UNIVERSITY OF
ILLINOIS
URBANA-CHAMPAIGN

Reducing Edge AI Footprint

Presented by: Tarek Abdelzaher

How to Reduce Edge AI Footprint?

How to Reduce Edge AI Footprint?

Optimizing for a single inference:

- Reduce input size (input approximation): Smaller inputs results in less downstream processing and communication.
- Reduce model size (model approximation):
  - Prune less important nodes/links
  - Perform quantization to reduce resolution
  - Exit early
- Improve processing efficiency (e.g., batch inputs)

Optimizing across multiple inferences:

- Break monolithic models into specialized experts
- Cache re-usable partial results
- Balance resource utilization

Inference Time

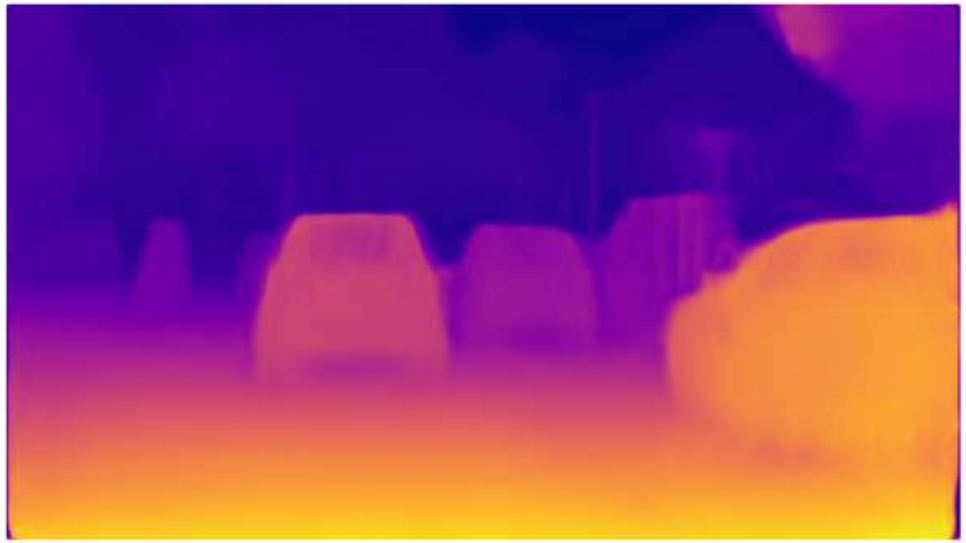
Optimizing for a Single Inference

Optimizing for a single inference:

- *Reduce input size* (input approximation):
  - Reduce input size for conserving computation
  - Reduce input size for conserving communication
- *Reduce model size* (model approximation):
  - Prune less important nodes/links
  - Perform quantization to reduce resolution
  - Input-dependent model reduction: Exit early
- *Improve processing efficiency* (e.g., batch inputs)

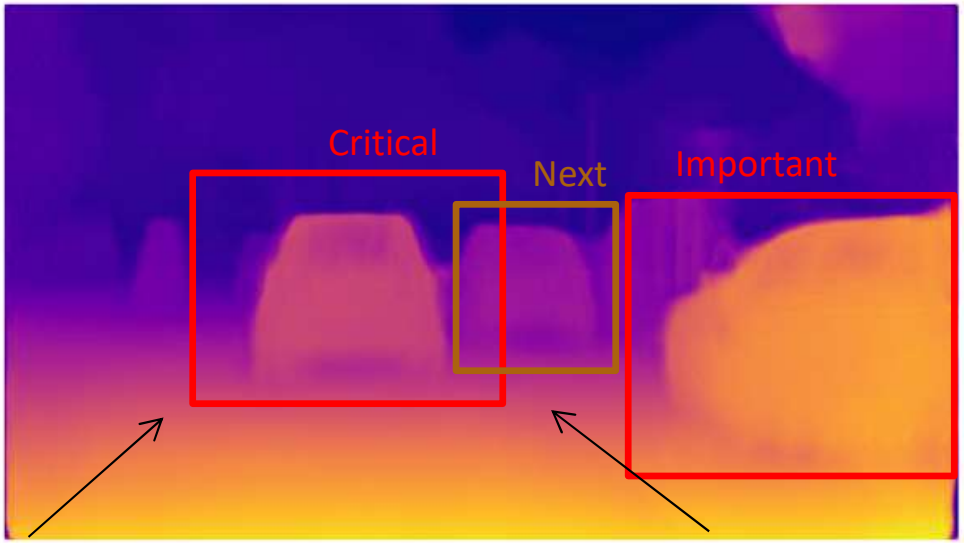
(i) Reducing Input Size for Conserving Computation: An Image Cropping Example

- Purpose of cueing:
  - Decide where to look (i.e., where to allocate computational attention)
  - Decide on (scene segment) prioritization and processing quality



(i) Reducing Input Size for Conserving Computation: An Image Cropping Example

- Purpose of cueing:
  - Decide where to look (i.e., where to allocate computational attention)
  - Decide on (scene segment) prioritization and processing quality



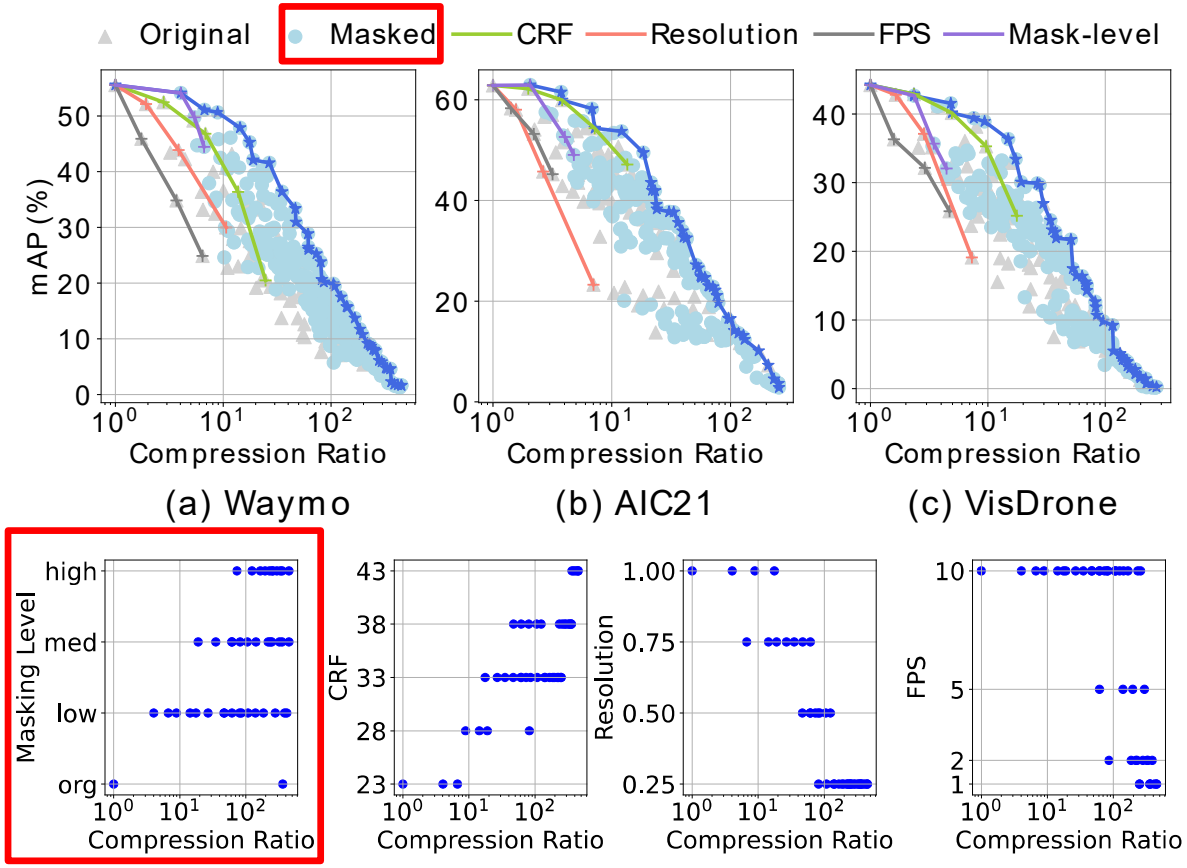
(i) Reducing Input Size for Conserving Communication: An MPEG Video Streaming Example

Figures:

- The upper figure shows the accuracy-bandwidth tradeoff for different configurations. The Pareto boundary, along with the impact of individual knobs are highlighted with curves.
- The lower figure shows the value change of each control knob on the Pareto boundary.

Analysis:

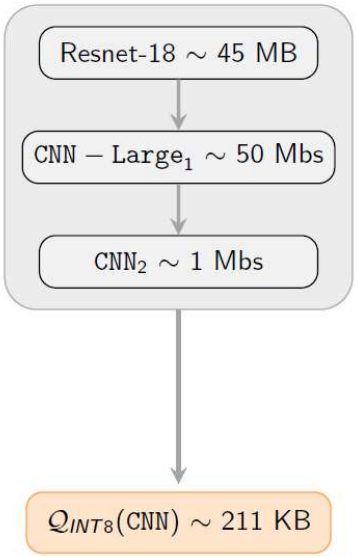
- Most points on the Pareto boundary use masked images (blue points).
- CRF (green curve) and masking level (purple curve) are better dimensions for trading less accuracy for higher compression ratios.



