Ekya – Continuous Learning of Video Analytics Models on Edge Compute Servers

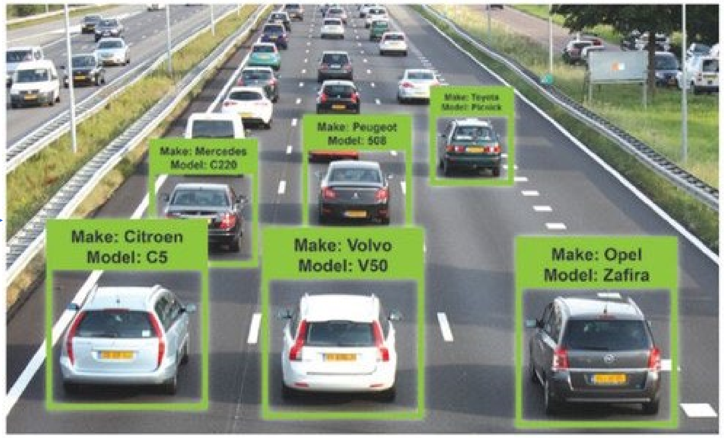
Romil Bhardwaj, Zhengxu Xia, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Nikolaos Karianakis, Kevin Hsieh, Victor Bahl, Ion Stoica

Video data is everywhere

Self-Driving Pipeline

Video Data | **Video Analytics** | **Economic Value**

Vehicle Counting

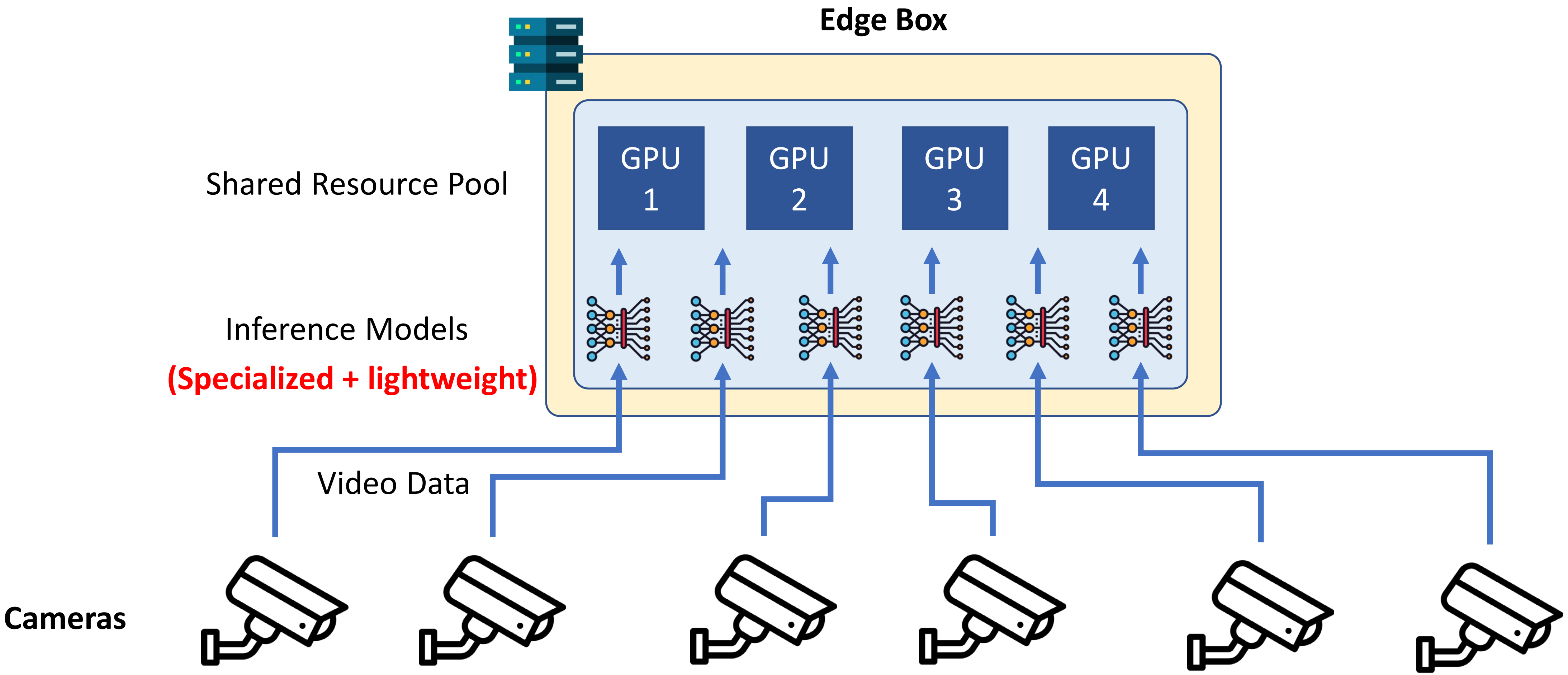


Why video analytics at the edge?

1. **Privacy**
   - Video data is sensitive
   - Regulations prohibit uploading to public cloud providers

2. **Bandwidth**
   - Expensive to provision
   - Not always feasible
   - Can be unreliable

