

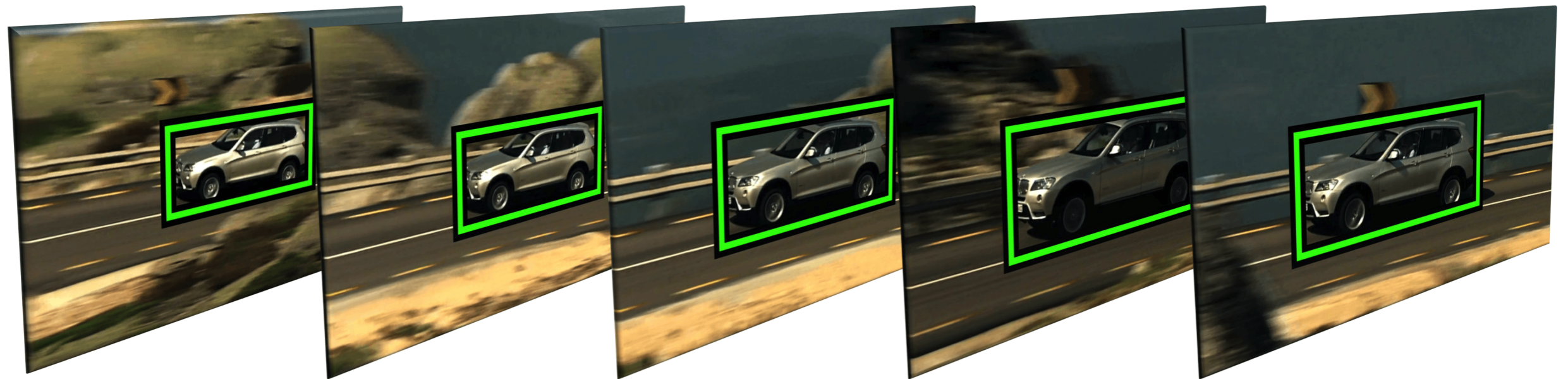


Task-Level Modeling

Case study: Continuous Vision Applications

Presenter: Yiming Gan

Application Domain

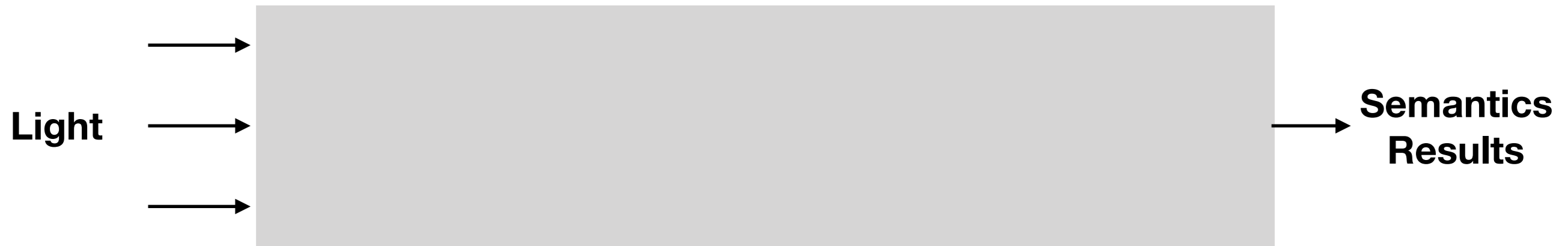


Continuous Vision

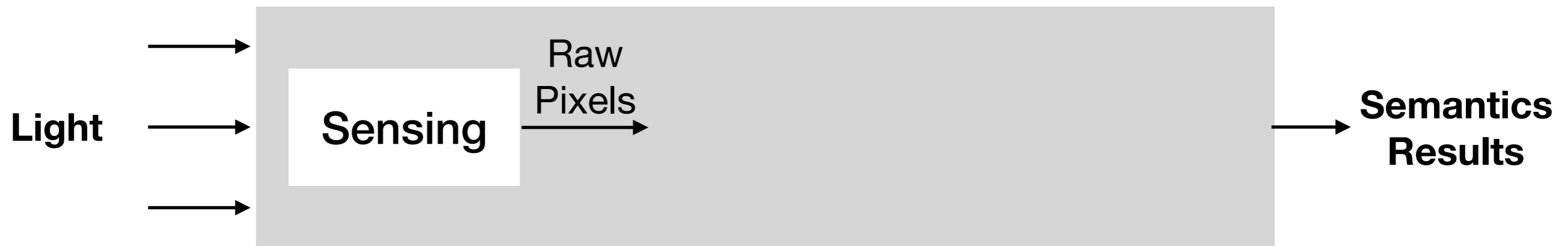
Bottlenecks

- Long Latency
 - 100 ms per frame
 - A. Censi, “Low-latency event-based visual odometry”, ICRA 2014.
- High Energy Consumptions
 - Drain battery in one hour using camera on iPhone 6s
 - Apple technique report, 2016
- High Memory Bandwidth Requirements
 - 1M Pixels image sensor running at 30 FPS needs 16 GB/s
 - Raj Parihar, “Frame Buffer Design For Image Sensor Array”.

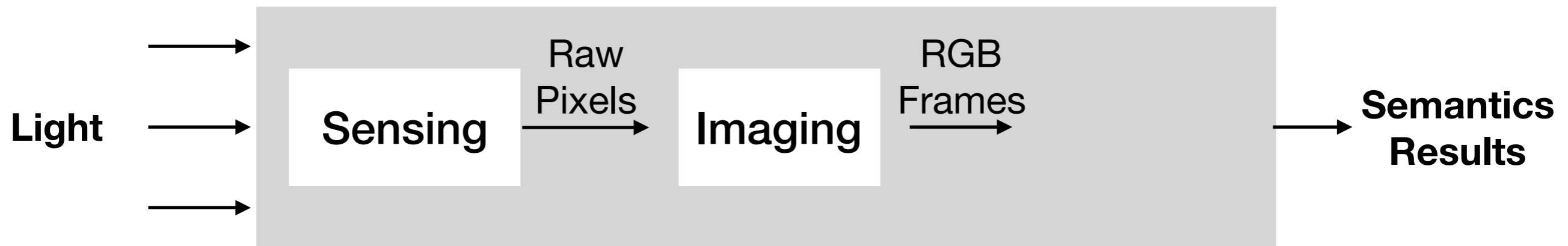
Continuous Vision Pipeline



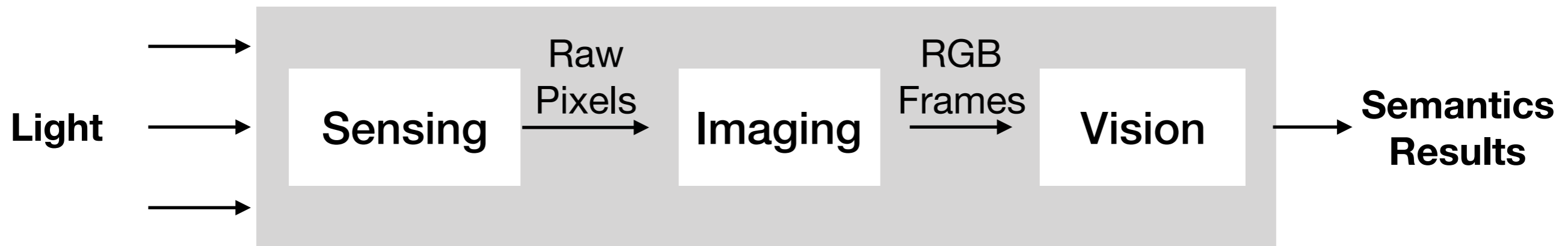
Continuous Vision Pipeline



Continuous Vision Pipeline



Continuous Vision Pipeline



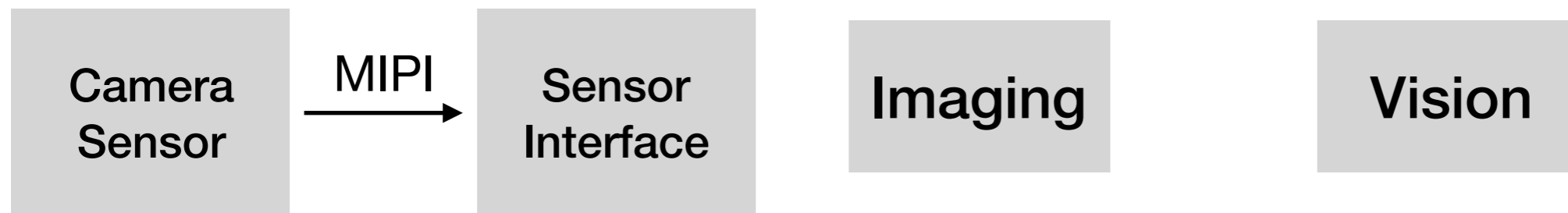
Mobile SoCs for Continuous Vision Pipeline

