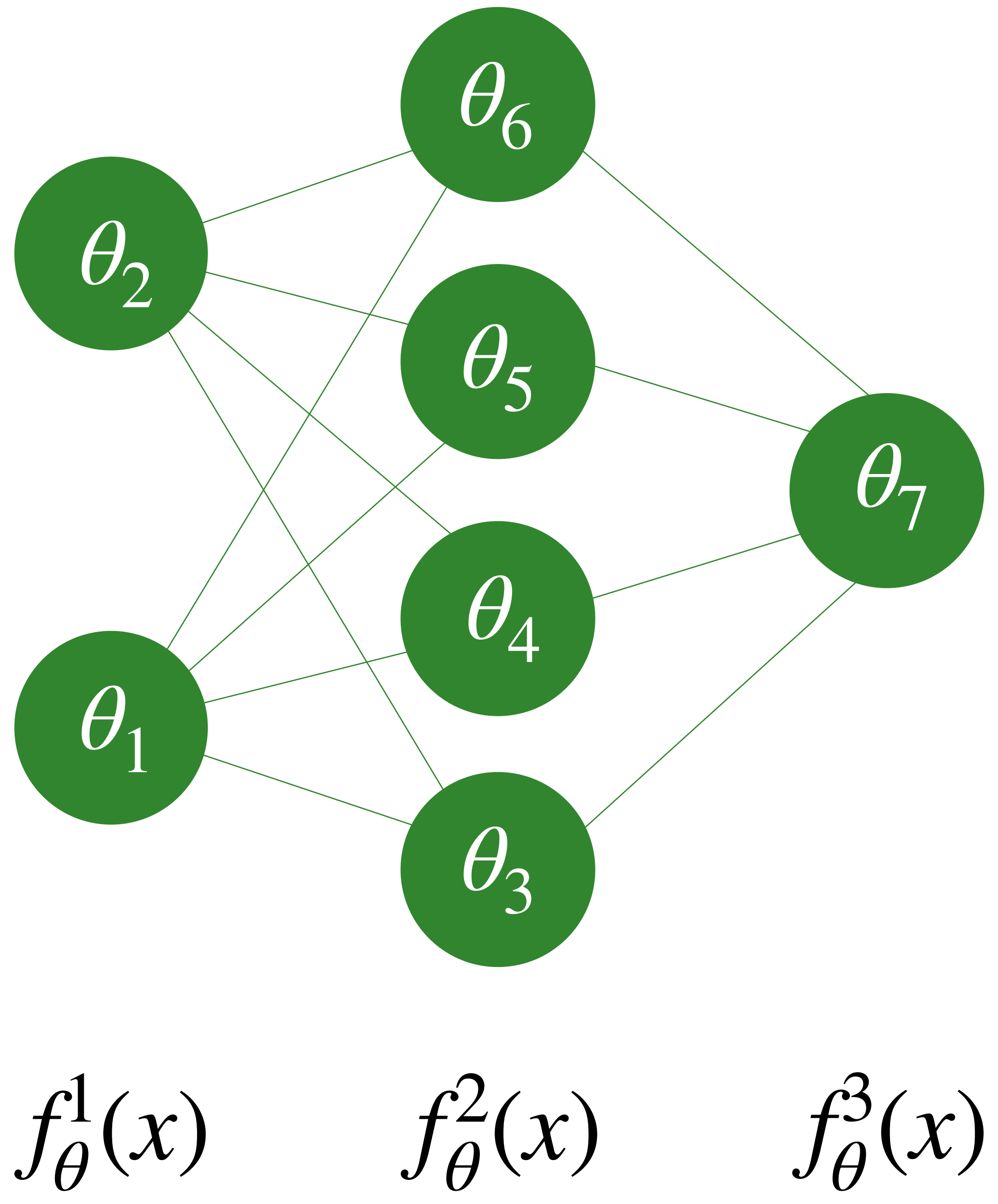
Gemel

How to reduce GPU memory bottlenecks in edge video analytics?

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

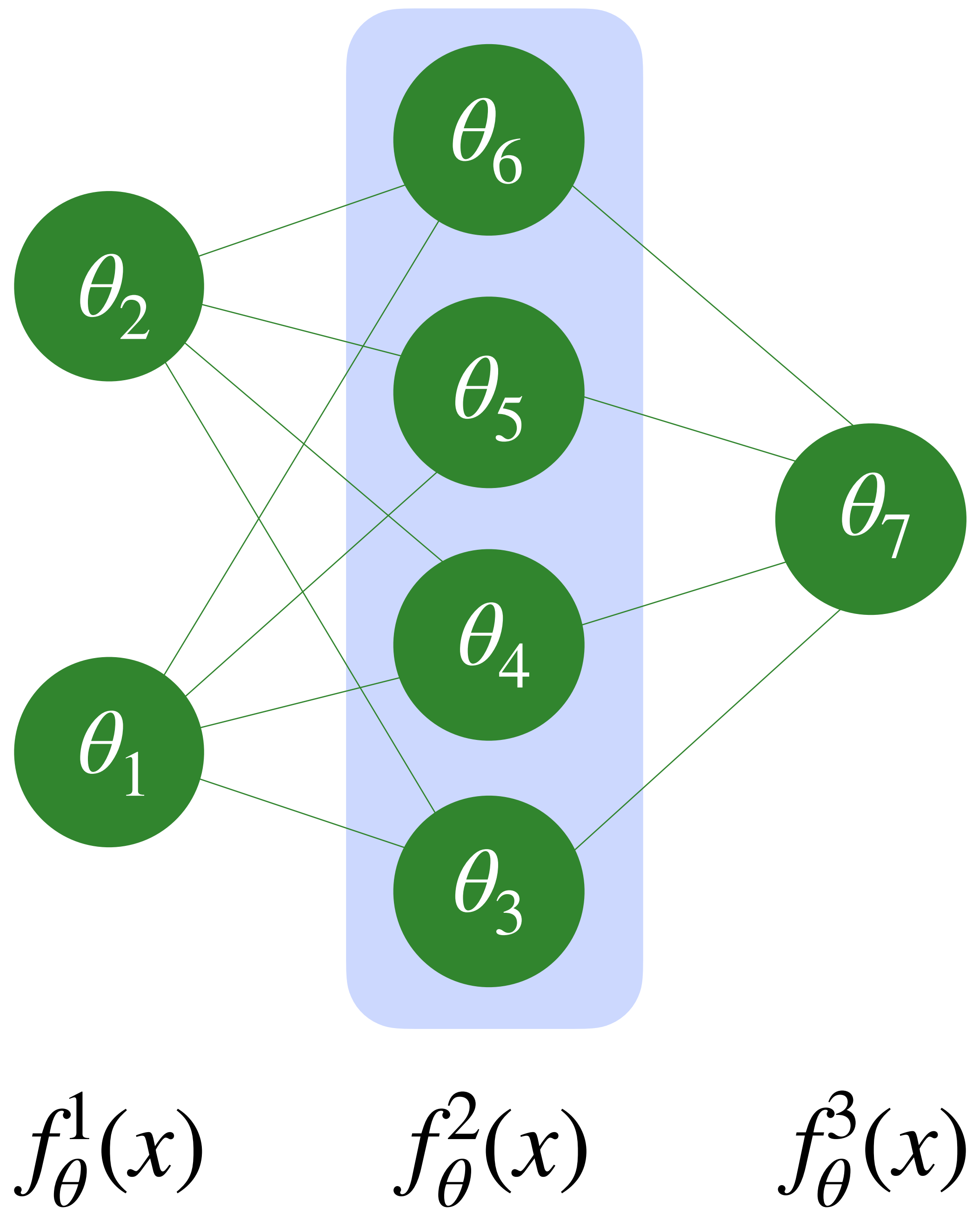
Observation: despite workload diversity, models often share many *layer definitions*

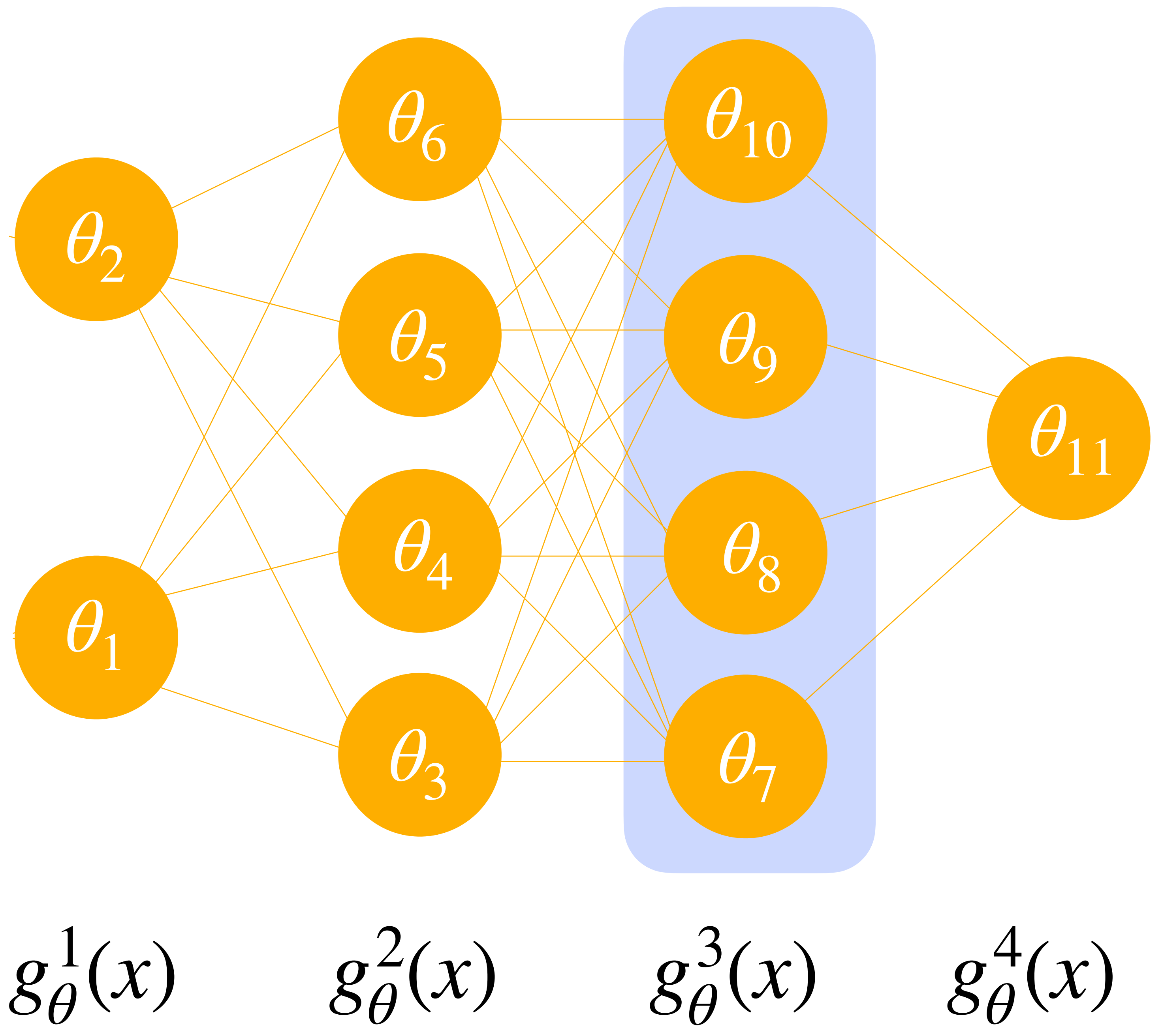
Shared Layer Definitions Across Models



Shared Layer Definitions Across Models

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

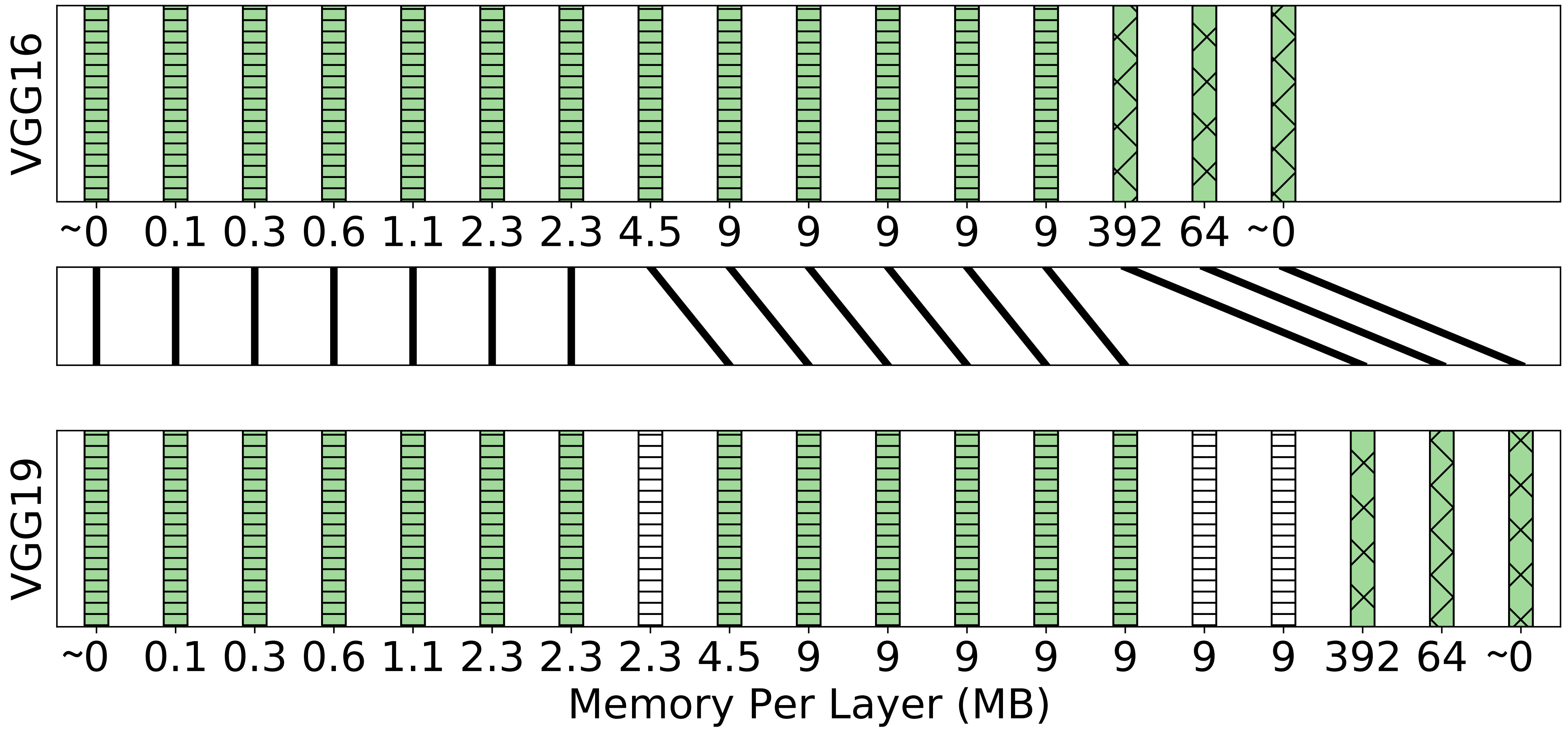




Shared layer definitions appear in…

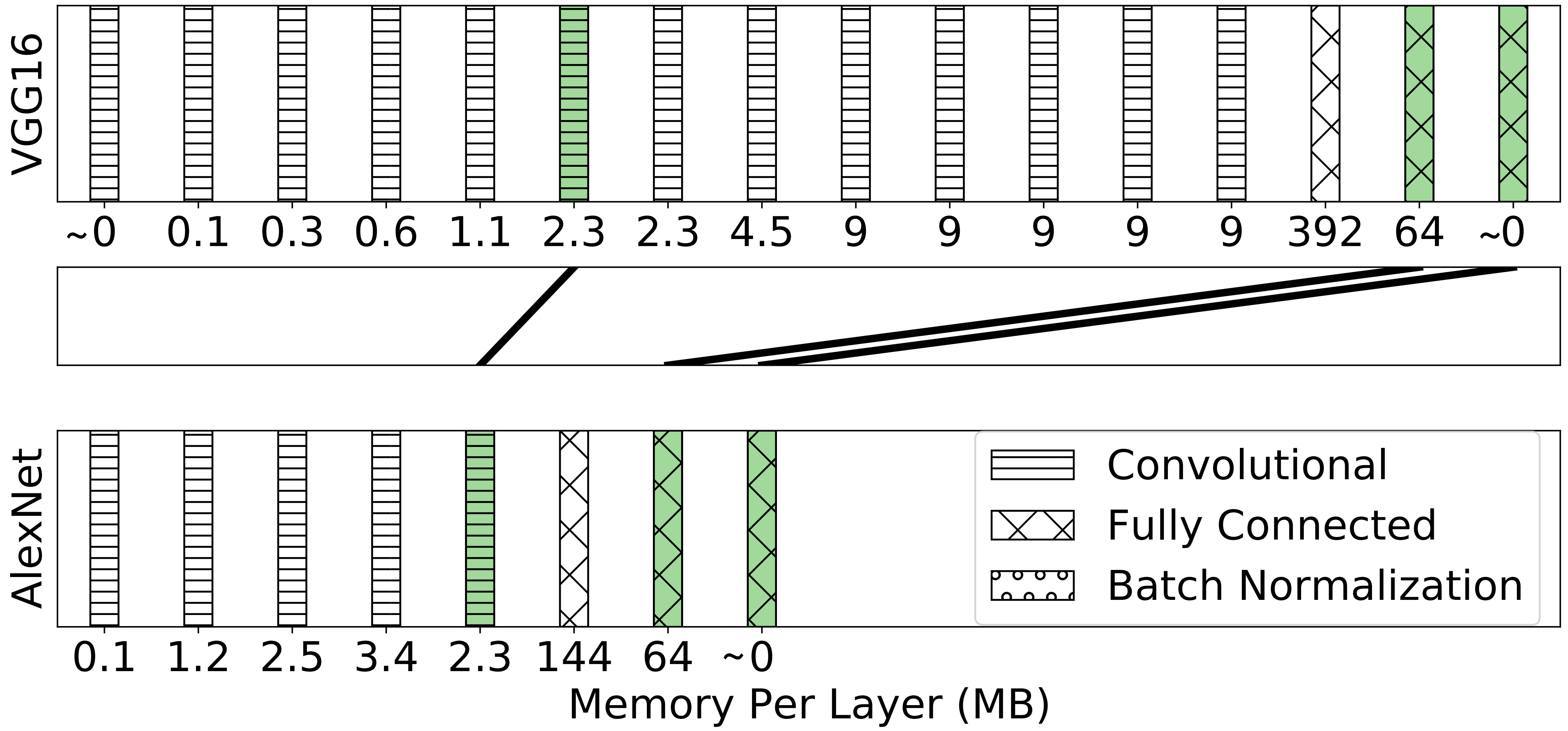
Models from the Same Architecture Family

e.g., VGG16 & VGG19



Models from Different Architecture Families

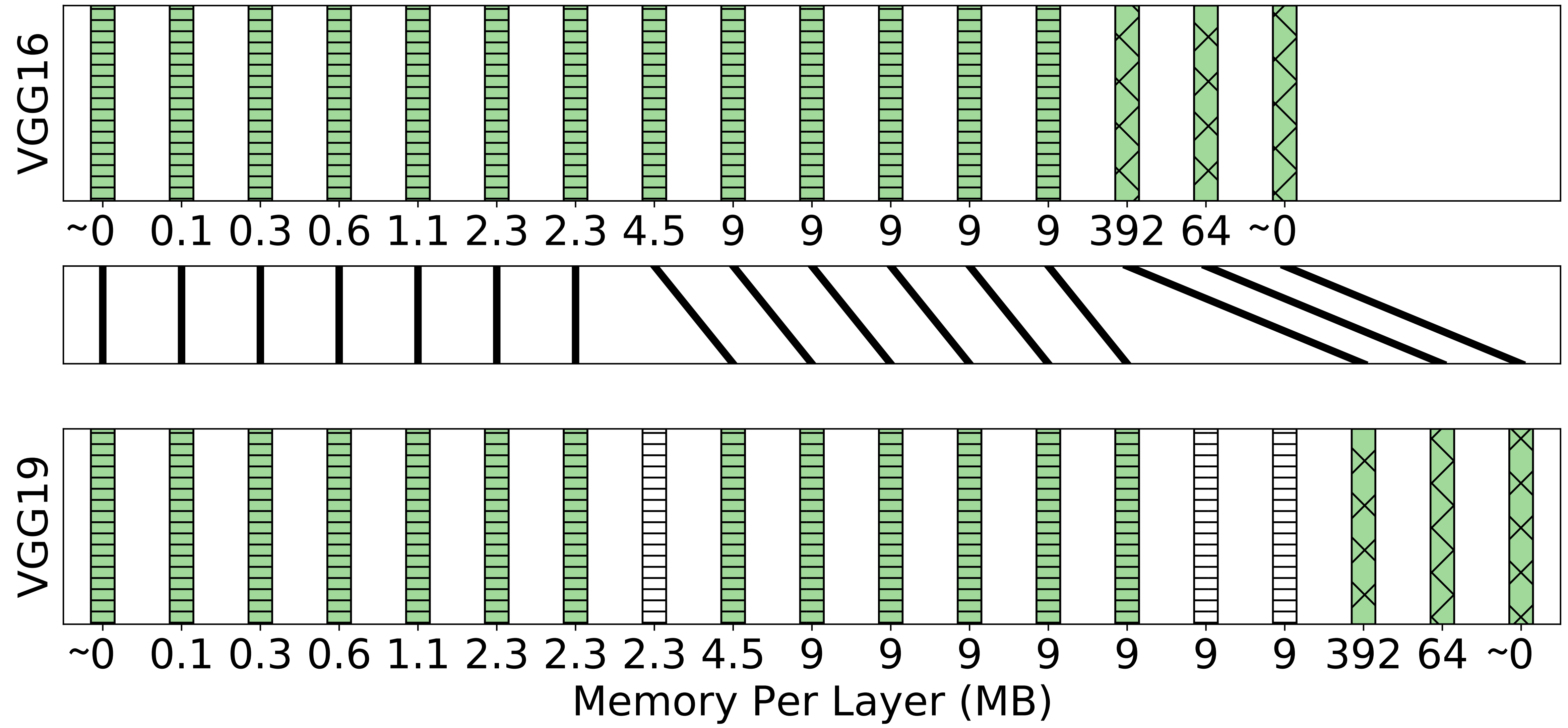
e.g., VGG16 & AlexNet



Shared layer definitions appear in…

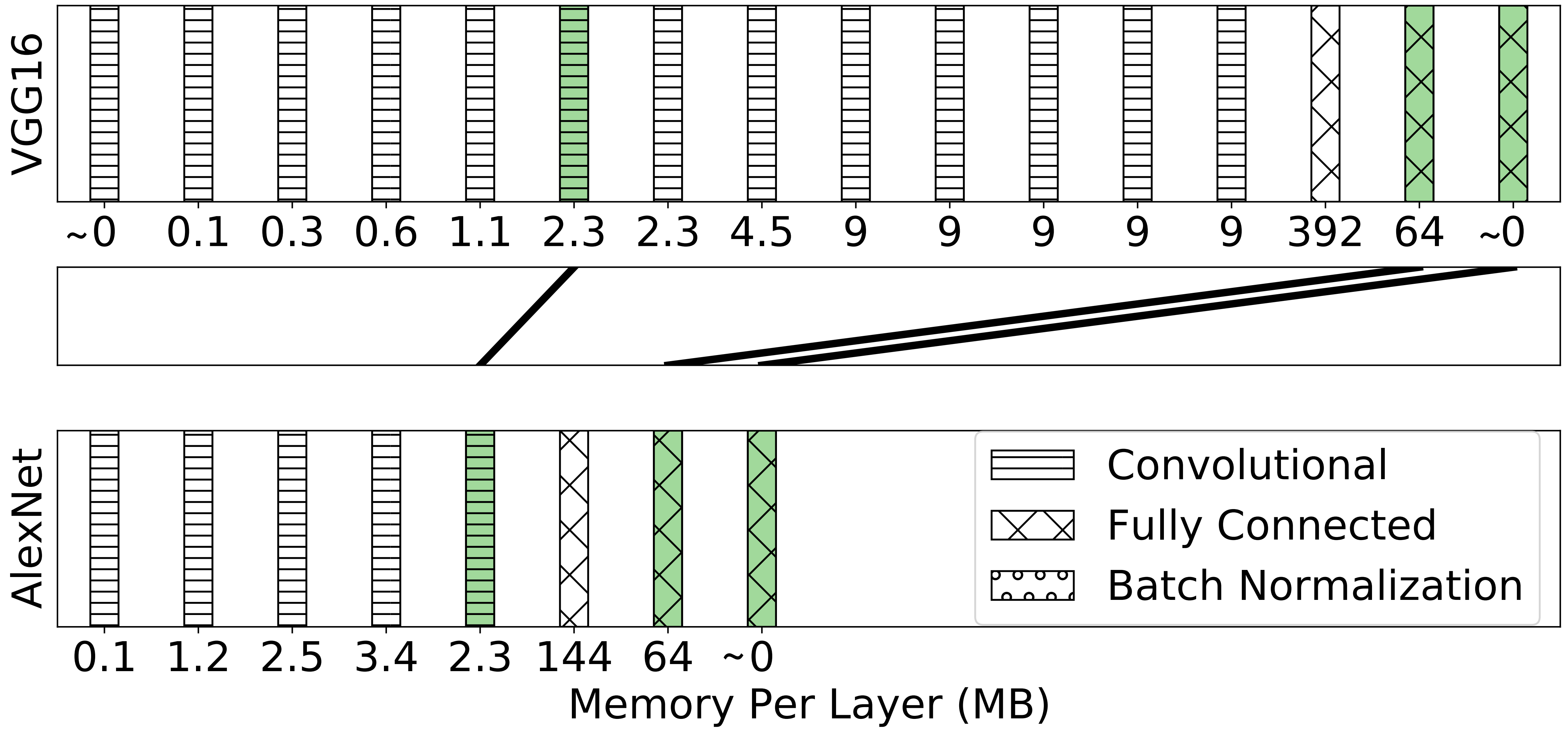
Models from the Same Architecture Family

e.g., VGG16 & VGG19



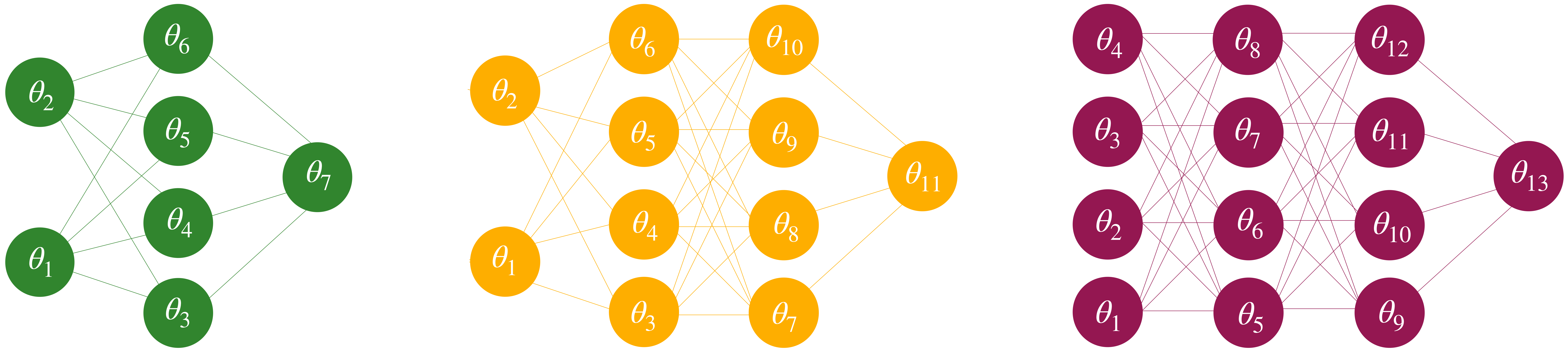
Models from Different Architecture Families

e.g., VGG16 & AlexNet

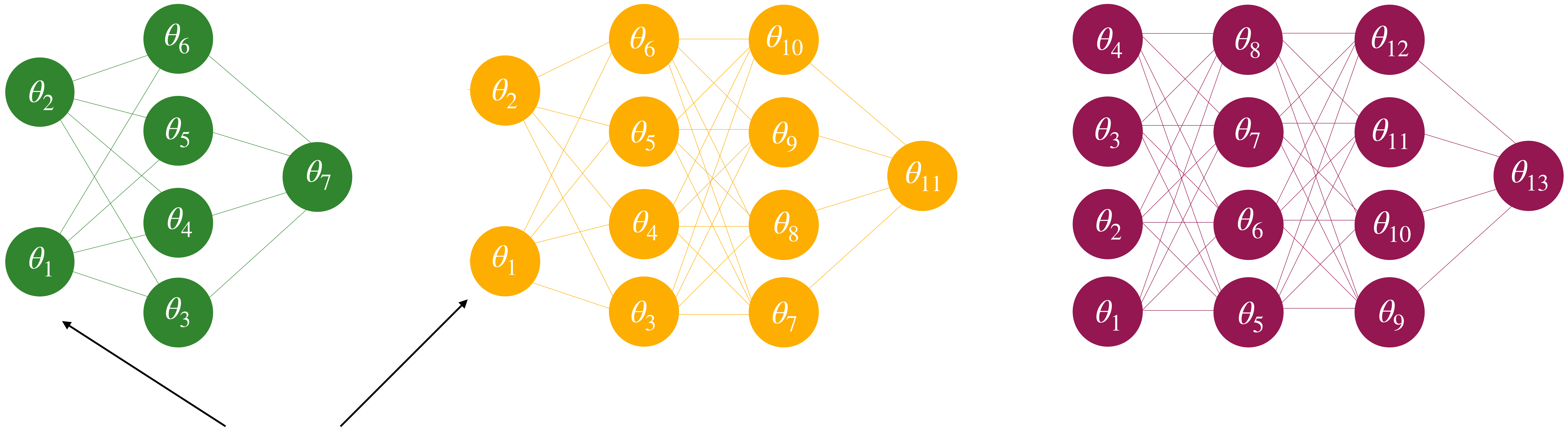


Across 24 different architectures, 43% of all pairs of different models have shared layers