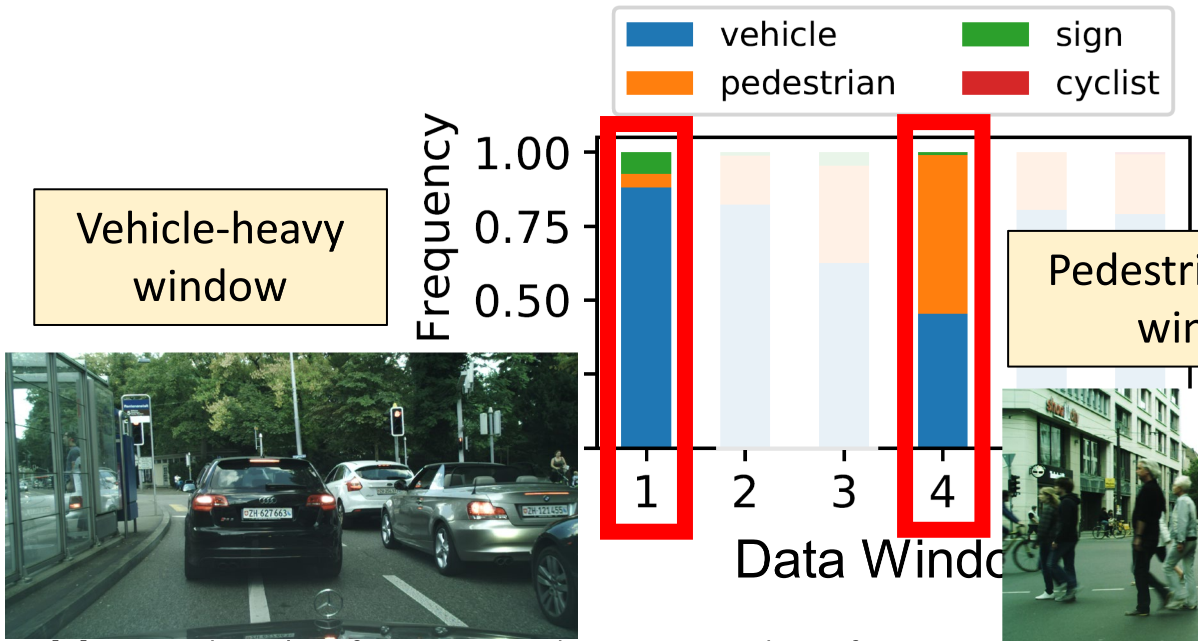
Edge Video Analytics Setup



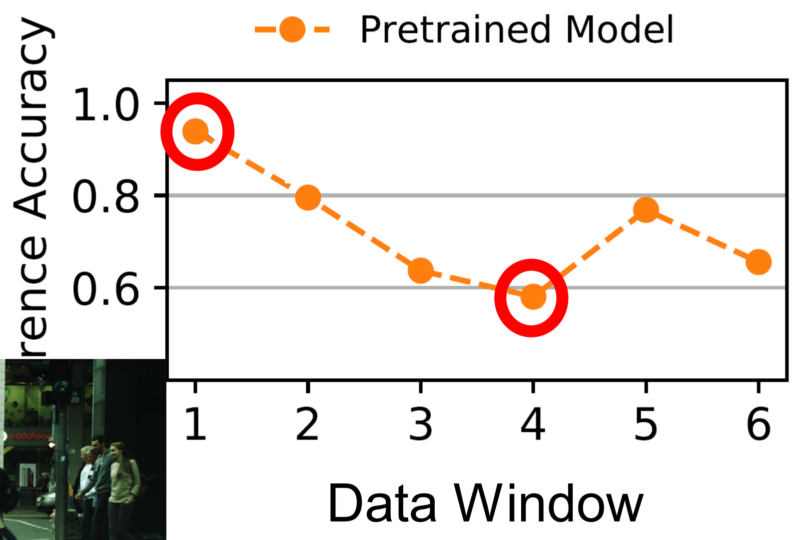
The challenge of data drift[1,2]

- Edge devices run lightweight models which have **limited generalizability**
- Observed data can be different than the training data, resulting in reduced accuracy
- Example – **Class Distribution Shifts**

Class Distribution



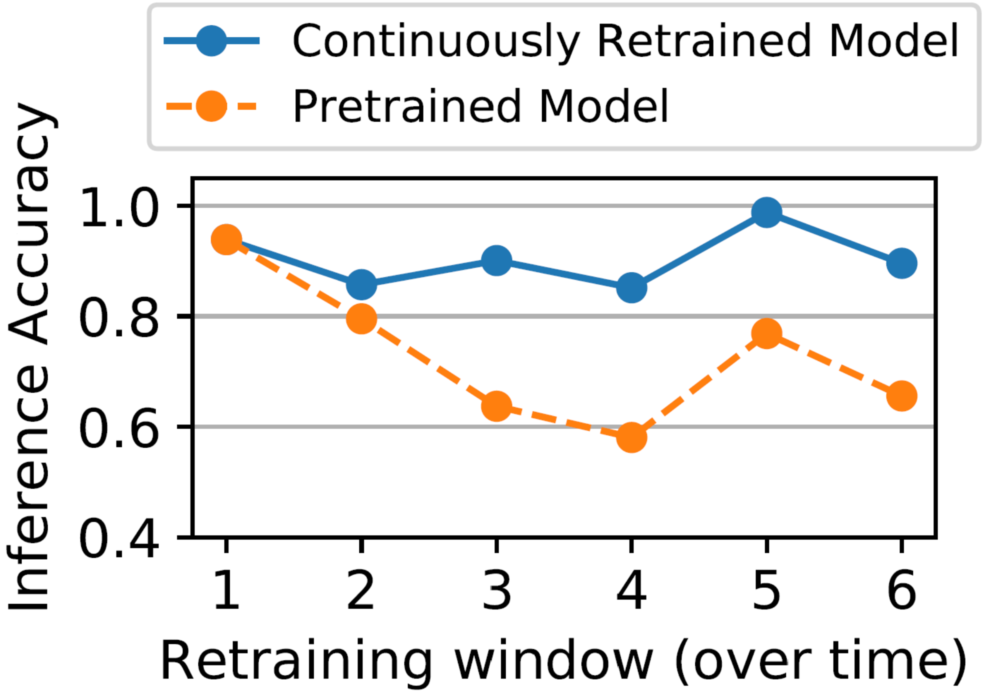
Inference Accuracy



[1] Fast Video Classification via Adaptive Cascading of Deep Mo
[2] Lifelong Robot Learning, Thrun et. al

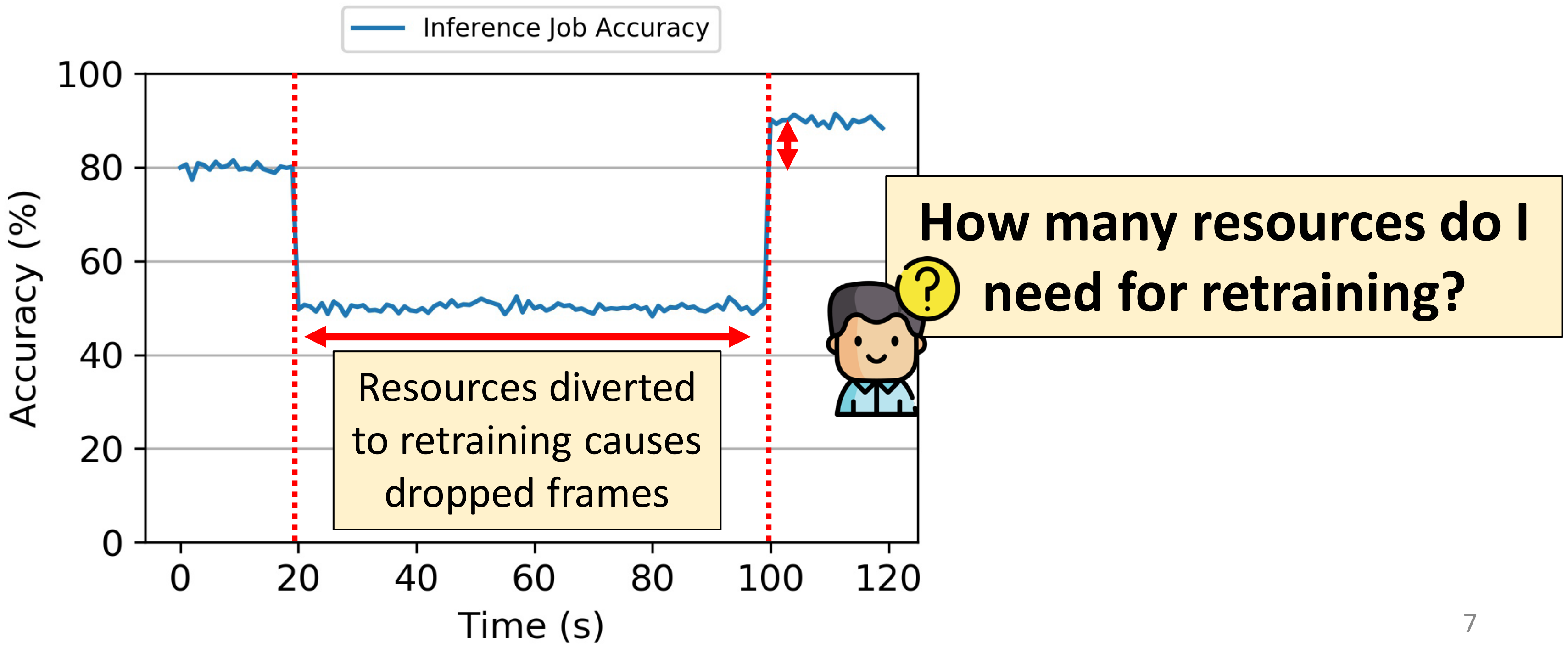
Tackling data drift with continuous learning