Sensing

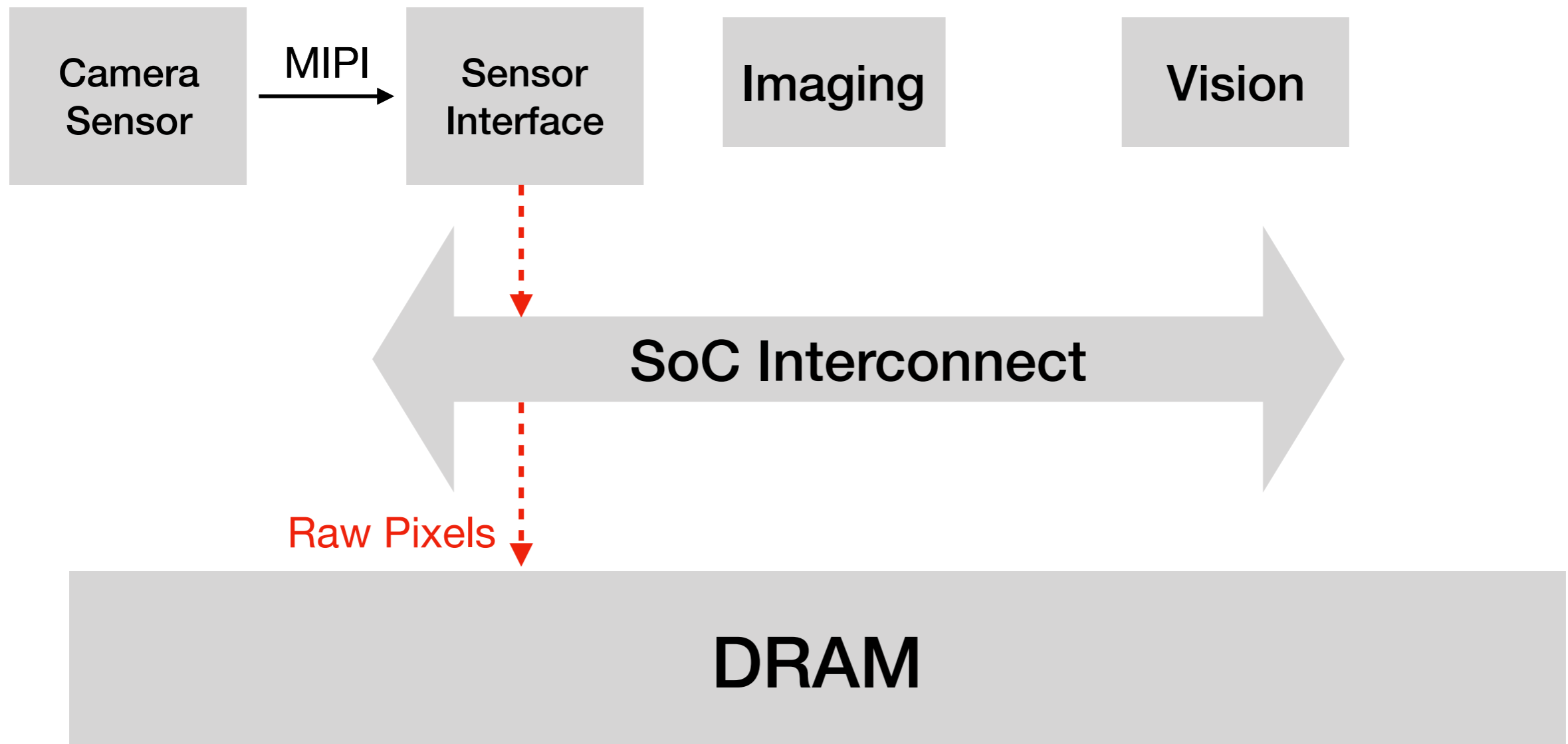
Imaging

Vision

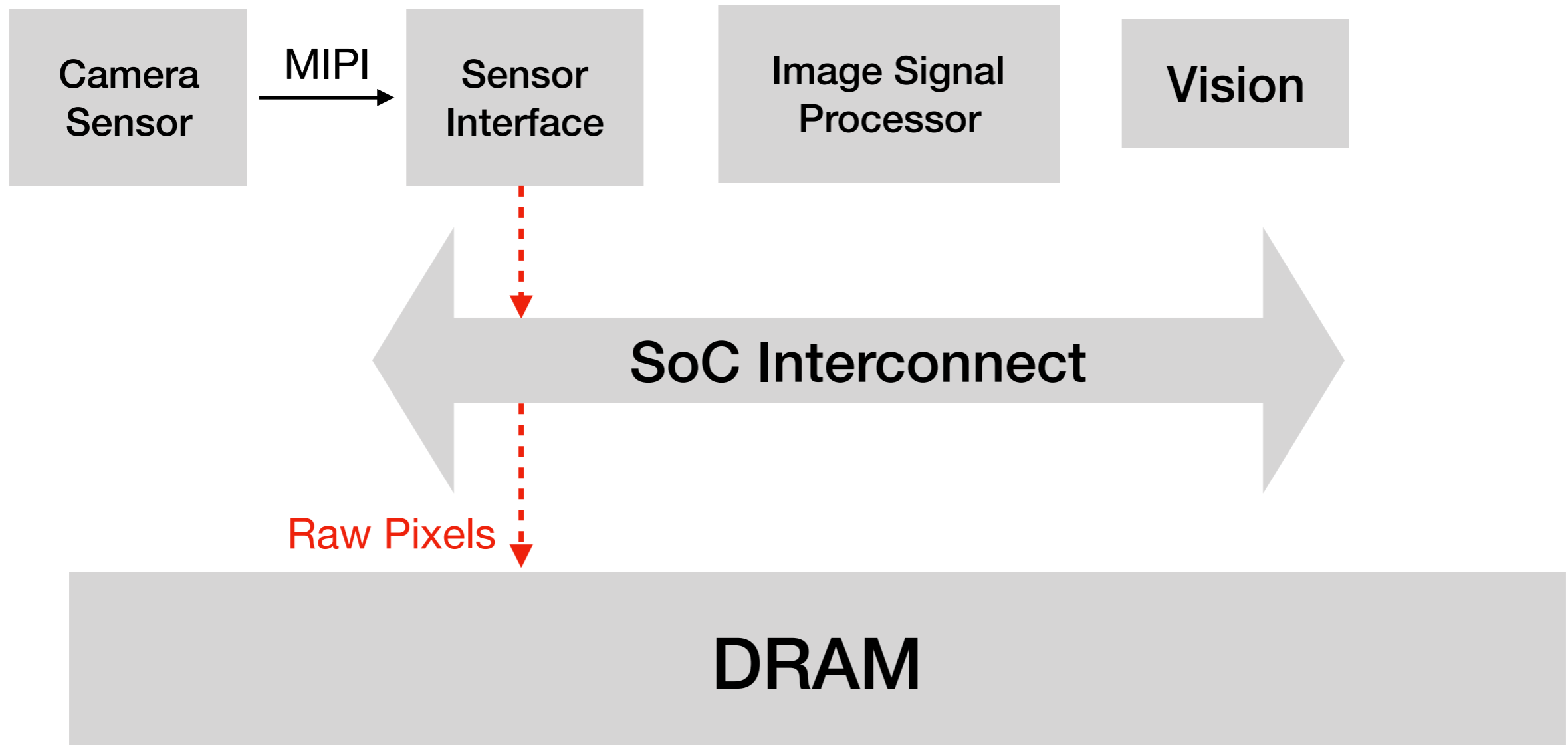
Mobile SoCs for Continuous Vision Pipeline



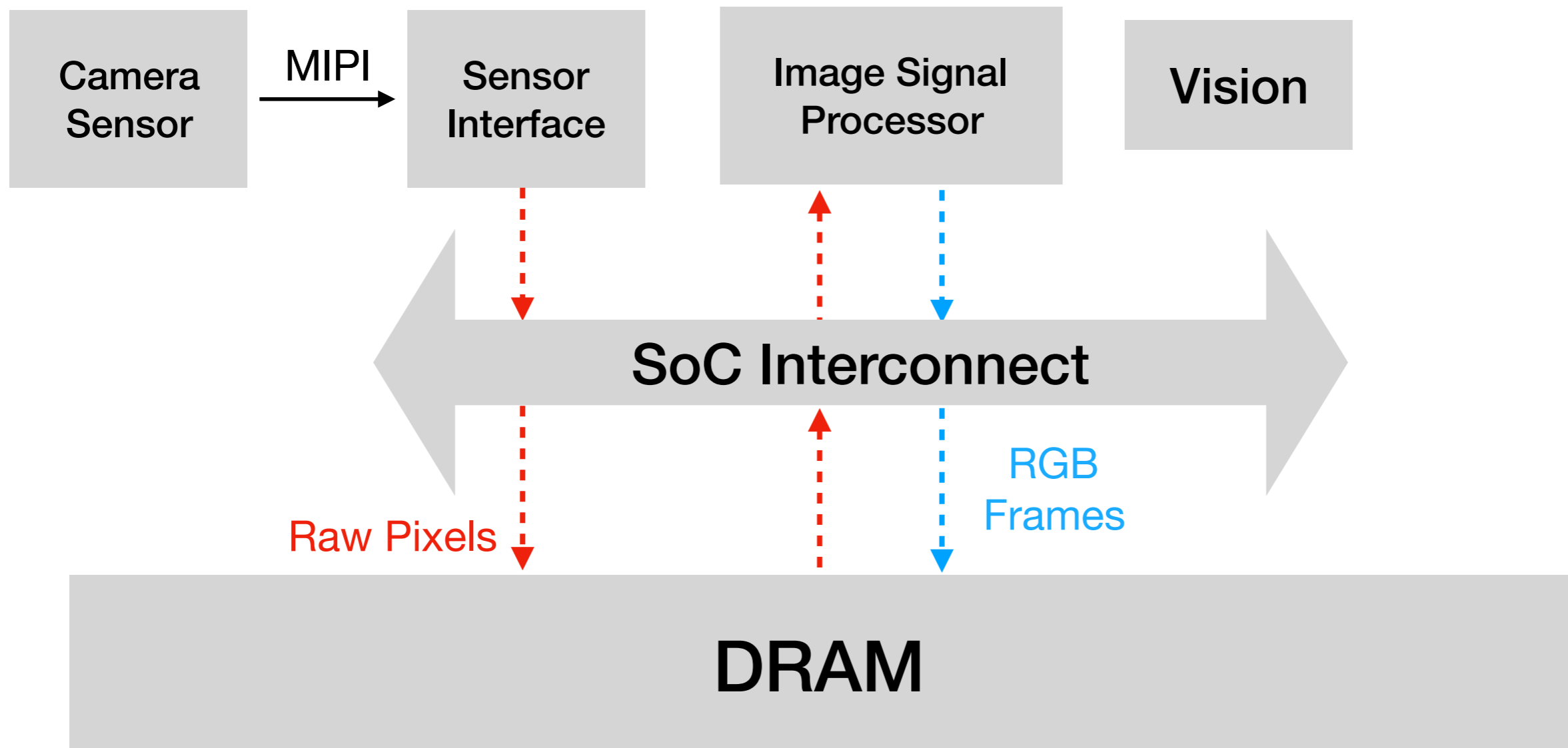
Mobile SoCs for Continuous Vision Pipeline



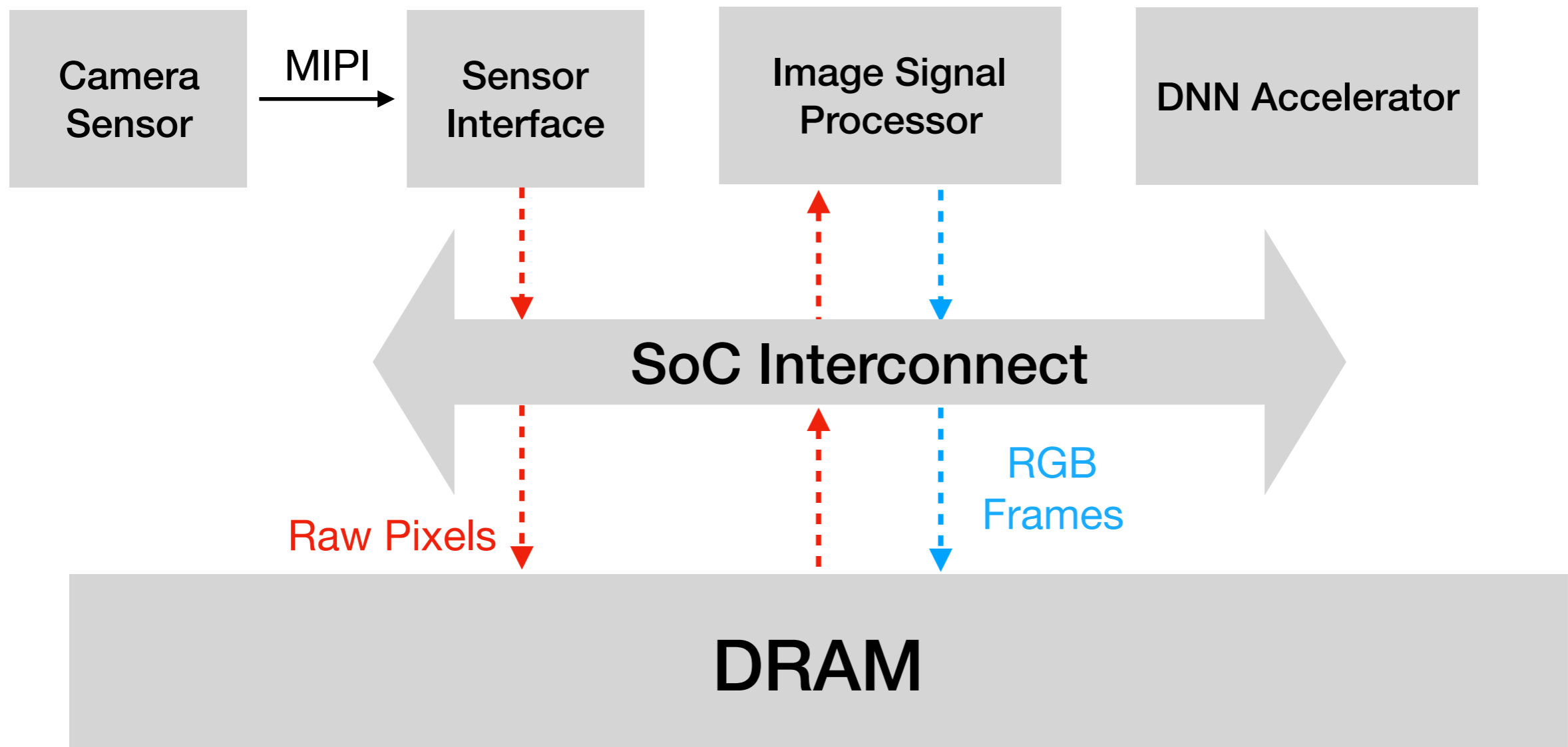
Mobile SoCs for Continuous Vision Pipeline