Idea: Find unified weights for shared layers

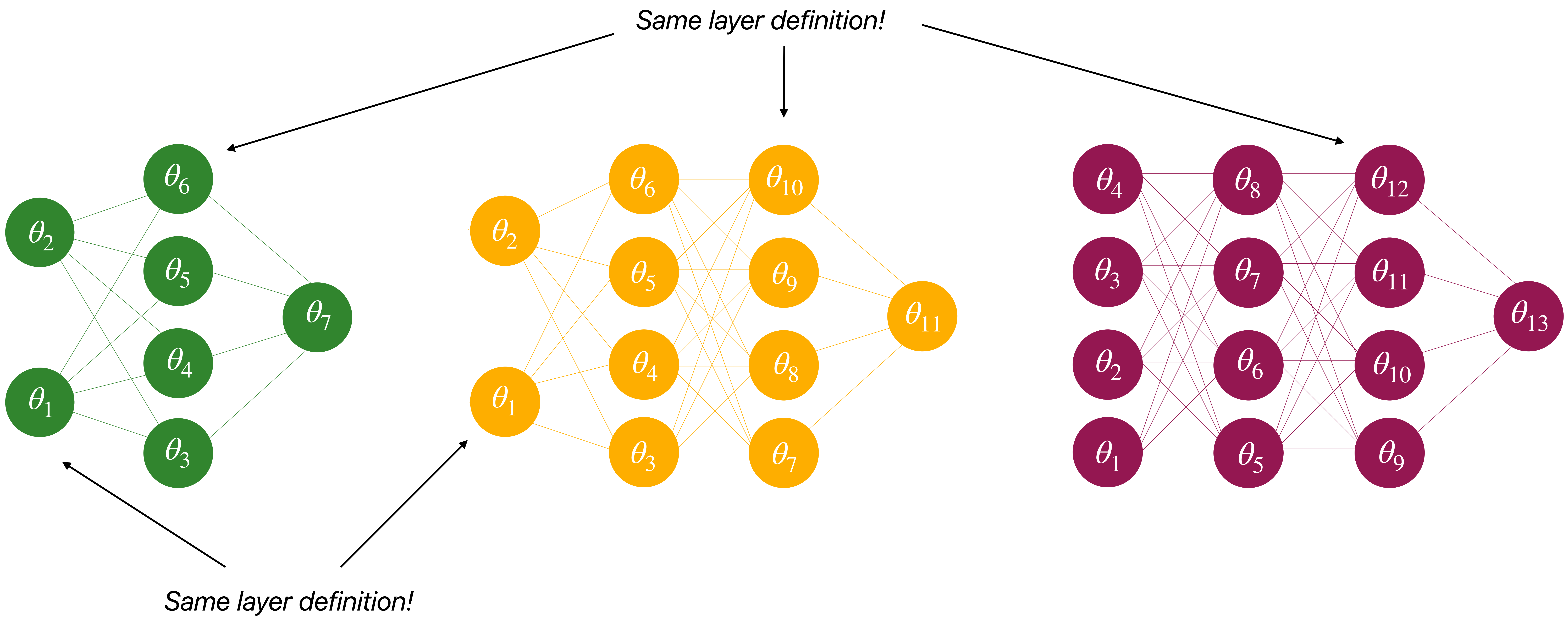


Idea: Find unified weights for shared layers

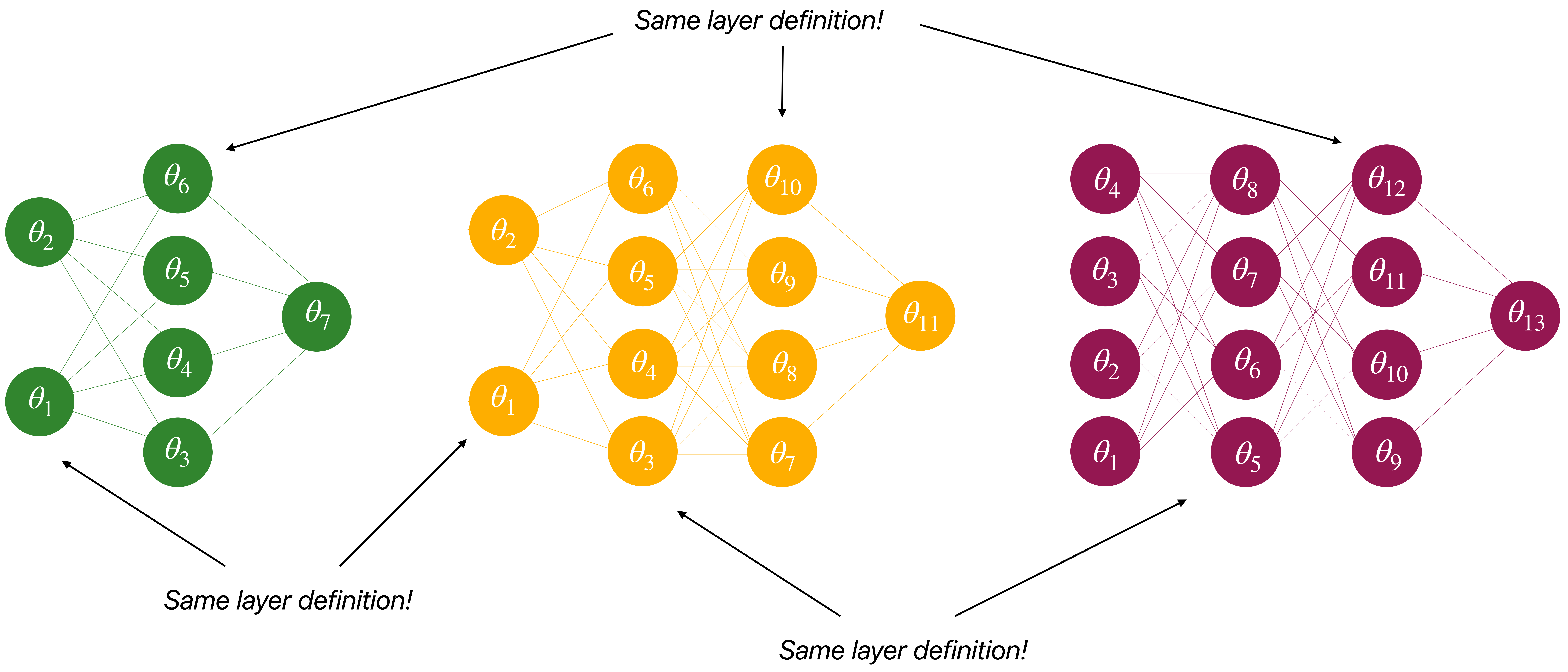


Same layer definition!

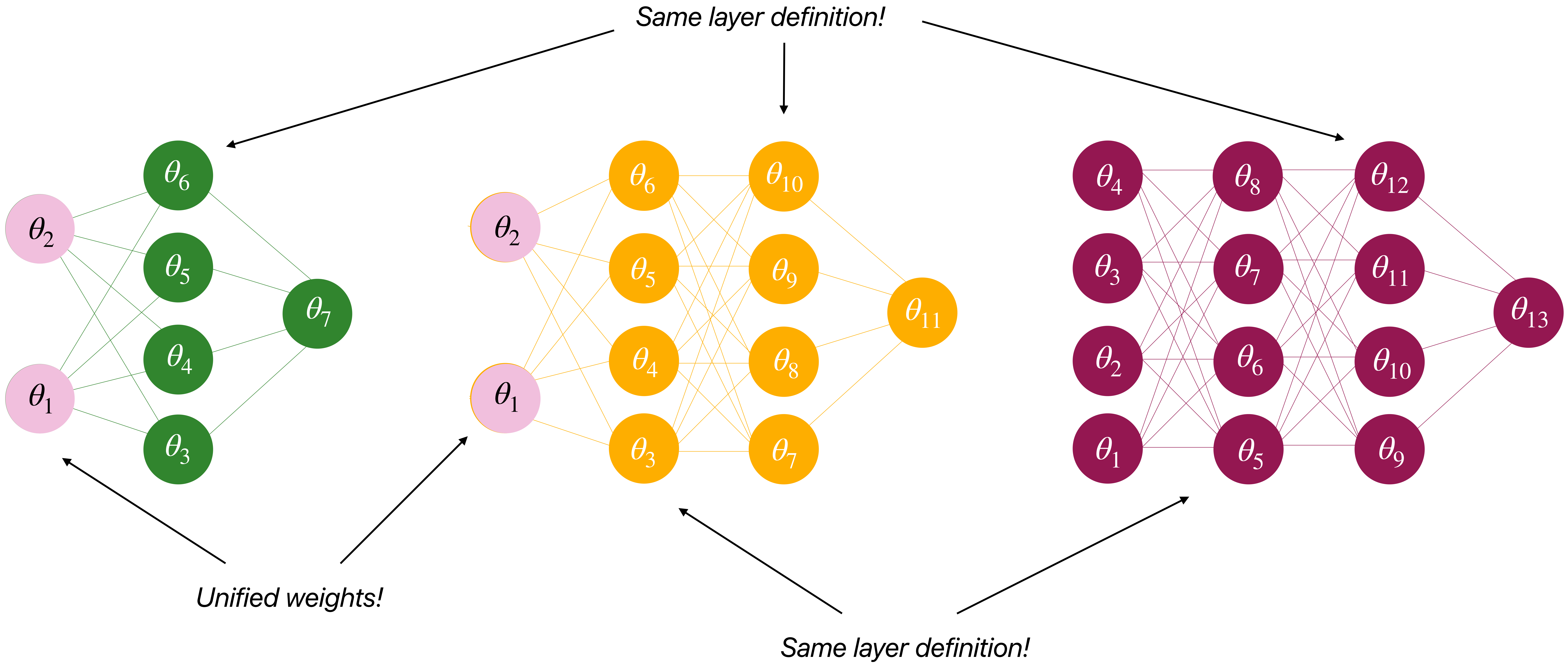
Idea: Find unified weights for shared layers



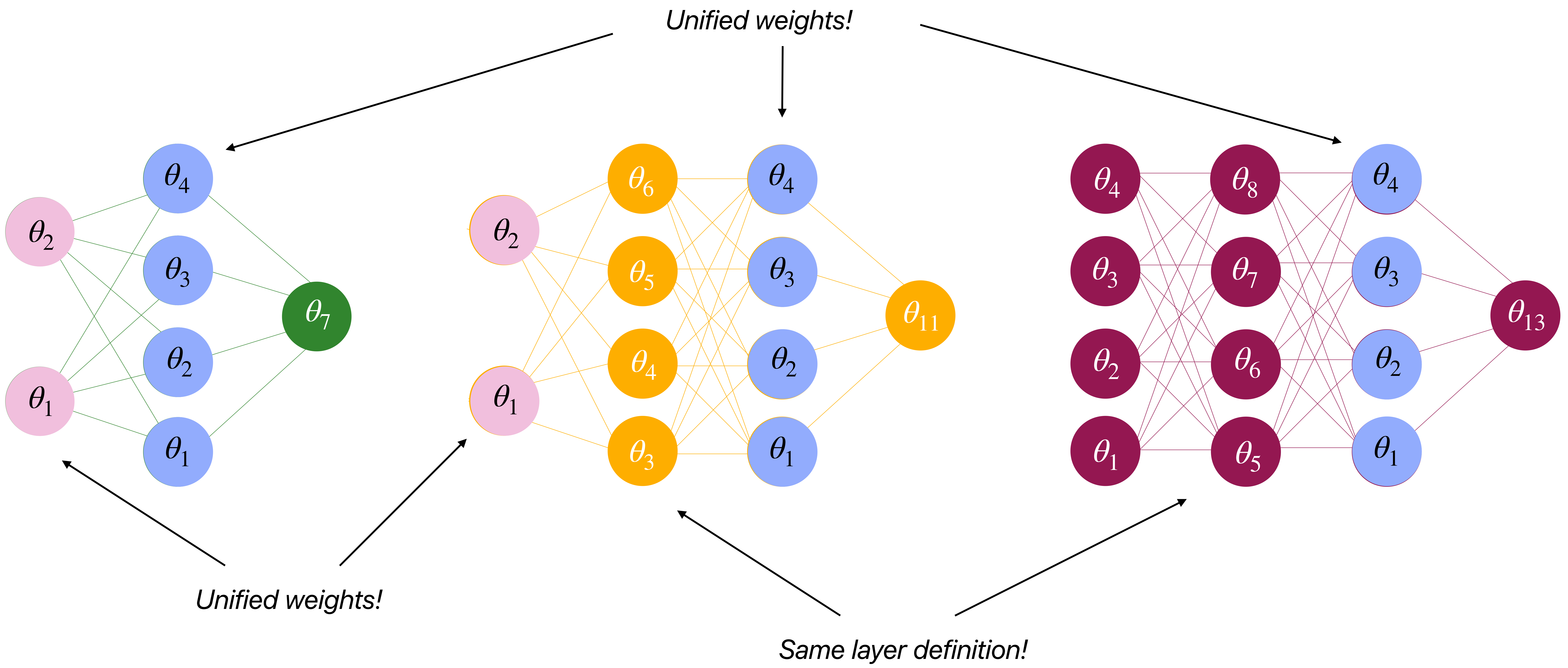
Idea: Find unified weights for shared layers



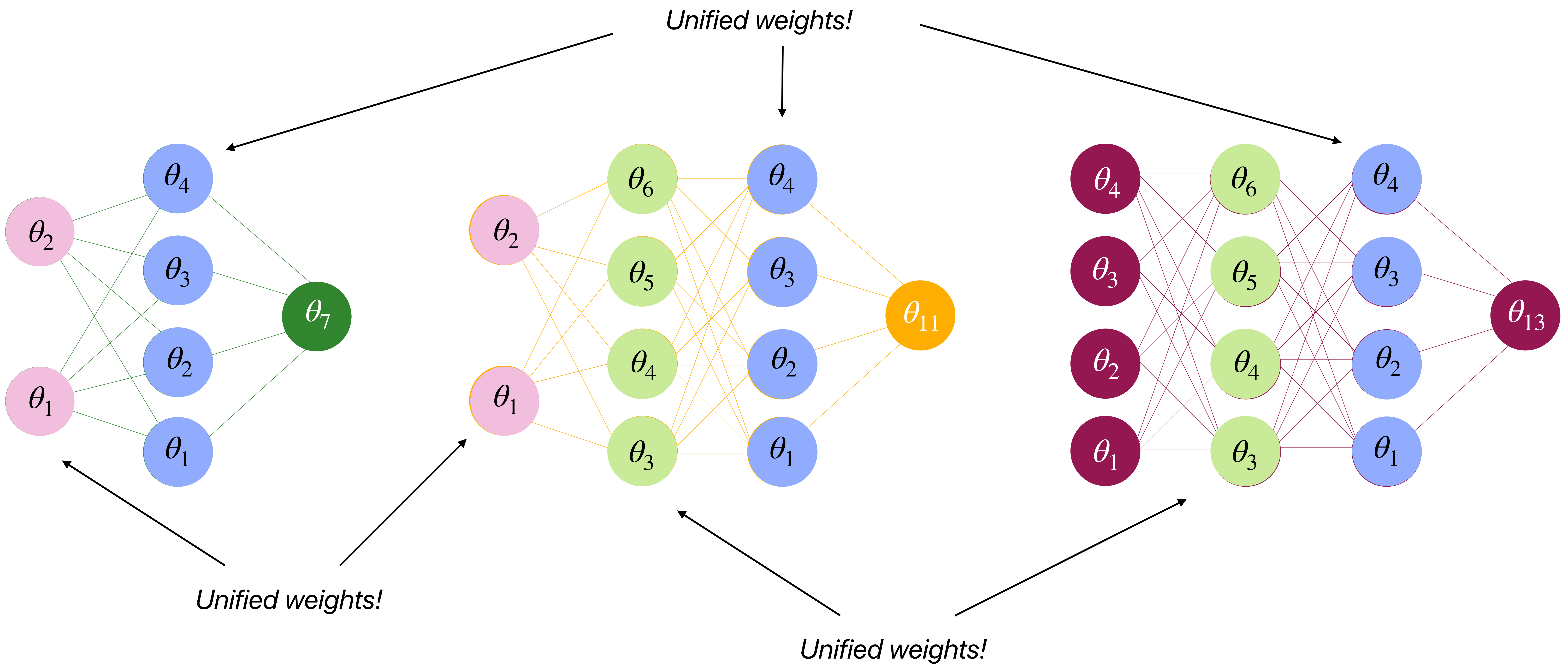
Idea: Find unified weights for shared layers



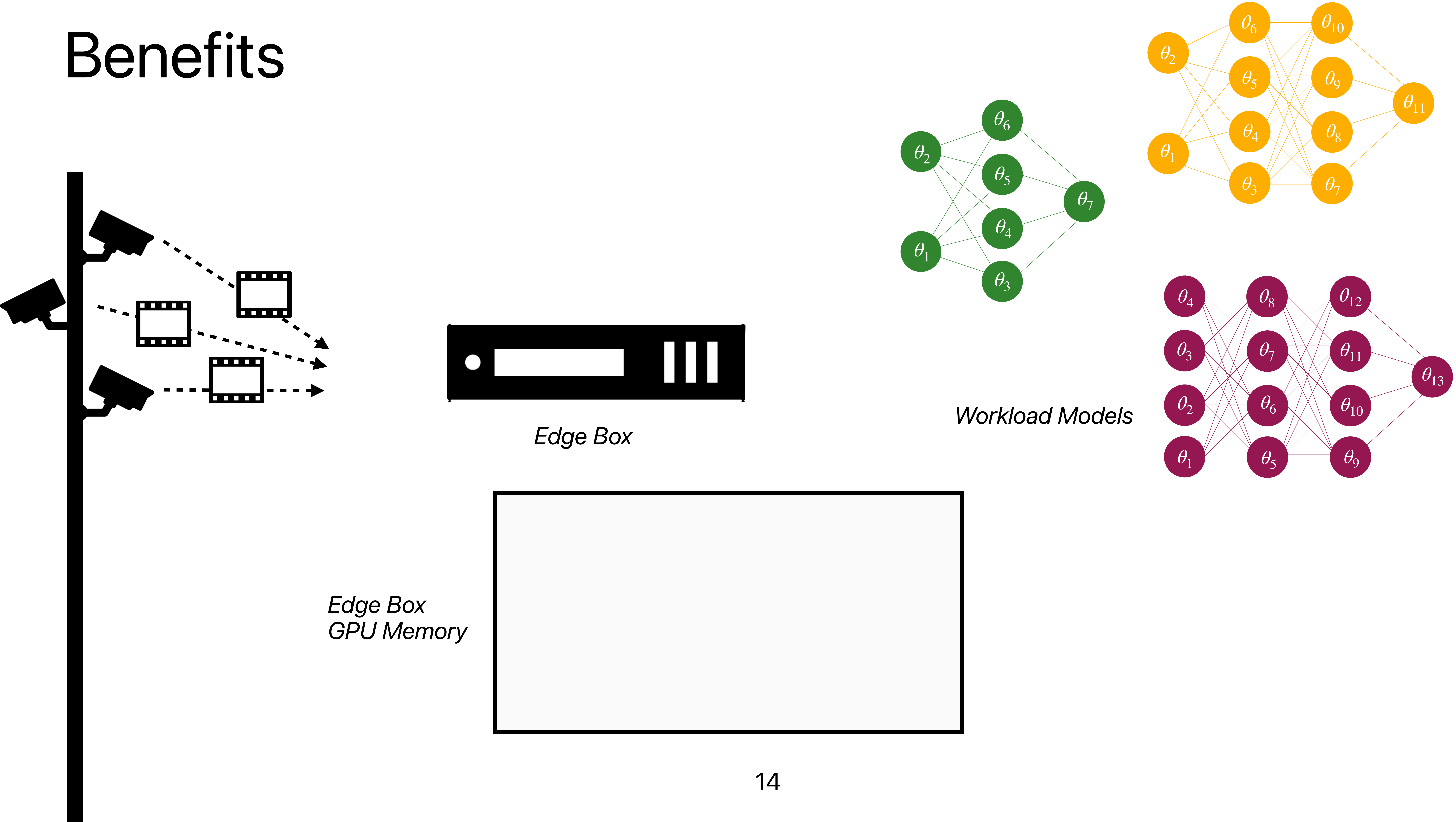
Idea: Find unified weights for shared layers



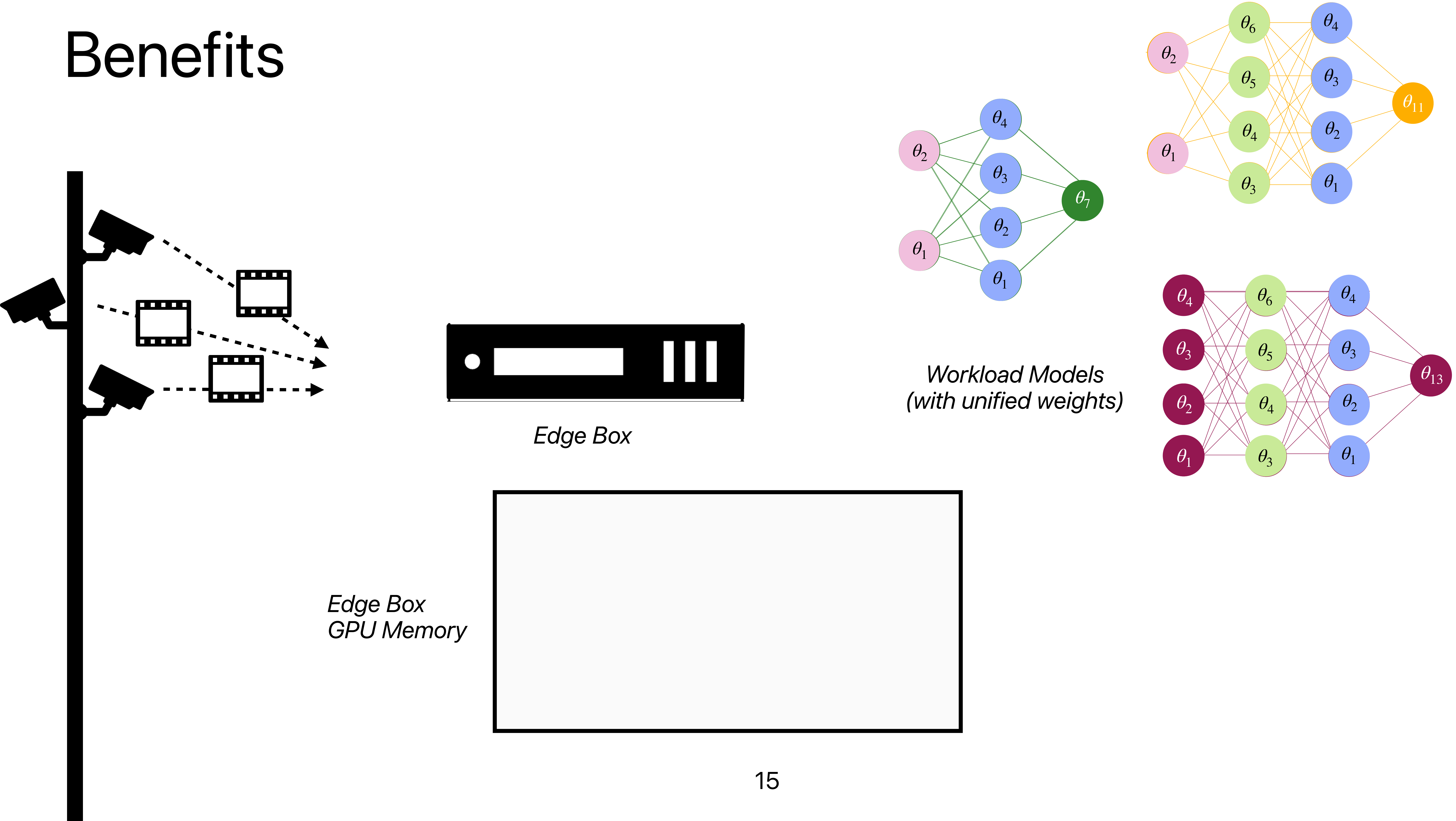
Idea: Find unified weights for shared layers