- To counter data drift, we can adapt our models by **continuously learning** on incoming data
- Retraining is done periodically (creating "retraining windows")



The cost of continuous learning

- Retraining models requires GPU-time, a precious quantity in resource constrained environments
- To retrain, we must **borrow resources** from inference and reallocate them to training
- Directly impacts inference accuracy because of **dropped frames and downsampling**

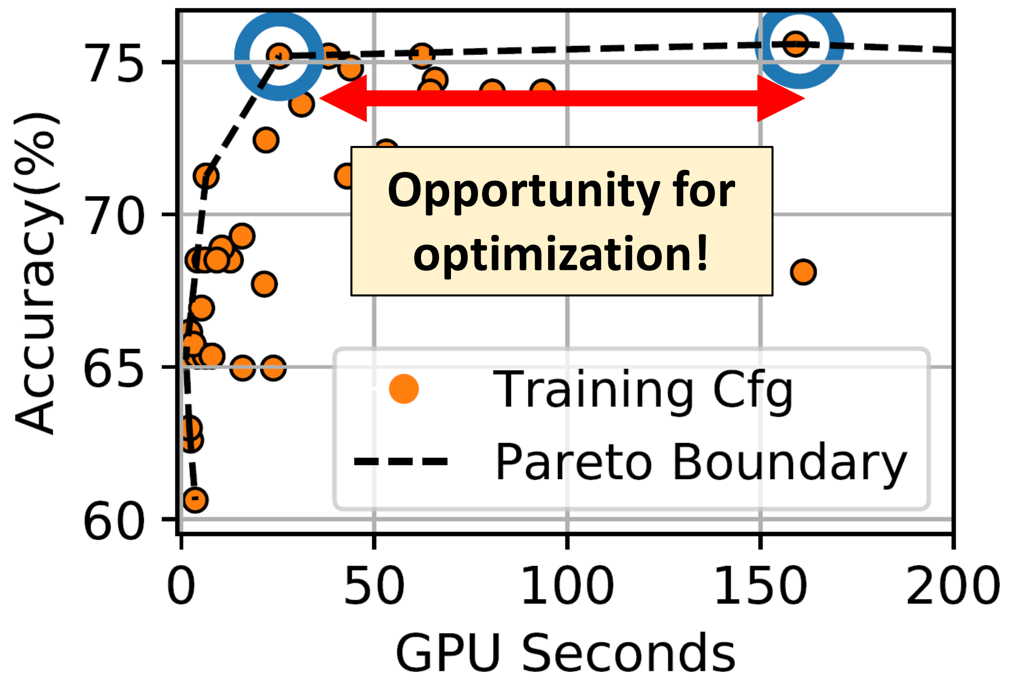


Resource demands of continuous learning

- The cost of retraining depends on the **configuration (hyperparameters)** chosen for retraining.

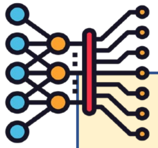
Hyperparameters:

- Layers to train
- Data sampling rate
- Learning rate
- Number of epochs
- Size of last hidden layer

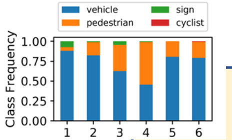


ResNet18 Hyperparameters vs Cost

Summary thus far

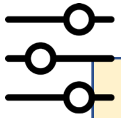


Edge devices use **compressed models**



Compressed models are sensitive to **data drift**

Data drift can be addressed by **continuous learning**



Continuous learning has **costs dependent on configuration**

Summary thus far

Subject to:

Resource capacity constraints
Minimum inference accuracy constraint

Scheduling Objective – Maximize mean inference accuracy across all cameras by picking configurations and resource allocations

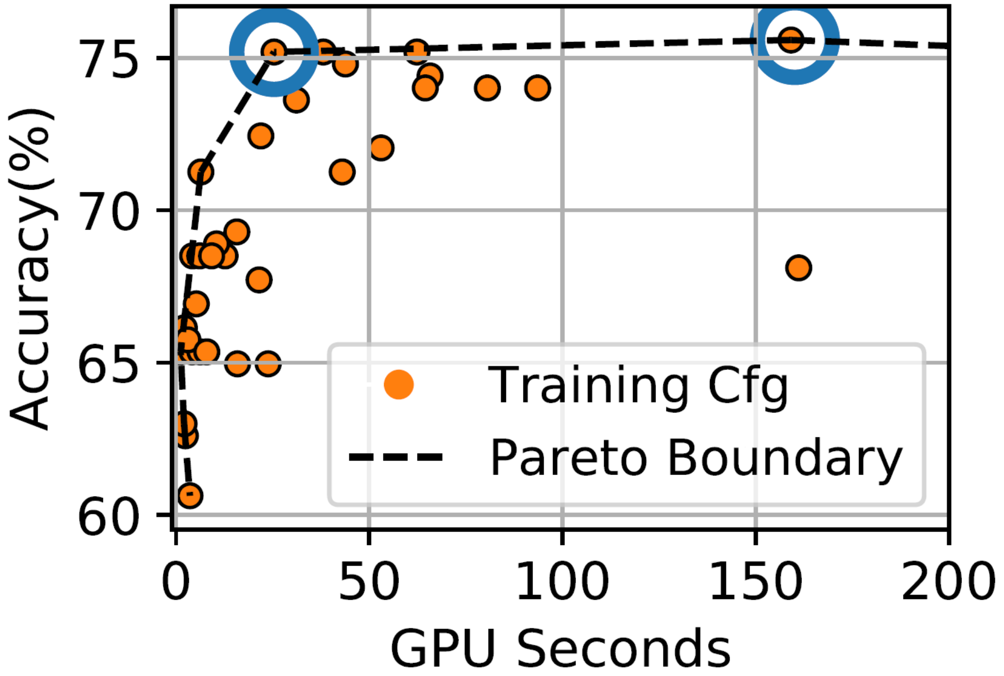
Continuous learning has **costs dependent on configuration**