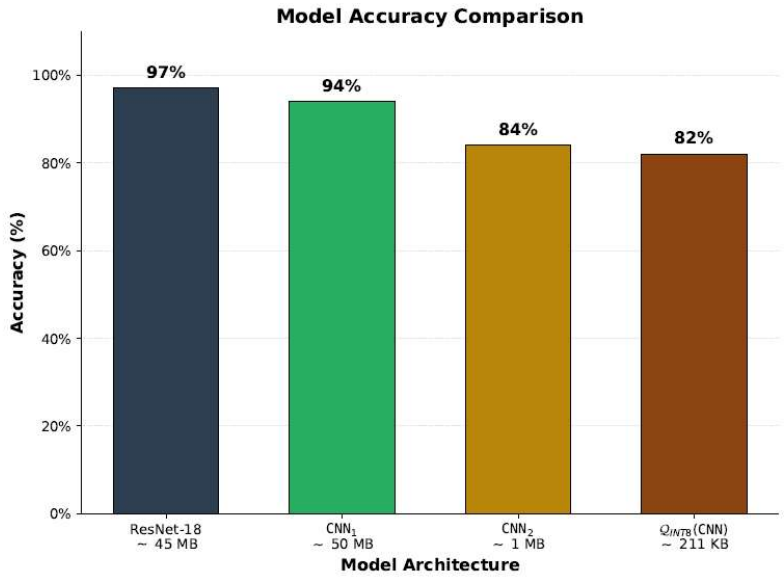
(ii) Reducing Model Size: Quantization

Knowledge Distillation
CE Loss of outputs

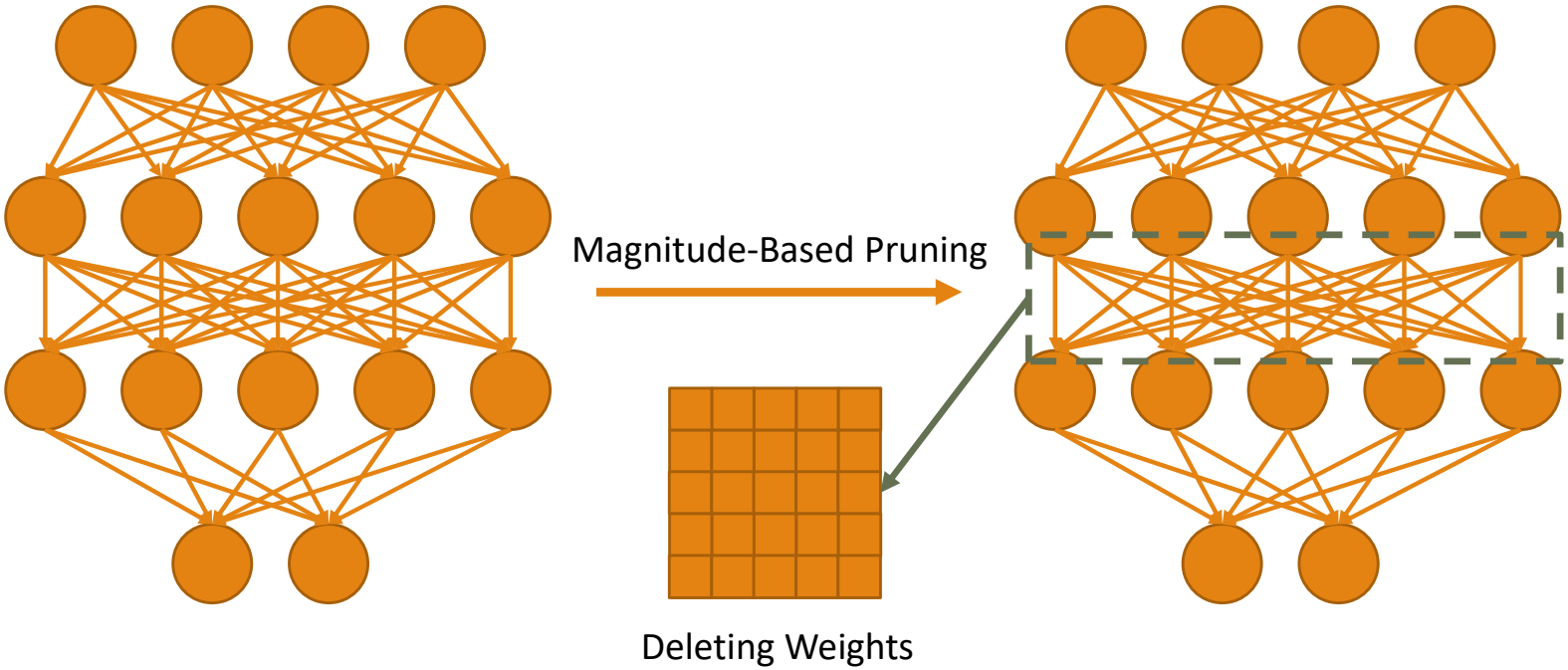
Knowledge Distillation +
Quantization Aware Training



Target classification dataset



(ii) Reducing Model Size: Pruning Less Important Links



Problem: Inefficiency of Sparse Matrix Operations

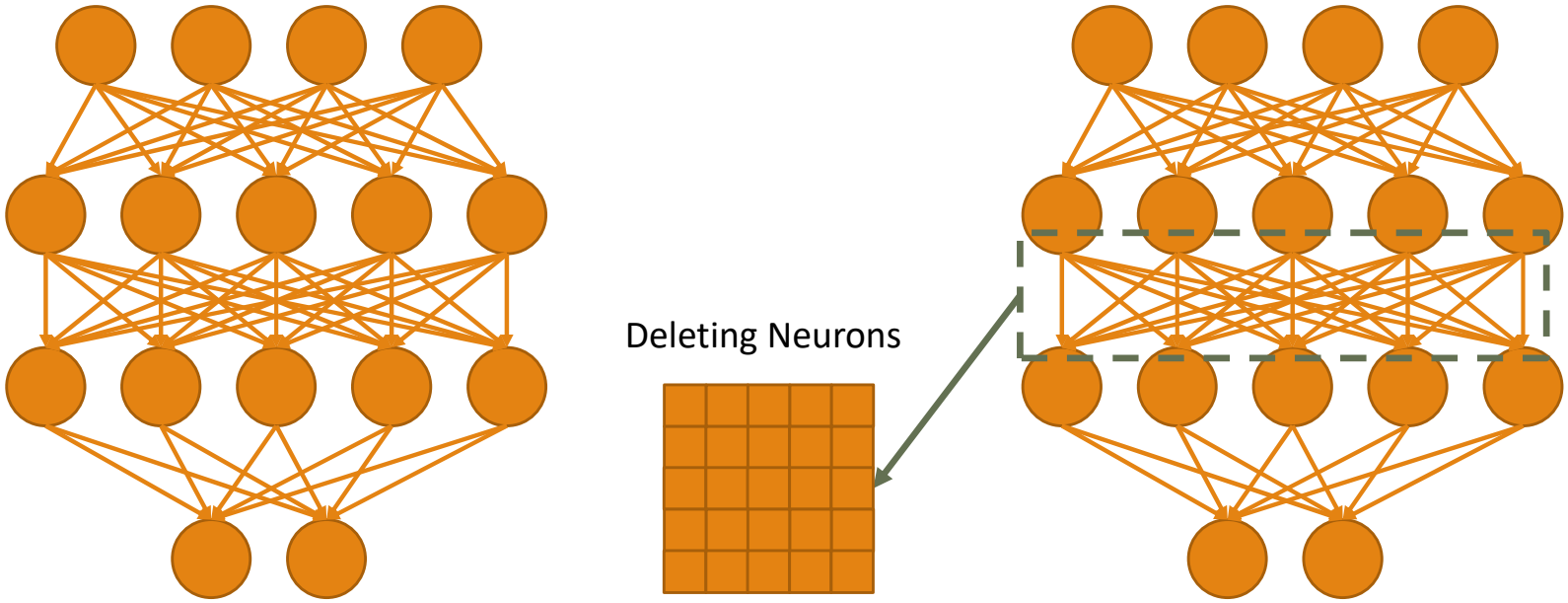
Example:

Multiplication cost: A matrix (m ✖ k) with ***1%*** nonzero elements by a vector (k ✖ 1)

m	k	Time(Sparse/Dense)
100	100	51.7%
100	1000	29.1%
1000	100	33.6%
1000	1000	11.7%

Problem: Sparse multiplication cost does not decrease proportionally to the fraction of zero entries (i.e., is much higher than 1% above).

(ii) Reducing Model Size: Pruning Less Important Nodes



Deciding the optimal number of elements in each layer (structure).