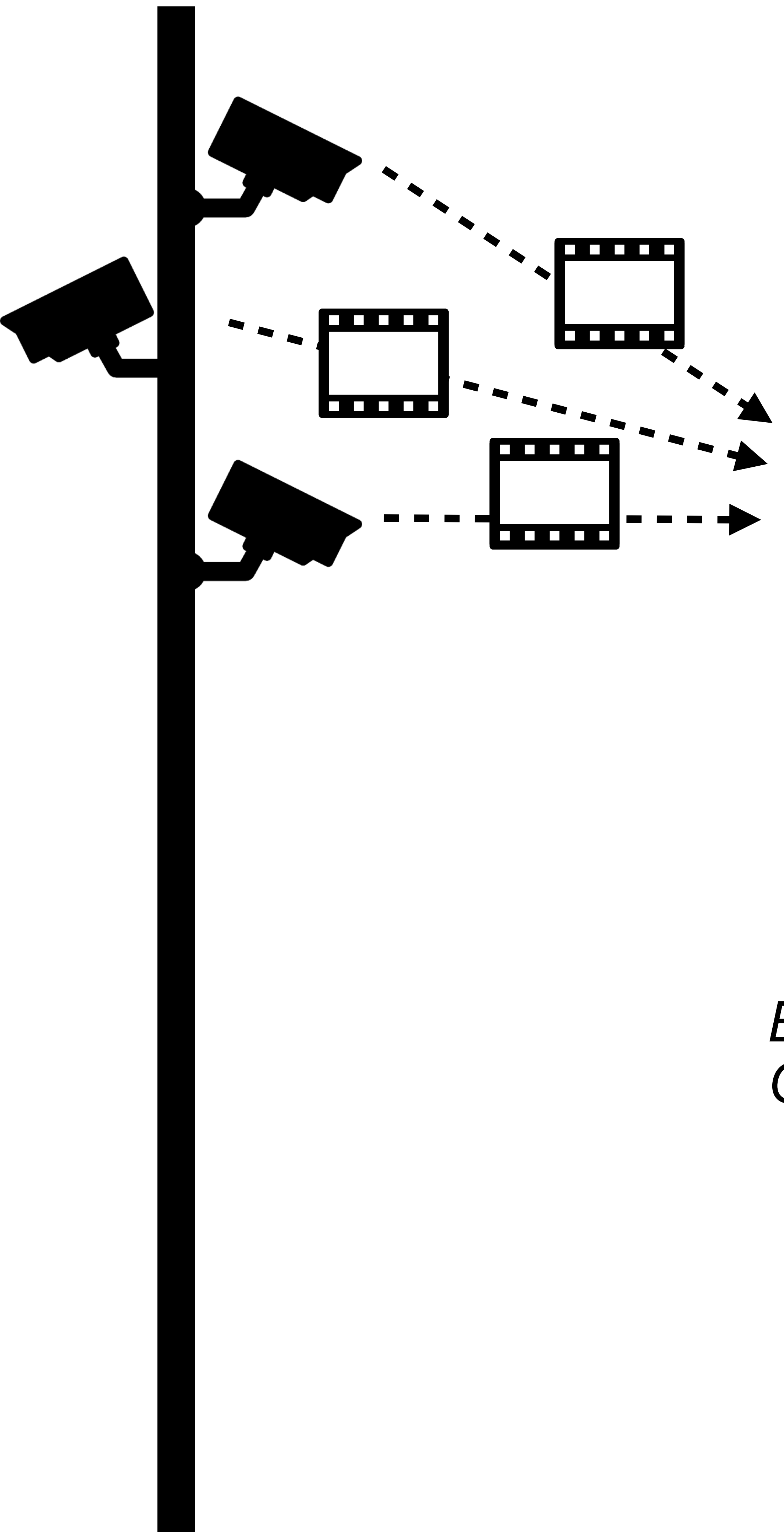
Benefits

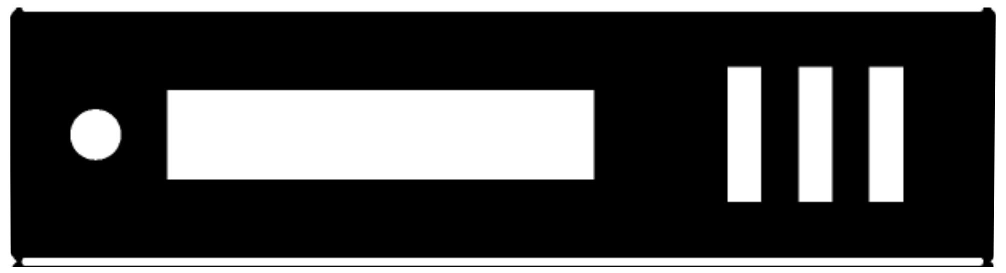


Benefits

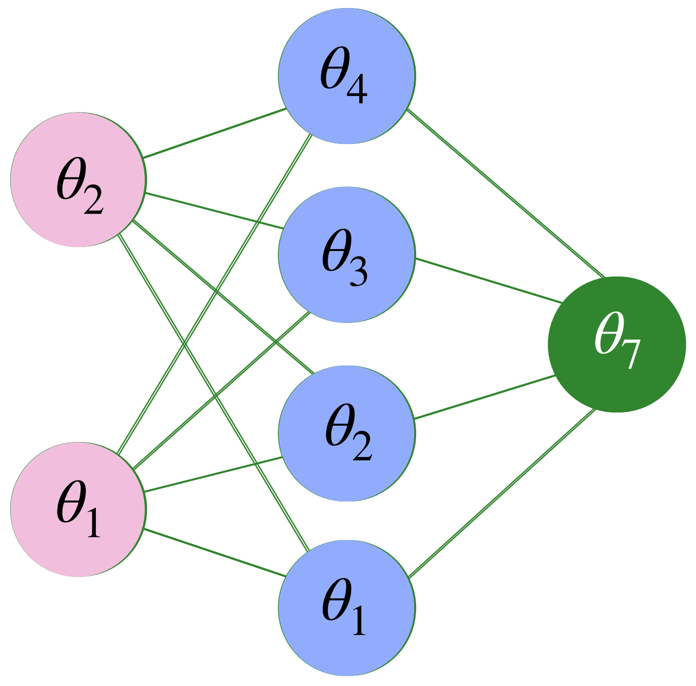

Edge Box
Workload Models
(with unified weights)
Edge Box
GPU Memory

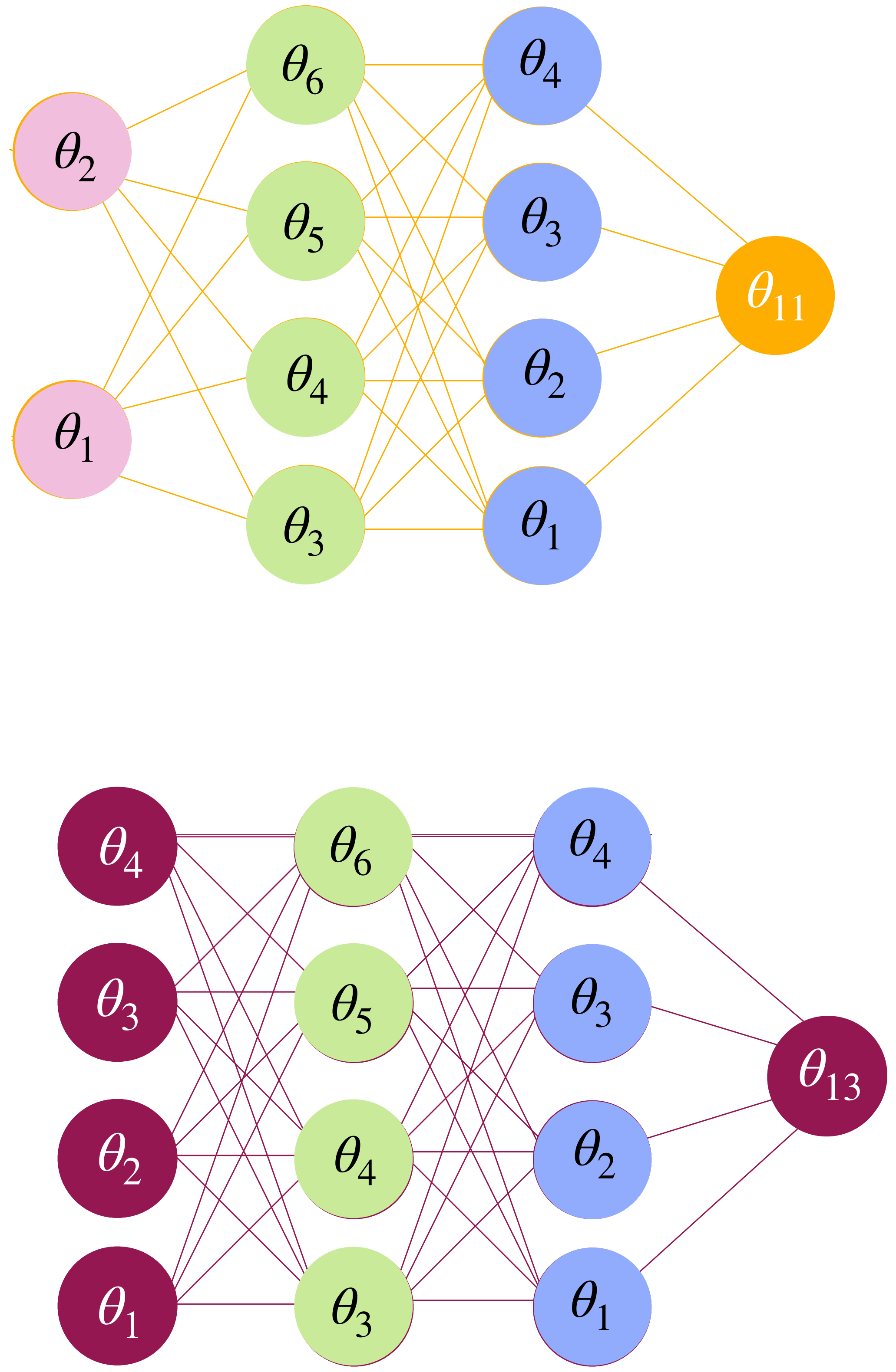
Benefits





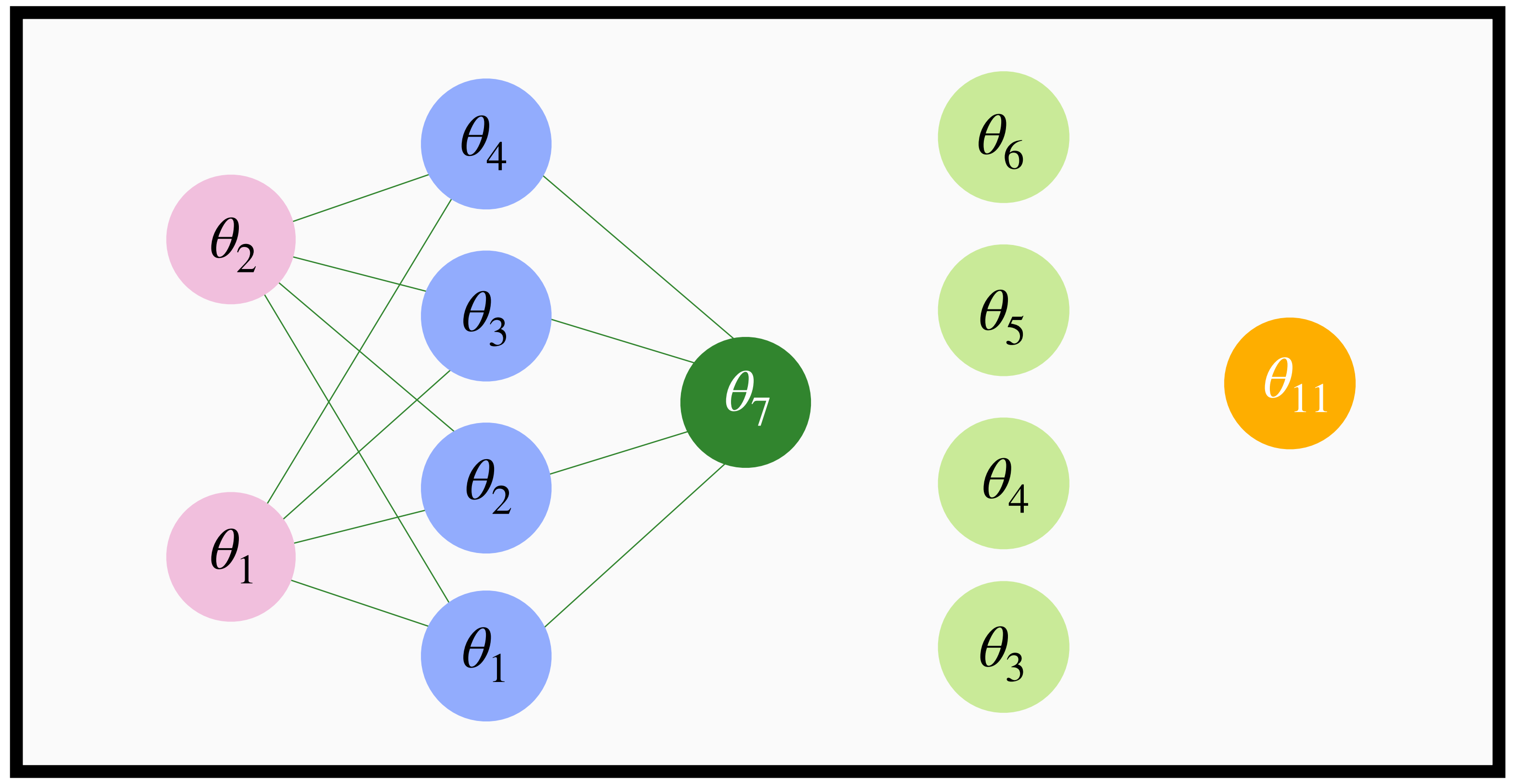
Edge Box



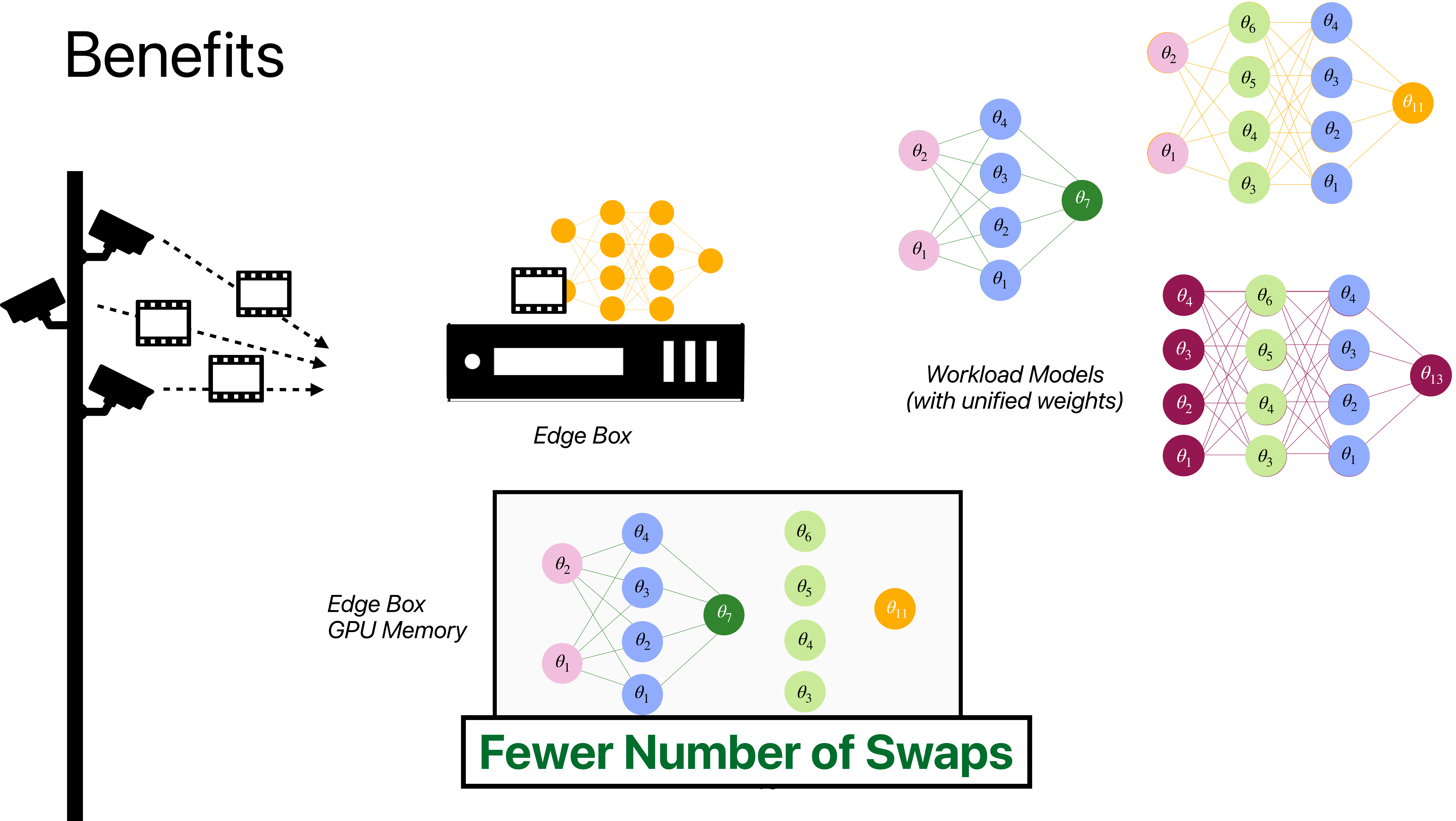


Workload Models
(with unified weights)

Edge Box
GPU Memory

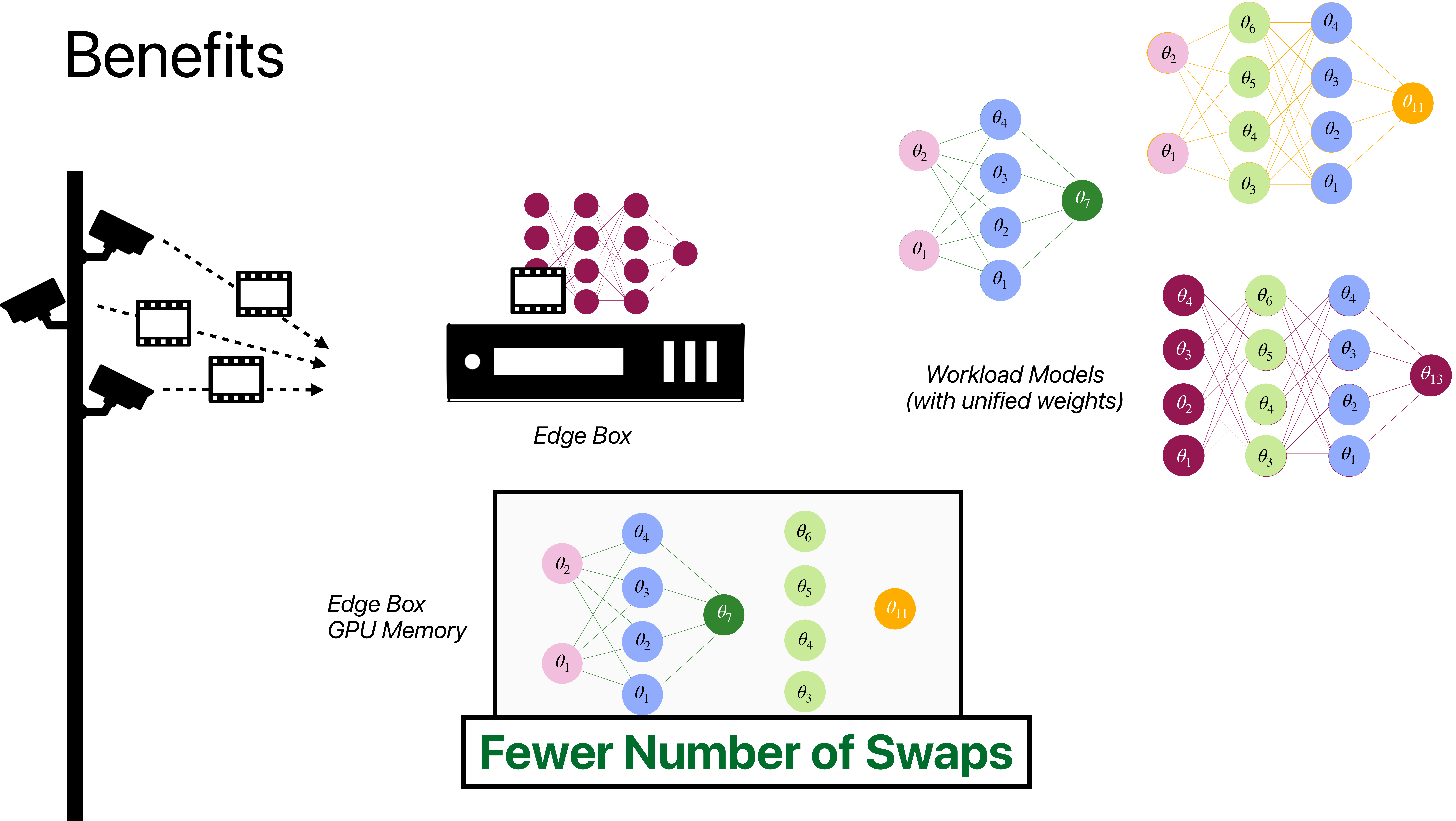


Benefits

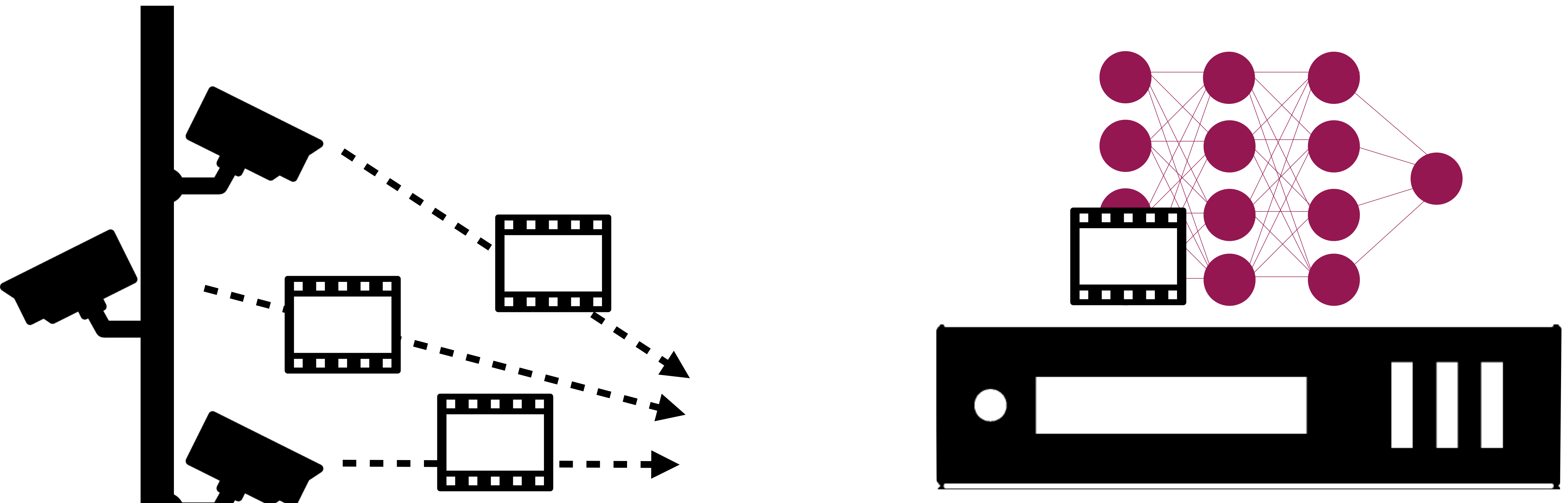


Edge Box
Workload Models
(with unified weights)
Edge Box
GPU Memory
Fewer Number of Swaps