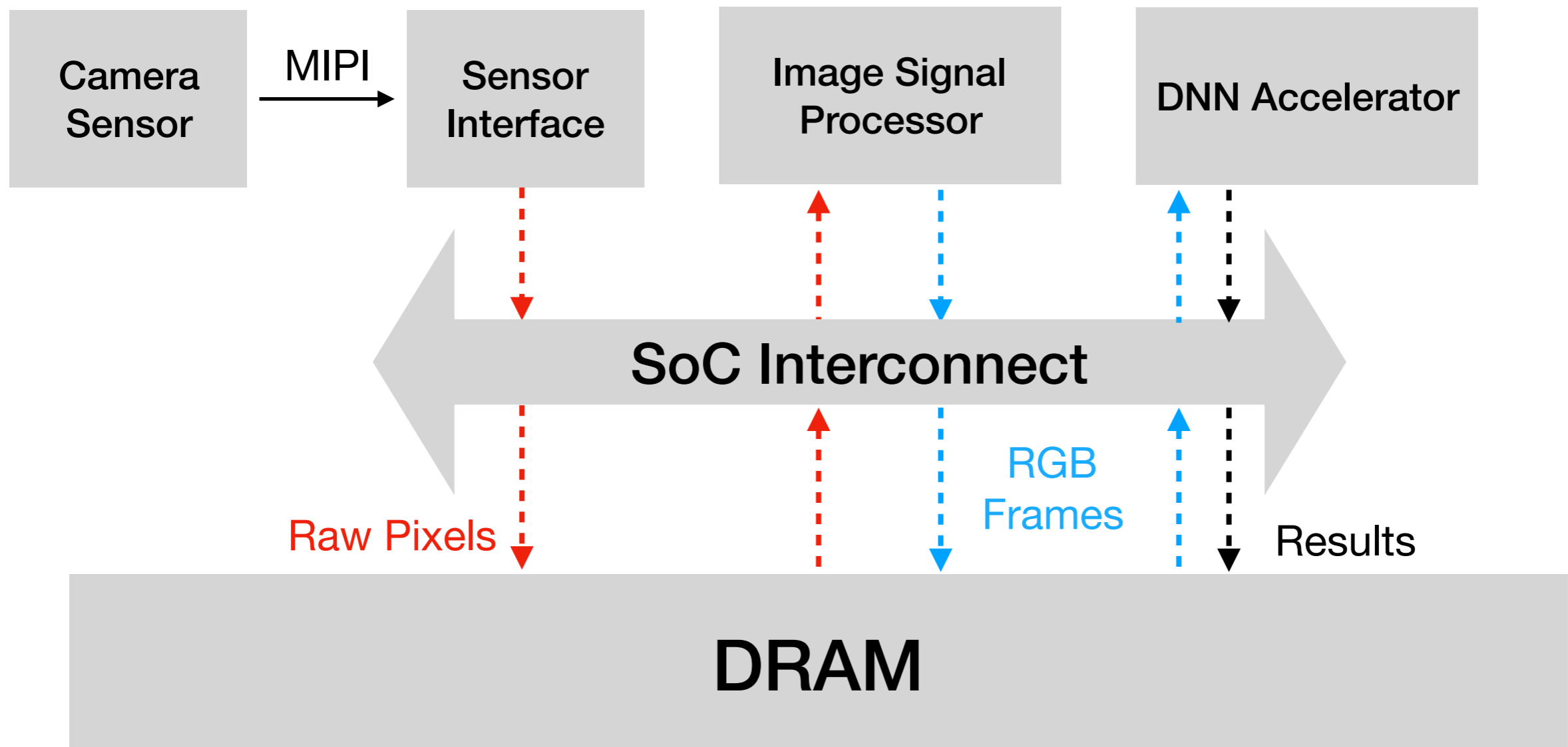
Mobile SoCs for Continuous Vision Pipeline



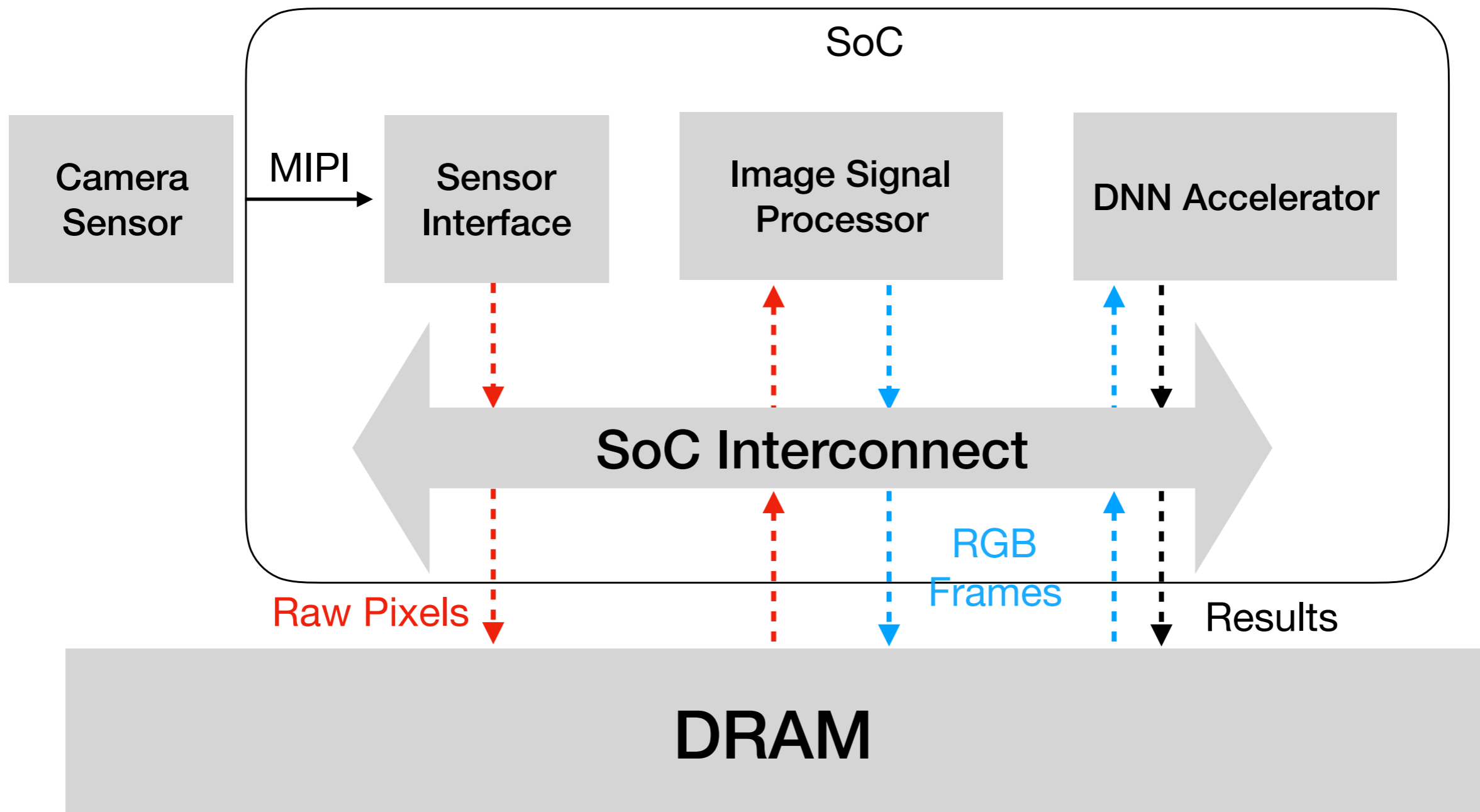
Mobile SoCs for Continuous Vision Pipeline



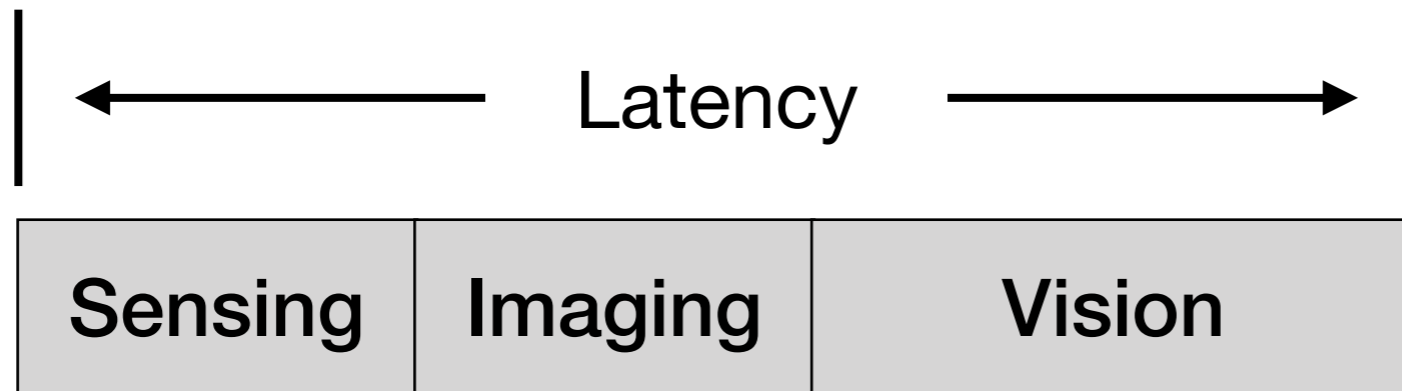
Mobile SoCs for Continuous Vision Pipeline