DeepIoT: Performance Overview

Original/Compressed/Savings(%)

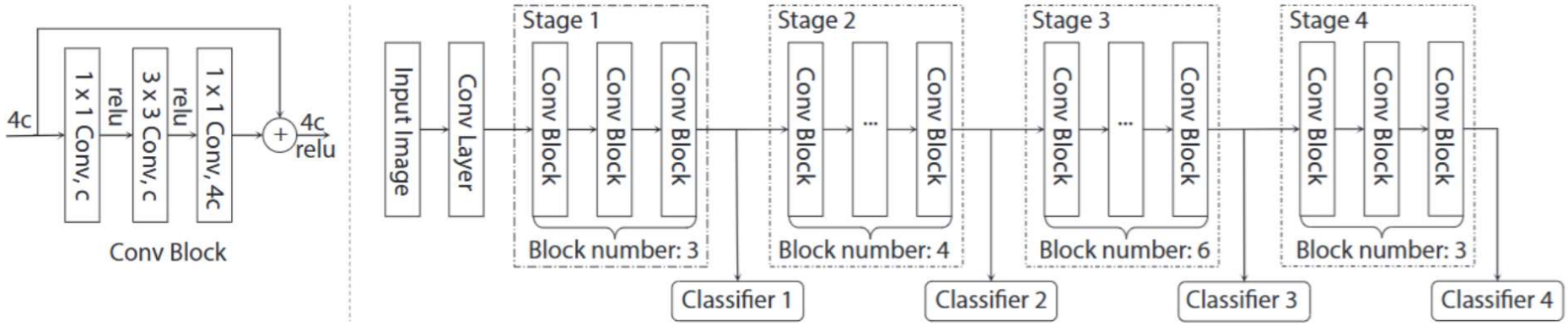
Network Model and Application	Size (MB)	Time (ms)	Energy (mJ)
LeNet5 (Handwriting recognition)	1.72/0.04/97.6%	50.2/14.2/71.4%	47.1/12.5/73.5%
VGGNet (Image recognition)	118.8/2.9/97.6%	1.5K/82.2/94.5%	1.7K/74/95.6%
Bi-LSTM (Speech recognition)	76.0/7.59/90.0%	71K/9.6K/86.5%	62.9K/8.1K/87.1%
DeepSense (Activity recognition)	1.89/0.12/93.7%	130/36.7/71.8%	99.6/27.7/72.2%
DeepSense (User identification)	1.89/0.02/98.9%	130/25.1/80.7%	105.1/18.1/82.8%

Problem with Static Model Size Reduction

The approach is implemented as a pre-processing step before neural network deployment. As such, it does not depend on the current input. It may be that the optimal pruning or quantization is dependent on the input data.

(ii) Reducing Model Size – Input Dependent: Early Exit Neural Networks

Break the neural network into multiple stages (a ResNet example):



Note: There is a classifier at the end of each stage. Confidence in classification results at a given stage decides whether to invoke the next stage. The “pruning” (of tail layers) is now input-dependent

(iii) Improving Efficiency – Batching (on Low-End GPUs)

Lower-end GPUs (such as those commonly used in embedded contexts) are optimized for processing a single neural network kernel at a time (i.e., all cores should run the same code).

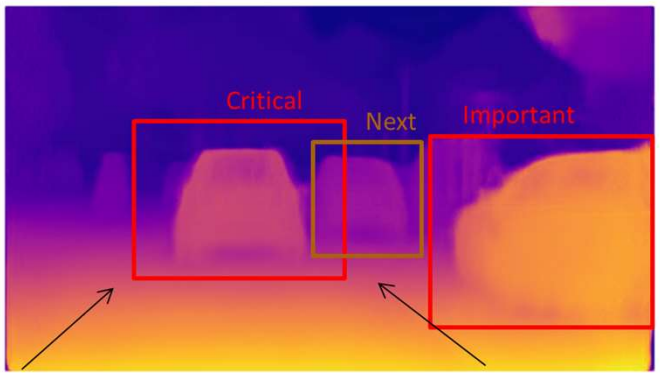
Challenge in the context of “reducing input size”?

(iii) Batching (on Low-End GPUs)

Lower-end GPUs (such as those commonly used in embedded contexts) are optimized for processing a single neural network kernel at a time (i.e., all cores should run the same code).

Challenge in the context of “reducing input size”?

- If smaller inputs are to be processed by smaller neural networks, then each input size has its own neural network kernels.
- Can’t batch inputs of different sizes together. Mitigation?

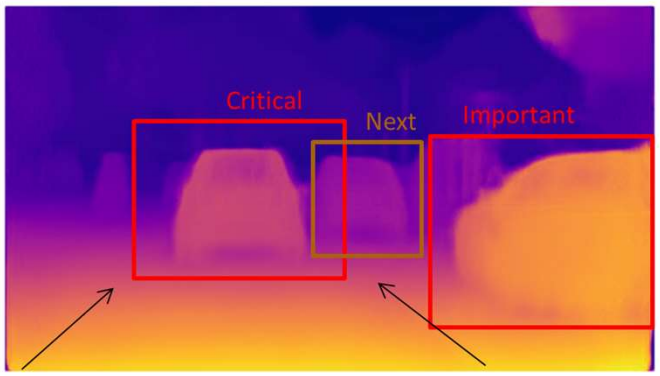


(iii) Batching (on Low-End GPUs)

Lower-end GPUs (such as those commonly used in embedded contexts) are optimized for processing a single neural network kernel at a time (i.e., all cores should run the same code).

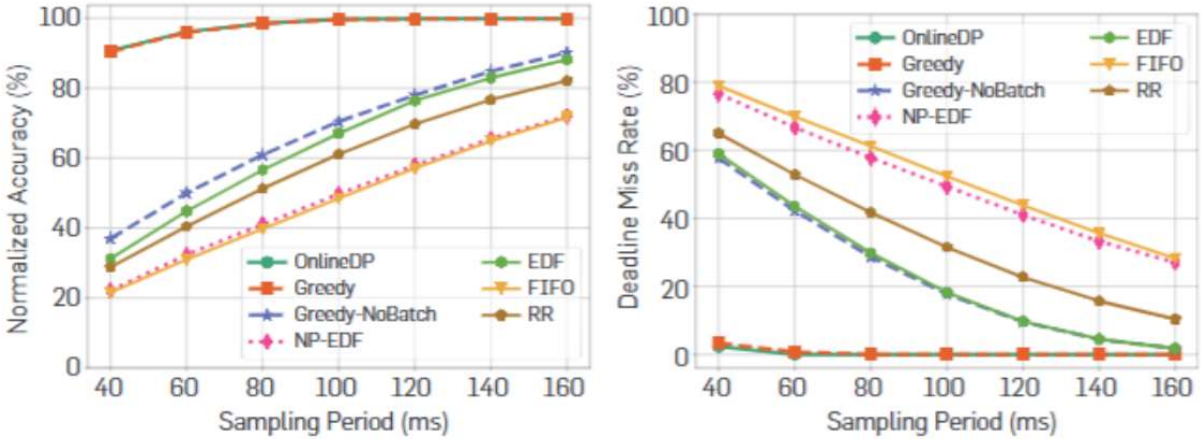
Challenge in the context of “reducing input size”?

- If smaller inputs are to be processed by smaller neural networks, then each input size has its own neural network kernels.
- Can’t batch inputs of different sizes together. Mitigation? → Quantize available sizes and/or resize.



A Joint (i) Cropping (Reducing Input Size), (ii) Early Exit (Reducing Model Size), and (iii) Batching Problem

Problem formulation: With online task arrivals, derive a schedule x to maximize the aggregate system utility. The schedule decides three outputs: (i) task stage execution order on the GPU, (ii) the number of stages to execute for each task, and (iii) task batching decisions.



(a) Normalized accuracy. (b) Deadline miss rate.

Classification accuracy and deadline miss rate on an autonomous car video processing dataset.

Yigong Hu, Shengzhong Liu, Tarek Abdelzaher, Maggie Wigness, Philip David, "On Exploring Image Resizing for Optimizing Criticality-based Machine Perception," *Journal of Real-time Systems*, August 2022.

Giving More Important Objects Better Service (with Differentiated Resizing)

Idea: To save on less important segments, resize them and use a smaller neural network

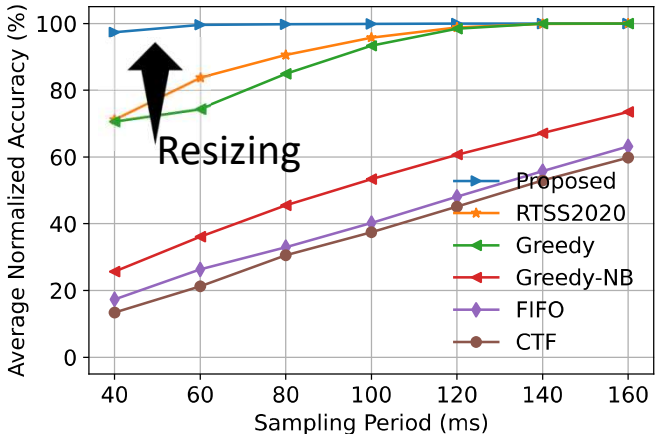
Observations:

Lowest deadline miss rate

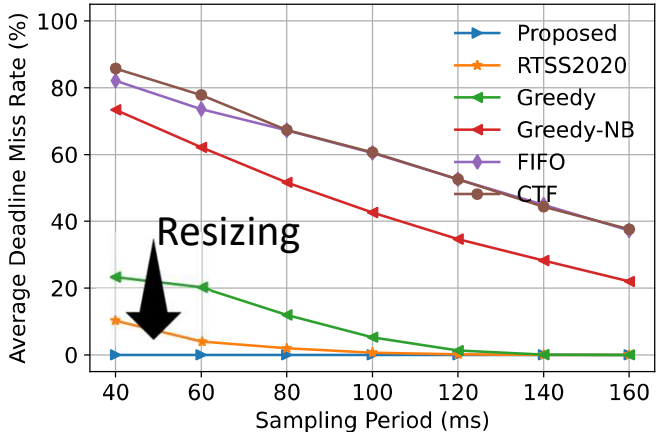
Highest accuracy

Lowest latency

Larger (better) batch size



(a) Normalized Accuracy



(b) Deadline Miss Rate

Trade-Offs

Reducing Input Size versus Early Exit (Reducing Computation)?