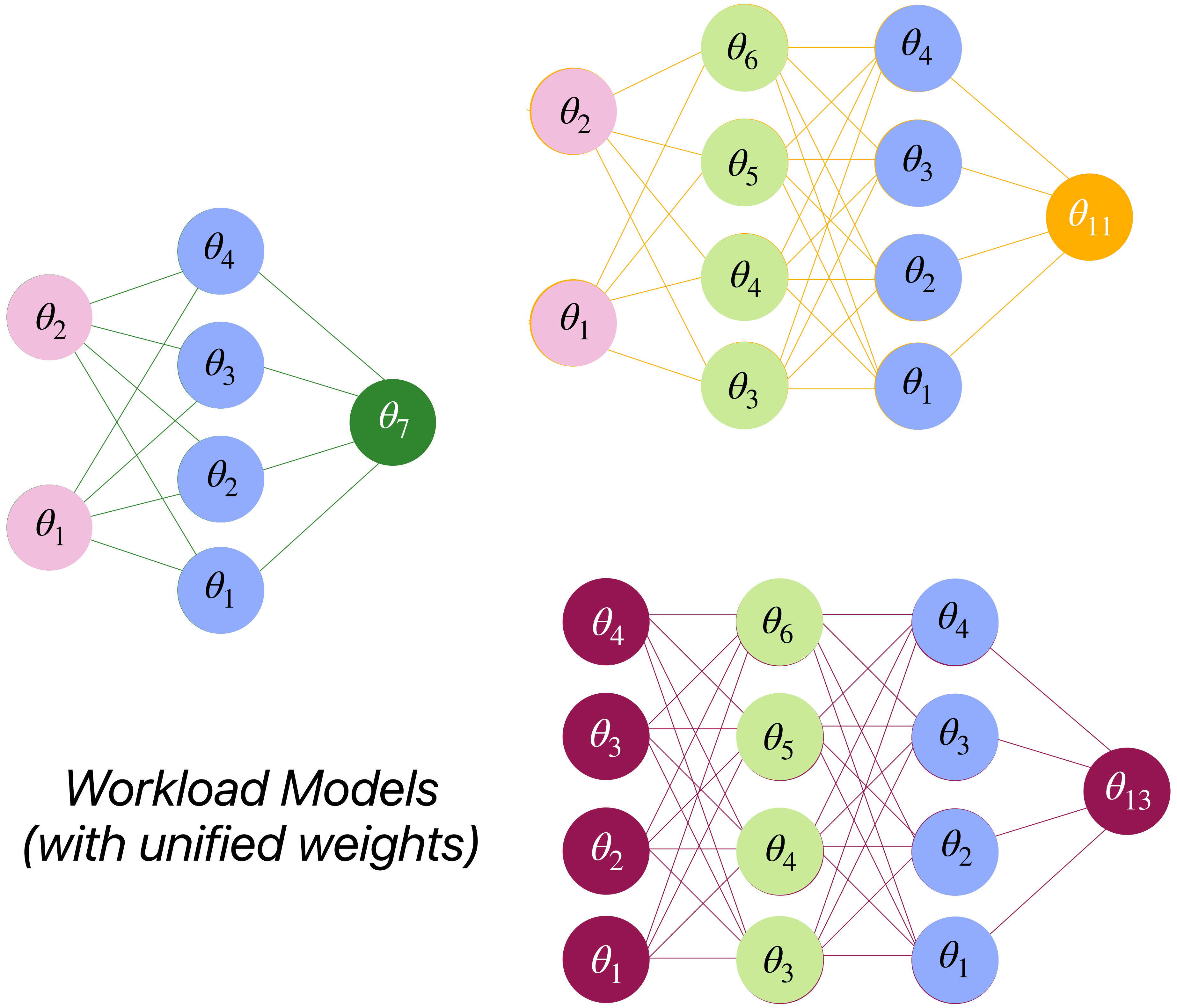
Benefits


Edge Box
Workload Models
(with unified weights)
Edge Box
GPU Memory
Fewer Number of Swaps

Benefits

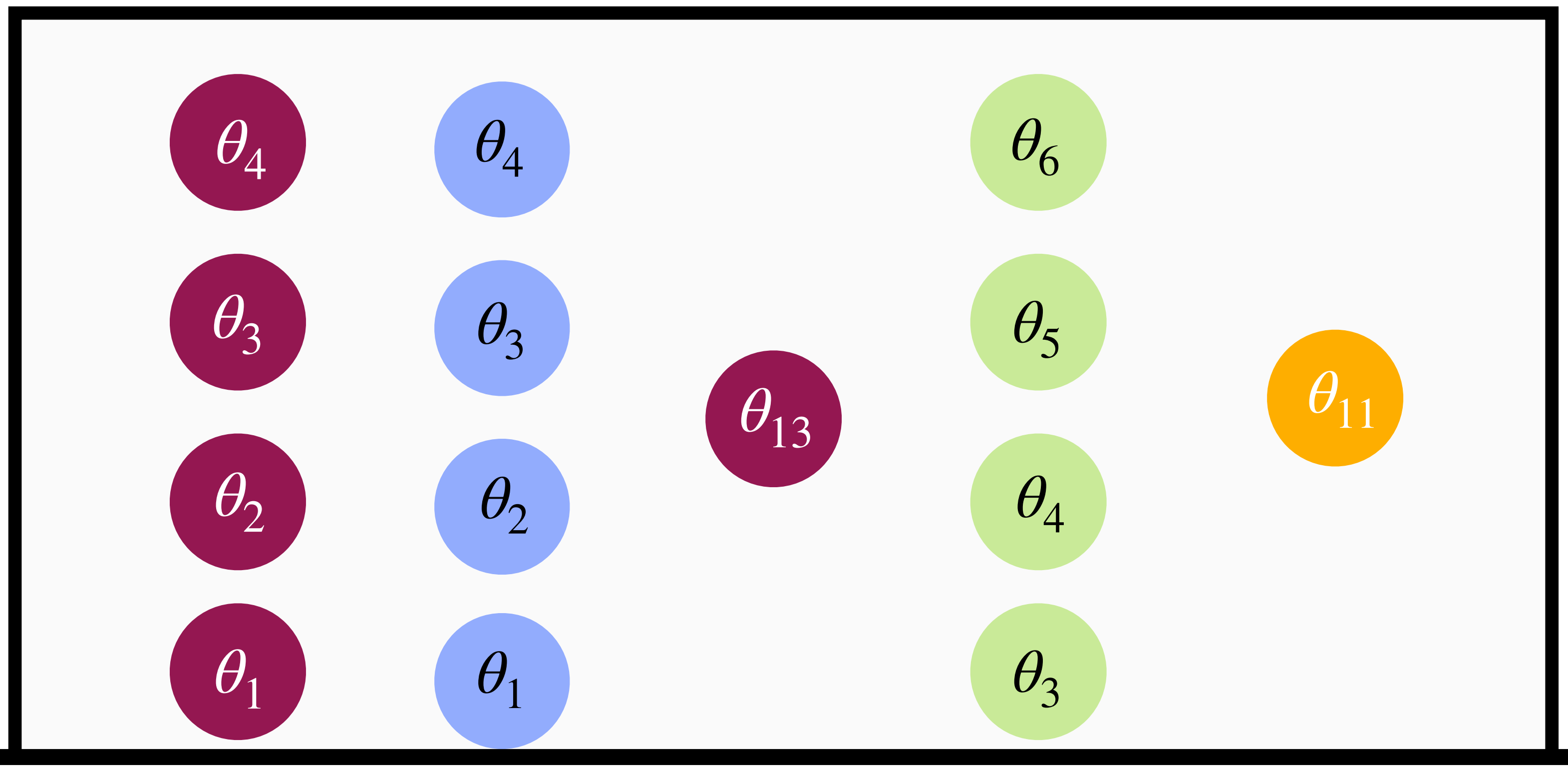


Edge Box



Workload Models
(with unified weights)

Edge Box
GPU Memory

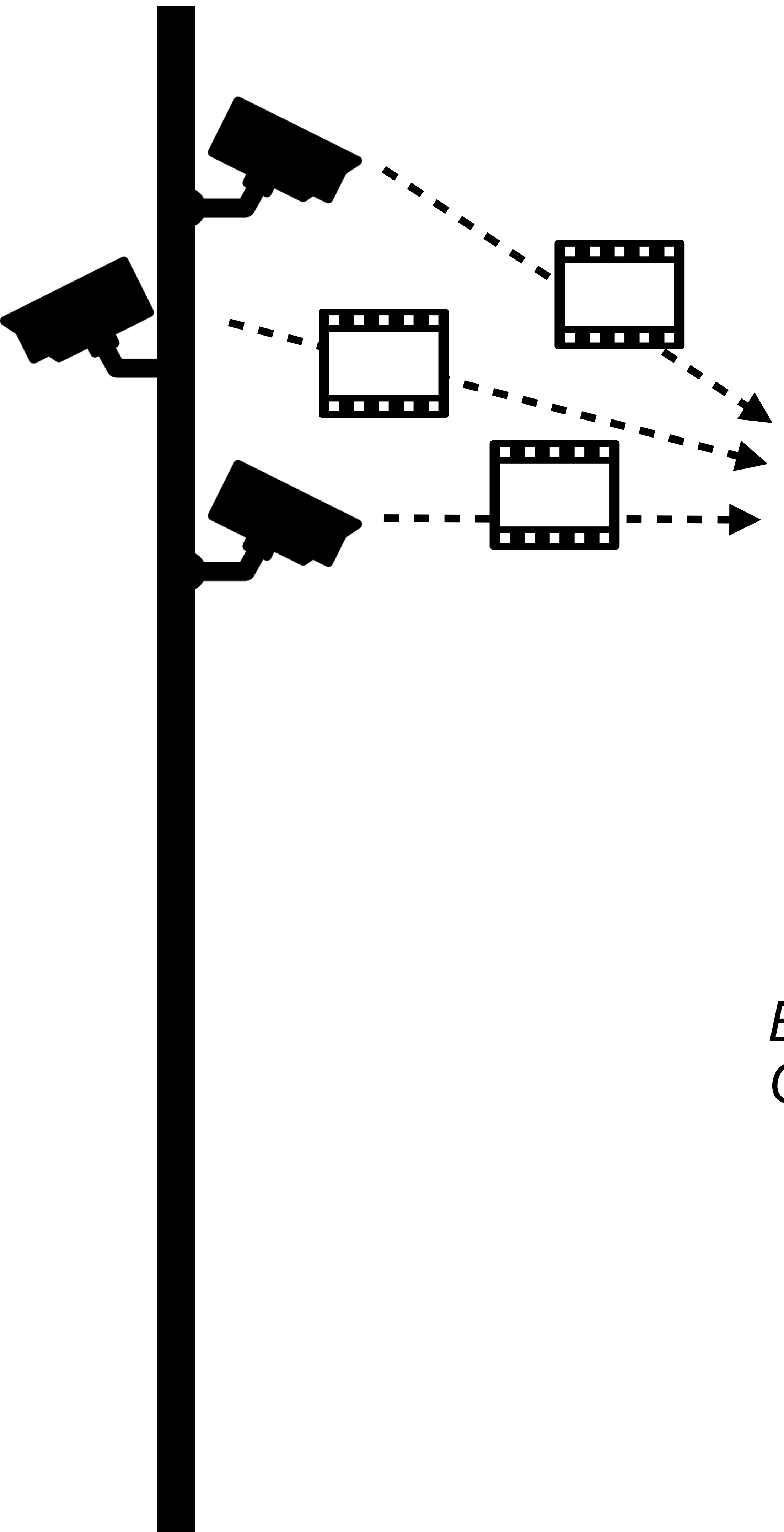


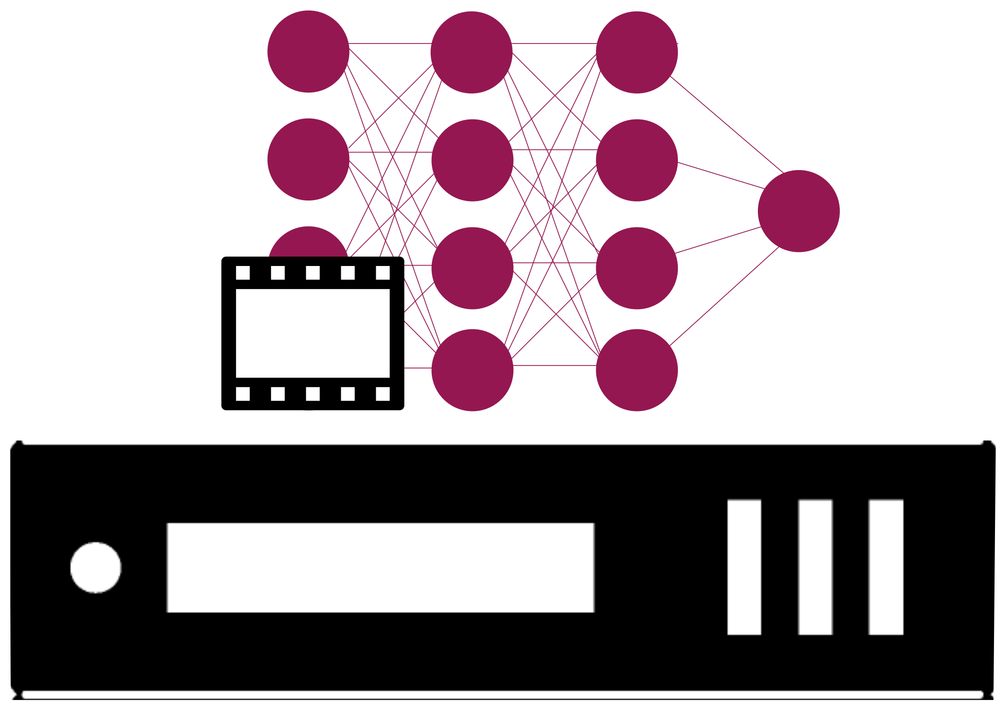
Fewer Number of Swaps

Remaining Swaps are Faster

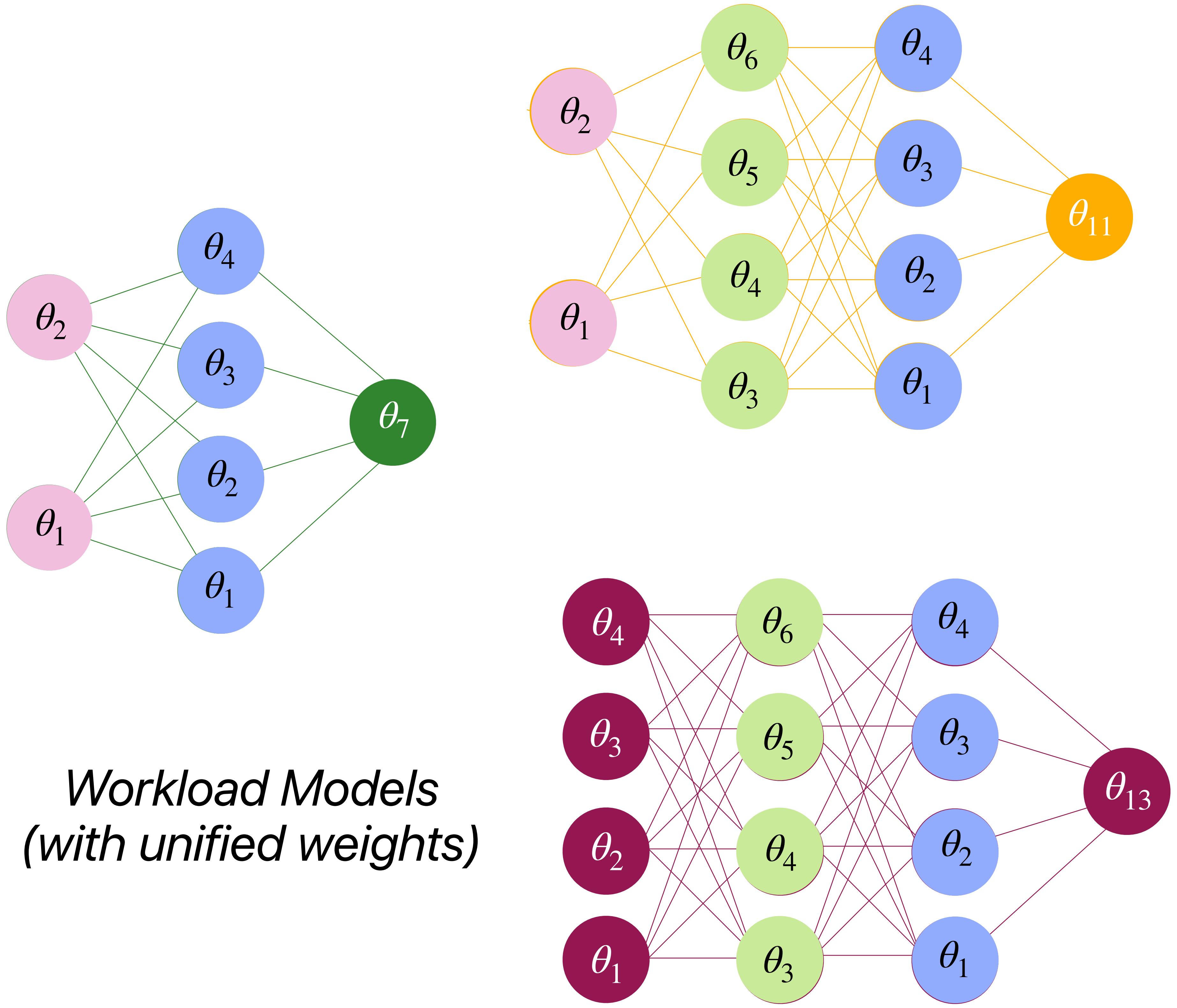
Benefits

Reduce per-workload memory usage by 17-86%



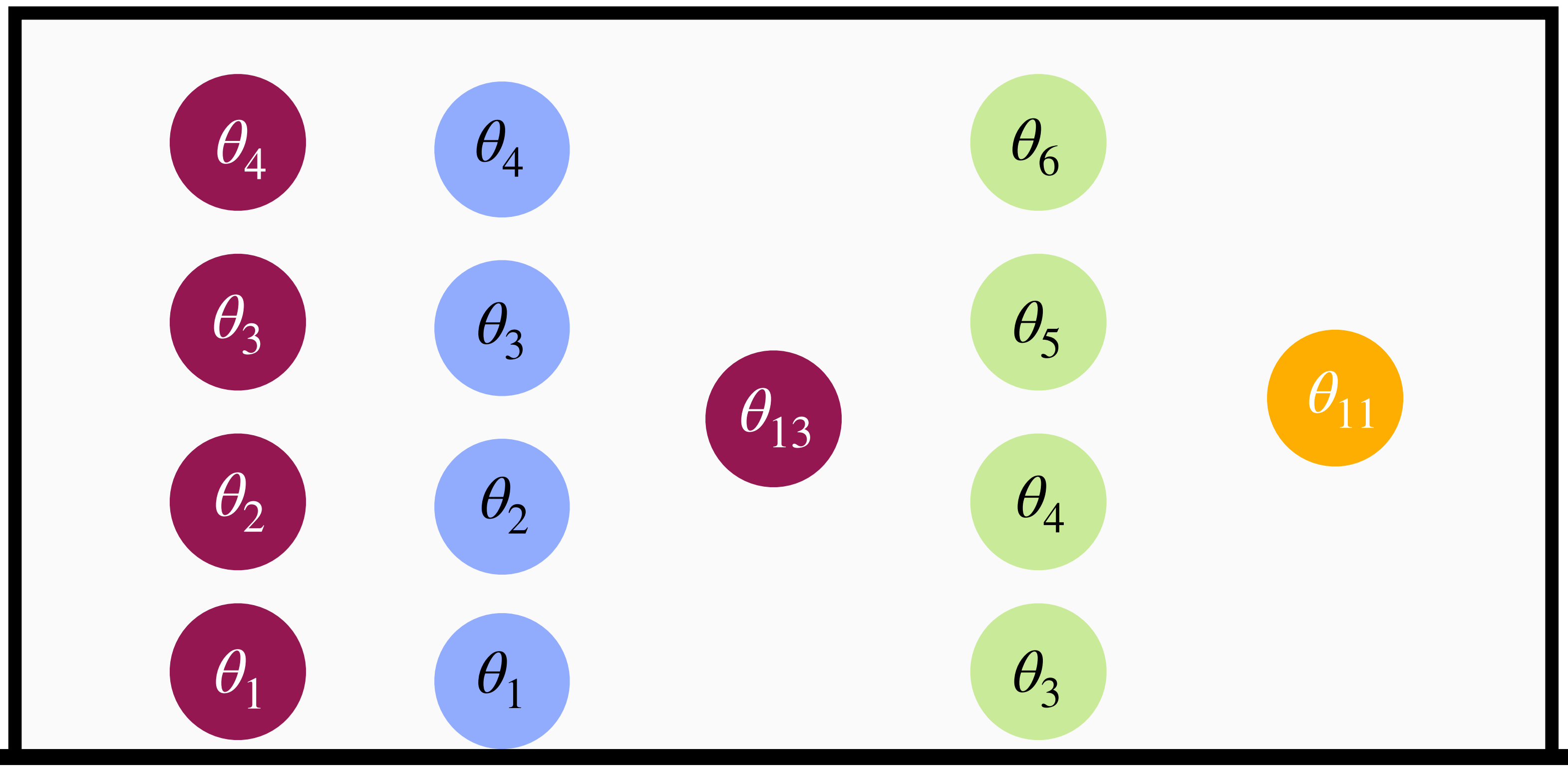


Edge Box



Workload Models (with unified weights)

Edge Box GPU Memory



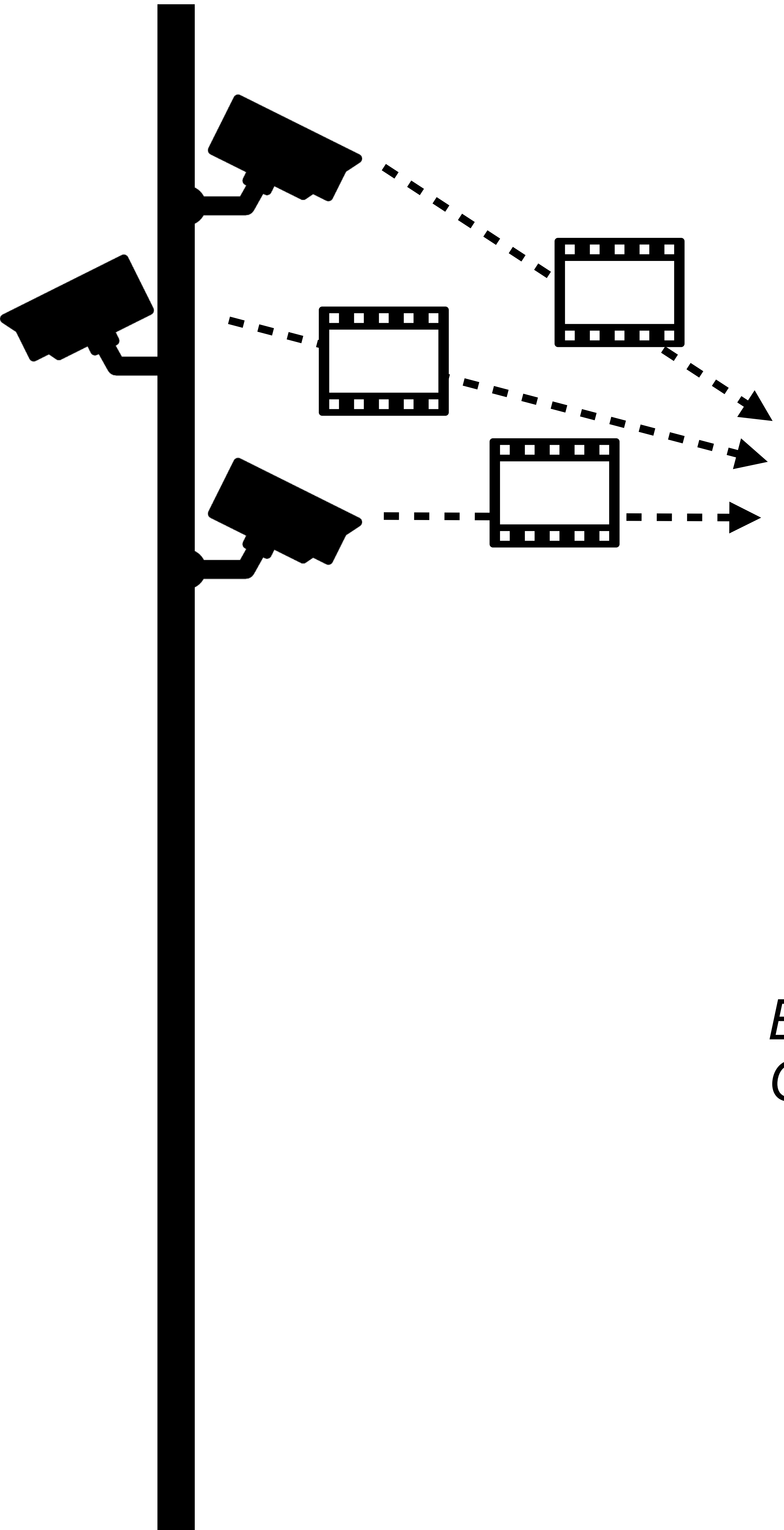
Fewer Number of Swaps

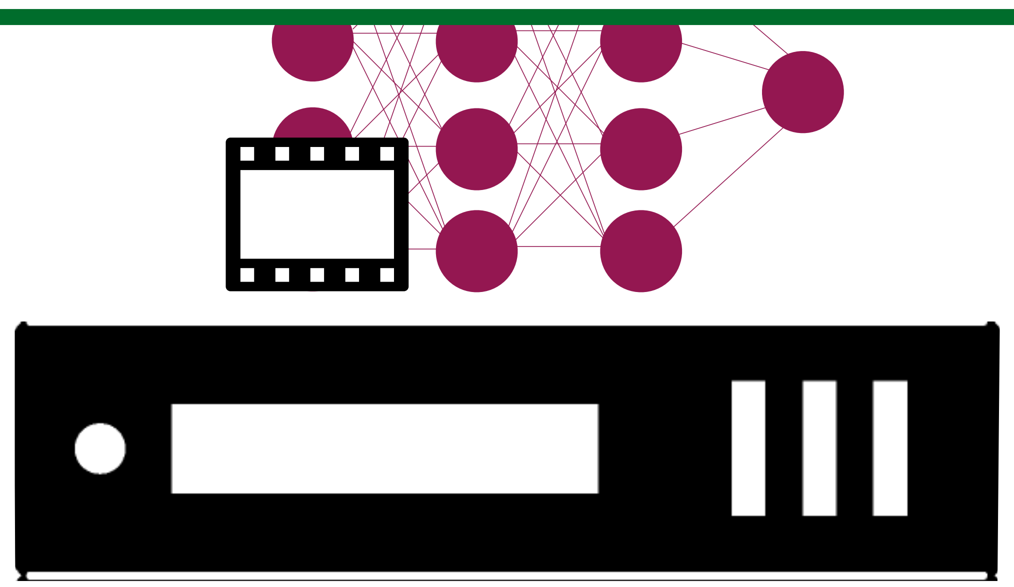
Remaining Swaps are Faster

Benefits

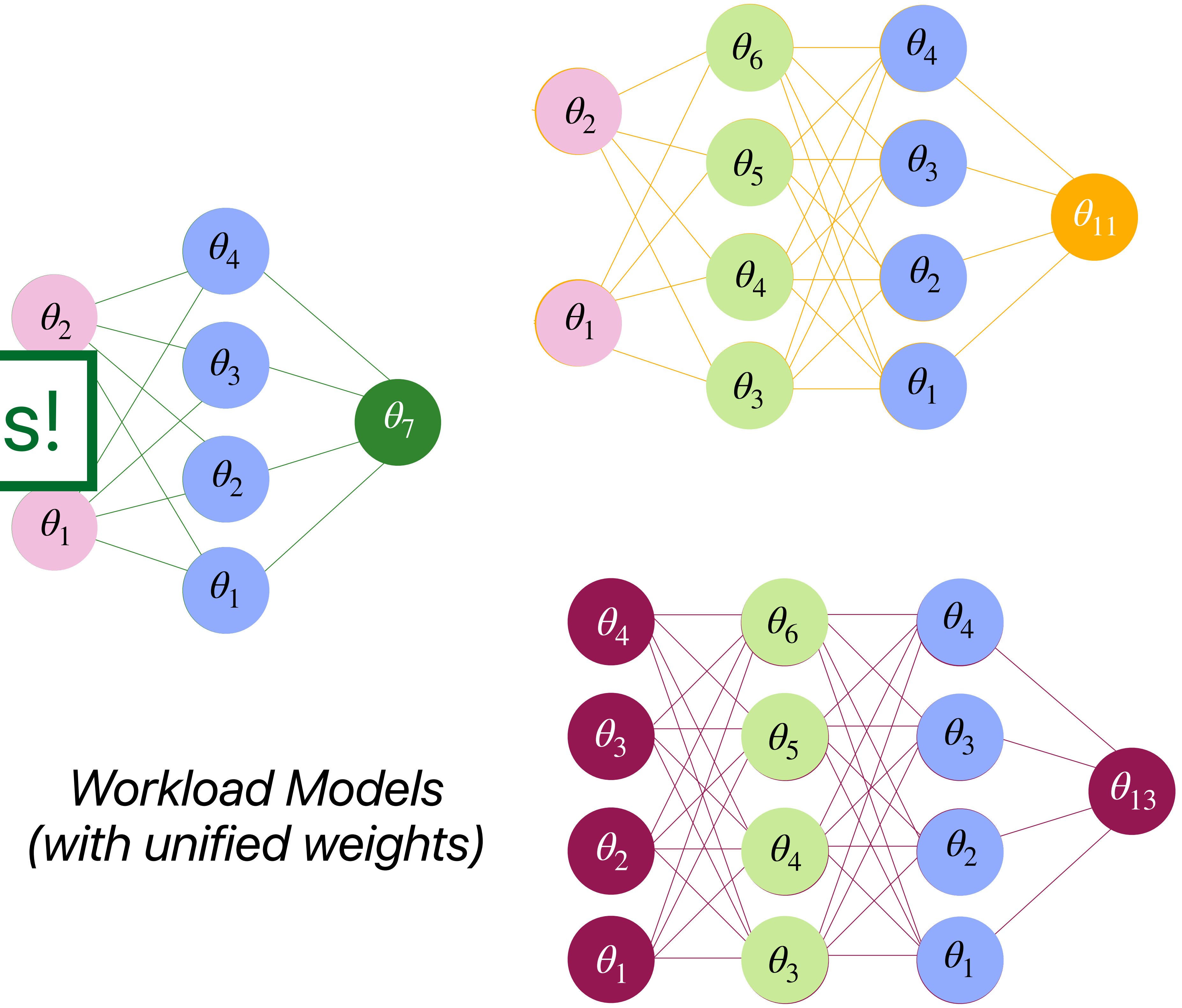
Reduce per-workload memory usage by 17-86%

Process 29-61% more frames!