Mobile SoCs for Continuous Vision Pipeline

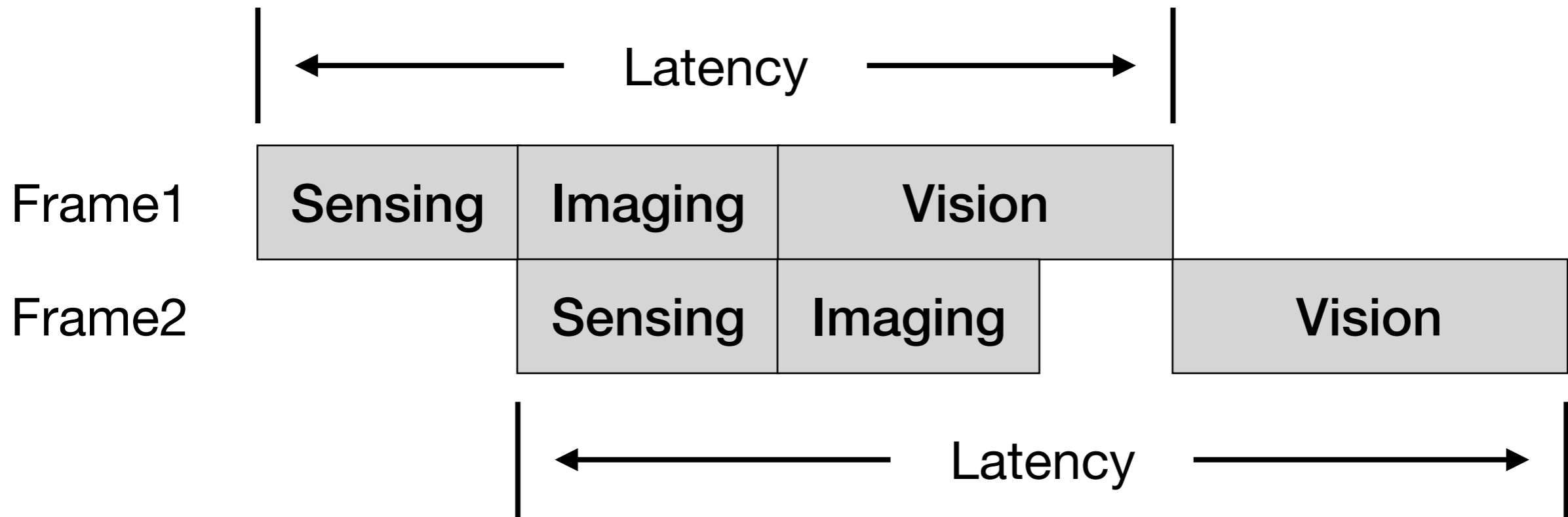


Latency in Continuous Vision Pipeline

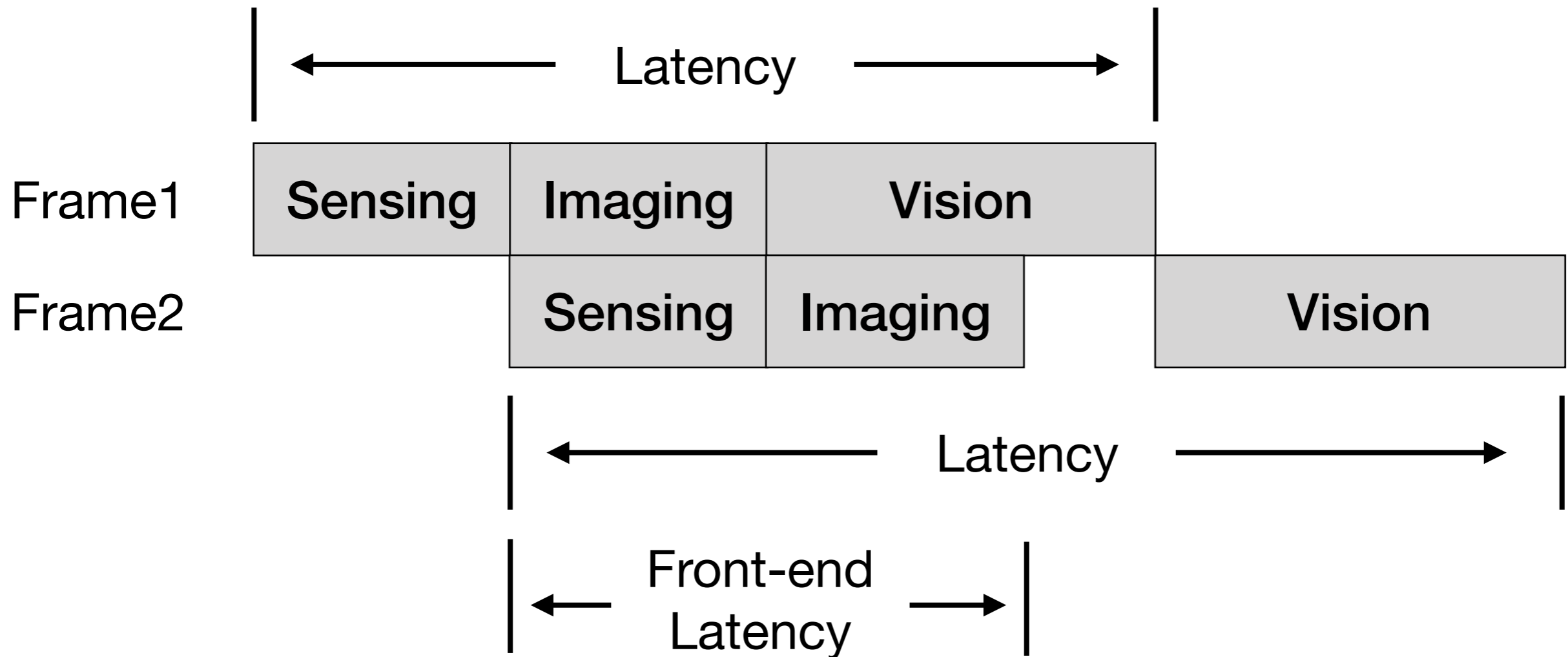


Frame1

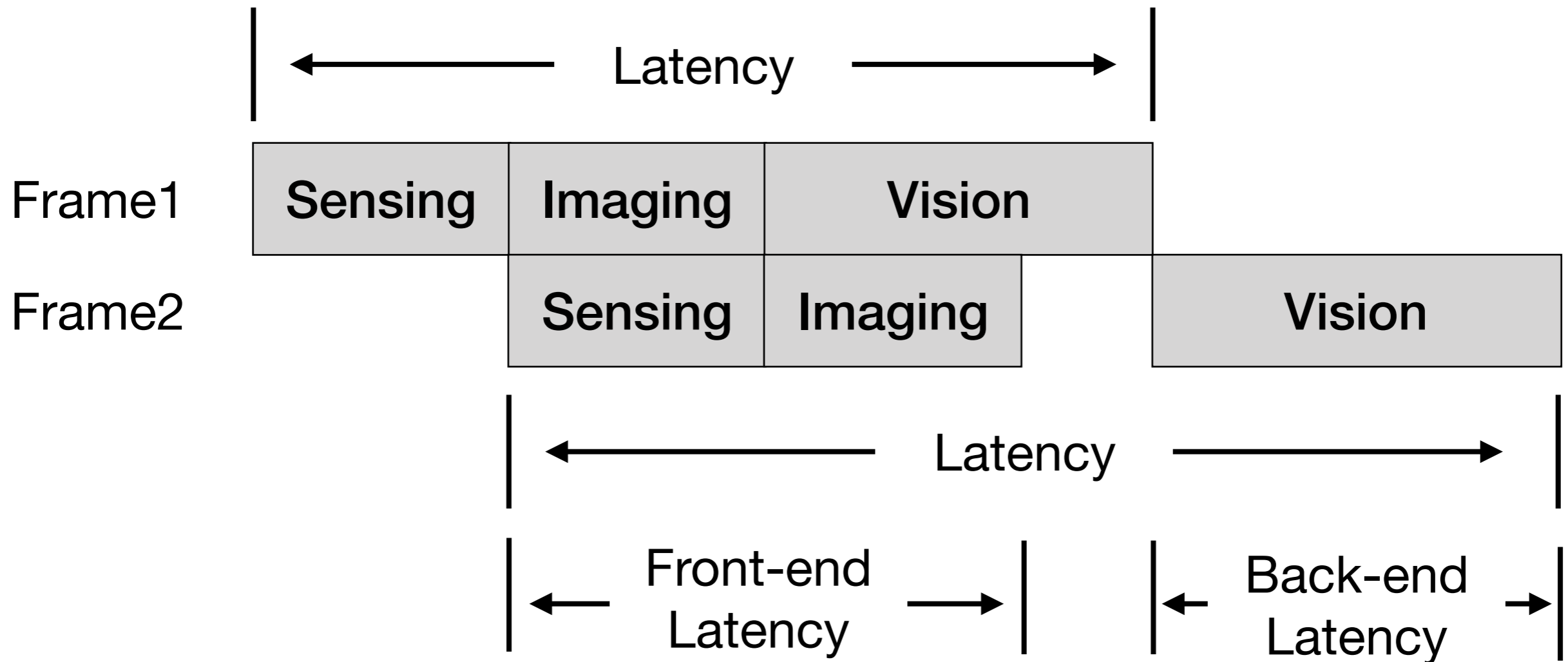
Latency in Continuous Vision Pipeline



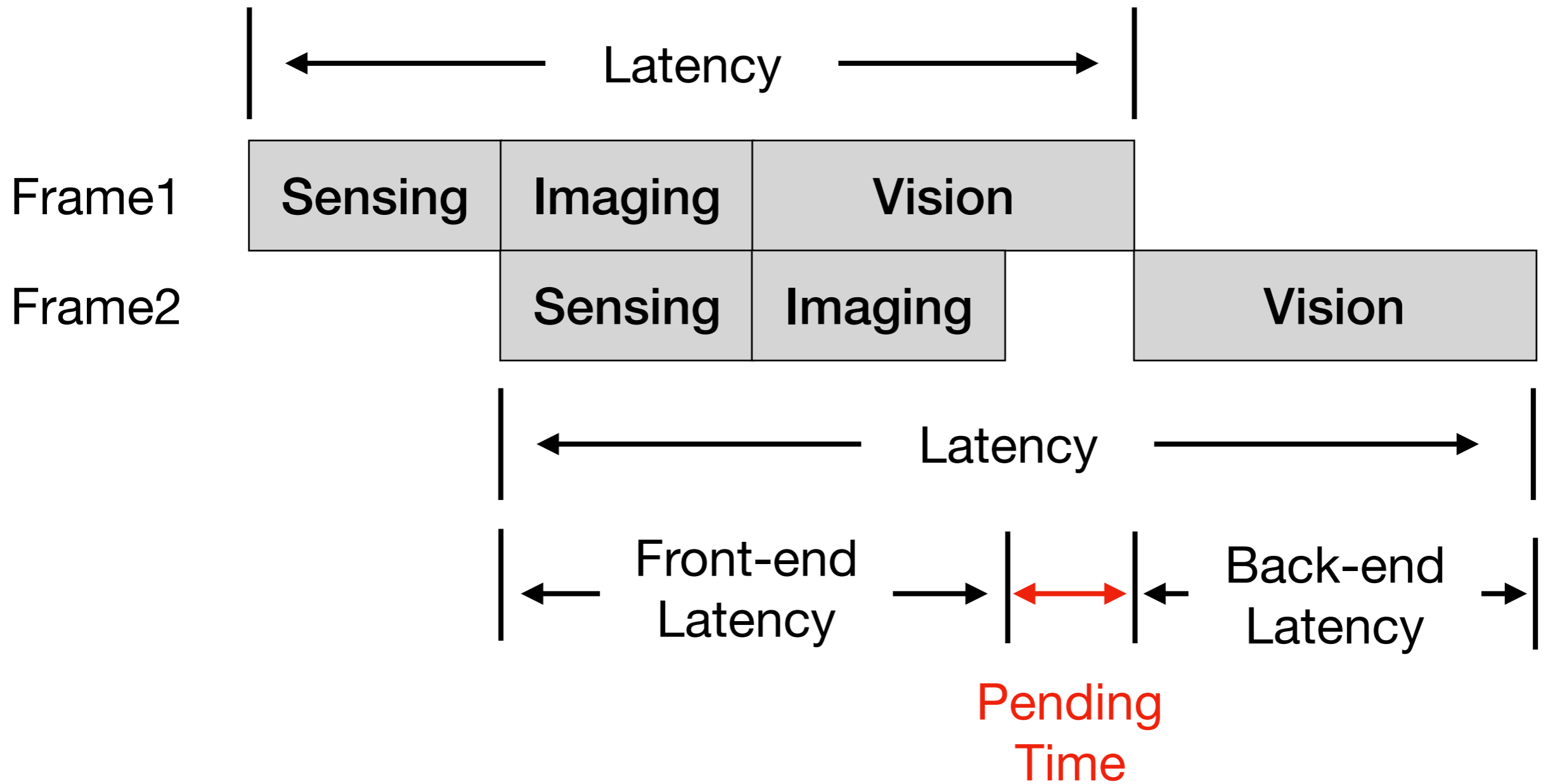
Latency in Continuous Vision Pipeline



Latency in Continuous Vision Pipeline



Latency in Continuous Vision Pipeline



Platform Architect Modeling

Sensor	AR 1335 Sensor, 30 FPS AR1335 Datasheet
ISP	Nvidia TX2 ISP, 30 FPS
DNN Accelerator	20x20 Systolic Array 1.5 MB SRAM, 500 MHz
Bus	AMBA4 AXI4, 32 GB/s

