Task-Level Modeling
Case study: Continuous Vision Applications

Presenter: Yiming Gan
Application Domain

Continuous Vision
Bottlenecks

• Long Latency
  • 100 ms per frame

• 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

Light → Semantics Results
Continuous Vision Pipeline

Light → Sensing → Raw Pixels → Semantics Results
Continuous Vision Pipeline

Light → Sensing → Raw Pixels → Imaging → RGB Frames → Semantics Results
Continuous Vision Pipeline

Light

Sensing → Raw Pixels → Imaging → RGB Frames → Vision → Semantics Results
Mobile SoCs for Continuous Vision Pipeline

Sensing  Imaging  Vision
Mobile SoCs for Continuous Vision Pipeline

- Camera Sensor
- Sensor Interface
- Imaging
- Vision

MIPI
Mobile SoCs for Continuous Vision Pipeline

Diagram showing the components of a continuous vision pipeline:
- **Camera Sensor** connected to **Sensor Interface** via **MIPI** interconnect.
- **Sensor Interface** connected to **Imaging** and **Vision** stages.
- **SoC Interconnect** connects **Imaging** and **Vision** stages.
- **DRAM** stores raw pixels.
Mobile SoCs for Continuous Vision Pipeline

- Camera Sensor
- Sensor Interface
- Image Signal Processor
- Vision
- SoC Interconnect
- Raw Pixels
- DRAM
Mobile SoCs for Continuous Vision Pipeline

Diagram:
- Camera Sensor
  - MIPI
  - Sensor Interface
- Image Signal Processor
- Vision
- SoC Interconnect
- Raw Pixels
- RGB Frames
- DRAM
Mobile SoCs for Continuous Vision Pipeline
Mobile SoCs for Continuous Vision Pipeline

- Camera Sensor
- Sensor Interface
- Image Signal Processor
- DNN Accelerator

SoC Interconnect

- Raw Pixels
- RGB Frames
- Results

DRAM
Mobile SoCs for Continuous Vision Pipeline

- Camera Sensor
- Sensor Interface
- Image Signal Processor
- DNN Accelerator
- SoC Interconnect
- Raw Pixels
- RGB Frames
- Results

SoC Interconnect:

- MIPI
- DRAM
Latency in Continuous Vision Pipeline
Latency in Continuous Vision Pipeline
Latency in Continuous Vision Pipeline

Frame 1
- Sensing
- Imaging
- Vision

Frame 2
- Sensing
- Imaging
- Vision

Front-end Latency

Latency
Latency in Continuous Vision Pipeline

Frame 1
- Sensing
- Imaging
- Vision

Frame 2
- Sensing
- Imaging
- Vision

Latency

Front-end Latency

Back-end Latency
Latency in Continuous Vision Pipeline

Frame 1: Sensing | Imaging | Vision
Frame 2: Sensing | Imaging | Vision

Latency: Front-end Latency → Back-end Latency
Pending Time:
## Platform Architect Modeling

<table>
<thead>
<tr>
<th>Component</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor</td>
<td>AR 1335 Sensor, 30 FPS AR1335 Datasheet</td>
</tr>
<tr>
<td>ISP</td>
<td>Nvidia TX2 ISP, 30 FPS</td>
</tr>
<tr>
<td>DNN Accelerator</td>
<td>20x20 Systolic Array 1.5 MB SRAM, 500 MHz</td>
</tr>
<tr>
<td>Bus</td>
<td>AMBA4 AXI4, 32 GB/s</td>
</tr>
</tbody>
</table>
Platform Architect Modeling
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

<table>
<thead>
<tr>
<th>Task</th>
<th>Sensing</th>
<th>Imaging</th>
<th>Vision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency (ms)</td>
<td>34</td>
<td>34</td>
<td>102</td>
</tr>
</tbody>
</table>
Latency Results for YOLOv2
Latency Results for YOLOv2

Average Frame Latency

Front-end Latency

Frame Numbers
Latency Results for YOLOv2

Average Frame Latency

Front-end Latency
Back-end Latency

Frame Numbers

Latency Results for YOLOv2

Average Frame Latency

Front-end Latency
Back-end Latency

Frame Numbers
Latency Results for YOLOv2

Frame Numbers

Average Frame Latency

- Front-end Latency
- Back-end Latency
- Pending Time
Latency Results for YOLOv2

![Chart showing average frame latency with frame numbers from 5 to 20. The chart includes categories for front-end latency, back-end latency, and pending time.]
Latency Results for YOLOv2

![Bar chart showing average frame latency for YOLOv2 with categories: Front-end Latency, Back-end Latency, and Pending Time.](chart.png)
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

<table>
<thead>
<tr>
<th>Task</th>
<th>Sensing</th>
<th>Imaging</th>
<th>Vision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latency (ms)</td>
<td>34</td>
<td>34</td>
<td>19.6</td>
</tr>
</tbody>
</table>
Latency Results for TinyYOLO

![Bar chart showing average frame latency for TinyYOLO]

- **Front-end Latency**
- **Back-end Latency**
- **Pending Time**

Frame Numbers:
- 5
- 10
- 15
- 20
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

<table>
<thead>
<tr>
<th>Normalized Latency</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Back-end Latency**  
**Frame Latency**
Comparison for 20 Frames

Normalized Latency

Back-end Latency

Frame Latency

Tiny YOLO

YOLOv2
Comparison for 20 Frames

![Bar graph comparing back-end latency and frame latency for Tiny YOLO and YOLOv2. Tiny YOLO has lower latency for both back-end and frame latency.](image-url)
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.