- *Reducing Input:* To leverage reductions in input size, one needs multiple neural network models, one per input size, which stresses storage.
- *Early Exit:* In contrast, early exit networks can do less work on easier inputs without having a separate model for each type of network.

- Do we even need to consider reducing input (and subsequent model switching) as a viable optimization policy for edge AI, instead of early exit?

Trade-Offs

Reducing Input Size versus Early Exit?

- *Reducing Input:* To leverage reductions in input size, one needs multiple neural network models, one per input size, which stresses storage.
- *Early Exit:* In contrast, early exit networks can do less work on easier inputs without having a separate model for each type of network.

- Do we even need to consider reducing input (and subsequent model switching) as a viable optimization policy for edge AI, instead of early exit?
  - There are problems with early exit.
  - Reducing input does not always imply having multiple models.

Problems with Early Exit Networks

Model Switching

Original Size		New Image Size			
		32	64	128	256
256	Accuracy (%)	67.3	88.1	98.5	100
	Time (ms)	4	8	13	28
128	Accuracy (%)	66.7	85.4	100	
	Time (ms)	4	8	13	
64	Accuracy (%)	76.0	100		
	Time (ms)	4	8		
32	Accuracy (%)	100			
	Time (ms)	4			

Early Exit

Original Size		Executed Stage			
		1	2	3	4
256	Accuracy (%)	43.2	63.0	89.9	100
	Time (ms)	6	11	19	28
128	Accuracy (%)	40.6	69.4	100	
	Time (ms)	4	8	13	
64	Accuracy (%)	54.6	100		
	Time (ms)	4	8		
32	Accuracy (%)	100			
	Time (ms)	4			

Results are for the ResNet neural network

Problems with Early Exit Networks

Compare results of processing for the same original input and the same computing time…

Model Switching

		New Image Size			
Original Size		32	64	128	256
256	Accuracy (%)	67.3	88.1	98.5	100
	Time (ms)	4	8	13	28
128	Accuracy (%)	66.7	85.4	100	
	Time (ms)	**4**	8	13	
64	Accuracy (%)	76.0	100		
	Time (ms)	4	8		
32	Accuracy (%)	100			
	Time (ms)	4			

Early Exit

		Executed Stage			
Original Size		1	2	3	4
256	Accuracy (%)	43.2	63.0	89.9	100
	Time (ms)	6	11	19	28
128	Accuracy (%)	40.6	69.4	100	
	Time (ms)	**4**	8	13	
64	Accuracy (%)	54.6	100		
	Time (ms)	4	8		
32	Accuracy (%)	100			
	Time (ms)	4			

Results are for the ResNet neural network

Problems with Early Exit Networks

Under the same conditions on input and processing time, model switching offers better accuracy than early exit… Why?

Model Switching

		New Image Size			
Original Size		32	64	128	256
256	Accuracy (%)	67.3	88.1	98.5	100
	Time (ms)	4	8	13	28
128	Accuracy (%)	**66.7**	85.4	100	
	Time (ms)	**4**	8	13	
64	Accuracy (%)	76.0	100		
	Time (ms)	4	8		
32	Accuracy (%)	100			
	Time (ms)	4			

Early Exit

		Executed Stage			
Original Size		1	2	3	4
256	Accuracy (%)	43.2	63.0	89.9	100
	Time (ms)	6	11	19	28
128	Accuracy (%)	**40.6**	69.4	100	
	Time (ms)	**4**	8	13	
64	Accuracy (%)	54.6	100		
	Time (ms)	4	8		
32	Accuracy (%)	100			
	Time (ms)	4			

Results are for the ResNet neural network

Problems with Early Exit Networks

Under the same conditions on input and processing time, model switching offers better accuracy than early exit… Why?

Model Switching

		New Image Size			
Original Size		32	64	128	256
256	Accuracy (%)	67.3	88.1	98.5	100
	Time (ms)	4	8	13	28
128	Accuracy (%)	66.7	**85.4**	100	
	Time (ms)	4	8	13	
64	Accuracy (%)	76.0	100		
	Time (ms)	4	8		
32	Accuracy (%)	100			
	Time (ms)	4			

Early Exit

		Executed Stage			
Original Size		1	2	3	4
256	Accuracy (%)	43.2	63.0	89.9	100
	Time (ms)	6	11	19	28
128	Accuracy (%)	40.6	**69.4**	100	
	Time (ms)	4	8	13	
64	Accuracy (%)	54.6	100		
	Time (ms)	4	8		
32	Accuracy (%)	100			
	Time (ms)	4			

Results are for the ResNet neural network

Problems with Early Exit Networks

Under the same conditions on input and processing time, model switching offers better accuracy than early exit… Why?

Model Switching

		New Image Size			
Original Size		32	64	128	256
256	Accuracy (%)	67.3	88.1	98.5	100
	Time (ms)	4	8	13	28
128	Accuracy (%)	66.7	85.4	100	
	Time (ms)	4	8	13	
64	Accuracy (%)	**76.0**	100		
	Time (ms)	4	8		
32	Accuracy (%)	100			
	Time (ms)	4			

Early Exit

		Executed Stage			
Original Size		1	2	3	4
256	Accuracy (%)	43.2	63.0	89.9	100
	Time (ms)	6	11	19	28
128	Accuracy (%)	40.6	69.4	100	
	Time (ms)	4	8	13	
64	Accuracy (%)	**54.6**	100		
	Time (ms)	4	8		
32	Accuracy (%)	100			
	Time (ms)	4			

Results are for the ResNet neural network

Problems with Early Exit Networks

Under the same conditions on input and processing time, model switching offers better accuracy than early exit… Why?

Individual models – needed for processing inputs of different size categories – can be optimally trained (i.e., their weights can be optimally selected) for the corresponding input size, whereas a single imprecise computation pipeline is essentially a compromise between optimal weight selections for each of the different exit points. No single weight assignments is simultaneously best for all exits.

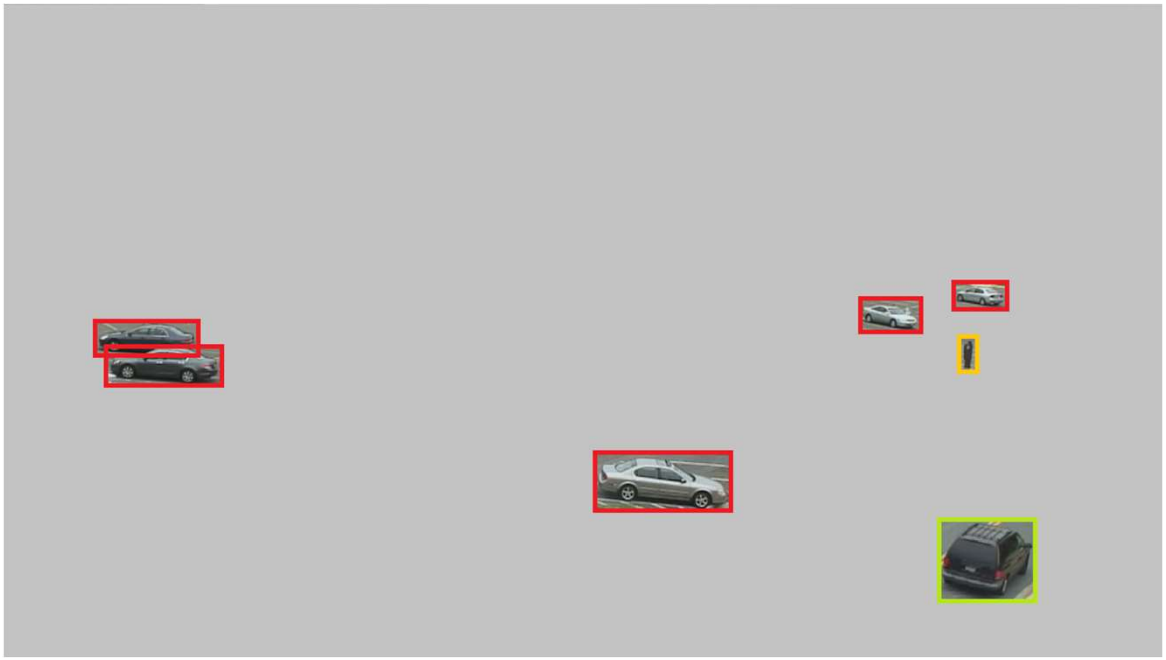
Problems with Early Exit Networks

Under the same conditions on input and processing time, model switching offers better accuracy than early exit… Why?

Individual models – needed for processing inputs of different size categories – can be optimally trained (i.e., their weights can be optimally selected) for the corresponding input size, whereas a single imprecise computation pipeline is essentially a compromise between optimal weight selections for each of the different exit points. No single weight assignments is simultaneously best for all exits.

Can we get the benefits of input size reduction without paying the cost of model multiplicity (to process the different input sizes)?