Platform Architect Modeling

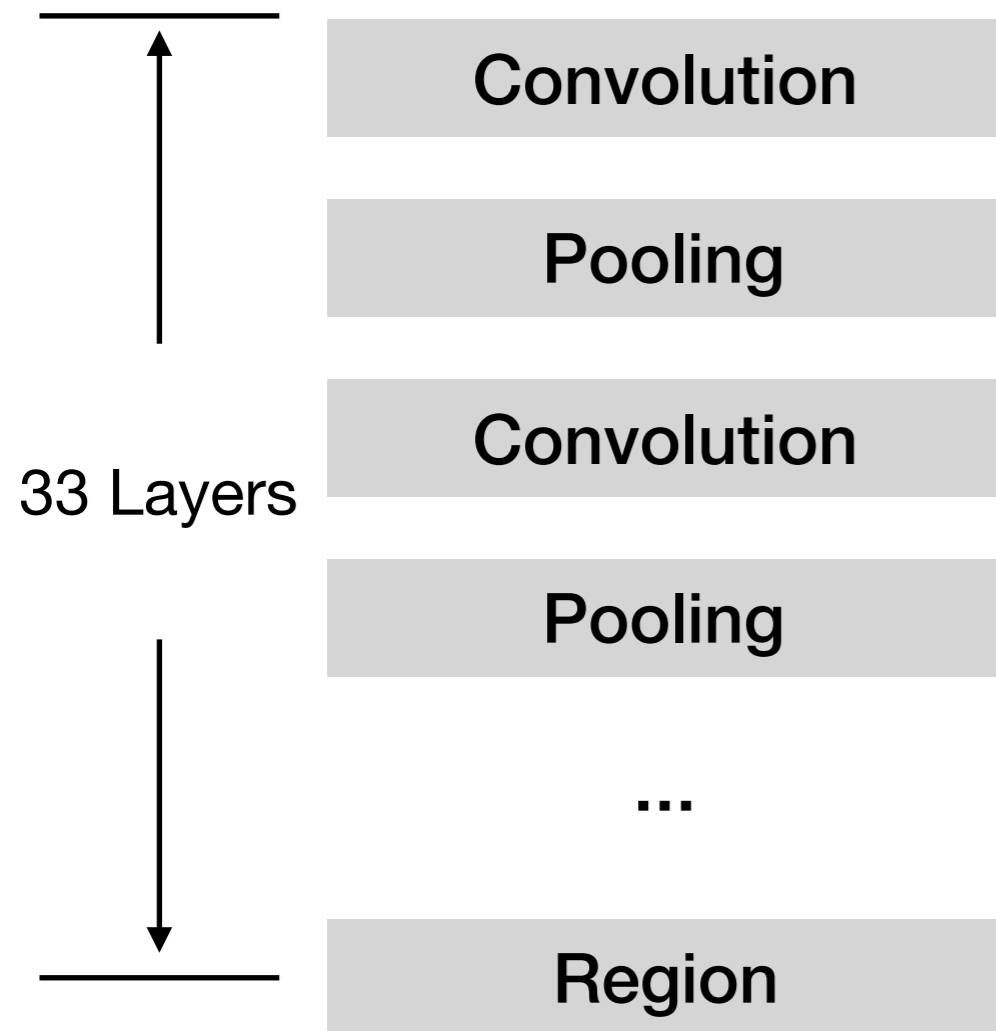
The screenshot displays a software modeling tool interface with the following components:

- Menu Bar:** File, Edit, Project, Diagram, Check, Export, Tools, View, Plugins, Help.
- Toolbar:** Includes icons for file operations and a 'Perspective: current' dropdown.
- Left Panel:** A tree view showing a project structure with folders for 'Creation' (HW, Workload, Mapping) and 'Exploration' (Simulation, Results).
- Central Panel:** Titled 'Creation and Exploration Domains - Flow: Workload - Domain: Workload'. It features tabs for 'Tasks' and 'Deadlines'. A filter is set to 'GTL_20:Task'. Below is a table of tasks:

Name	priority	iterations	traffic_type	processing_cyc	rd_bytes	rd_buffer	rd_buffer_sz
1 Sensing		0	5 Stochastic	200	0	0	40
2 Imaging		0	5 Stochastic	200	6291456	0	40
3 Vision		0	5 Stochastic	500	2000000	0	40

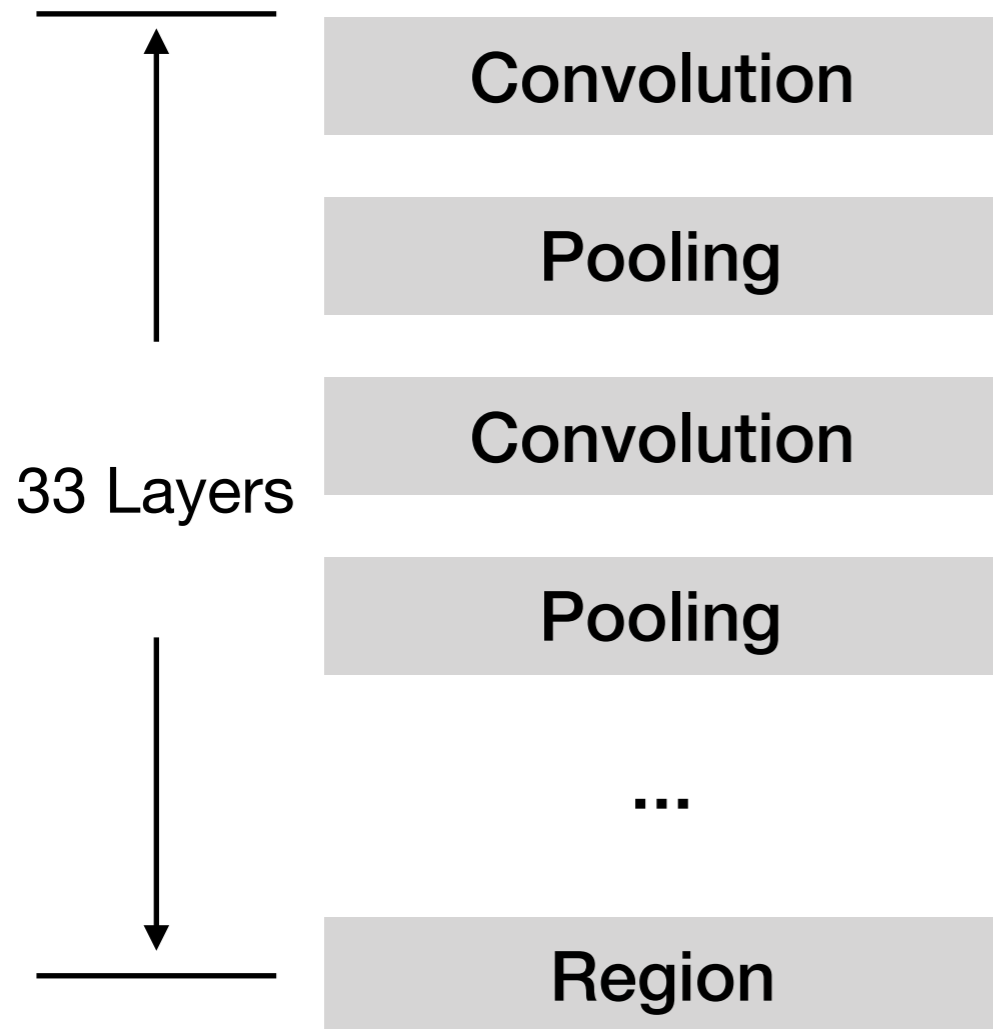
- Bottom Left Panel:** 'Connections' and 'Advanced' tabs, both with filter fields and empty data tables.
- Right Panel:** A diagram window titled '/HARDWARE/Workload/Workload' showing a data flow graph on a grid. It contains three orange blocks: 'Sensing', 'Imaging', and 'Vision'. 'Sensing' is connected to 'Imaging' via a channel labeled 'p_put[0]_get[0] C', and 'Imaging' is connected to 'Vision' via a channel labeled 'p_put[0]_get[0] C_1'.
- Status Bar:** Shows '85%' zoom, '1:1' aspect ratio, and '497x382' dimensions.