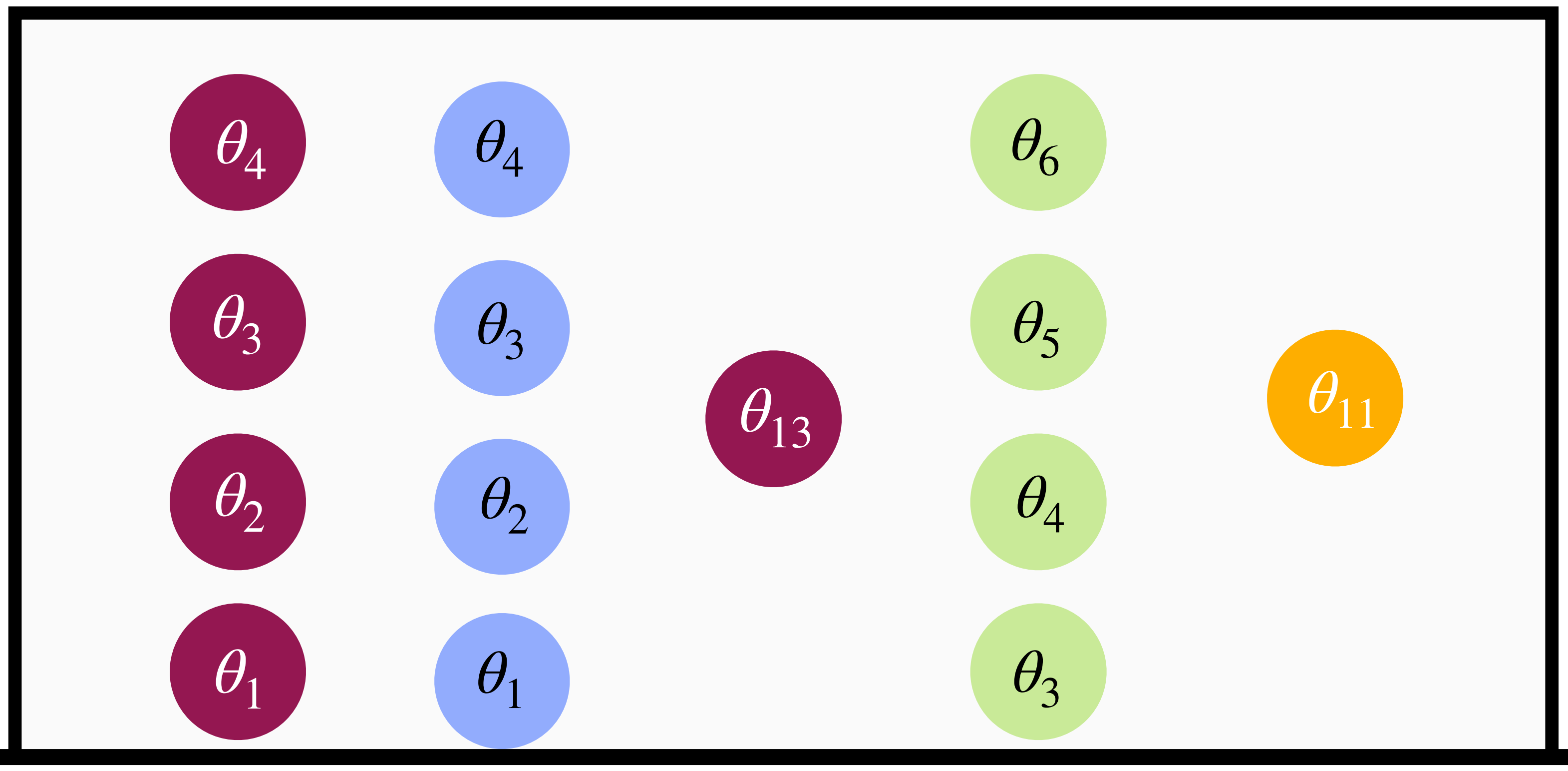


Edge Box



Workload Models (with unified weights)

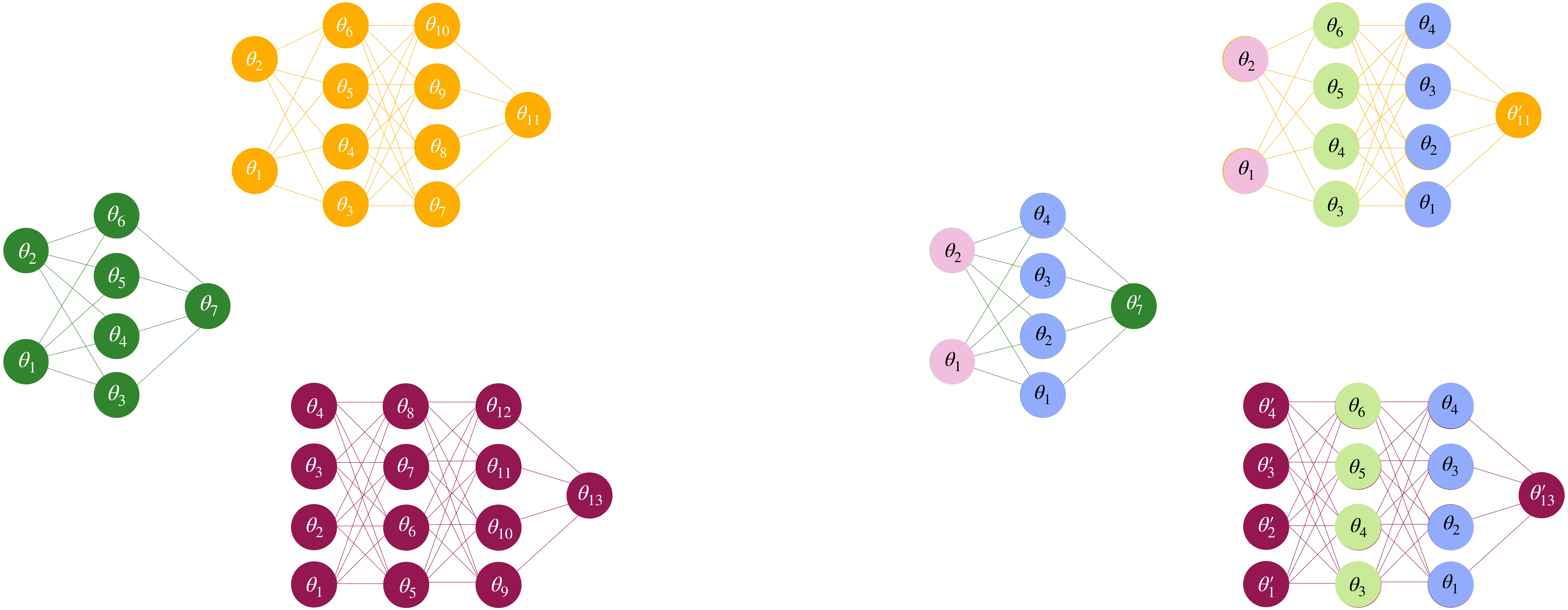
Edge Box GPU Memory



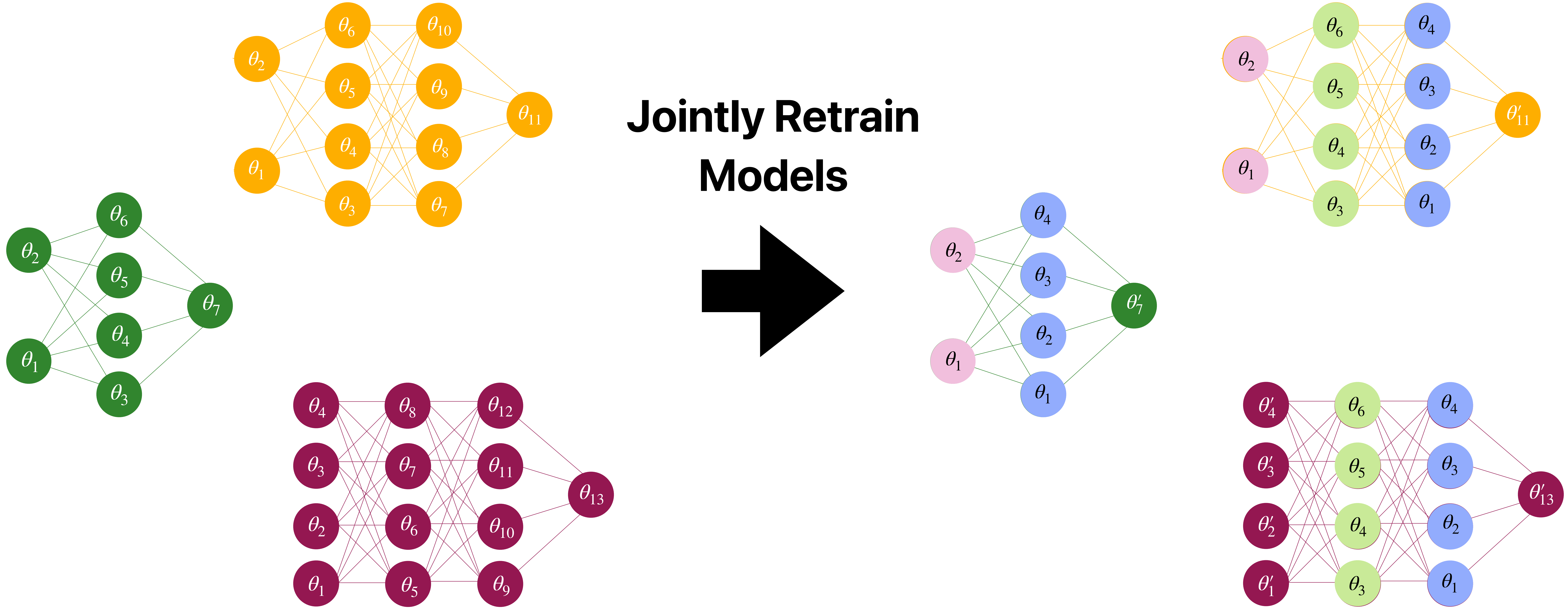
Remaining Swaps are Faster

Fewer Number of Swaps

Model Merging



Model Merging



Model Merging Challenges

Model Merging Challenges

- The more layers you merge, the lower the accuracy achieved during retraining

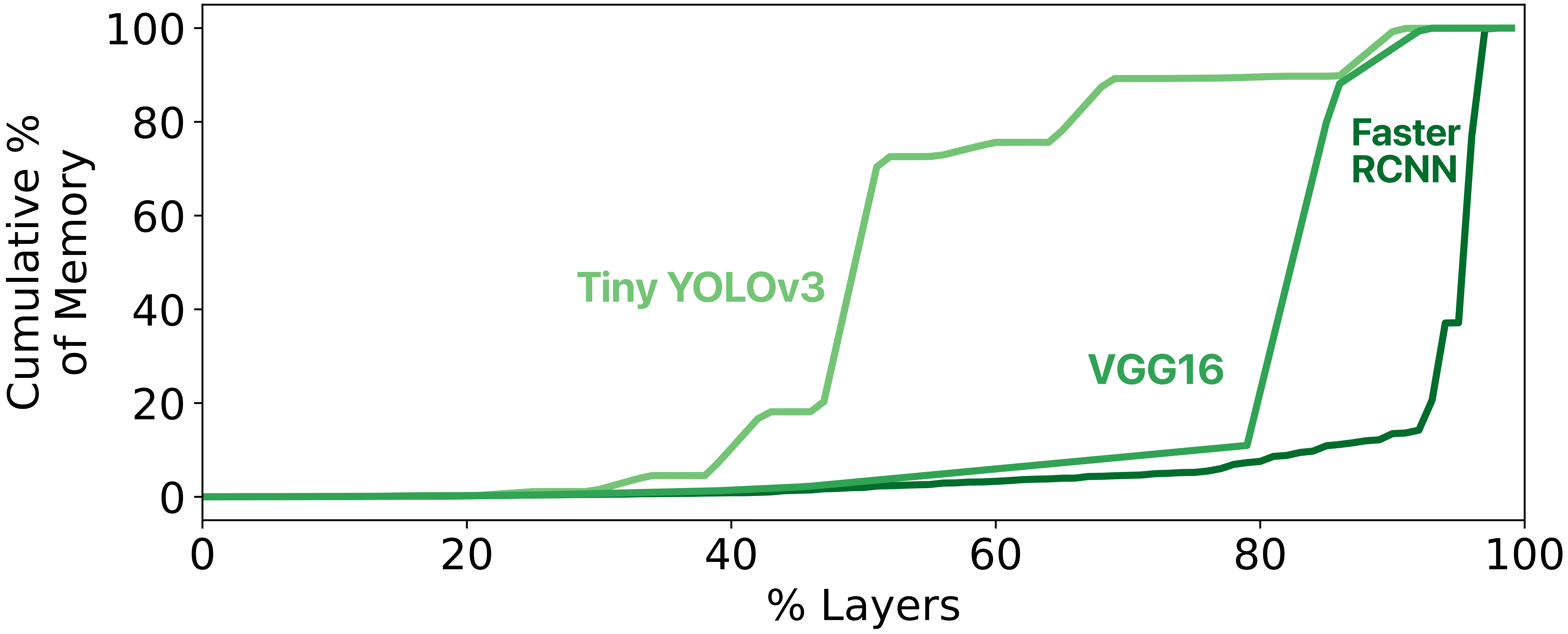
Model Merging Challenges

- The more layers you merge, the lower the accuracy achieved during retraining
- Difficult to predict precisely how many layers will be mergeable before accuracy violations occur

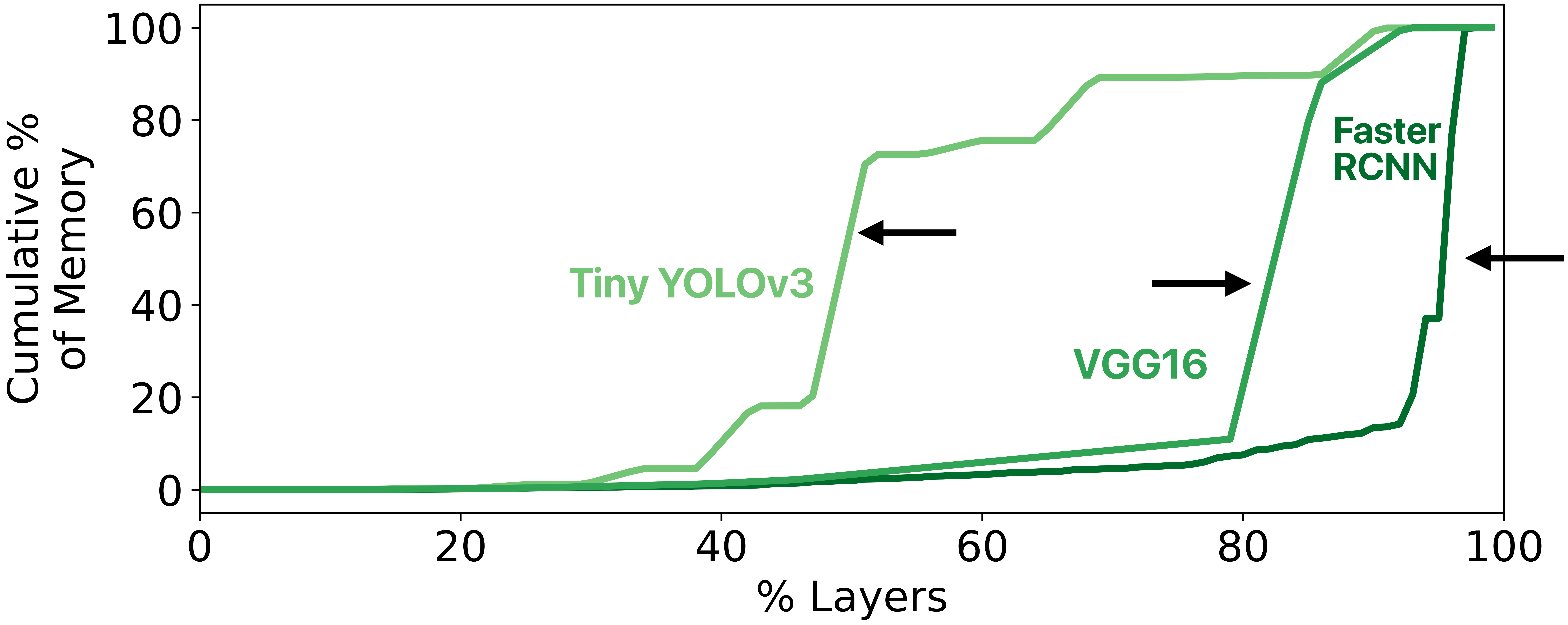
Model Merging Challenges

- The more layers you merge, the lower the accuracy achieved during retraining
- Difficult to predict precisely how many layers will be mergeable before accuracy violations occur
- Each instance of retraining is costly

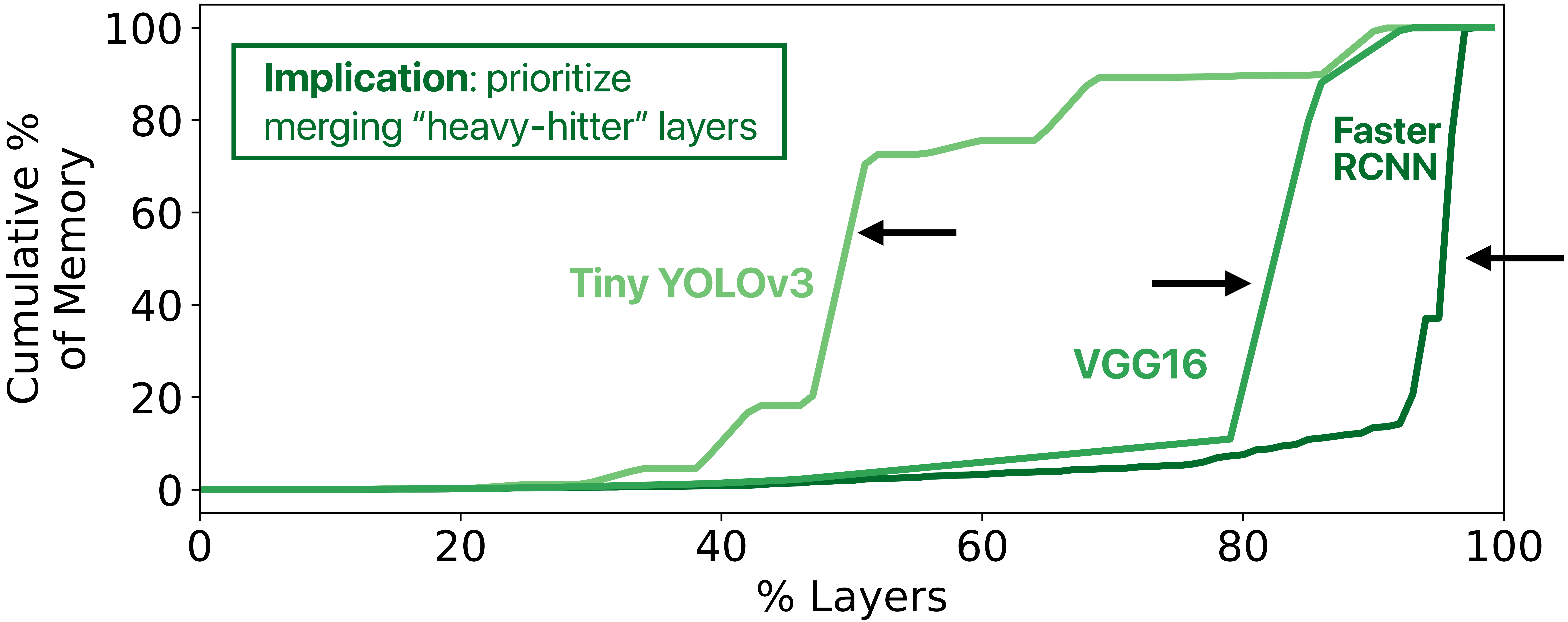
Observation 1: Presence of “Heavy-Hitter” Layers



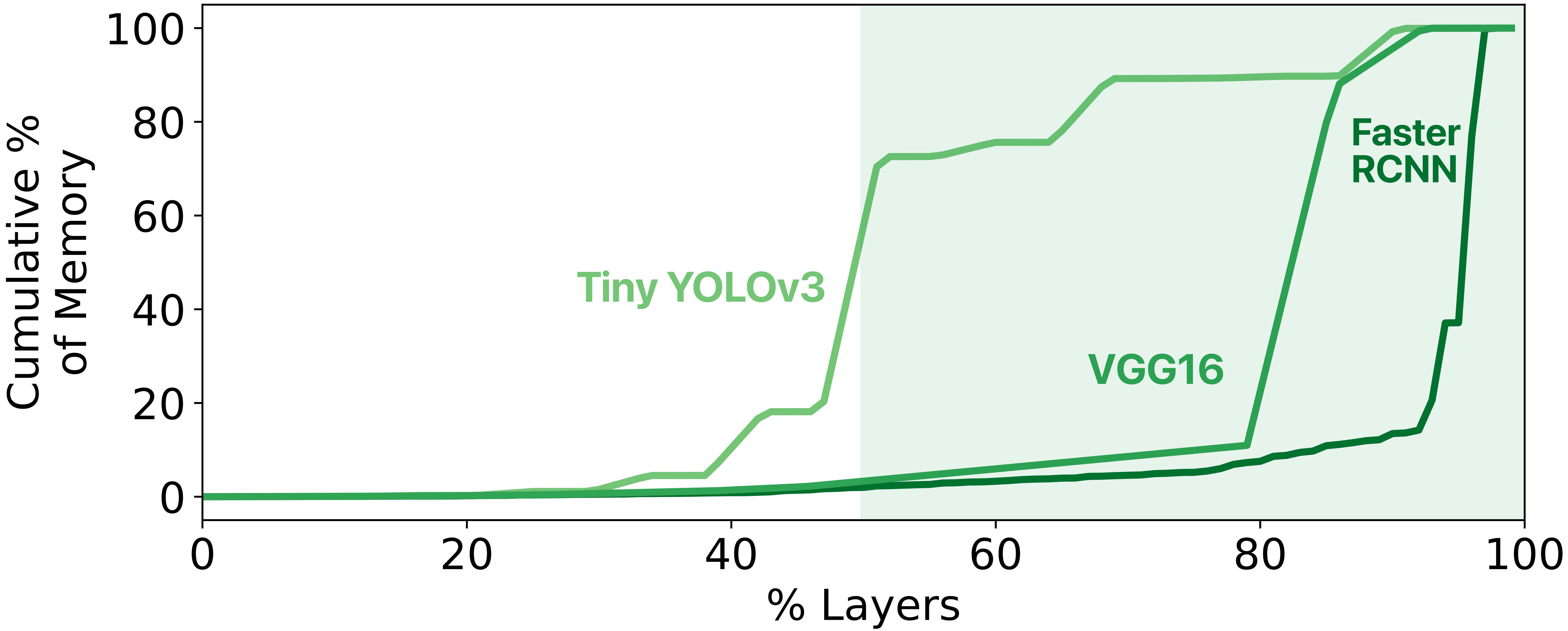
Observation 1: Presence of "Heavy-Hitter" Layers