Canvas Scheduling



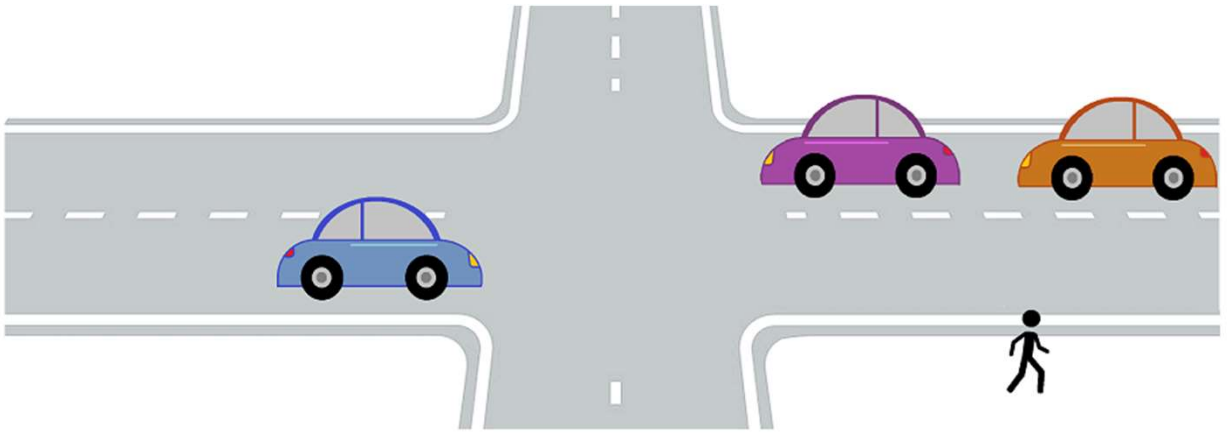
Region of Interest

Canvas Scheduling

Batching: T = 97ms

Canvas: T = 43ms

Scheduling Problem: an Example

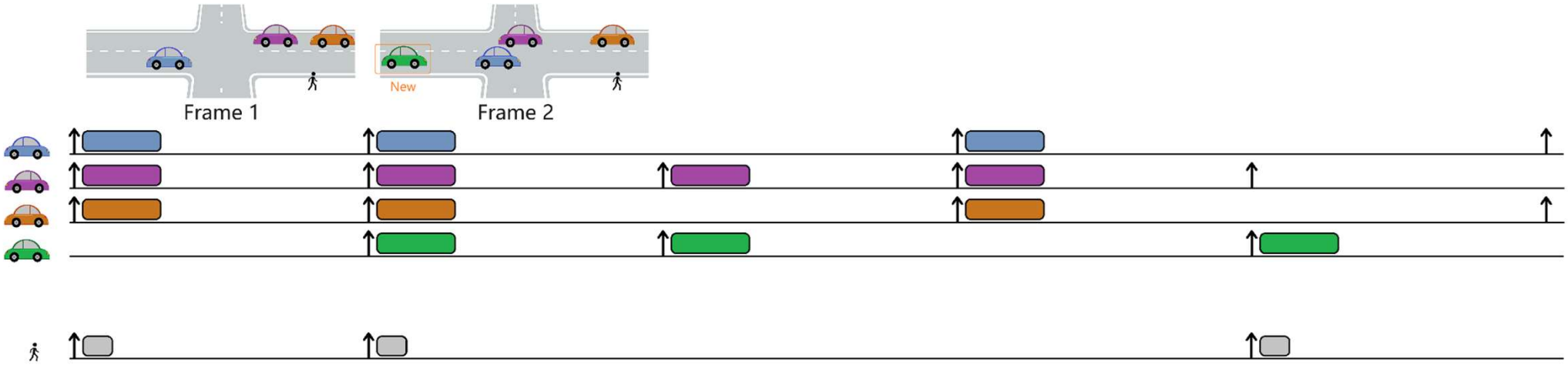


Scheduling Problem: an Example

Frame 1

Scheduling Problem: an Example

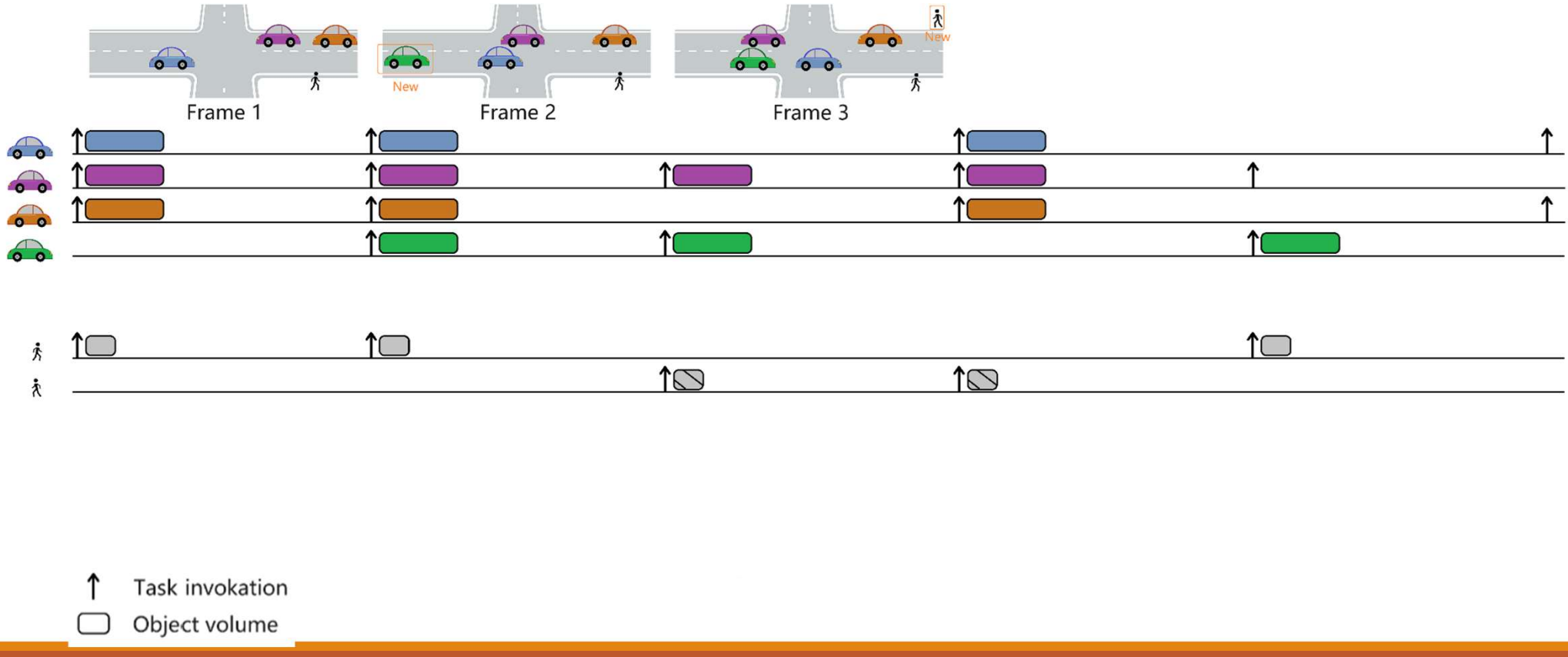
Scheduling Problem: an Example



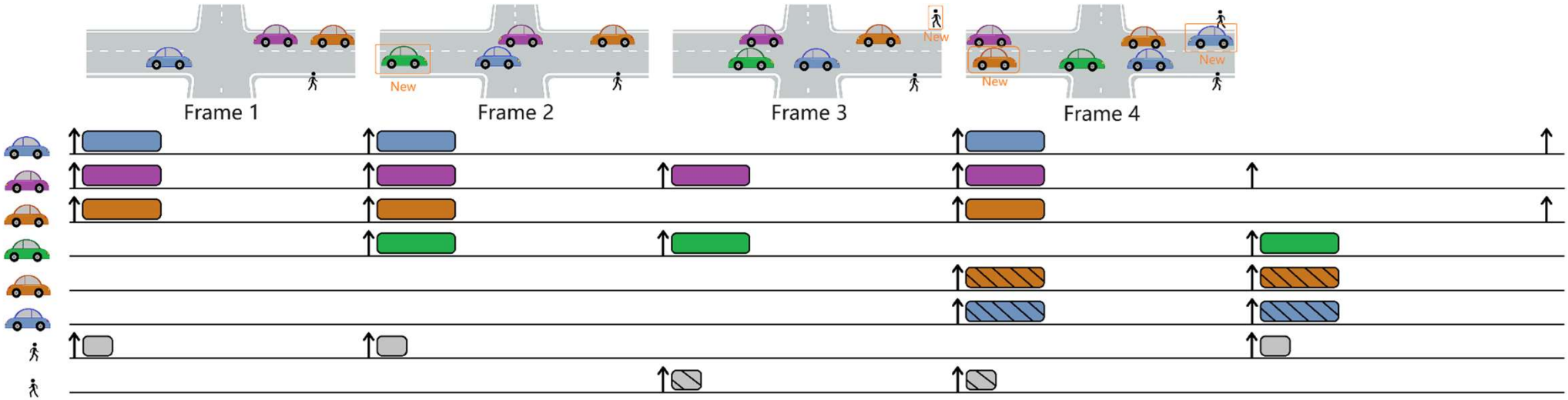
Task invokation

Object volume

Scheduling Problem: an Example



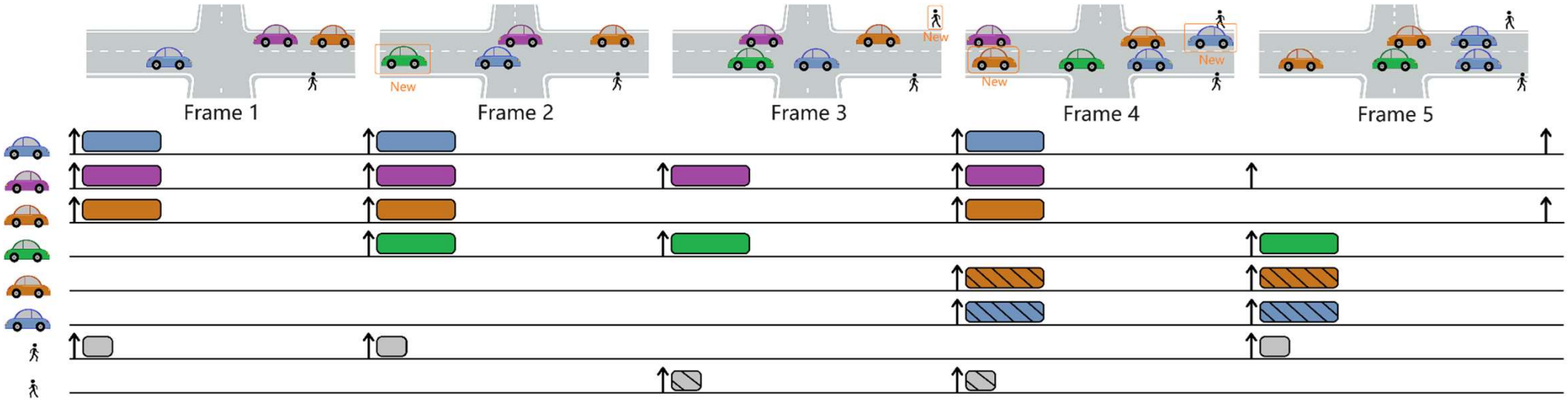
Scheduling Problem: an Example



↑ Task invokation

▭ Object volume

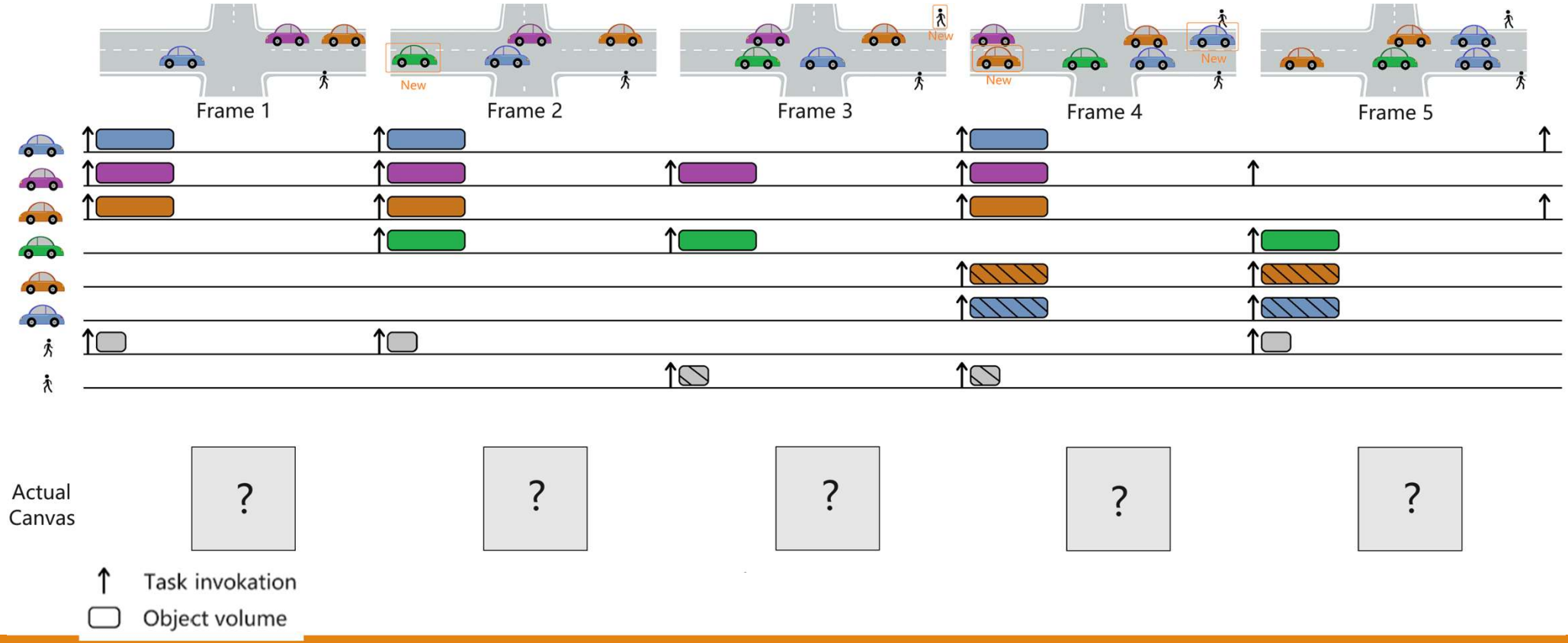
Scheduling Problem: an Example



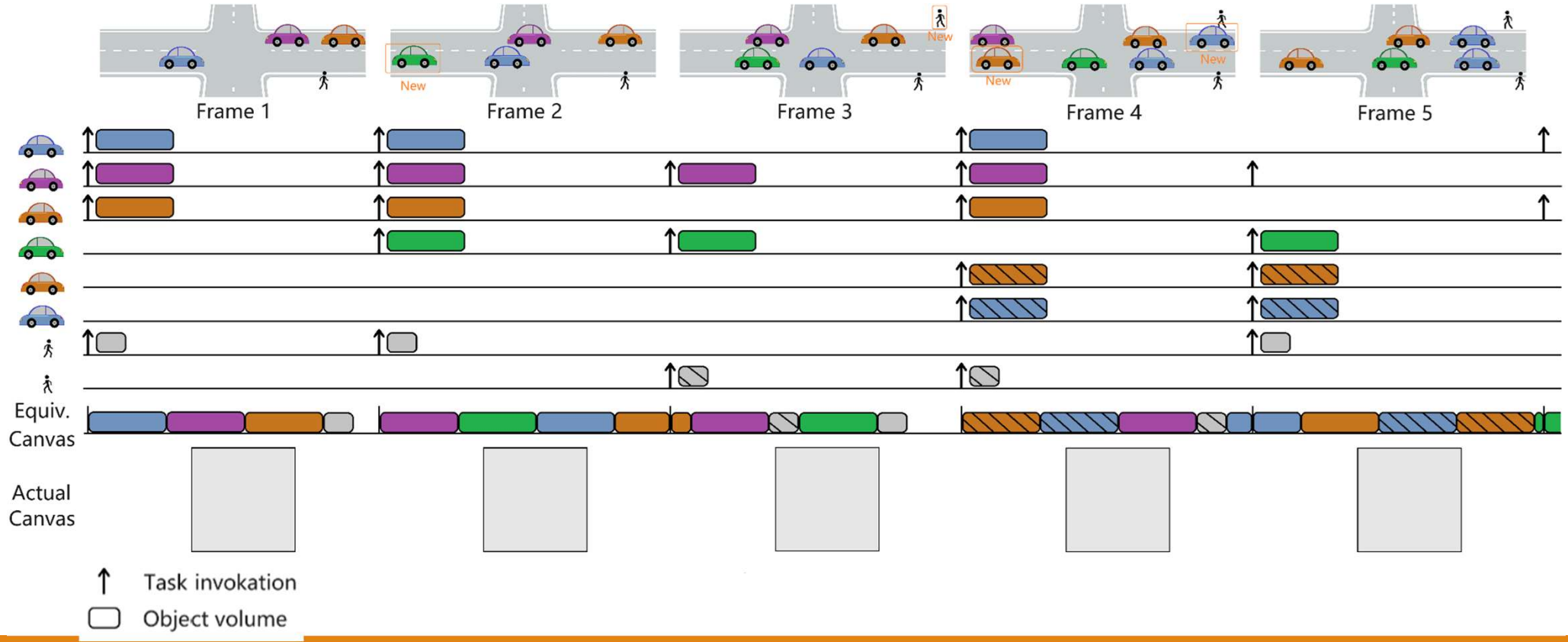
↑ Task invokation

▭ Object volume

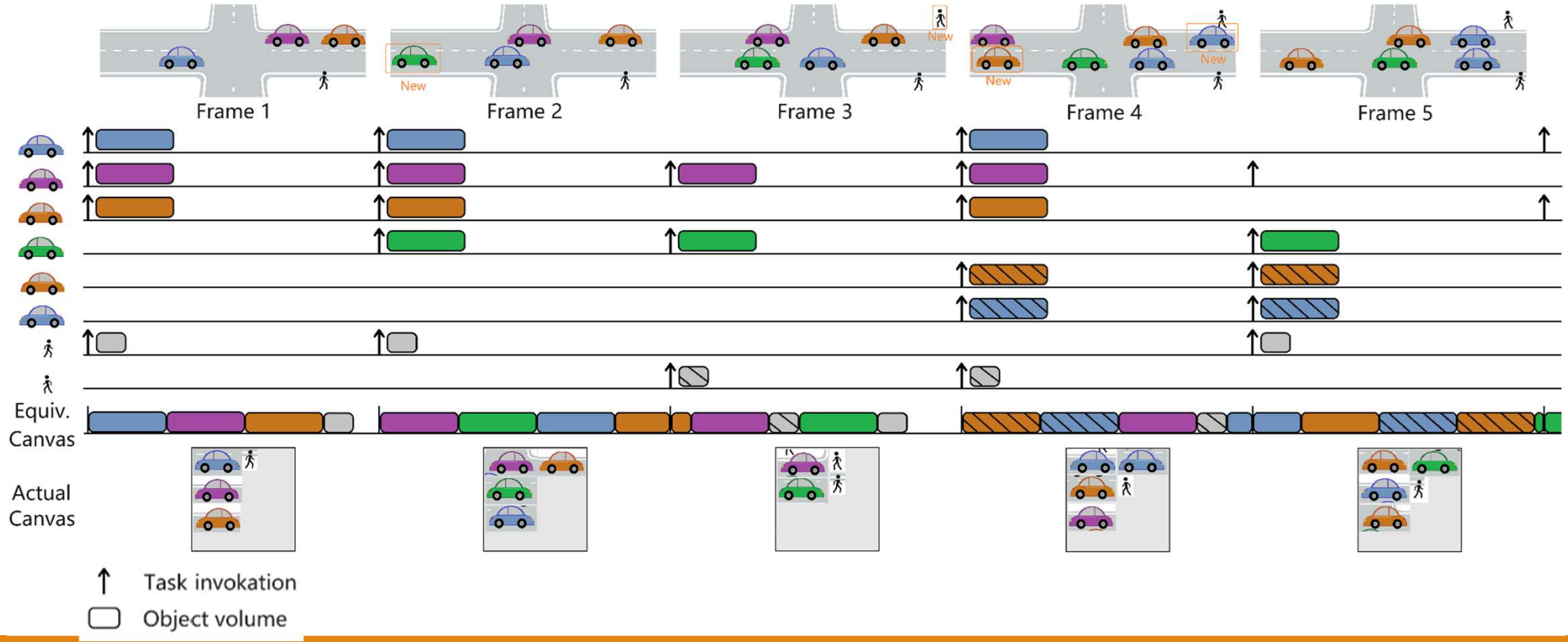