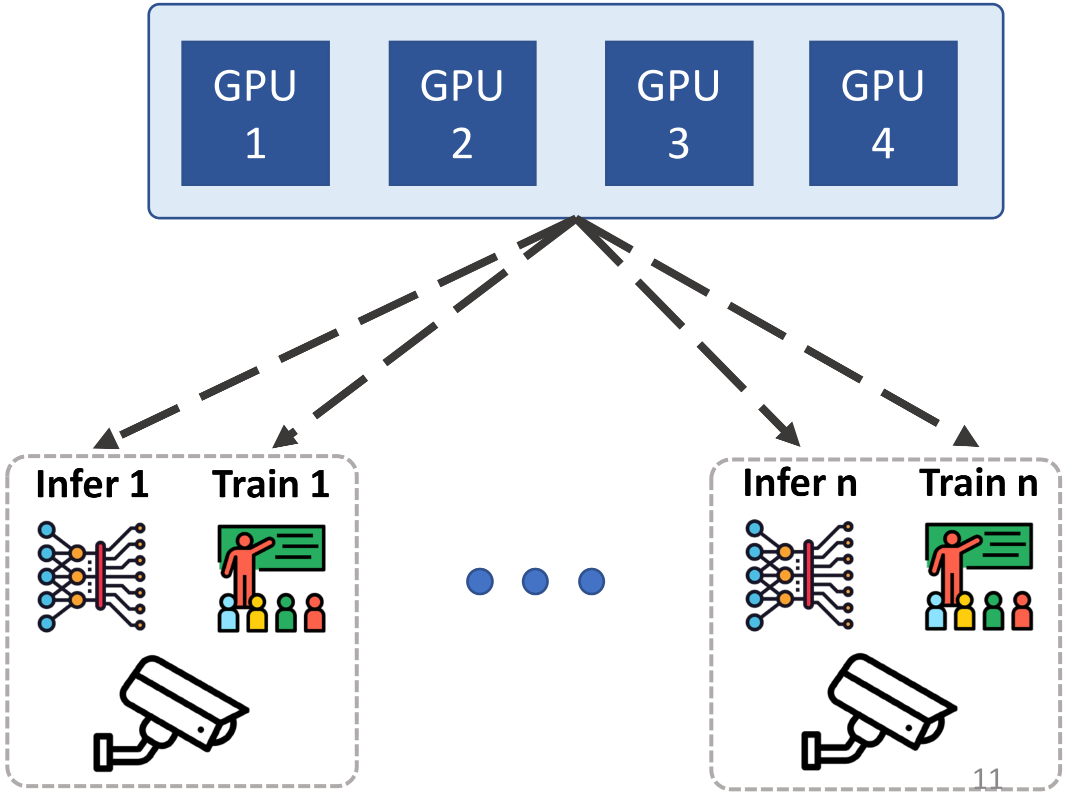
Scheduling decisions to make

Must be done jointly!