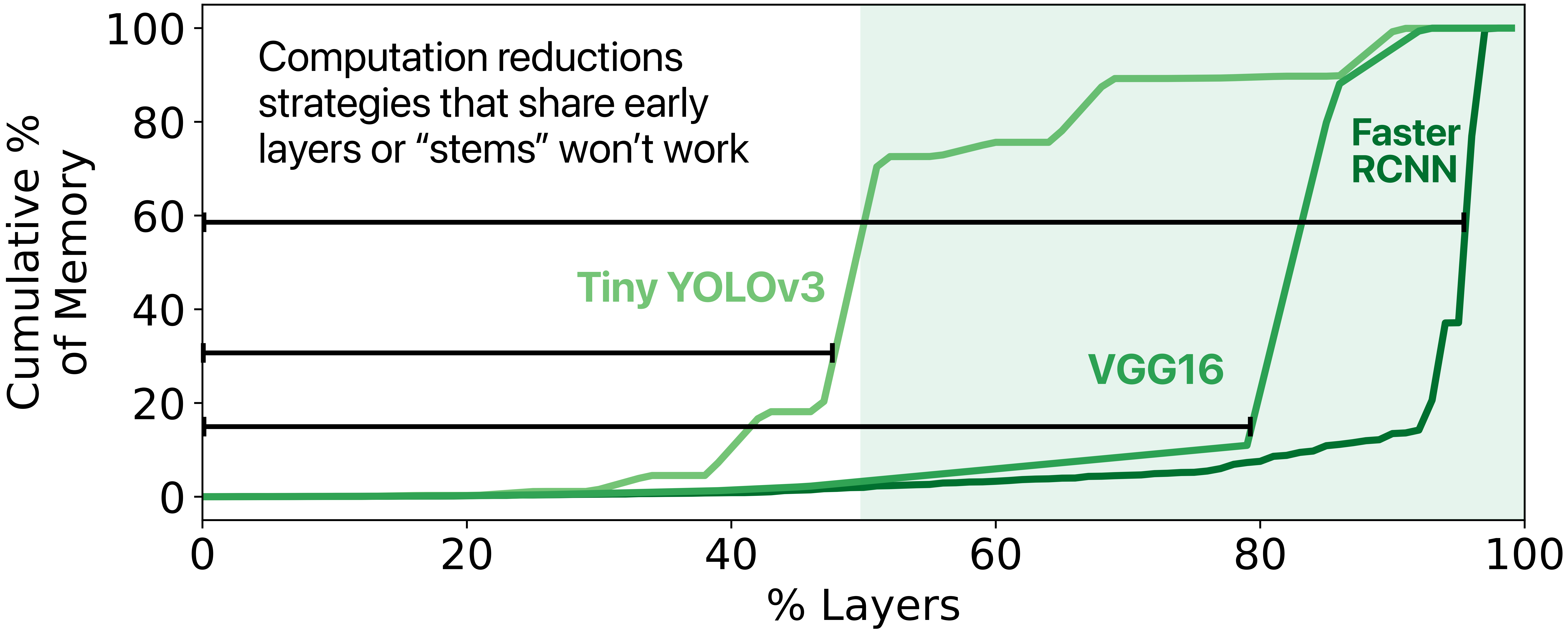
Observation 1: Presence of “Heavy-Hitter” Layers



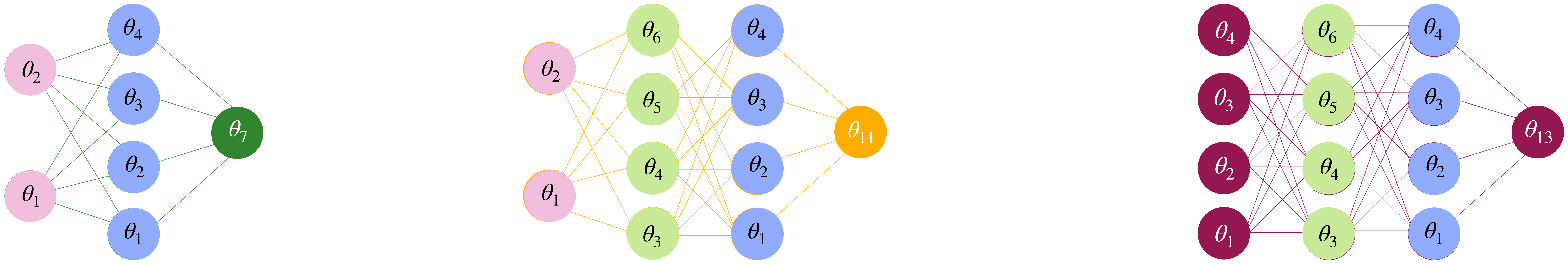
Observation 1: Presence of "Heavy-Hitter" Layers



Observation 1: Presence of "Heavy-Hitter" Layers

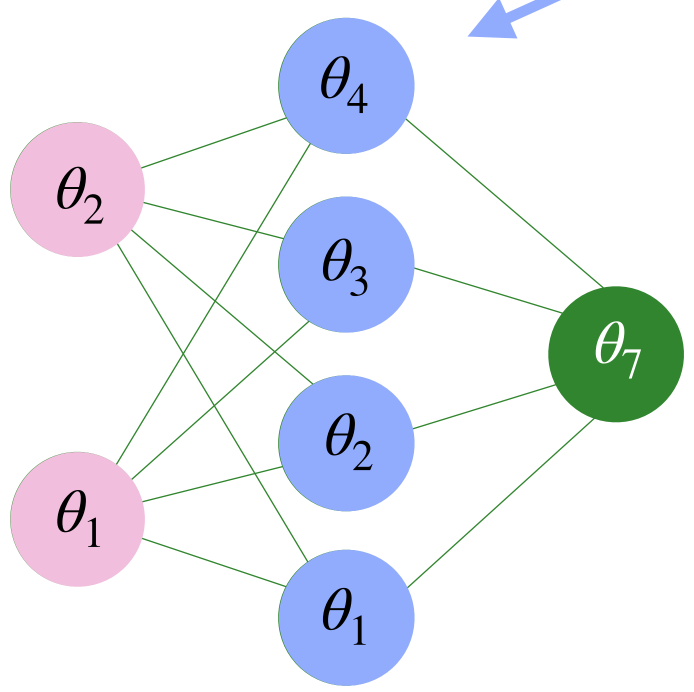


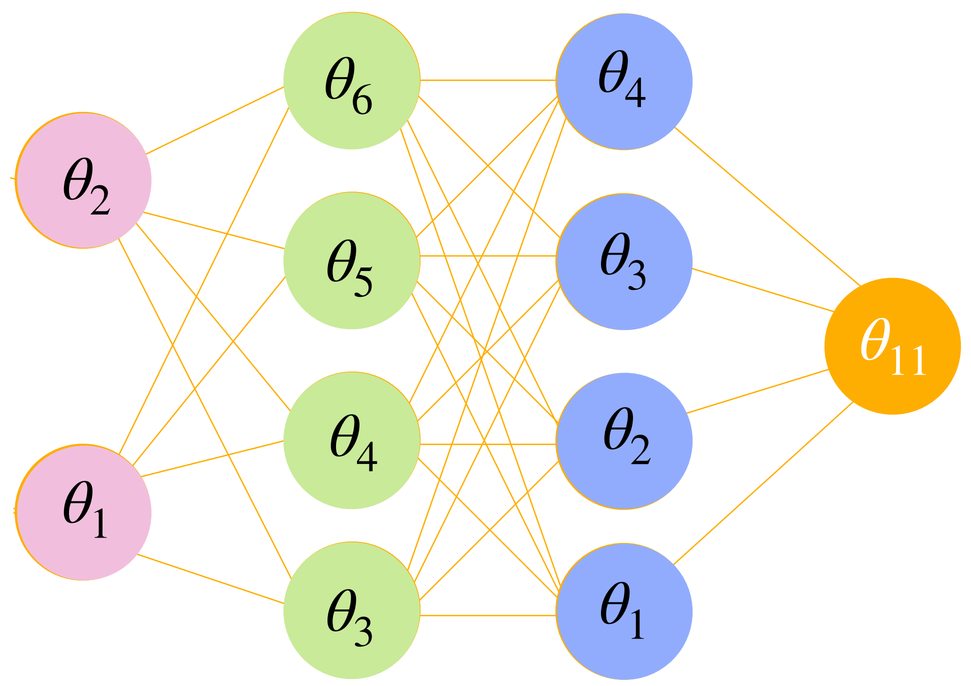
Observation 2: Per-layer merging decisions can be made in isolation

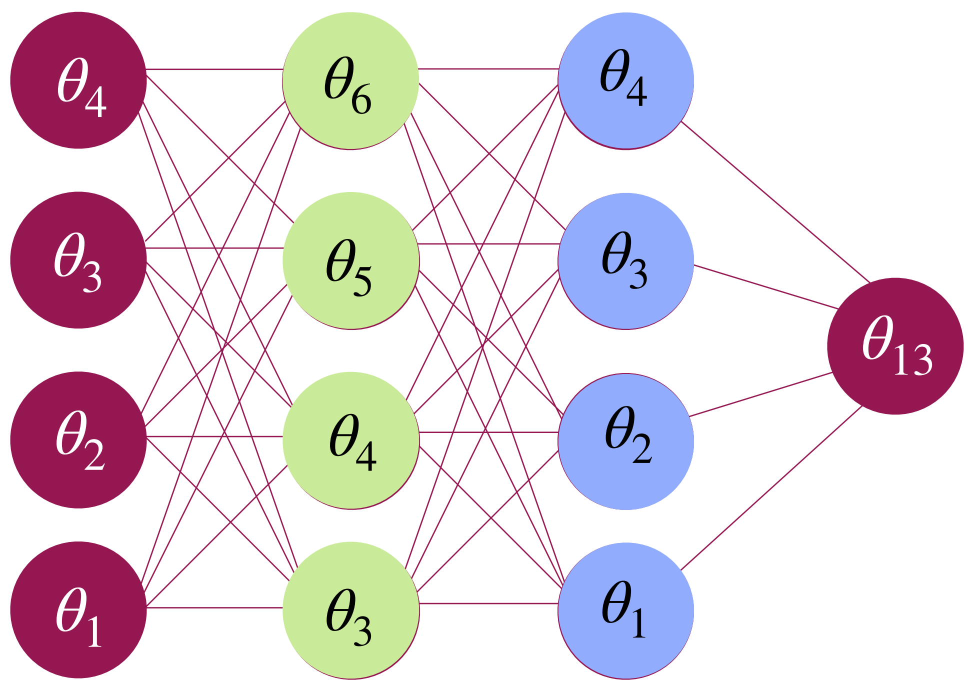


Observation 2: Per-layer merging decisions can be made in isolation

Ability to successfully retrain (i.e., preserve accuracy) when merging these layers...

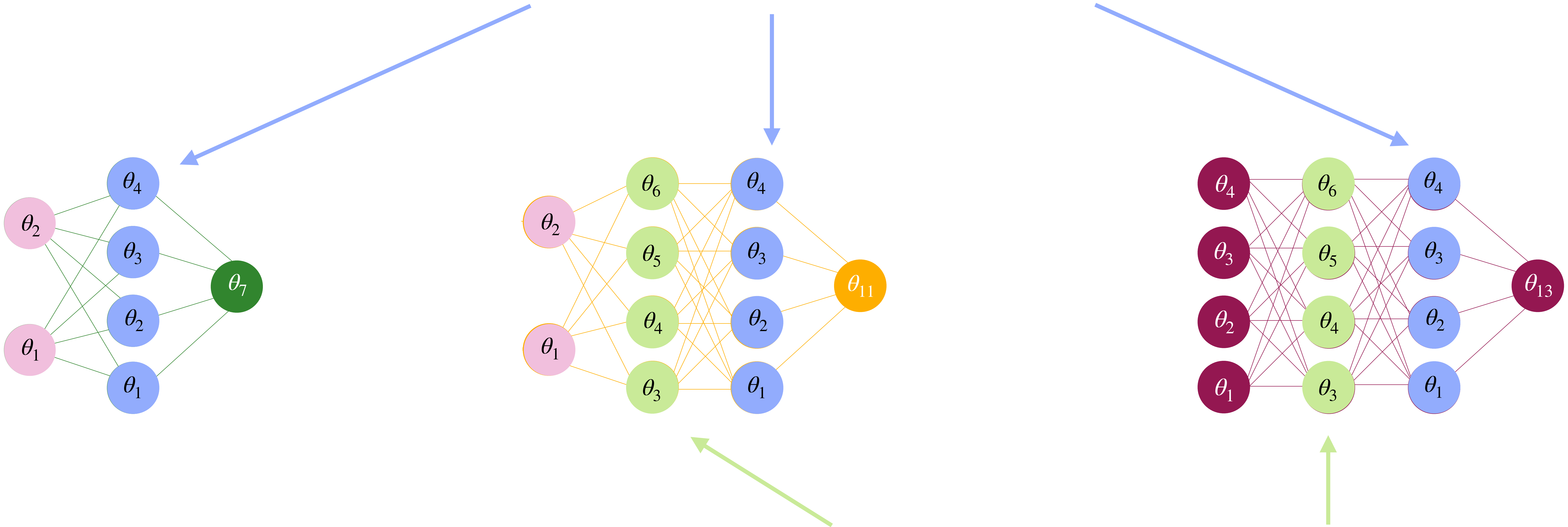






Observation 2: Per-layer merging decisions can be made in isolation

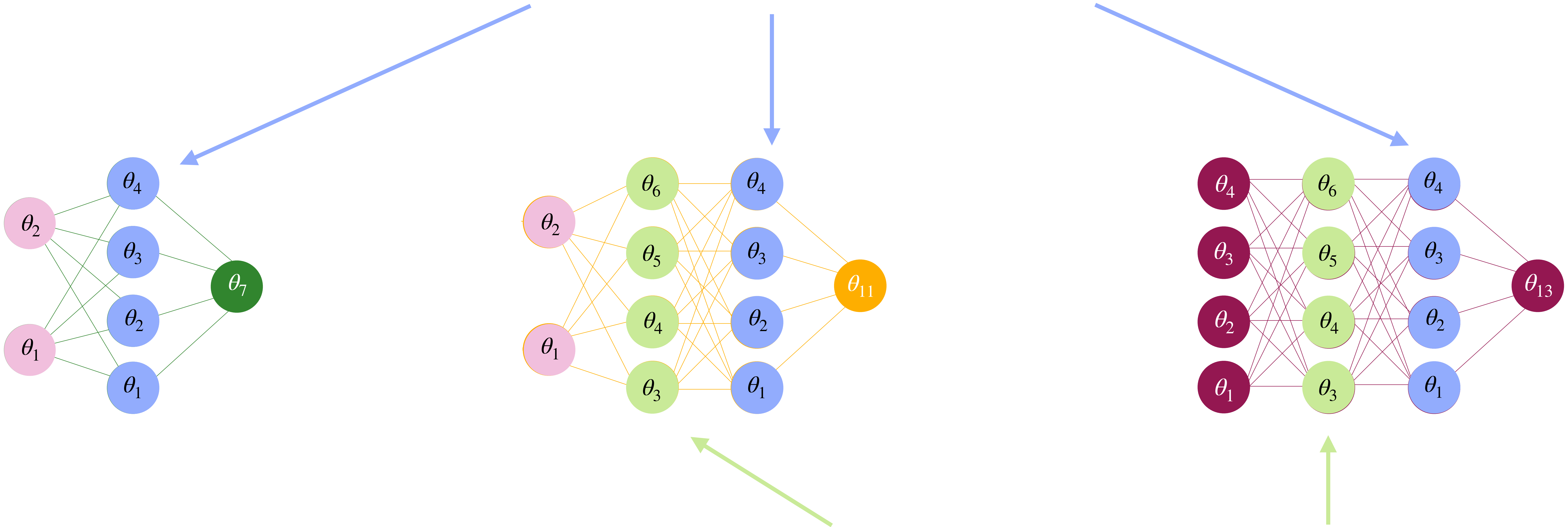
Ability to successfully retrain (i.e., preserve accuracy) when merging these layers…



…does not depend on also merging these layers.

Observation 2: Per-layer merging decisions can be made in isolation

Ability to successfully retrain (i.e., preserve accuracy) when merging these layers…



…does not depend on also merging these layers.

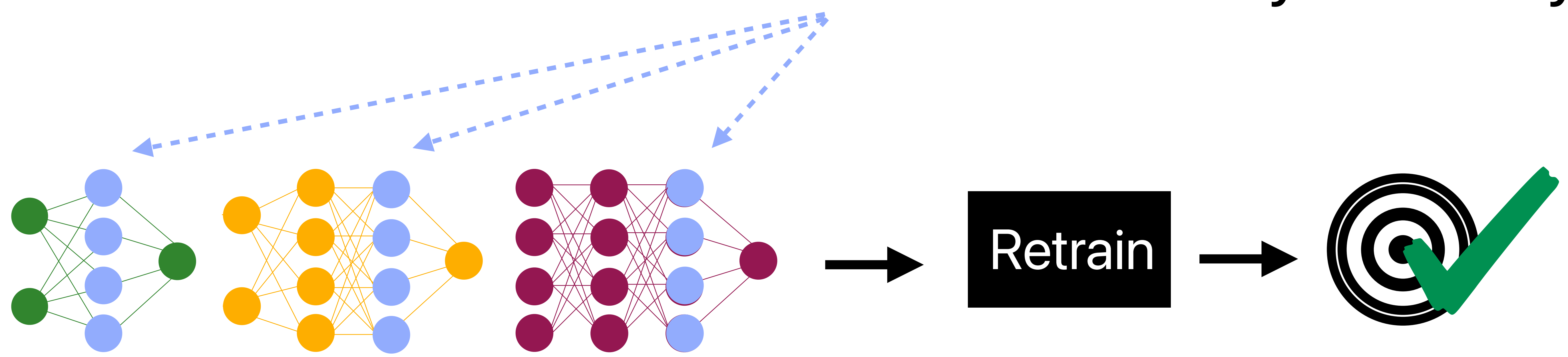
Implication: we can try merging layers independently one at at time

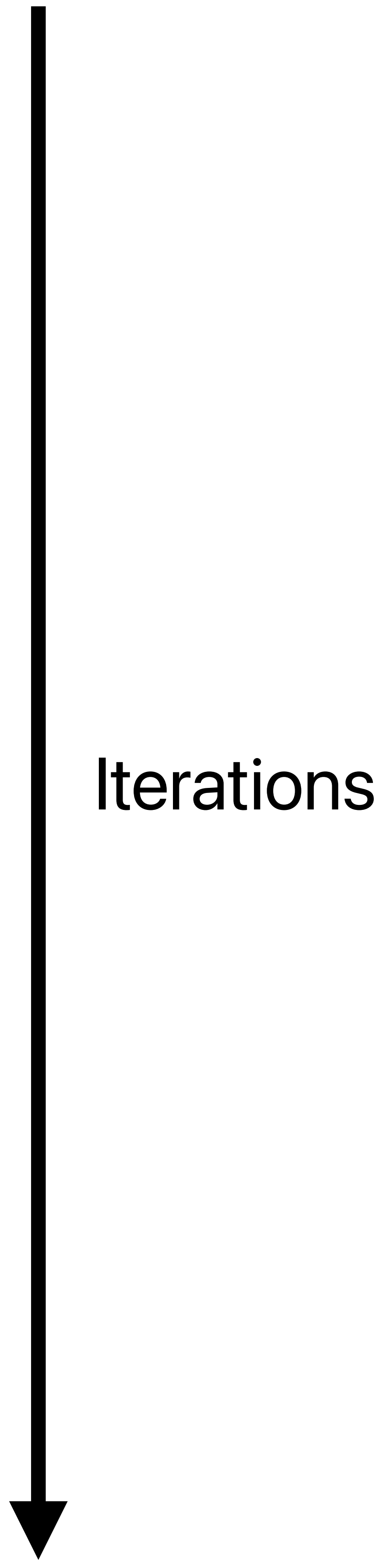
Model Merging Strategy

Model Merging Strategy

Start with heavy-hitter layers

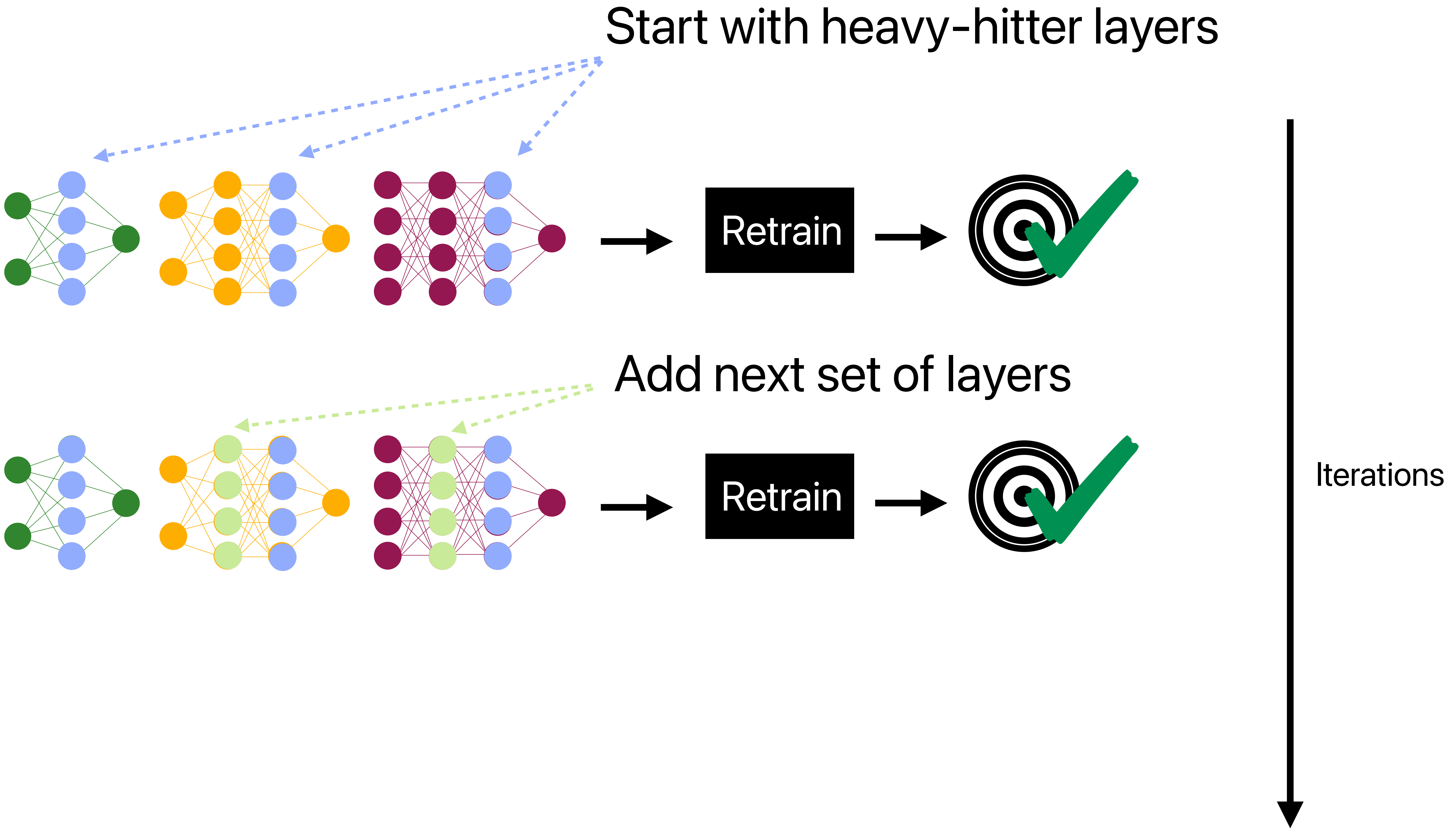
Merge one
additional layer
per iteration