Scheduling Problem: an Example



Scheduling Problem: an Example



Scheduling Problem: an Example

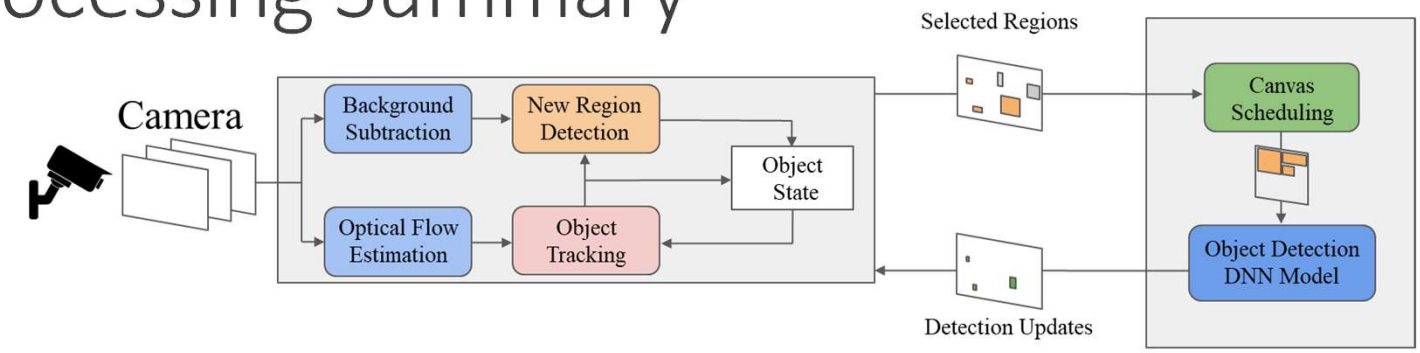


Canvas Scheduling Challenges

Two constraints must be satisfied:

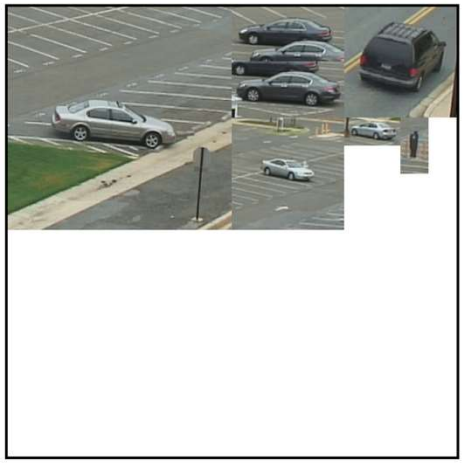
- Object inspection deadlines must be met (a real-time scheduling problem)
- Each canvas capacity constraints must be met (a knapsack packing problem)
- Challenge: Optimal order of packing to maximize utilized capacity is not the same as the optimal order for meeting deadlines.

Canvas-based Processing Summary



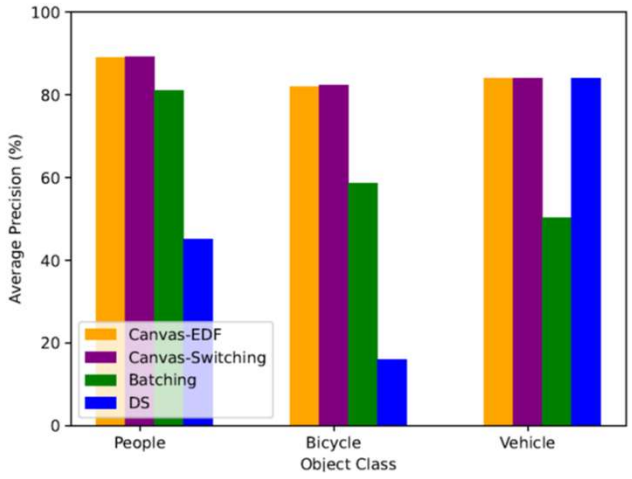
+

+



1 x 256·256 4 x 128·128 2 x 64·64 1 x 512·512

Batching: T = 97ms Canvas: T = 43ms



How to Reduce Edge AI Footprint?