Different Vision Algorithms

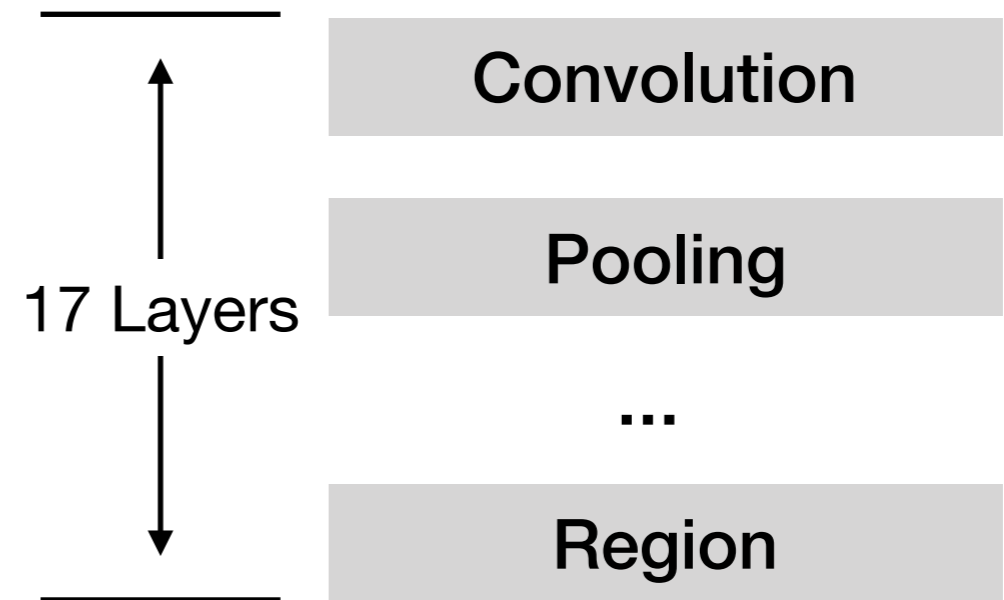


YOLOv2: 448x448 Input
36 Billion MAC Ops

Different Vision Algorithms



**YOLOv2: 448x448 Input
36 Billion MAC Ops**

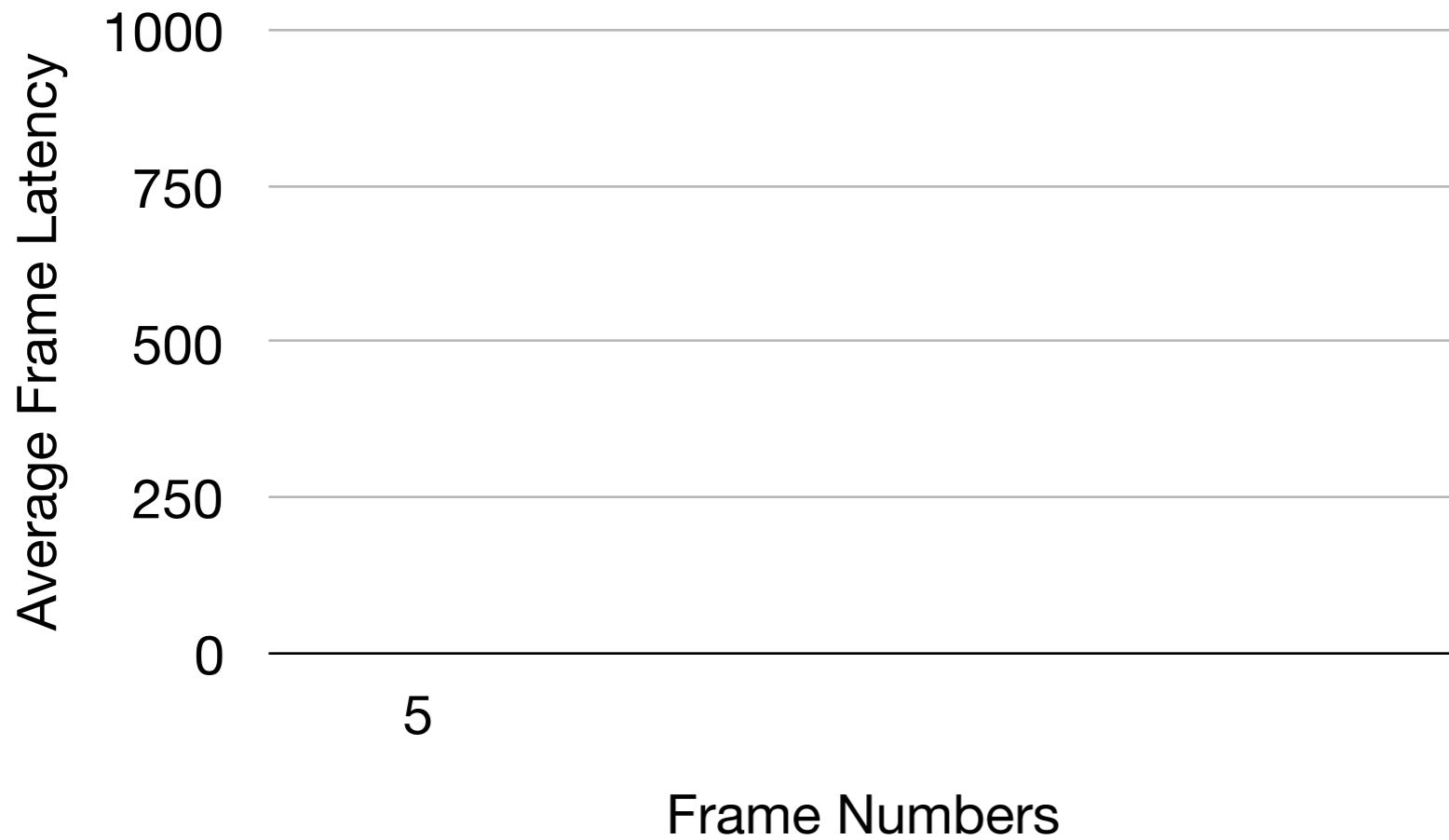


**TinyYOLO: 448x448 Input
7 Billion MAC Ops**

Task Modeling for YOLOv2

Task	Sensing	Imaging	Vision
Latency (ms)	34	34	102

Latency Results for YOLOv2



Latency Results for YOLOv2



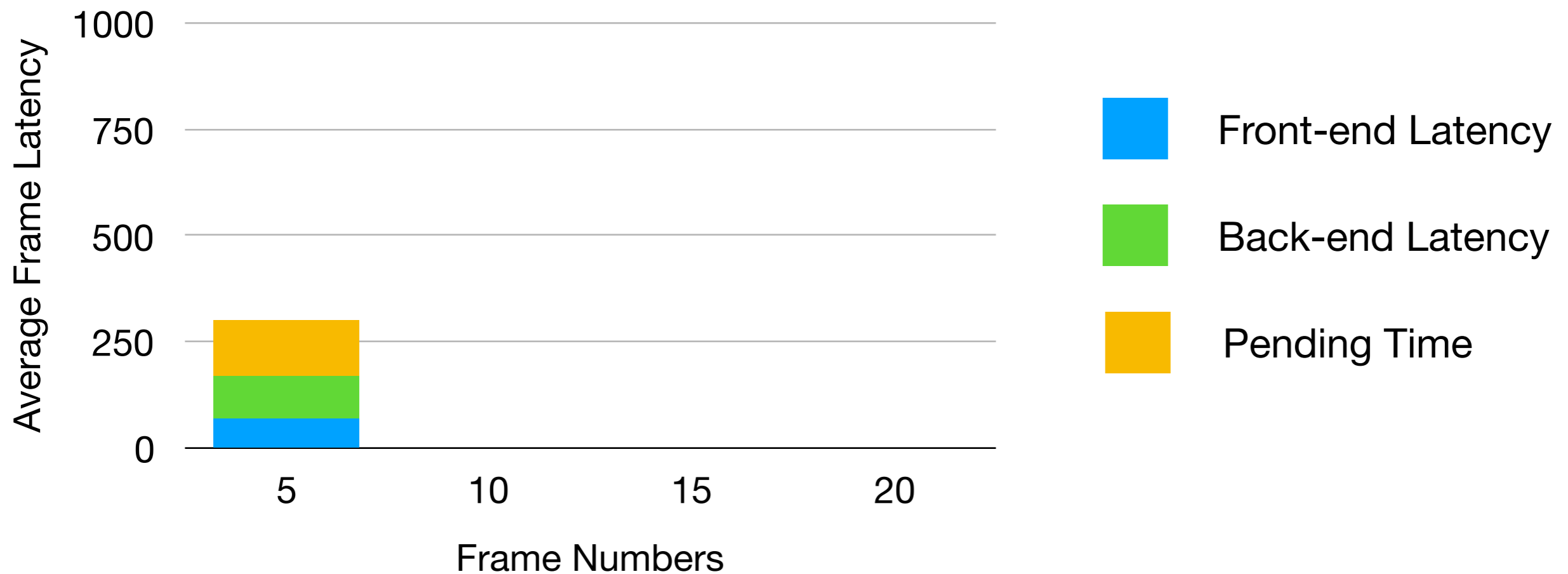