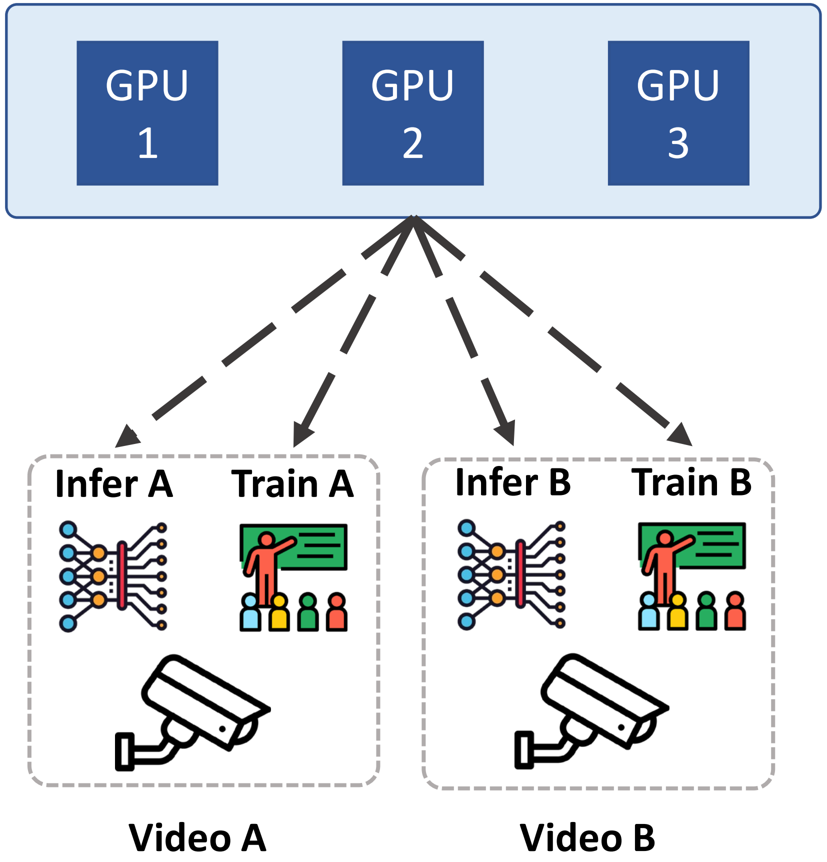
Pick Hyperparameters to train

Allocate resources between Training and Inference across video streams





Working Example



2 retraining windows
Retraining Period = 120s

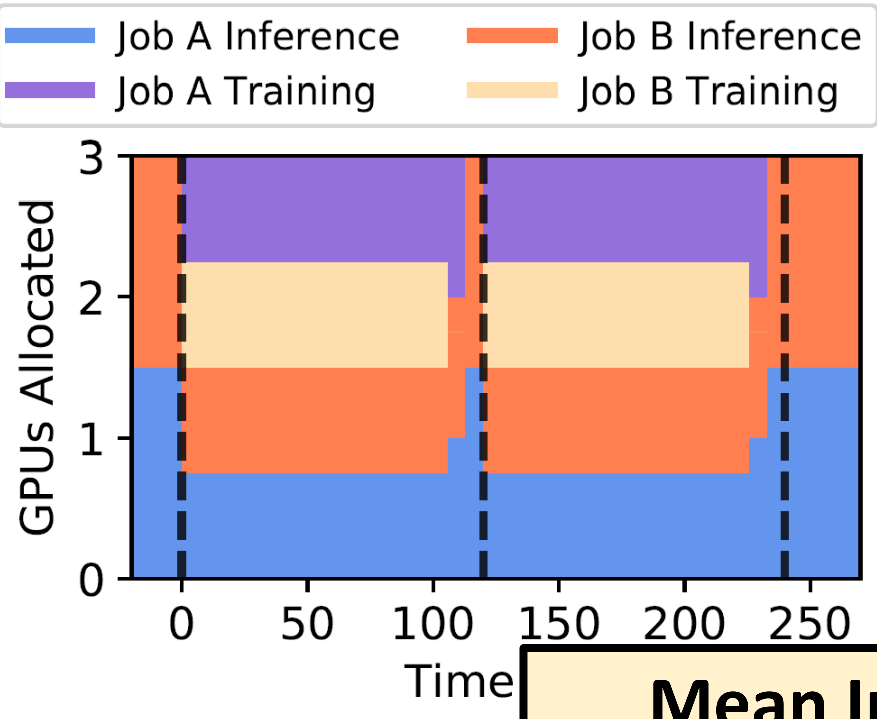
Retraining Configurations

Window	Video	Configuration	End Accuracy	GPU Seconds
Window 1	Video A	**Cfg1A**	**75**	**85**
		Cfg2A	70	65
	Video B	**Cfg1B**	**90**	**80**
		Cfg2B	85	50
Window 2	Video A	**Cfg1A**	**95**	**90**
		Cfg2A	90	50
	Video B	**Cfg1B**	**98**	**80**
		Cfg2B	90	70

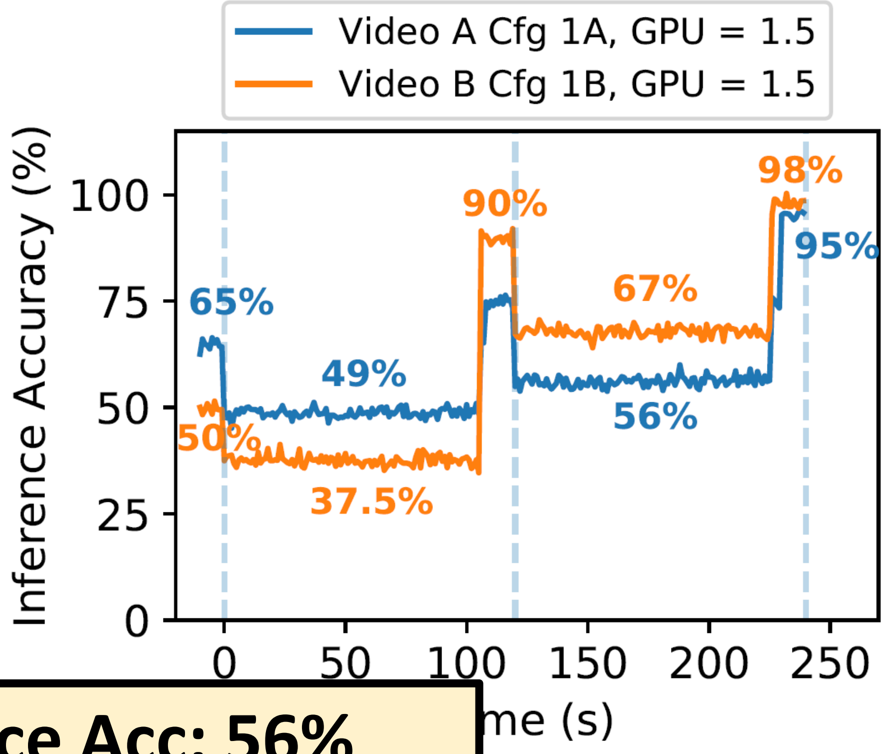
Example – Fair Scheduler

Window	Video	Configuration	End Accuracy	GPU Seconds
Window 1	Video A	Cfg1A	75	85
		Cfg2A	70	65
	Video B	Cfg1B	90	80
		Cfg2B	85	50
Window 2	Video A	Cfg1A	95	90
		Cfg2A	90	50
	Video B	Cfg1B	98	80
		Cfg2B	90	70