Optimizing for a single inference:

- ✔ Reduce input size (input approximation): Smaller inputs results in less downstream processing and communication.
- ✔ Reduce model size (model approximation):
  - ✔ Prune less important nodes/links
  - ✔ Perform quantization to reduce resolution
  - ✔ Exit early
- ✔ Improve processing efficiency (e.g., batch inputs)

Optimizing across multiple inferences:

- Break monolithic models into specialized experts
- Cache re-usable partial results
- Balance resource utilization

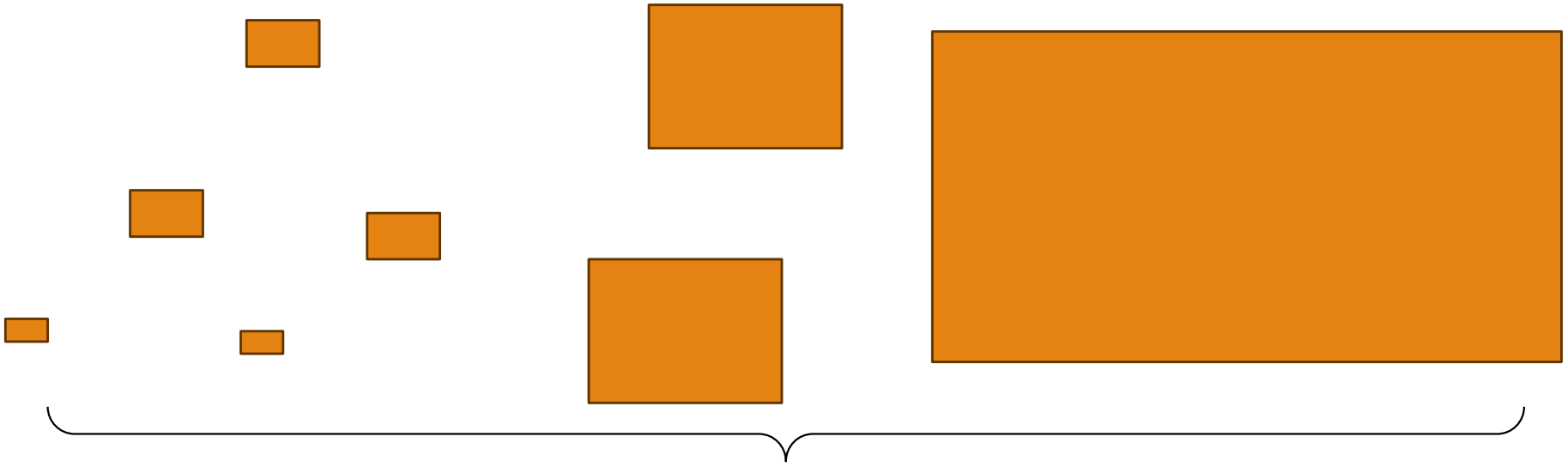
Inference Time

Optimizing across multiple inferences

Optimizing across multiple inferences:

- Break monolithic models into specialized experts
- Cache reusable partial results
- Balance resource utilization