Latency Results for YOLOv2



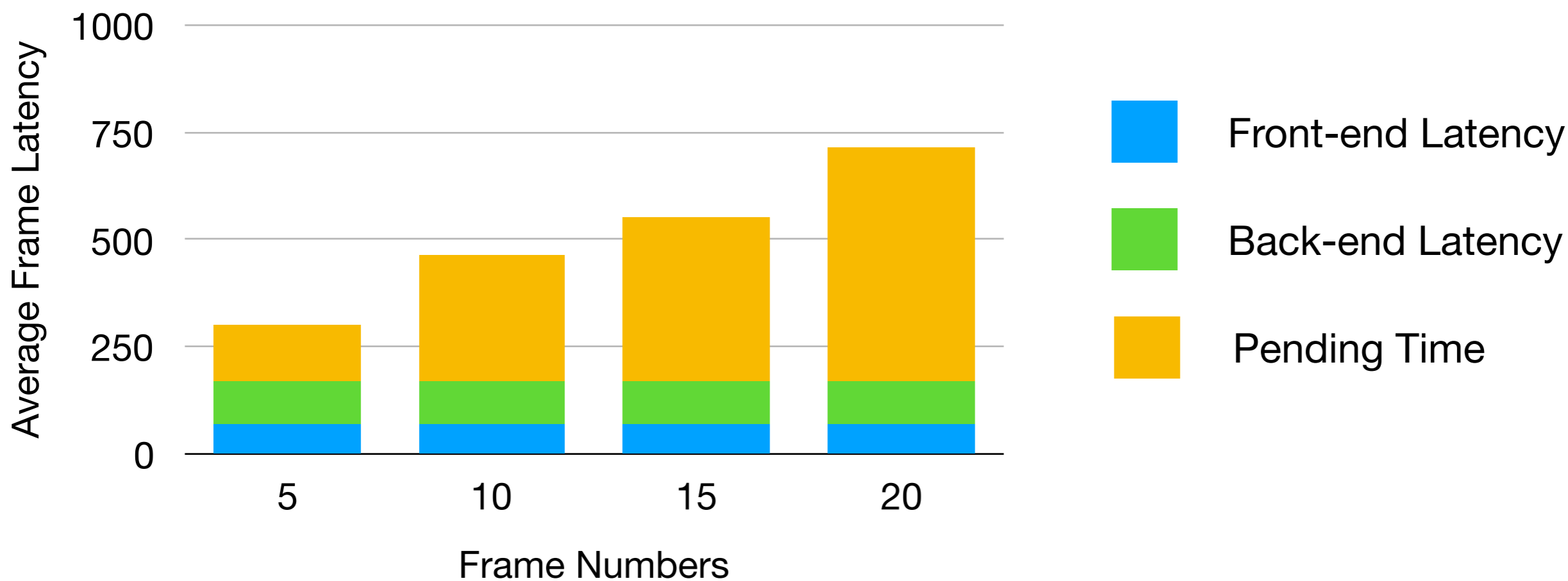
Latency Results for YOLOv2



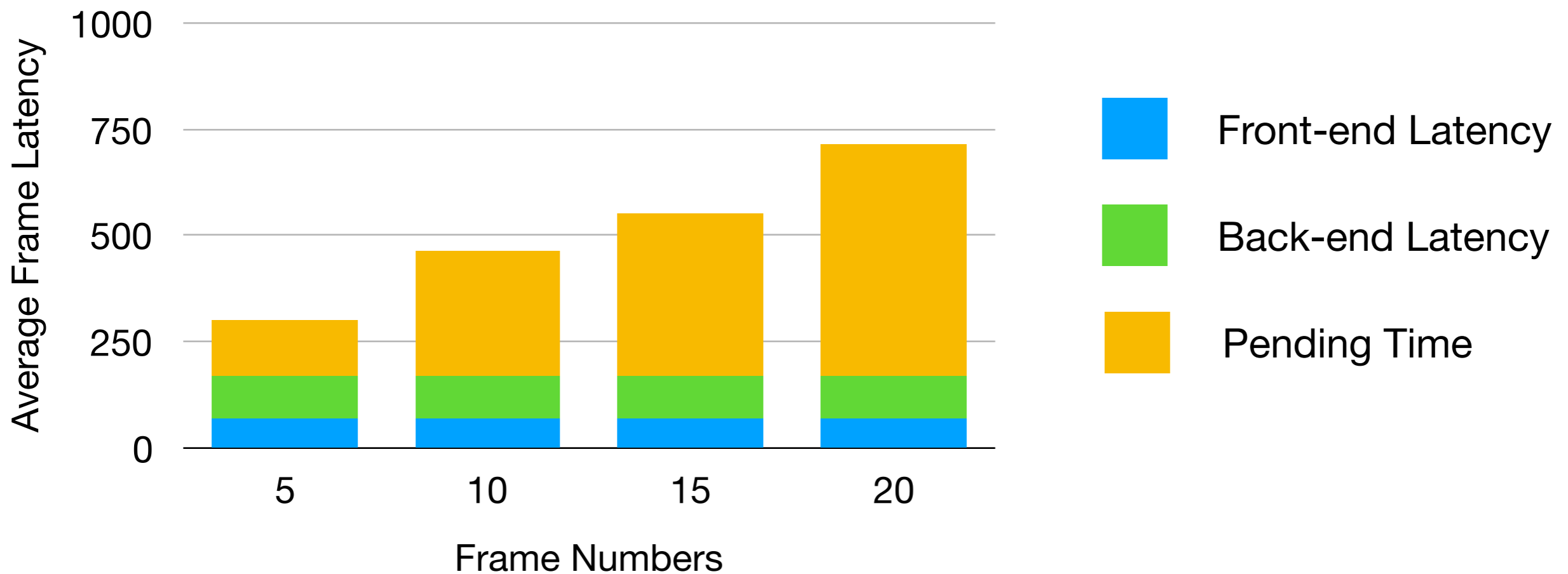
Latency Results for YOLOv2



Latency Results for YOLOv2



Latency Results for YOLOv2

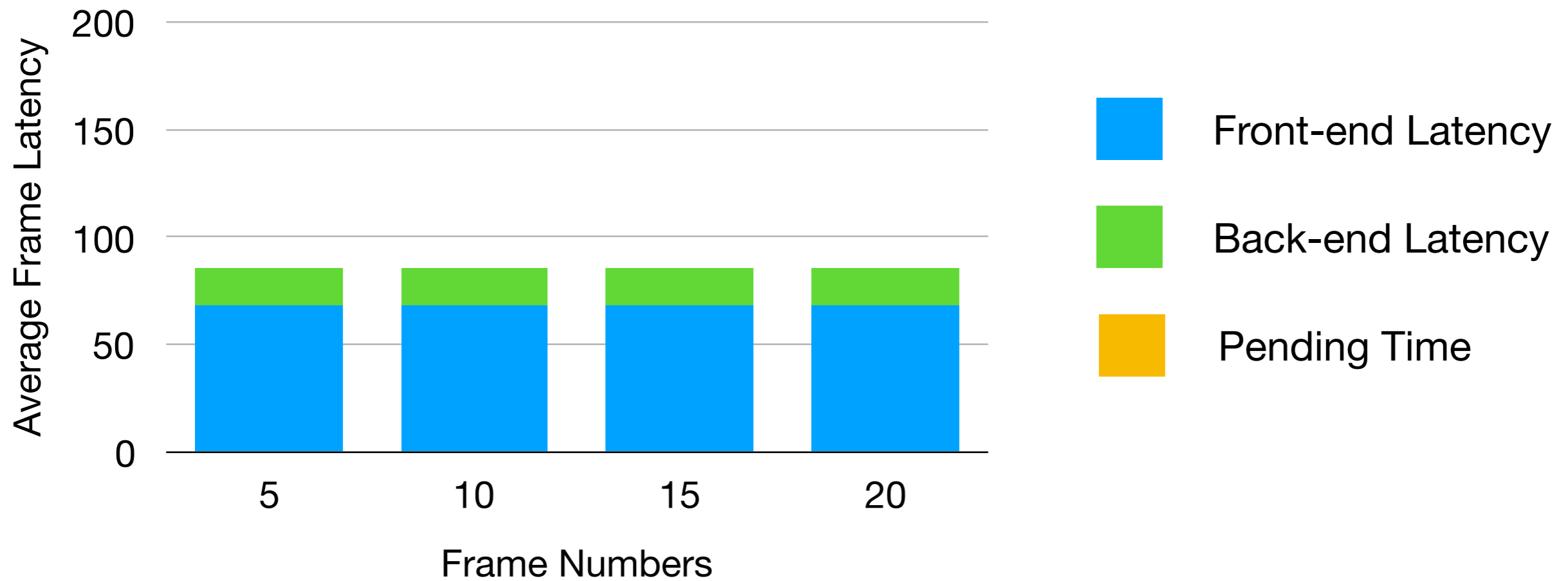


Conclusion 1: When the front-end is faster than the back-end, pending time is the bottleneck. Reducing pending time can significantly reduce the per frame latency.