- Allocates equal resources, picks configurations which give highest accuracy



Resource Allocation



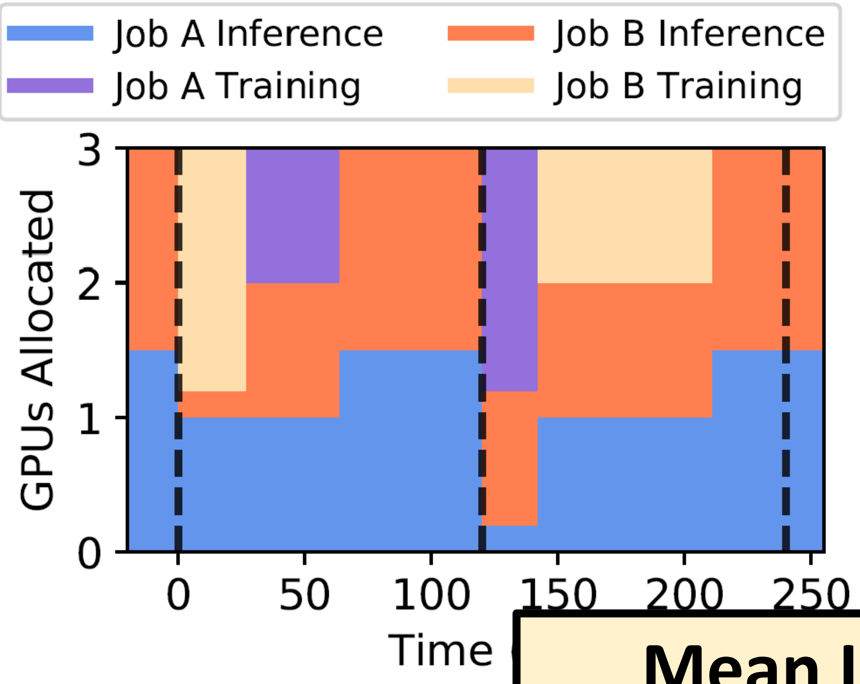
Inference Accuracy

Mean Inference Acc: 56%

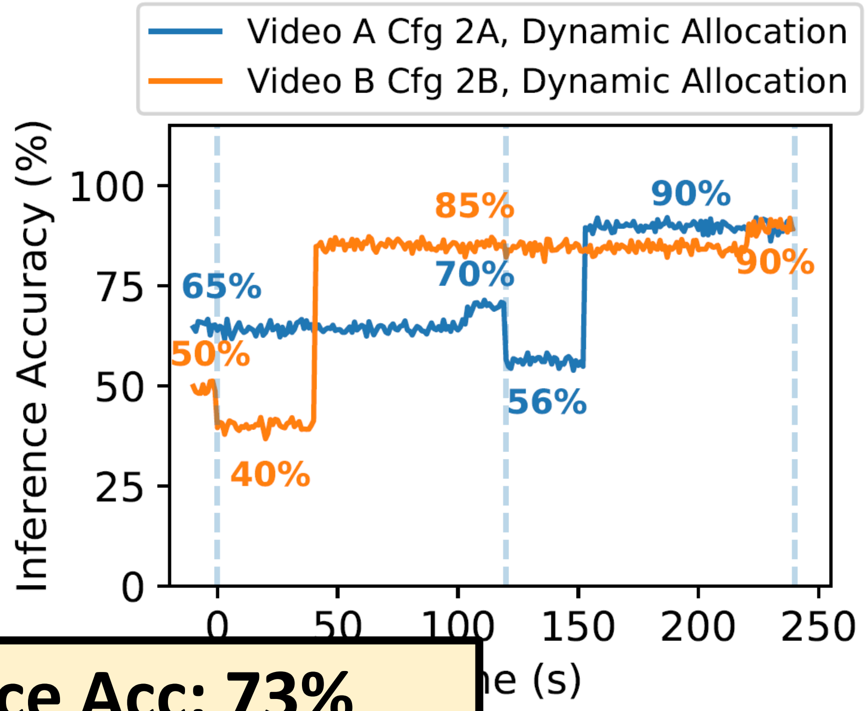
Example – a smarter schedule

Window	Video	Configuration	End Accuracy	GPU Seconds
Window 1	Video A	Cfg1A	75	85
		Cfg2A	**70**	**65**
	Video B	Cfg1B	90	80
		Cfg2B	**85**	**50**
Window 2	Video A	Cfg1A	95	90
		Cfg2A	**90**	**50**
	Video B	Cfg1B	98	80
		Cfg2B	**90**	**70**

- Picks more efficient hyperparameters, prioritizes Job B first



Resource Allocation



Inference Accuracy

Mean Inference Acc: 73%

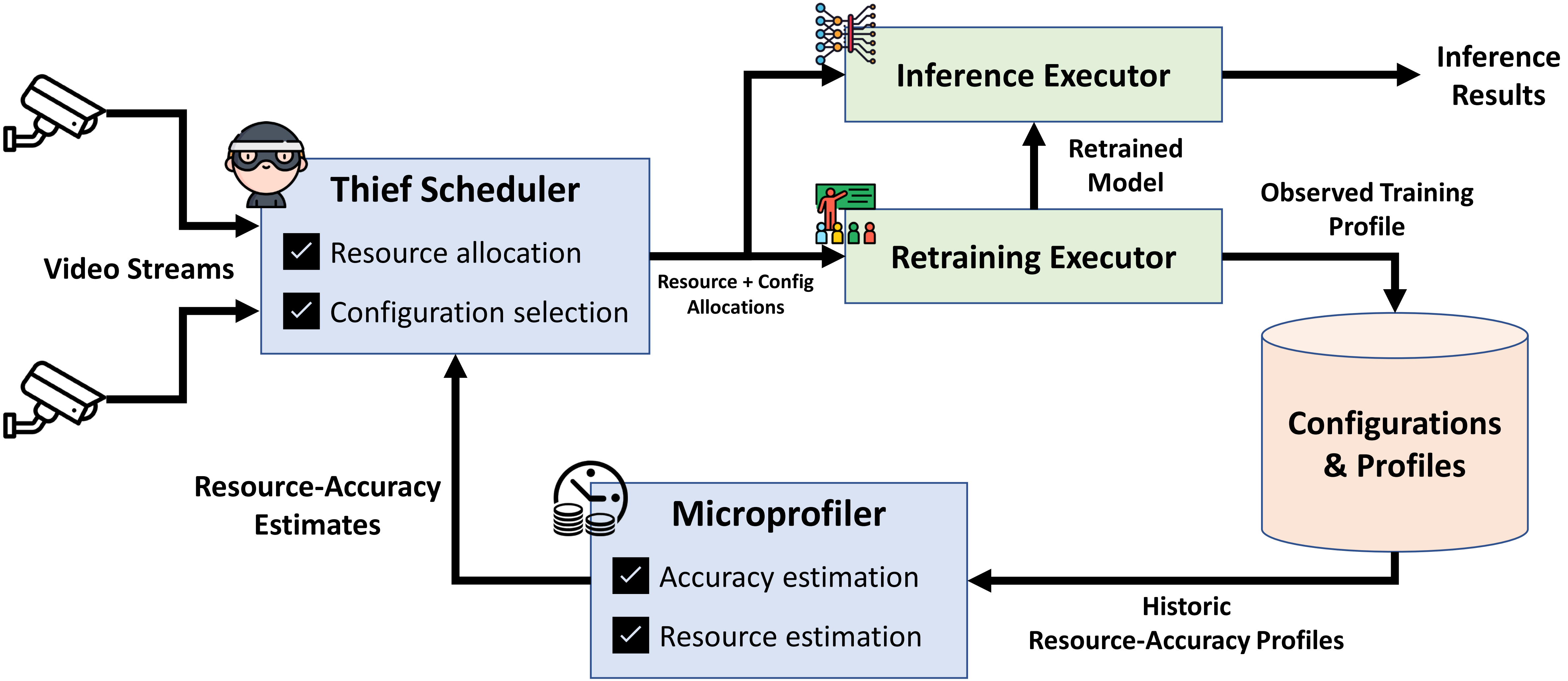
Key Takeaways

Select configurations which
maximize accuracy improvements relative to GPU cost

Prioritize retraining jobs that yield higher accuracy

Reduce accuracy drops by balancing retraining time and resources taken from inference

Ekya Design


Inference Executor
Inference Results
Retrained Model
Observed Training Profile
Retraining Executor
Video Streams
Thief Scheduler
Resource allocation
Configuration selection
Resource + Config Allocations
Configurations & Profiles
Resource-Accuracy Estimates
Microprofiler
Accuracy estimation
Resource estimation
Historic Resource-Accuracy Profiles

Ekya Thief Scheduler

- **Goal**: Maximize mean inference accuracy across all jobs
- Start with a fair allocation to all video streams V
- For each camera, evaluate all pairs of candidate configurations and pick the one which gives highest predicted accuracy
- For each thief job j ∈ J:
  - For victim job k ∈ {J - j}:
    - Steal a quantum of resource δ from **k** and allocat
    - Repeat configuration picking with new resource a
    - If expected mean inference does not improve, stop stealing from k. Else, steal again.