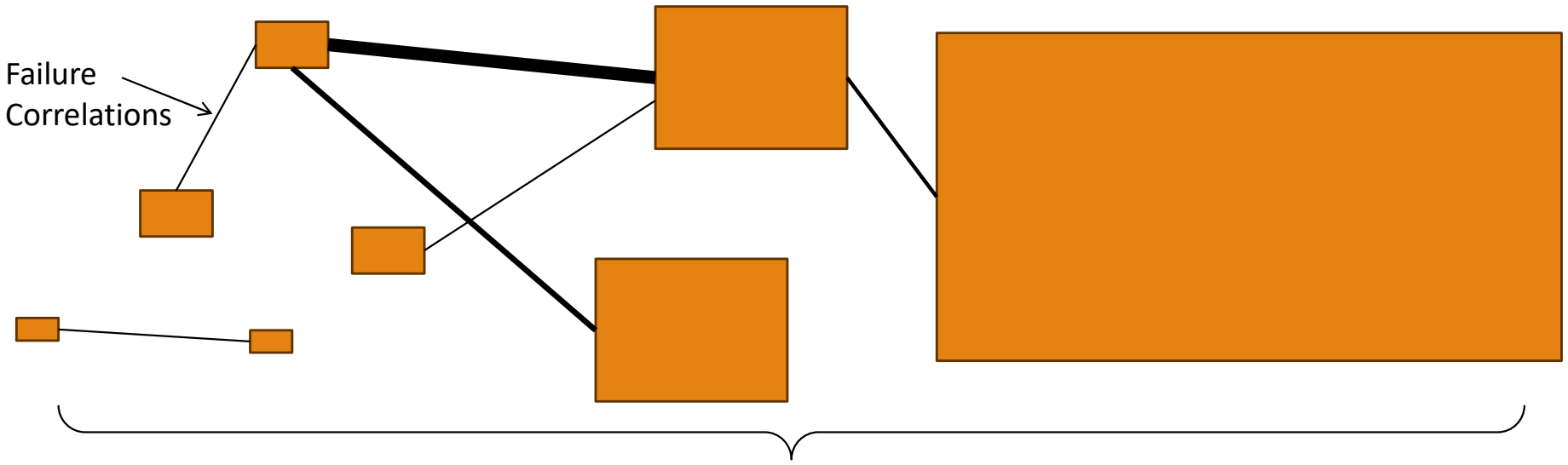
Break Monolithic Models into Specialized Experts



AI models of different quality and different computational complexity

What is the optimal sequence of models to try in order to minimize average latency to successful decision?

Break Monolithic Models into Specialized Experts



What is the optimal sequence of models to try in order to minimize average latency to successful decision?

Example: Request Routing to (Classification) Experts

Simple heuristic: Try a quick and dirty classifier first. If it fails, invoke progressively more advanced models.

Other considerations?

Example: Request Routing to (Classification) Experts

Simple heuristic: Try a quick and dirty classifier first. If it fails, invoke progressively more advanced models.

A more general observation: Failure of a classifier to properly classify an input offers insights into the selection of the next classifier to try (e.g., try a "better" classifier, or try a classifier based on a different modality, or try a classifier that has minimum correlations with the failed classifiers, etc.)

Leveraging partial knowledge:

- A classifier that fails to return the specific target class (e.g., Toyota Camry) can nevertheless suggest a higher-level category to which the class belongs (e.g., Sedan)
- The partial knowledge can help inform the selection of the next classifier (e.g., one specialized in Sedans, not one specialized in trucks or busses)

Example: Request Routing to (Classification) Experts

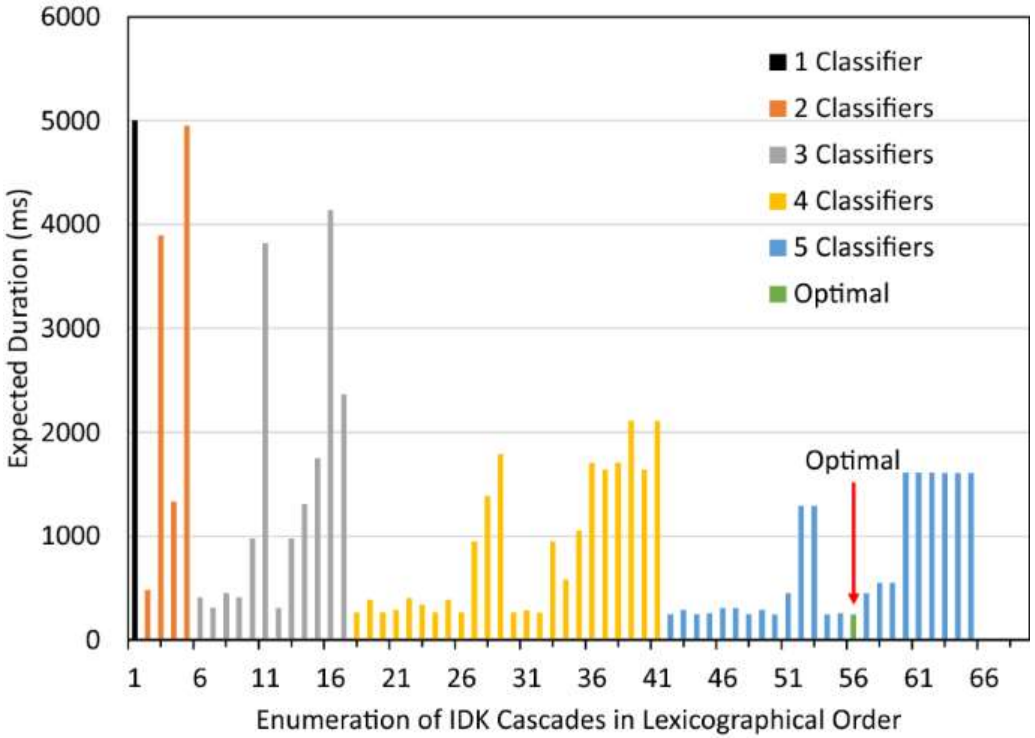
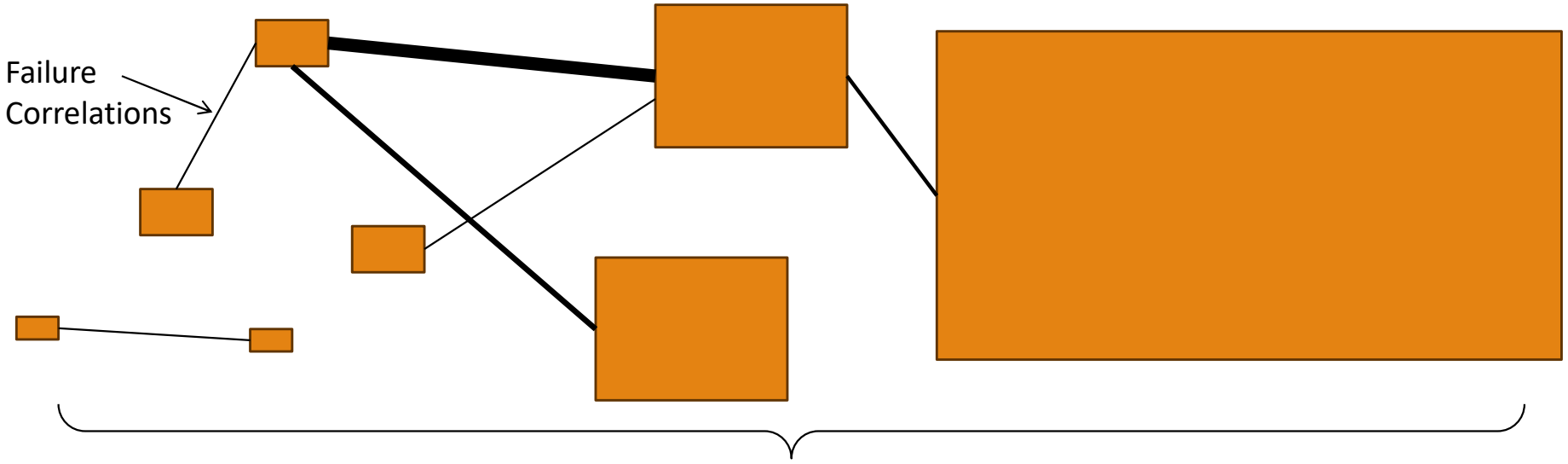


Figure shows expected durations of execution of classifier sequences made of acoustic, seismic, and camera-based object classifiers.

Significant average latency reductions are possible without jeopardizing expected accuracy by optimally ordering the execution sequence of different classifiers (where each escalates to the next when unsure)

Adding Cache Size Constraints



What is the optimal sequence of models to try in order to minimize average latency to successful decision subject to constraint on total model storage?

Considerations in Resource Utilization

- In general, different experts may need different amounts of each of a number of resources (memory, CPU threads, GPUs threads, communication bandwidth, etc).
- The section of the experts to use, as well as request scheduling/routing algorithms must ensure that no individual resource is overloaded.

Conclusions

The embedded systems community can substantially contribute to the development of edge AI and domain specific foundation models for (different categories of) intelligent embedded applications

Foundation model development for embedded time-series data requires innovations in self-supervised pretraining. Examples include:

- (i) Handling frequency domain data (e.g., attention design across harmonics)
- (ii) Handling multimodal data (e.g., attention design across modalities)
- (iii) Handling multi-vantage data

Resulting models can be used to reduce reliance on labeled data, thus removing one of the key impediments to deploying robust AI for the embedded systems domain

Future directions:

- A “micro foundation model” pre-training architecture
- Extensions to multi-vantage sensing
- Integration with LLMs for improved higher-level reasoning (about activities, intents, causes, roles, etc)