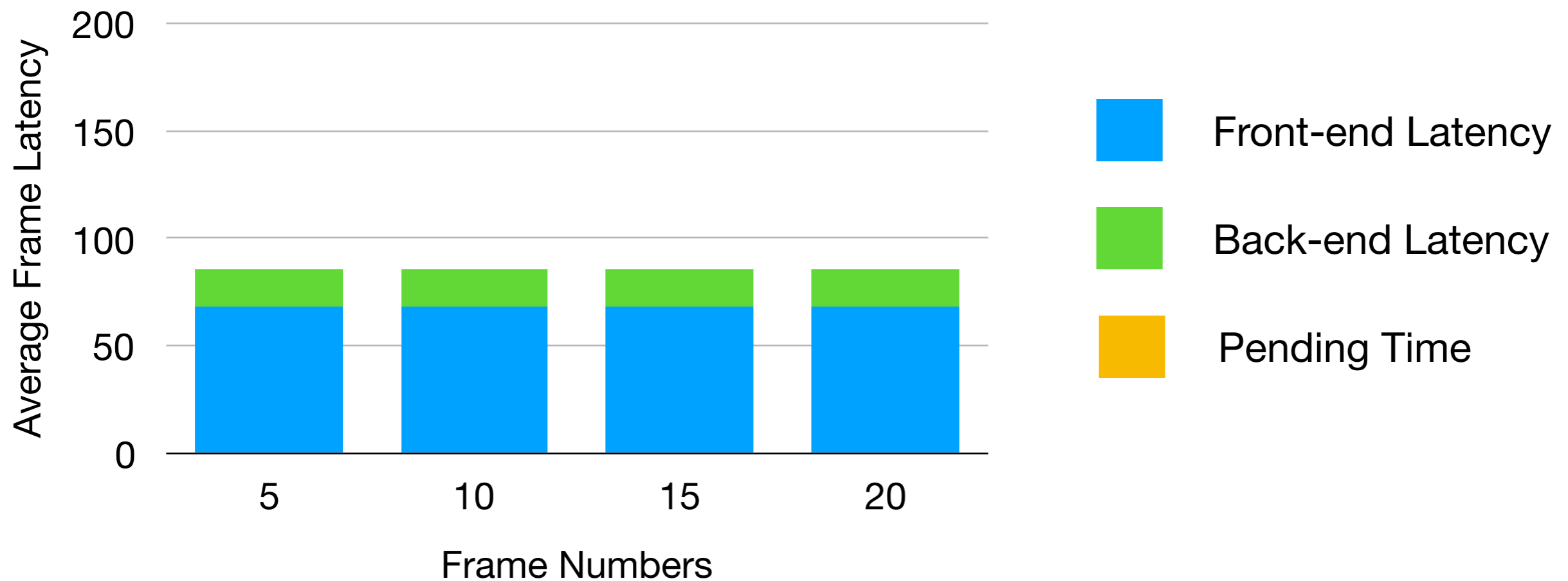
Task Modeling for TinyYOLO

Task	Sensing	Imaging	Vision
Latency (ms)	34	34	19.6

Latency Results for TinyYOLO

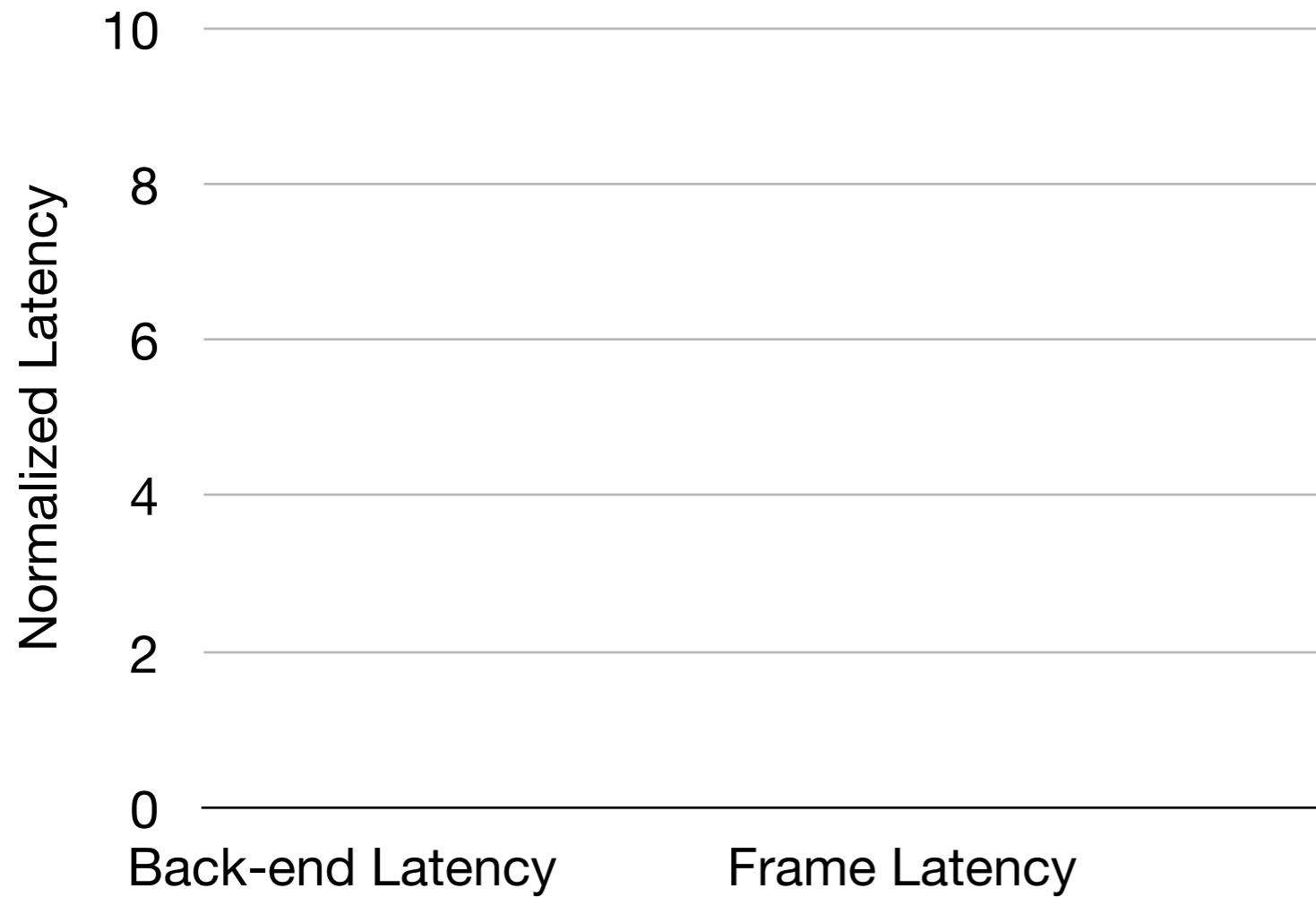


Latency Results for TinyYOLO

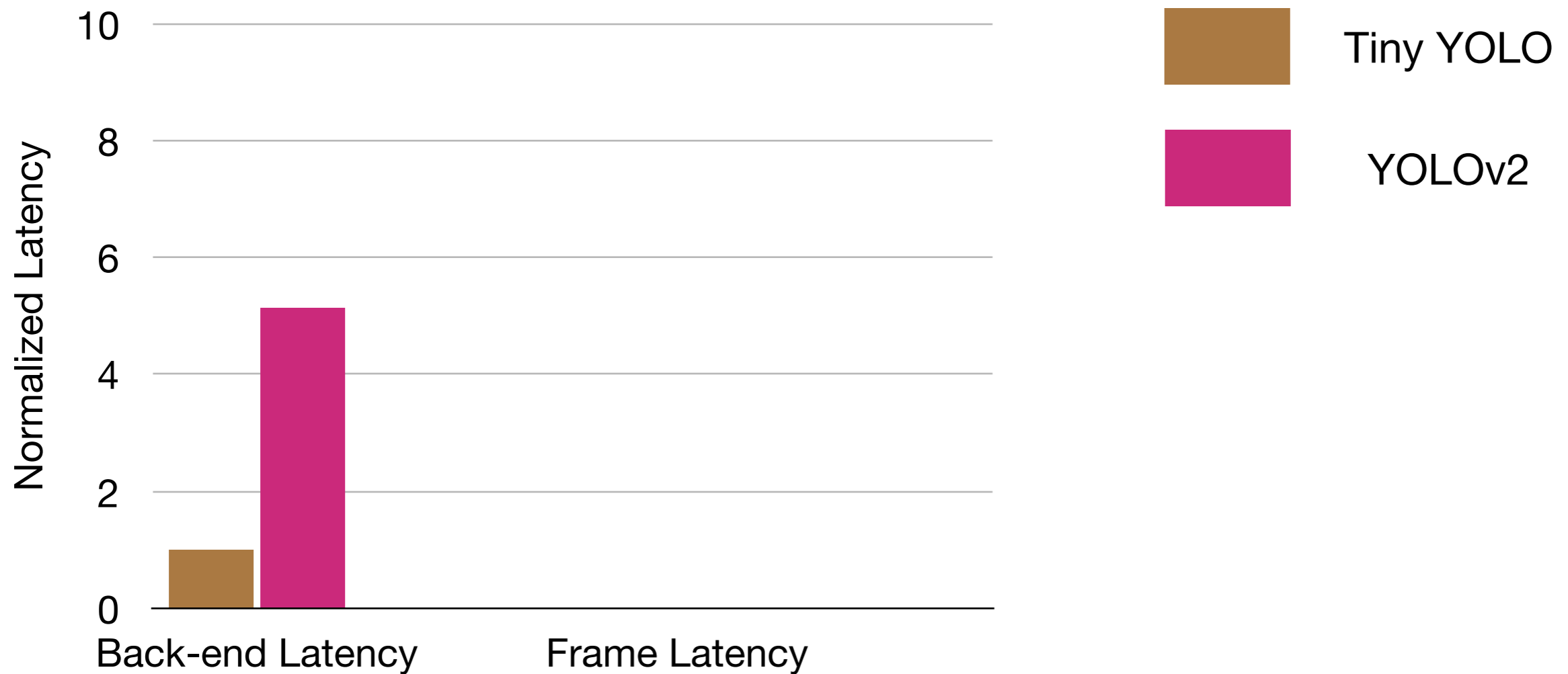


Conclusion 2: When the front-end is slower than the back-end, NPU is not fully utilized. The front-end is the bottleneck and optimizing the vision stage has marginal effect.

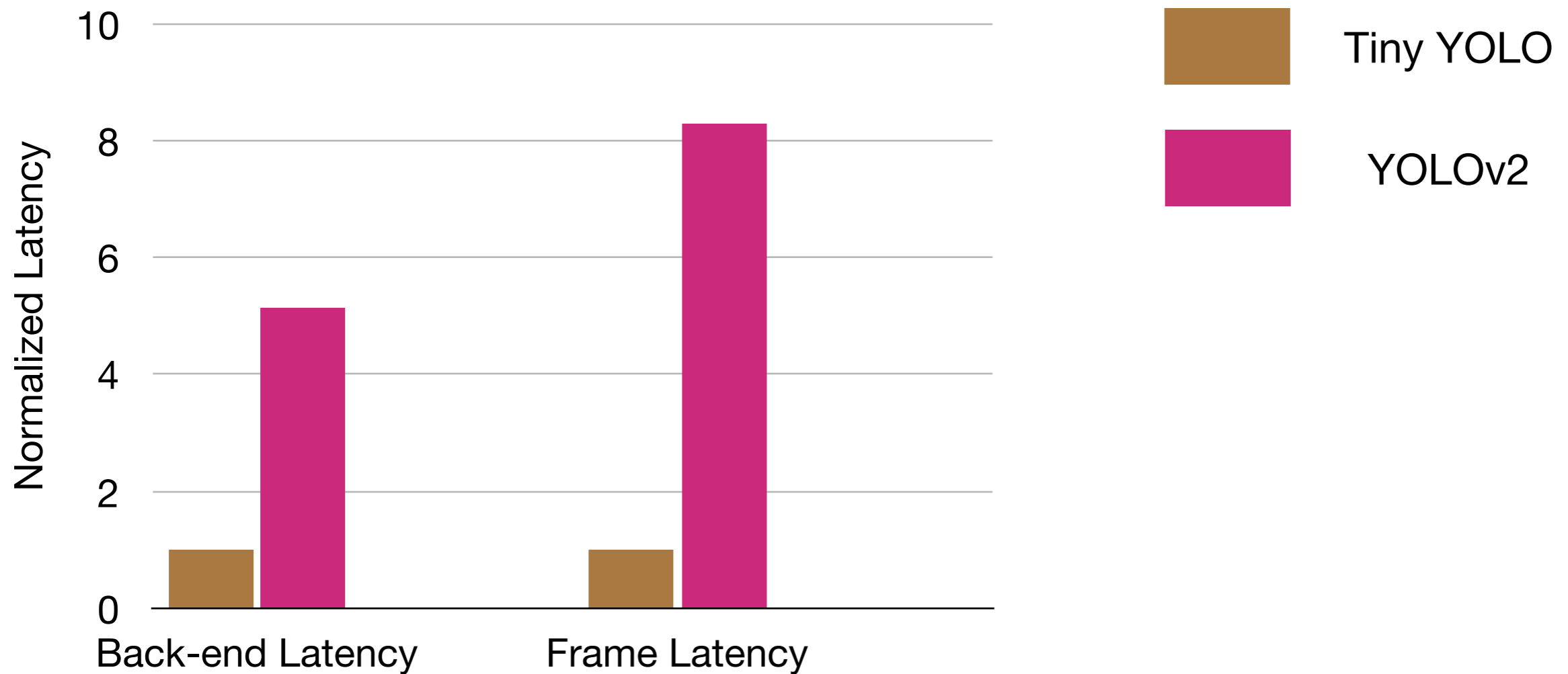
Comparison for 20 Frames



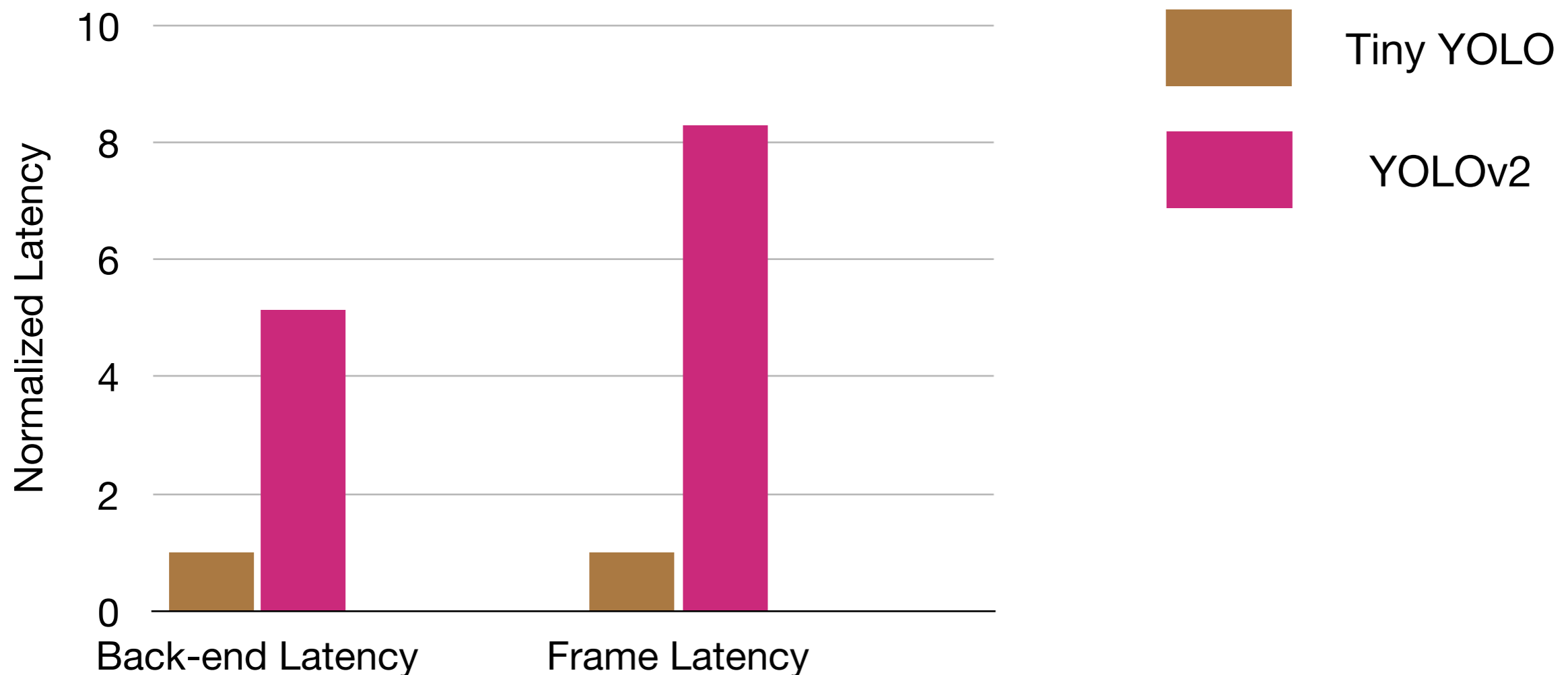
Comparison for 20 Frames



Comparison for 20 Frames



Comparison for 20 Frames



Conclusion 3: Frame latency is not proportional to back-end latency, even with the same front-end. We have to focus on end-to-end latency not just vision stage.