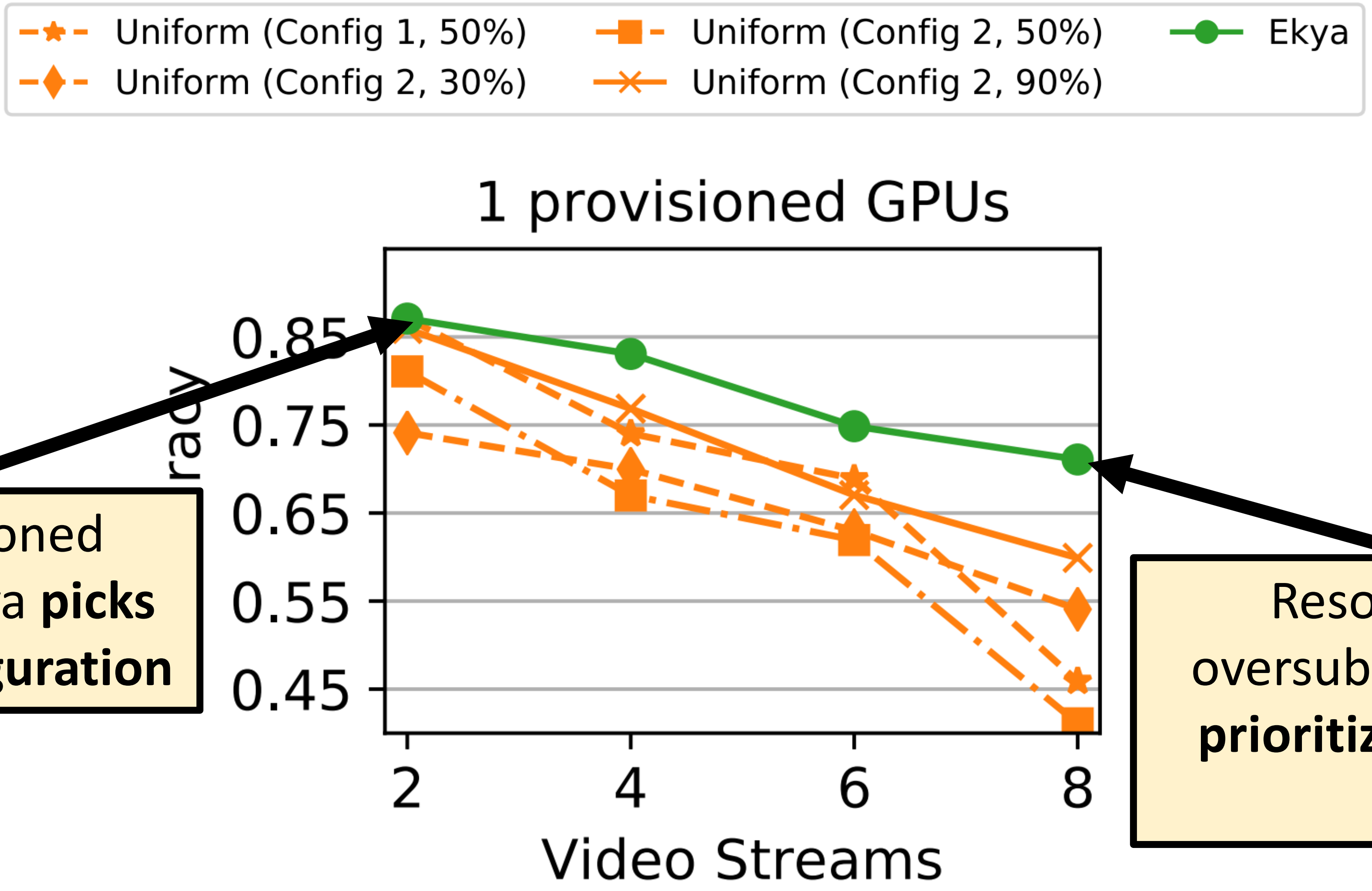
Microprofiler

Accuracy+Resource estimation through hyperparameter sweep and early stopping

Evaluation

- Implement Ekya using Ray’s actor model
  - Nvidia MPS as the resource broker, pytorch for training and inference
- Four datasets –
  - Cityscapes, Waymo – both dashcam videos of driving in different cities
  - UrbanTraffic and UrbanBuilding (New!) – Cameras from Bellevue and Las Vegas
- Baselines – fair scheduler picking different configurations