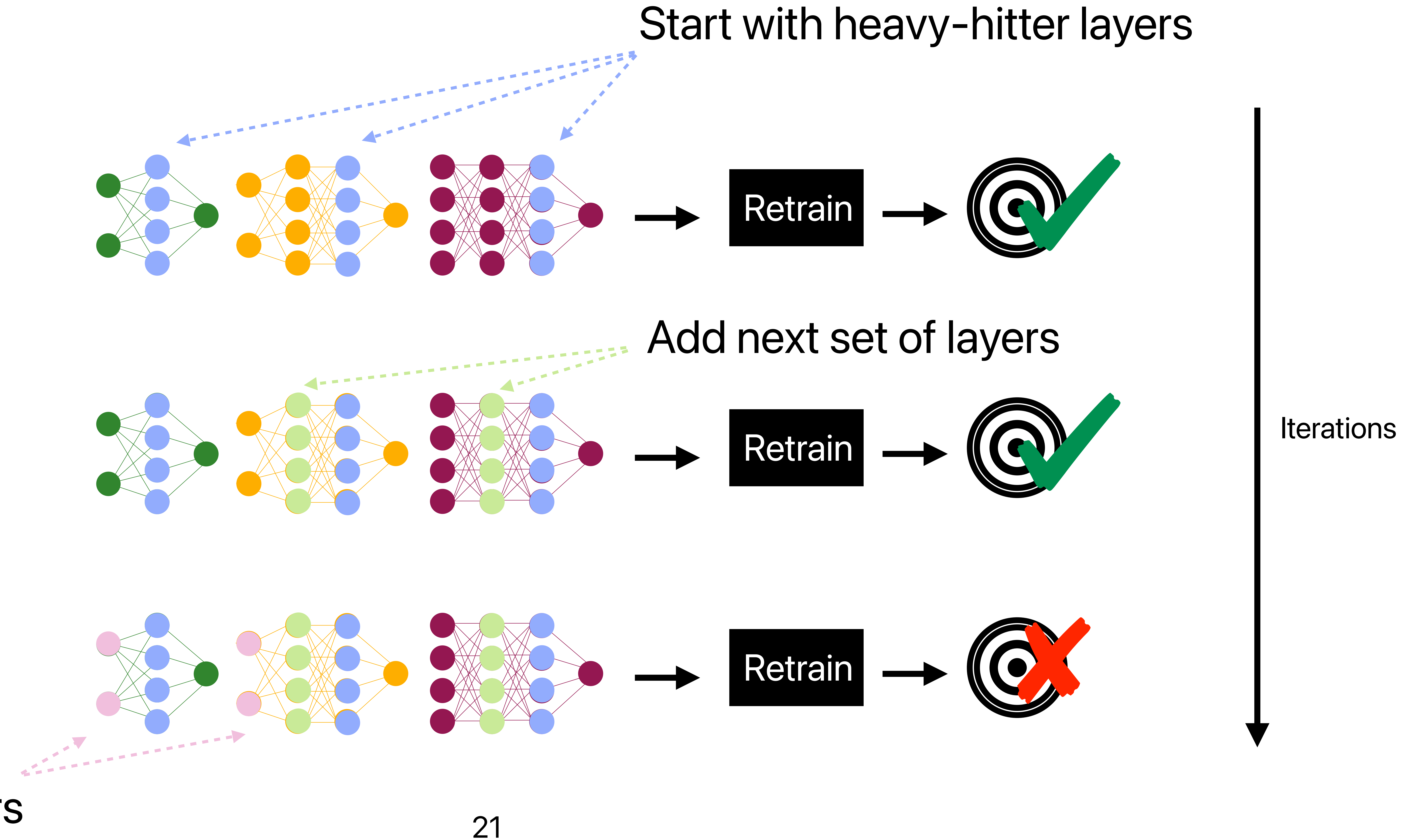
Model Merging Strategy

Merge one additional layer per iteration

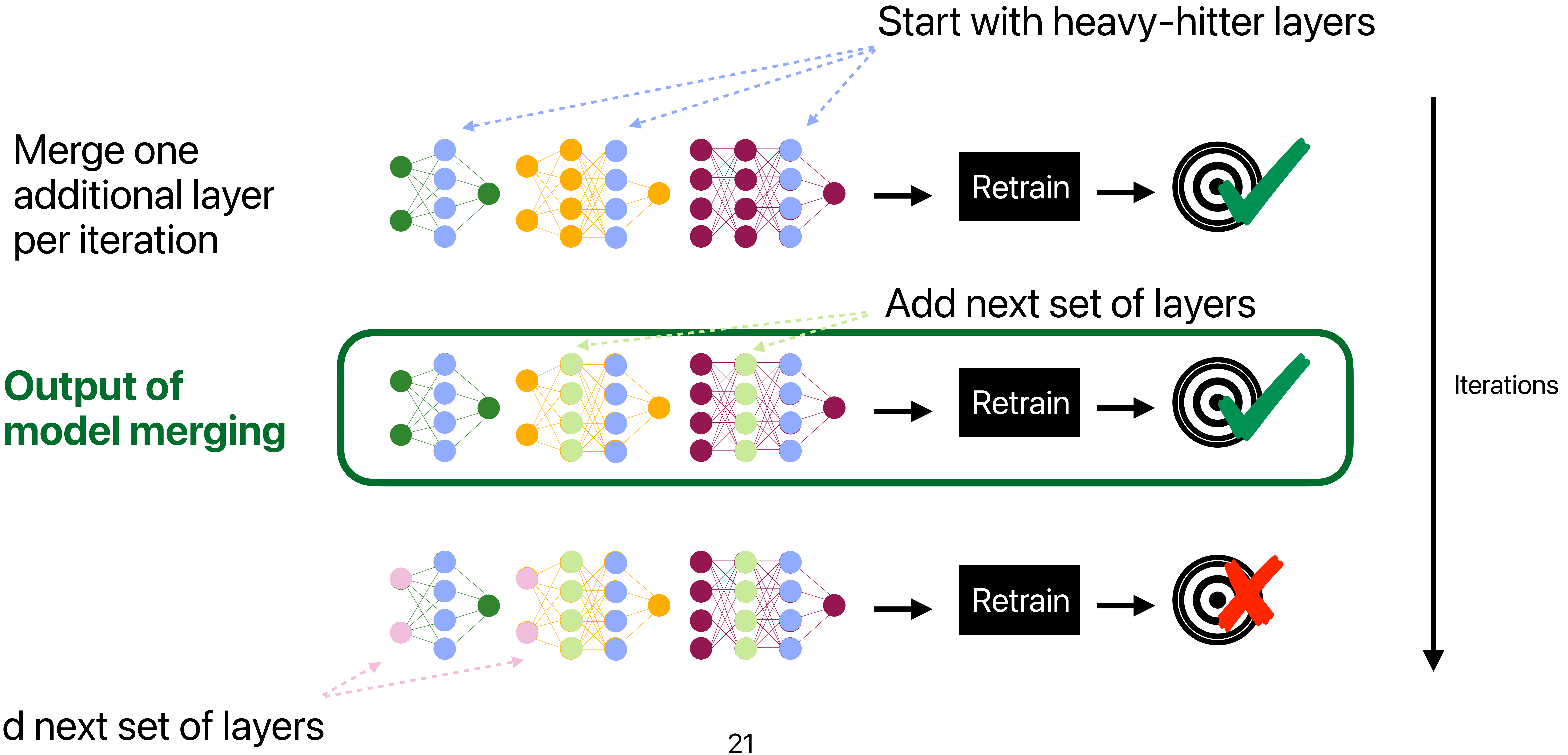


Model Merging Strategy

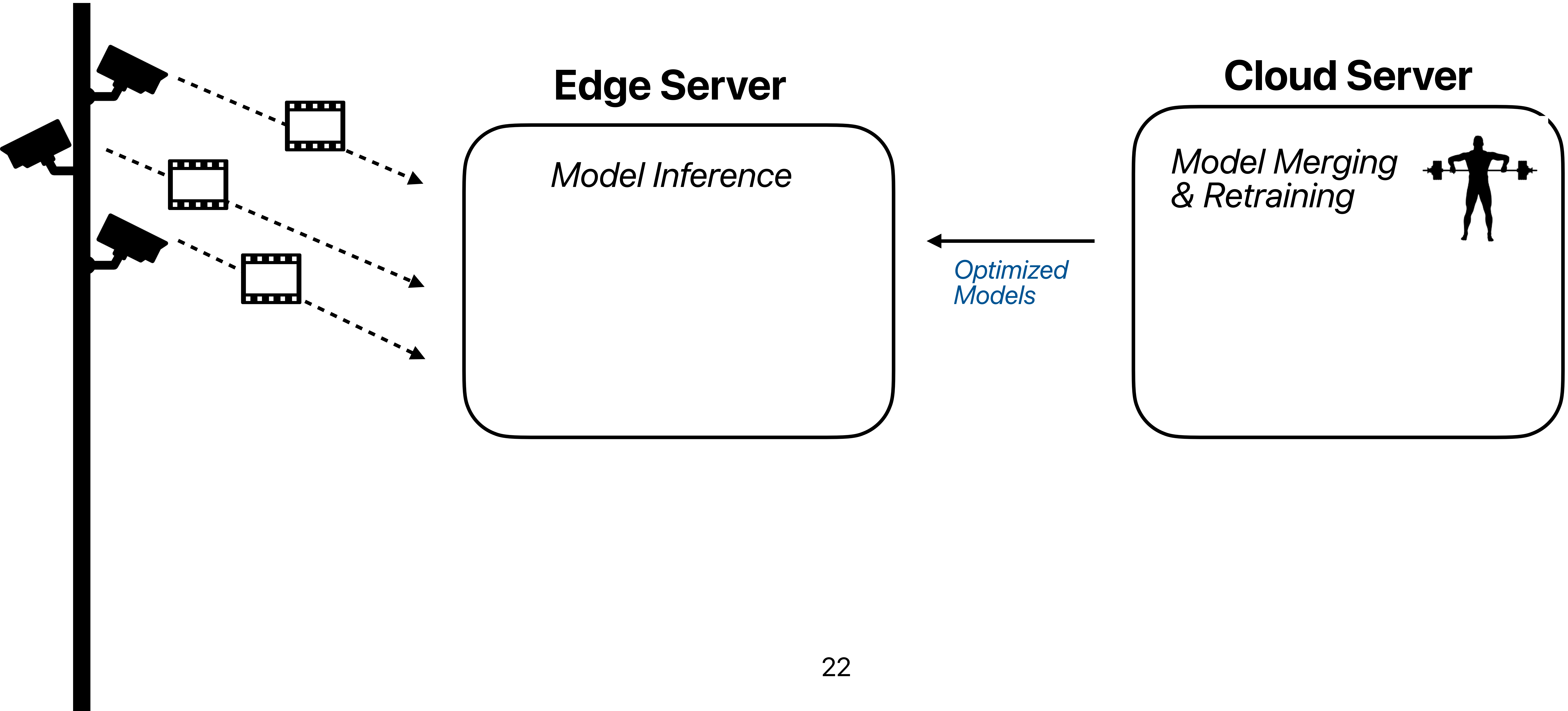
Merge one
additional layer
per iteration



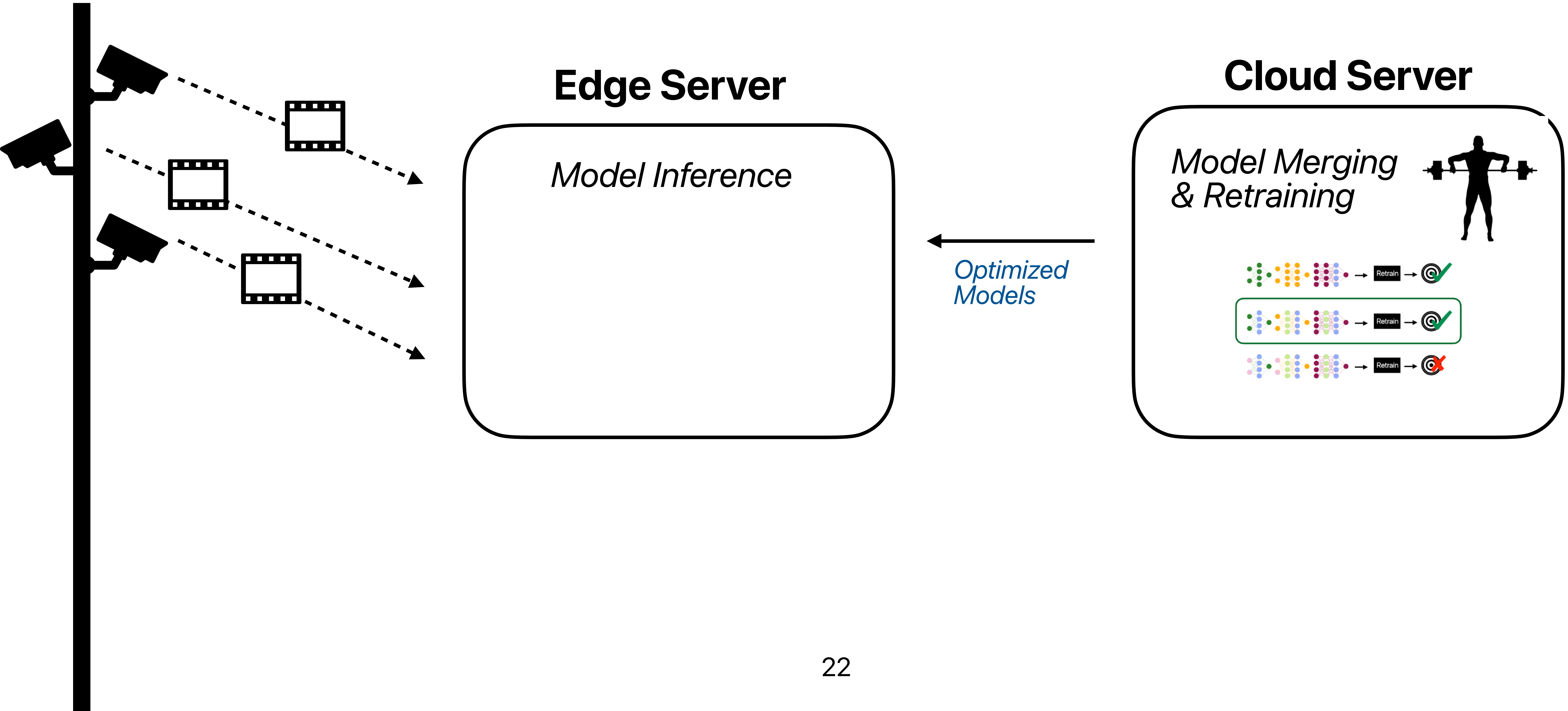
Model Merging Strategy



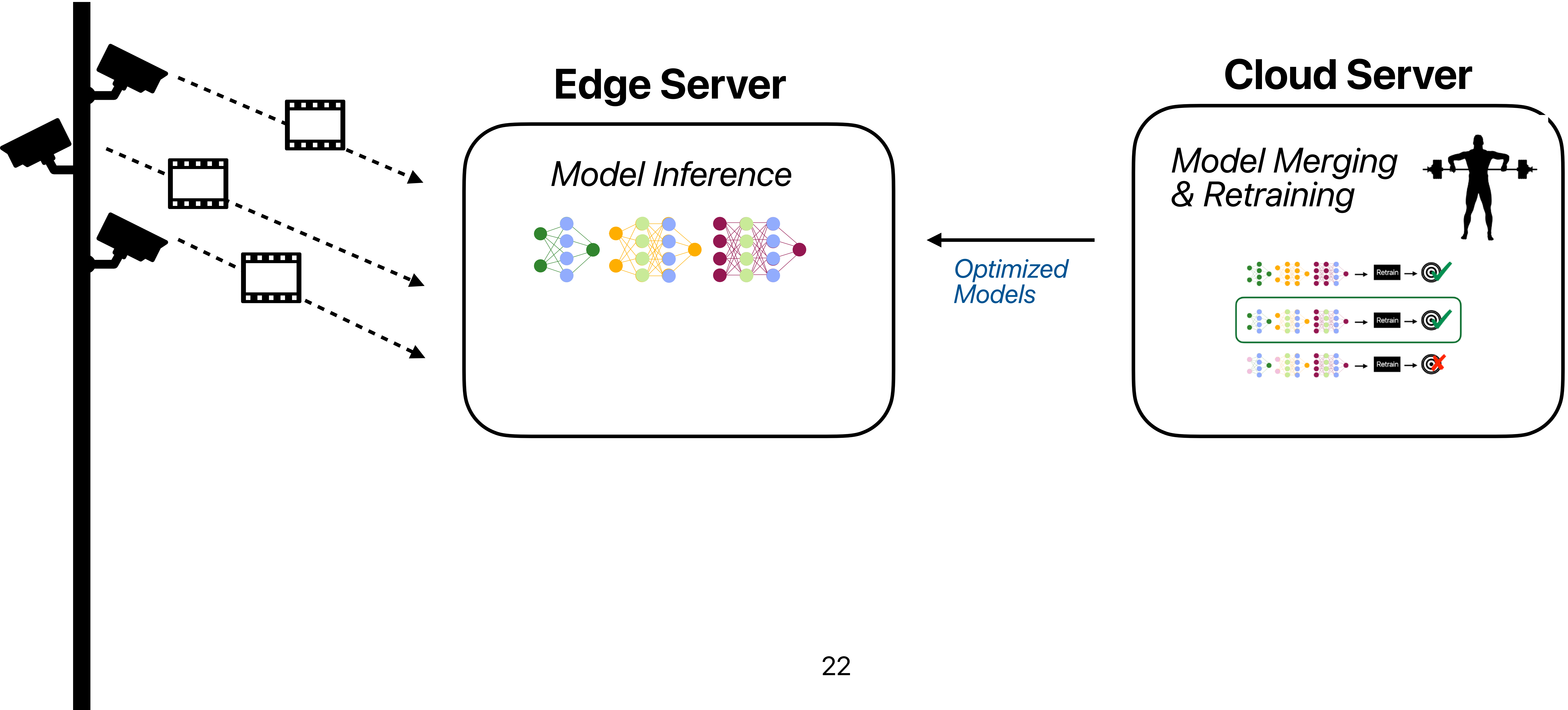
System Design



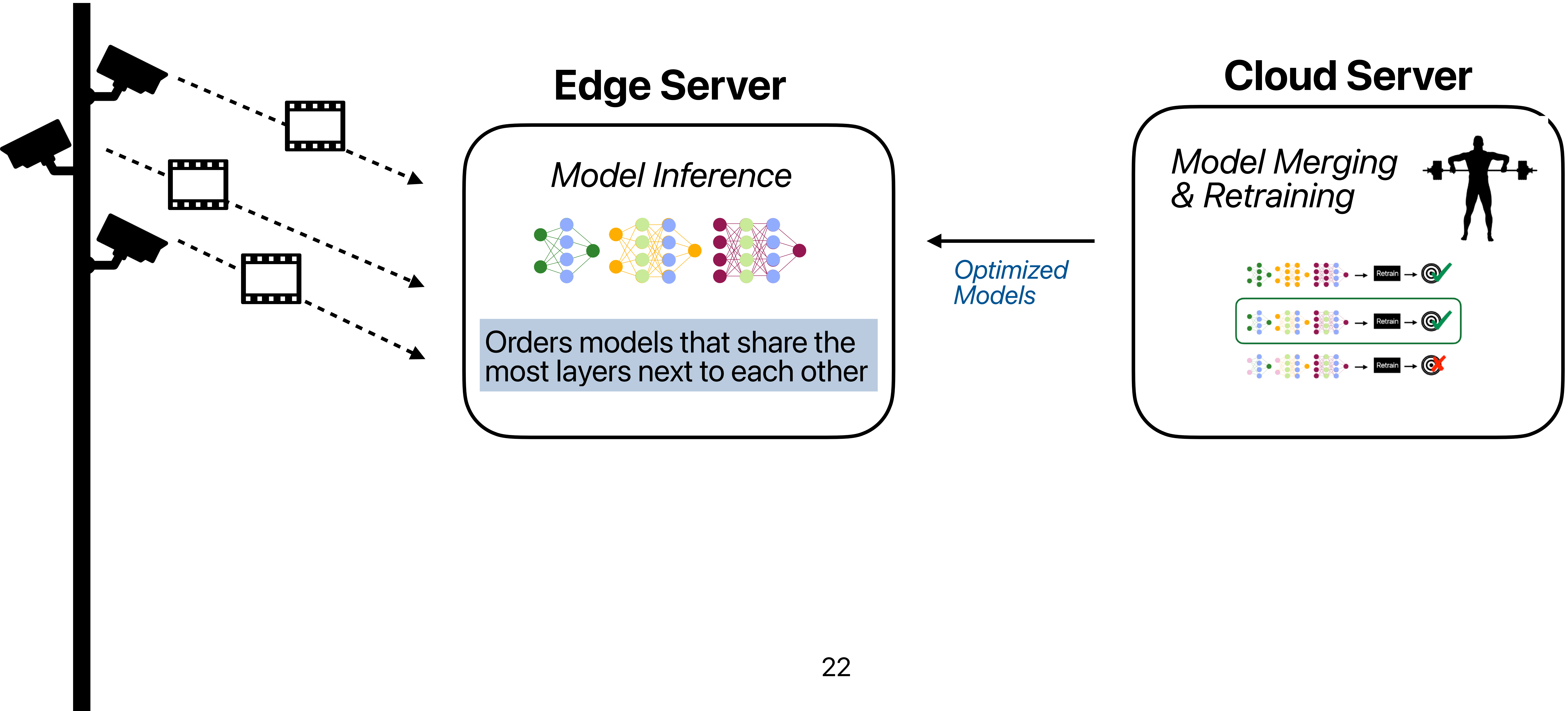
System Design



System Design



System Design



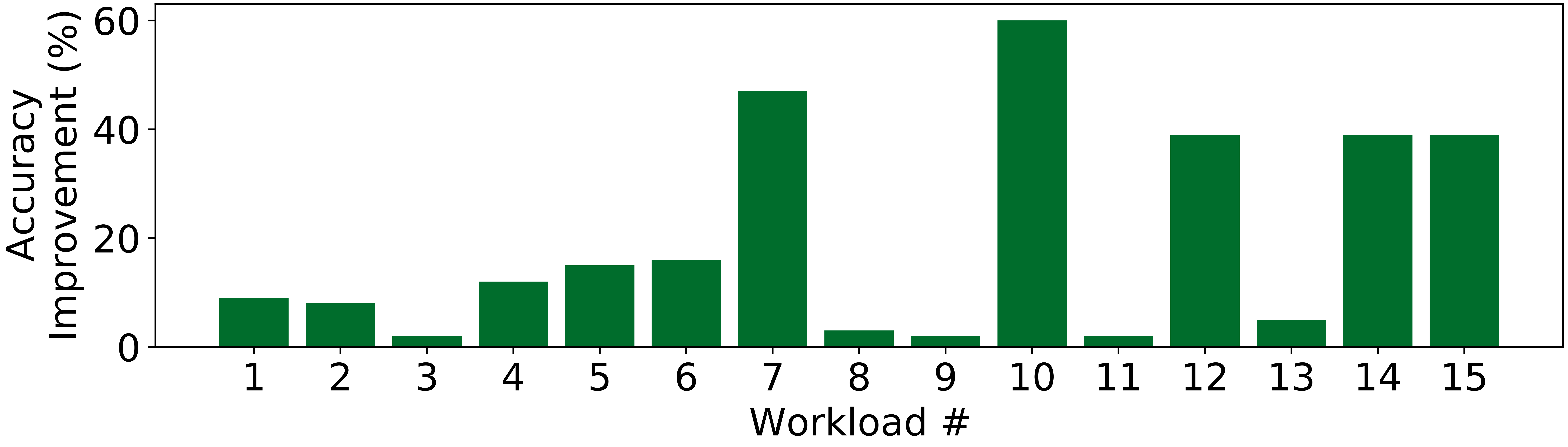
Gemel Evaluation

- Accuracy Improvements
- GPU Memory Savings
- Varying FPS, Accuracy, SLA
- Comparison to Stem-Sharing Approach
- Incremental Memory Savings
- Merging Heuristic Ablation
- Microbenchmarks
- Generalizability

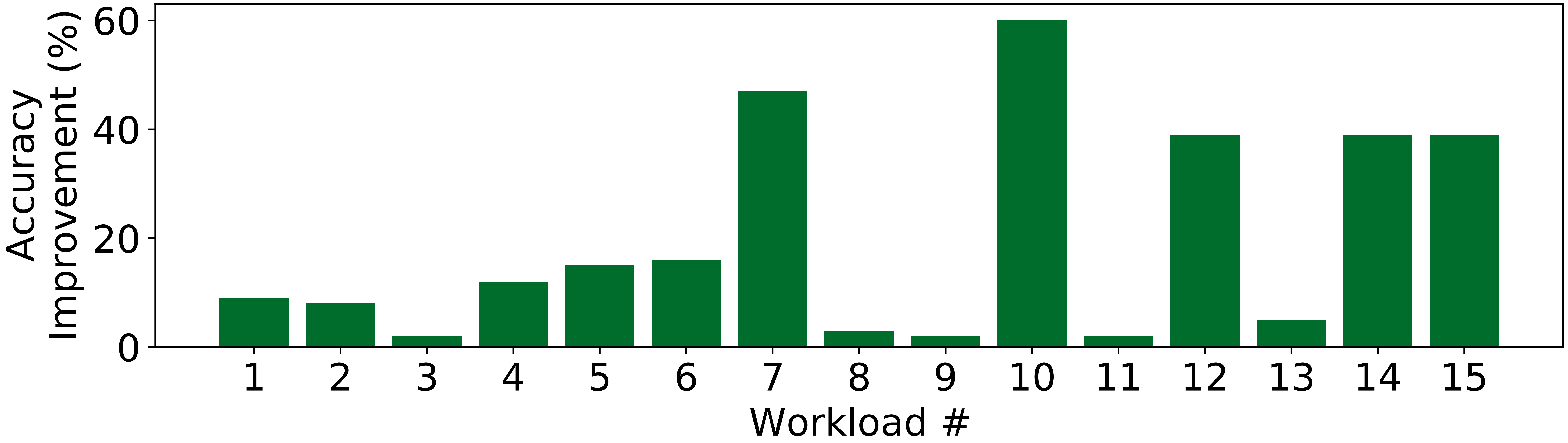
Gemel Evaluation

- Accuracy Improvements
- GPU Memory Savings
- Varying FPS, Accuracy, SLA
- Comparison to Stem-Sharing Approach
- Incremental Memory Savings
- Merging Heuristic Ablation
- Microbenchmarks
- Generalizability

Improvement in Query Accuracy

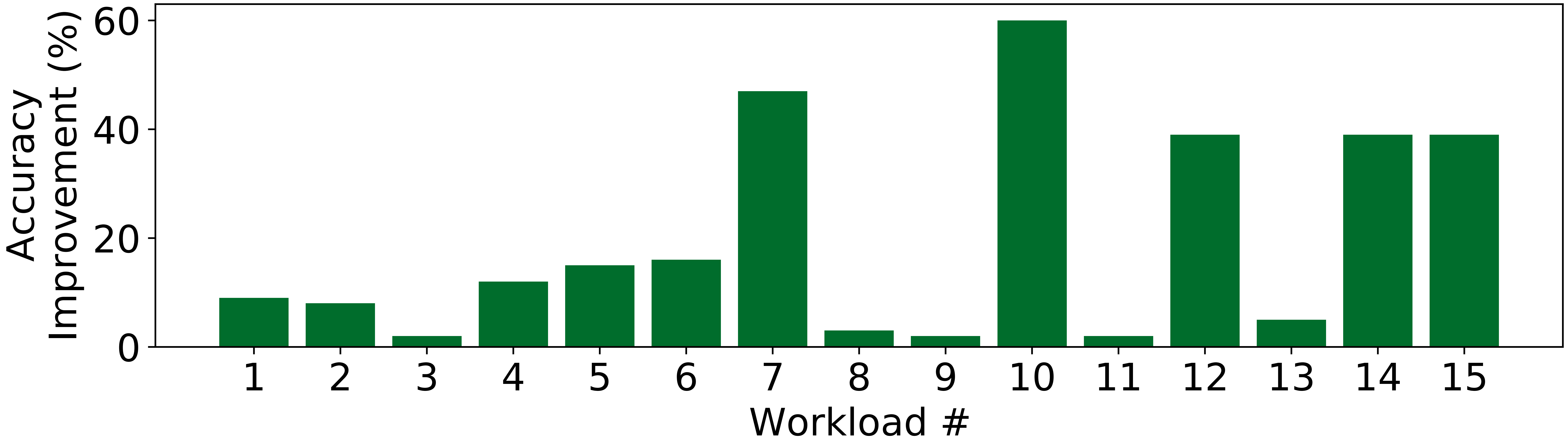


Improvement in Query Accuracy



Gemel improves per-workload query accuracy by 2-60%.