Scaling with increasing video streams

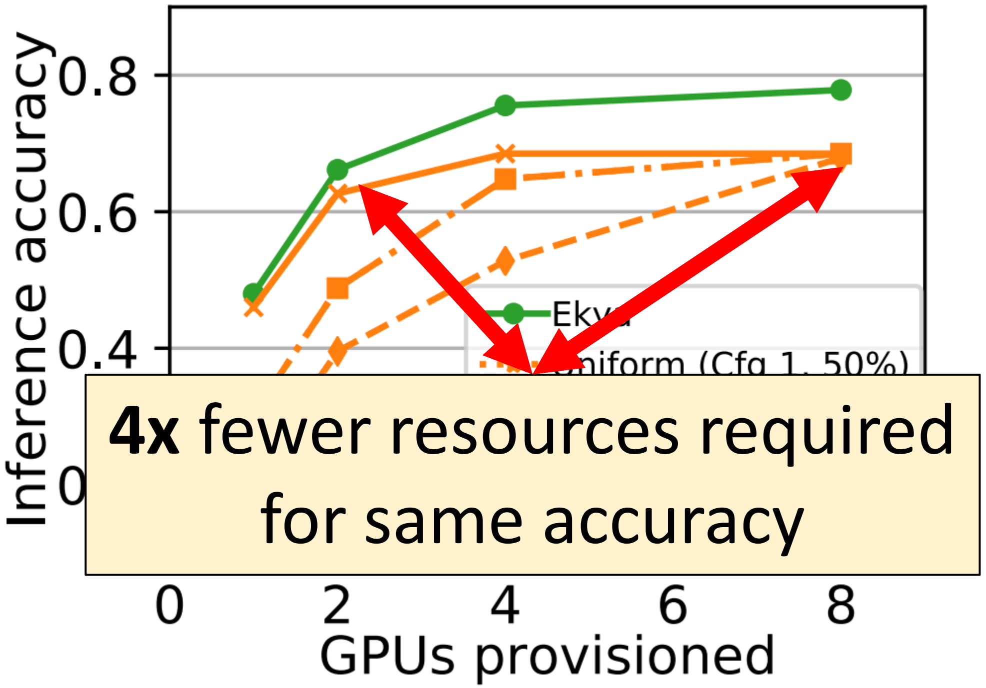


Overprovisioned resources, Ekya **picks the best configuration**

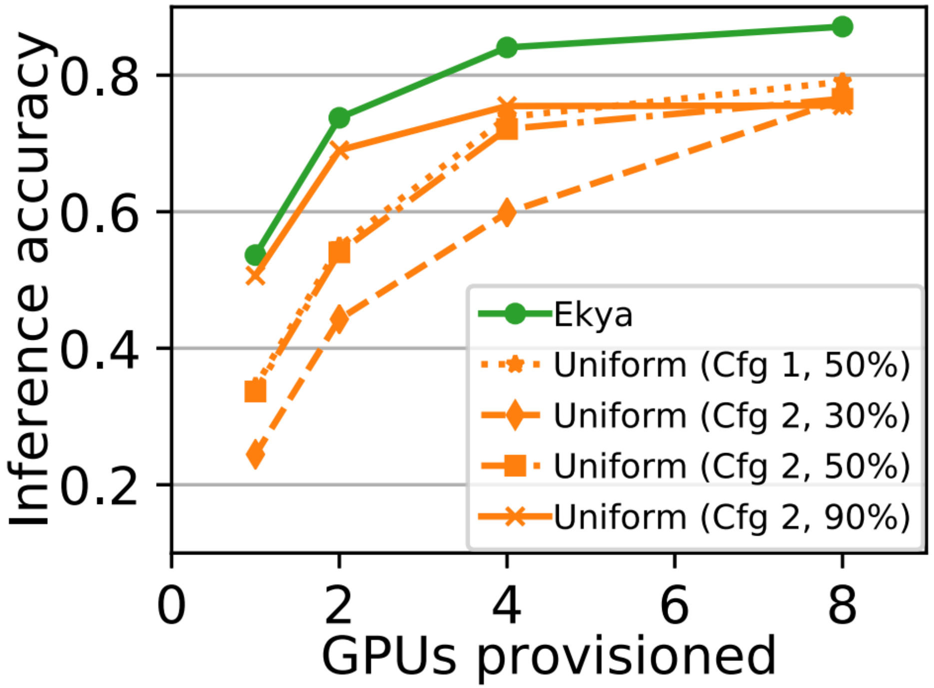
Resources are oversubscribed, Ekya **prioritizes inference jobs**

Scaling with resources and datasets

- 0 to 8 GPUs running on 8 video streams on Cityscapes and Waymo



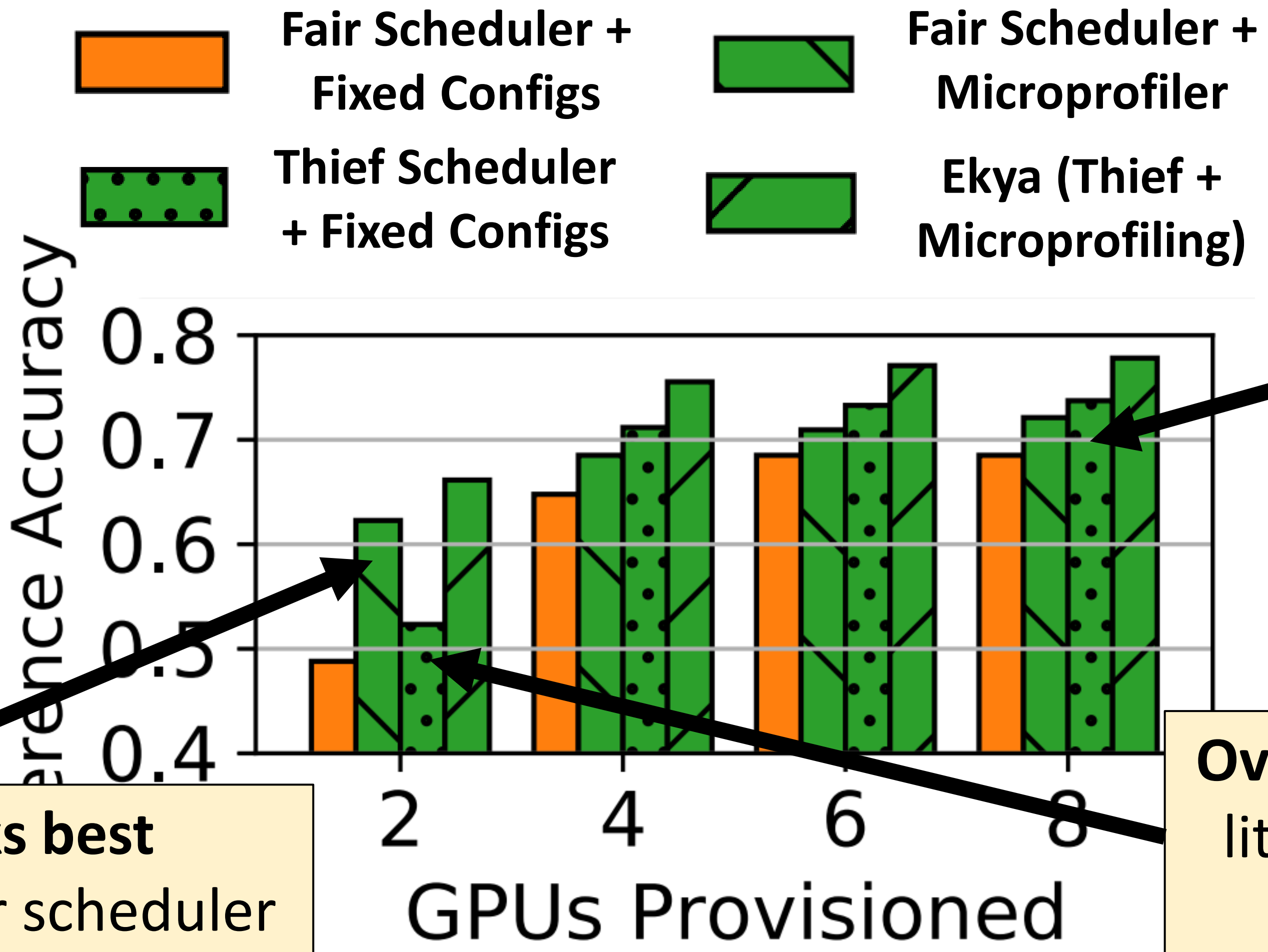
(a) Cityscapes



(b) Waymo

Ekya Ablation

Which components of Ekya contribute to its performance?



Surplus of resources, thief scheduler can significantly improve accuracy

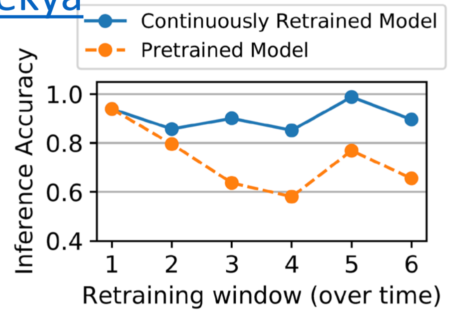
Microprofiler picks best configurations, but fair scheduler equally allocates resources

Oversubscribed GPUs, little opportunity for thief scheduler

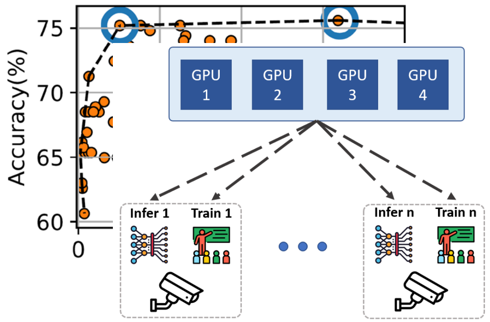
Summary

Code + Datasets:
aka.ms/ekya

- **Continuous learning** counters data drift for lightweight models, but has a resource cost



- Ekya **intelligently allocates resources** and **picks efficient hyperparameters for retraining** to enable continuous learning in resource constrained settings



- Ekya requires **4x fewer resources** to achieve the same inference accuracy as baselines