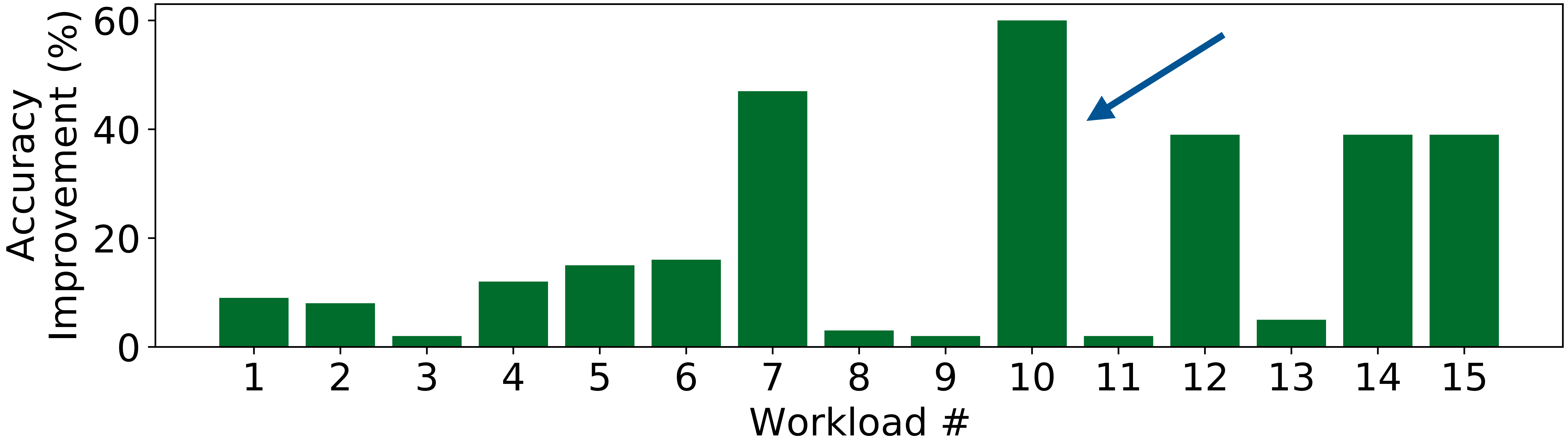
Improvement in Query Accuracy



Gemel improves per-workload query accuracy by 2-60%.

Reduce time blocked on swapping delays by 17-84% → process 13-44% more frames

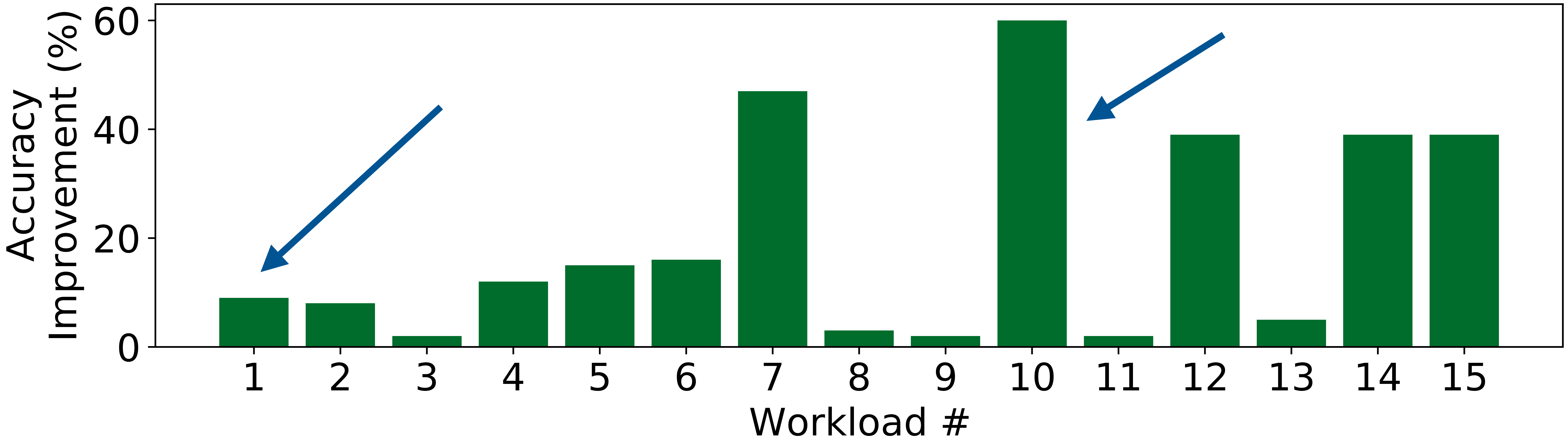
Improvement in Query Accuracy



Gemel improves per-workload query accuracy by 2-60%.

Reduce time blocked on swapping delays by 17-84% → process 13-44% more frames

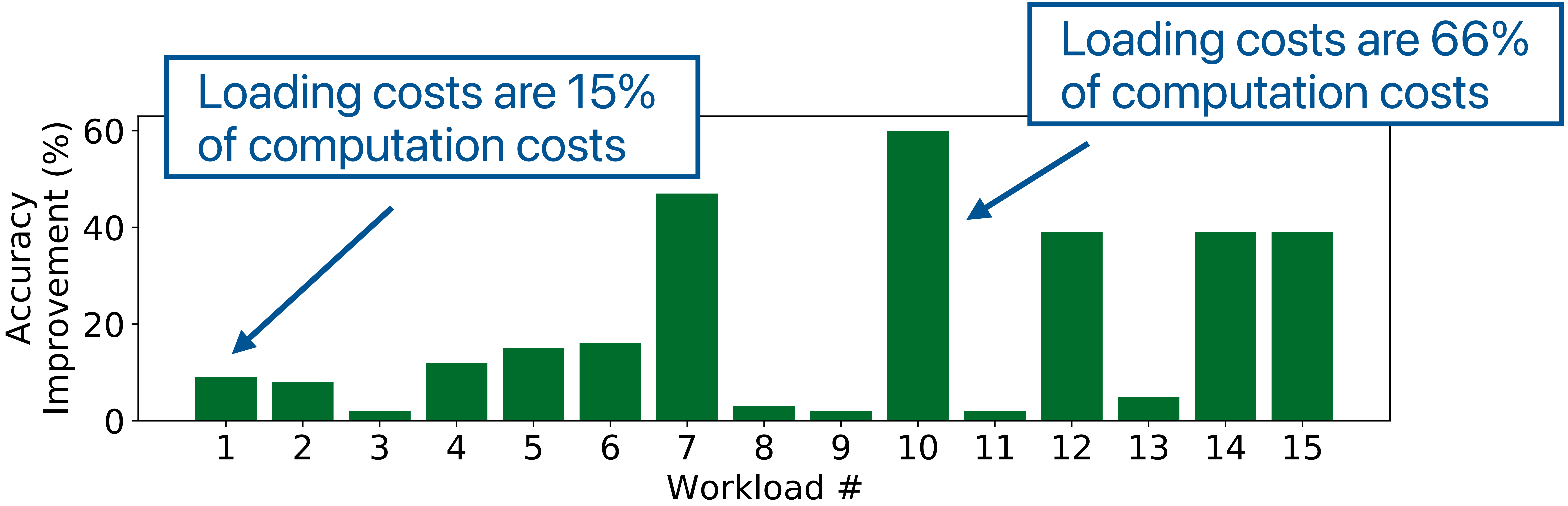
Improvement in Query Accuracy



Gemel improves per-workload query accuracy by 2-60%.

Reduce time blocked on swapping delays by 17-84% → process 13-44% more frames

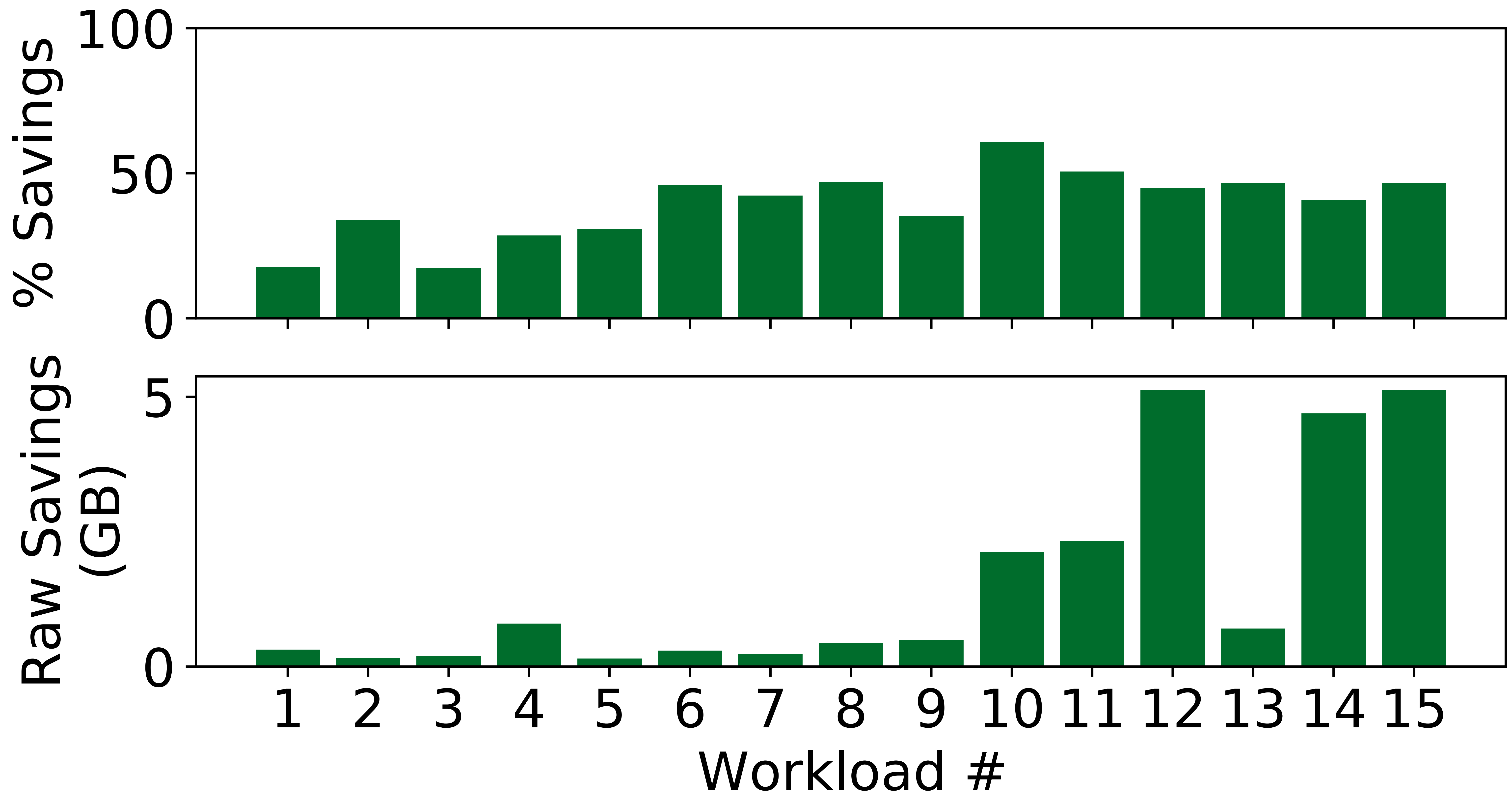
Improvement in Query Accuracy



Gemel improves per-workload query accuracy by 2-60%.

Reduce time blocked on swapping delays by 17-84% → process 13-44% more frames

Memory Savings Achieved



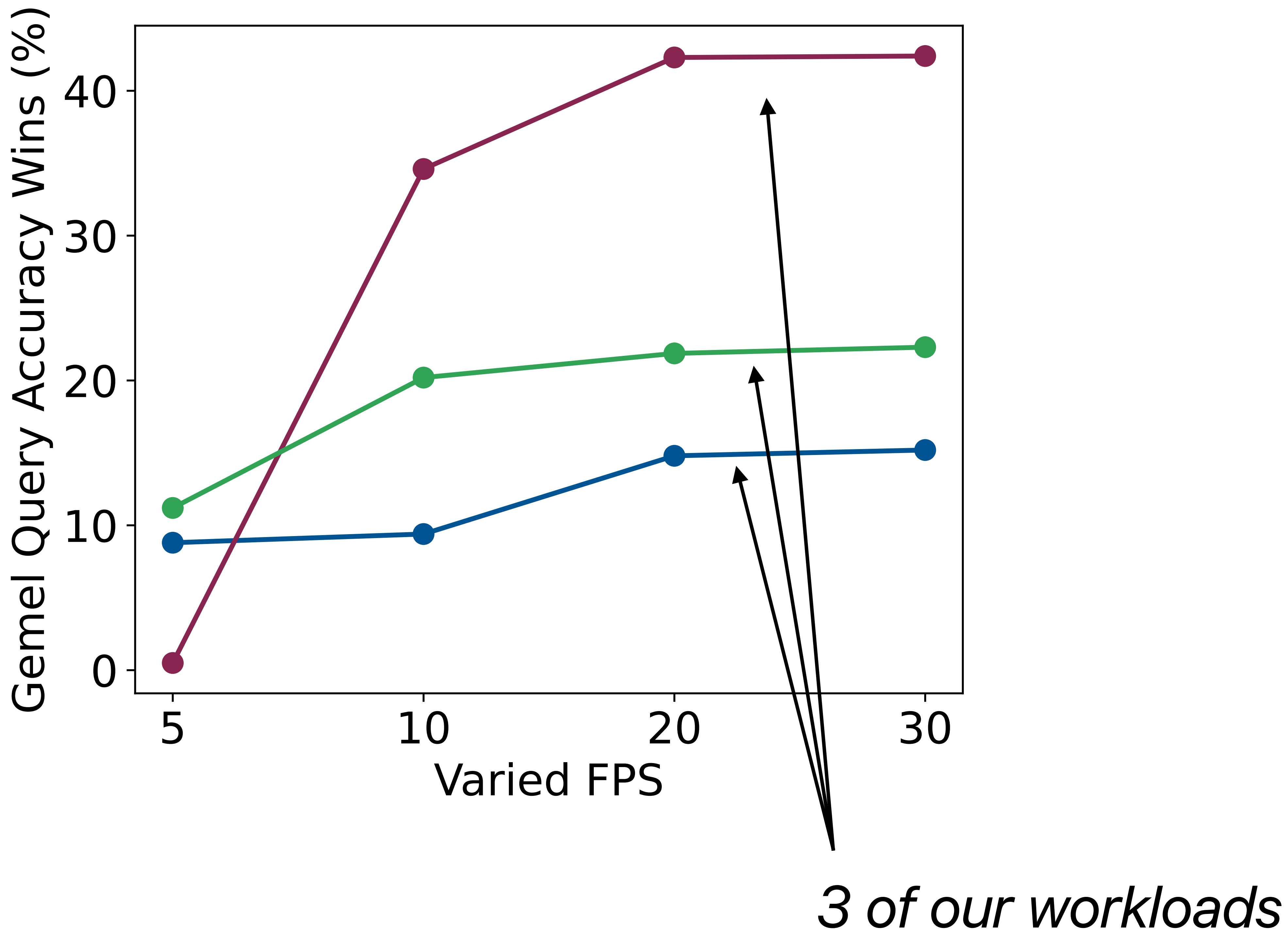
Gemel reduces per-workload memory usage by 18-61%.

Which translates to 150MB to 5.12 GB in raw savings

Varying FPS, Accuracy Target, SLA

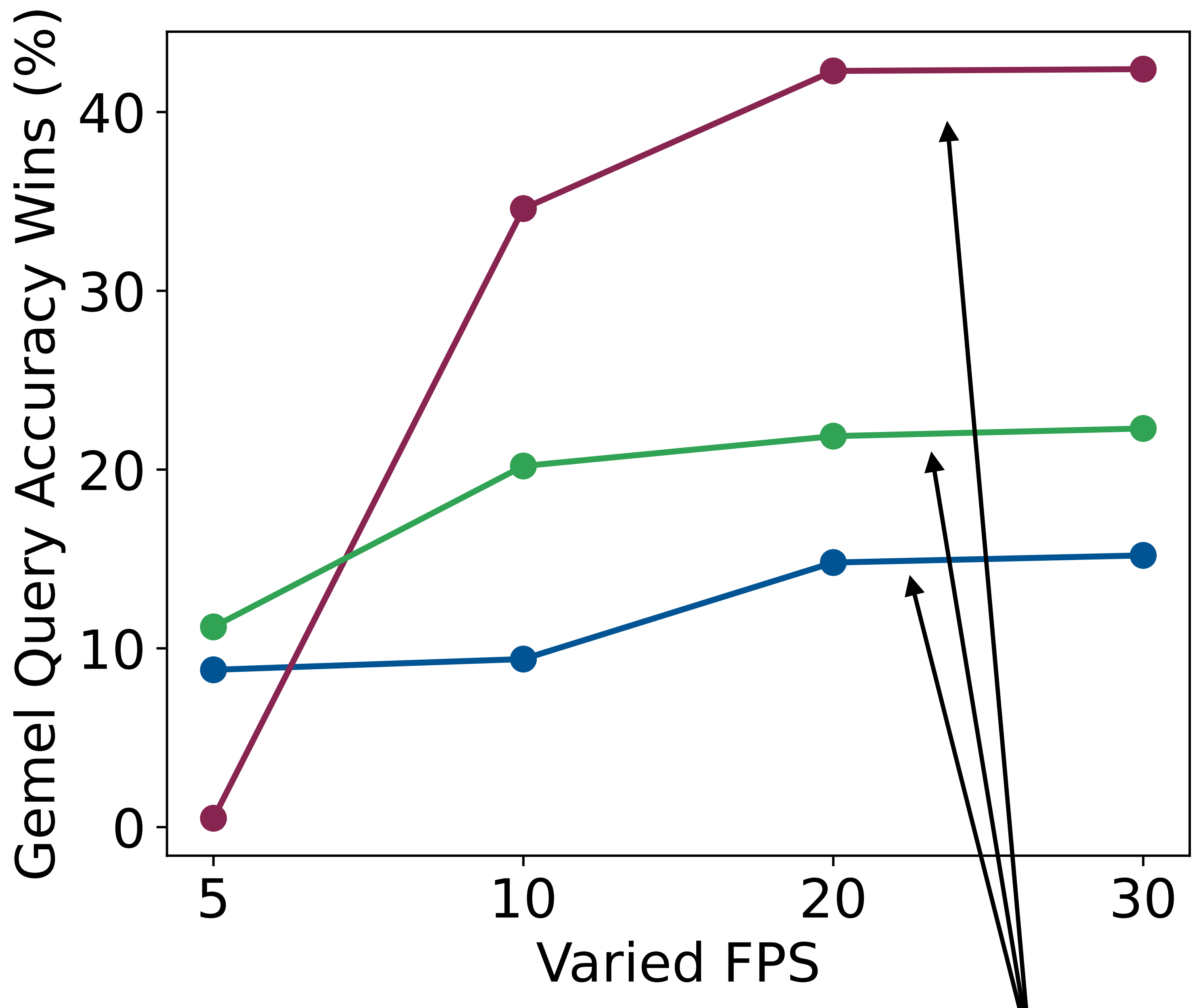
Varying FPS, Accuracy Target, SLA

↑ FPS, Gemel's wins ↑

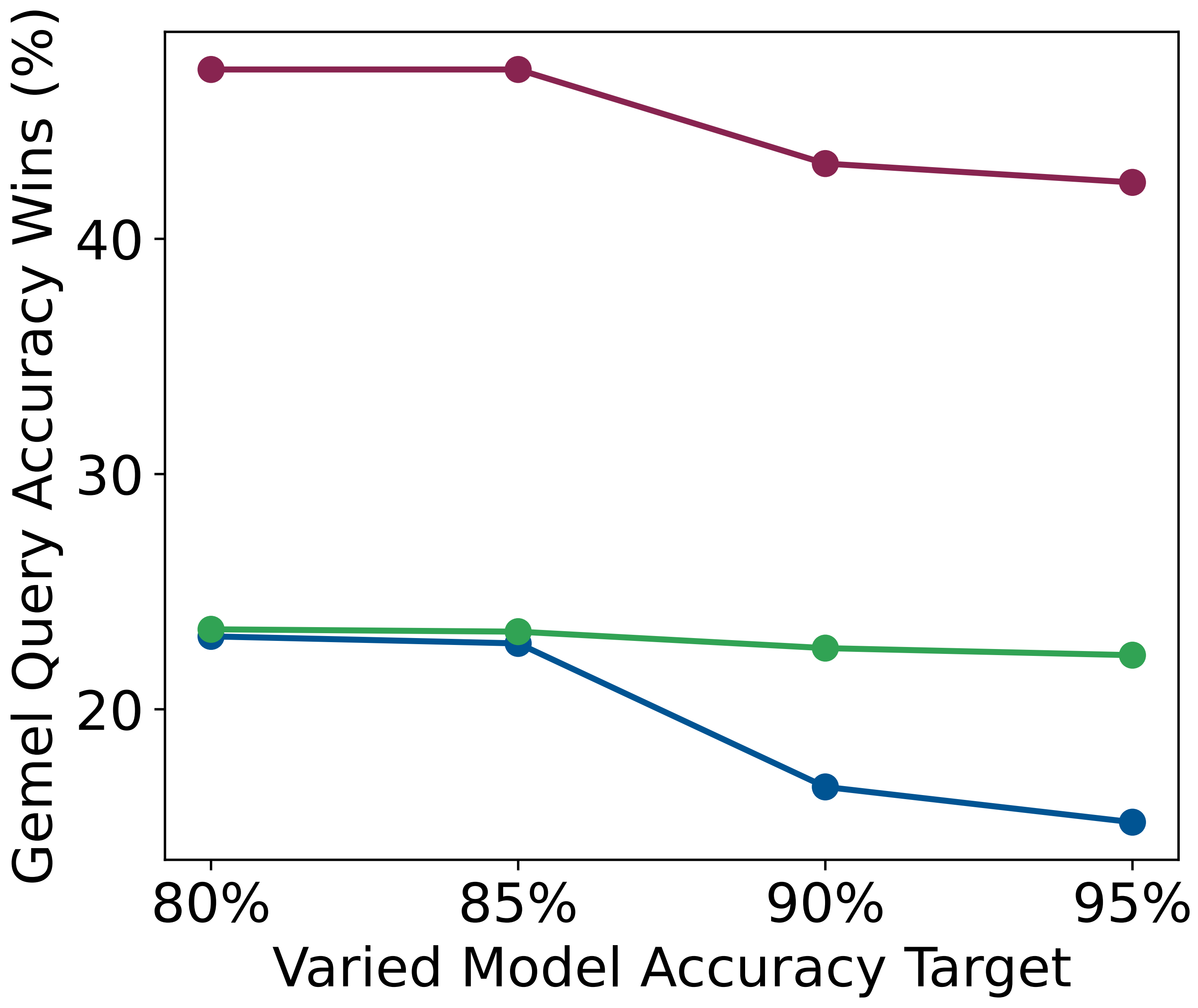


Varying FPS, Accuracy Target, SLA

↑ FPS, Gemel's wins ↑



↓ Model Acc. Target, Gemel's wins ↑



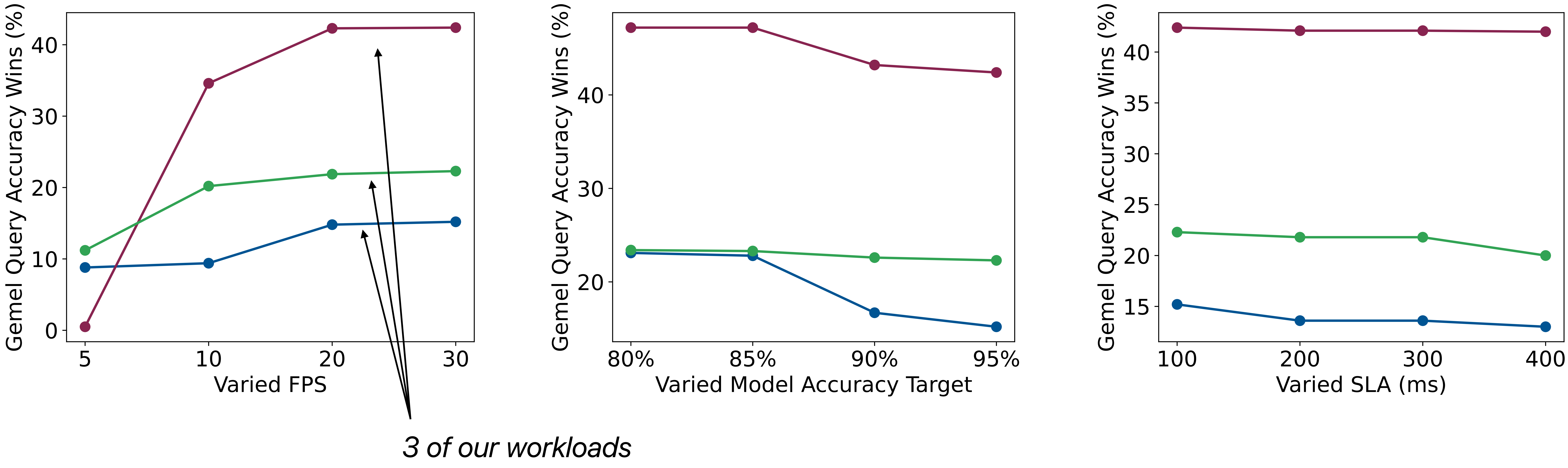
3 of our workloads

Varying FPS, Accuracy Target, SLA

↑ FPS, Gemel's wins ↑

↓ Model Acc. Target, Gemel's wins ↑

More stringent SLAs, Gemel's wins ↑



Gemel Evaluation

- Accuracy Improvements
- GPU Memory Savings
- Varying FPS, Accuracy, SLA
- Comparison to Stem-Sharing Approach
- Incremental Memory Savings
- Merging Heuristic Ablation
- Microbenchmarks
- Generalizability

Gemel

- Tackles GPU memory bottlenecks for real-time video analytics at the edge
- Exploits redundancies across models to find unified weights for layers with shared definitions
- Achieves considerable memory savings and application accuracy improvements

Source code available at github.com/artpad6/gemel_nsdi23