Updating the Edge ML Development Process

Jim Steele
VP, Embedded Software
Samsara, Inc.

2023 embedded VISION SUMMIT

samsara

Samsara: Connect IoT Assets to Our Connected Operations Cloud

Video-Based Safety »
AI cameras, driver coaching, safety reports, in-cab alerts

Equipment Monitoring »
Location tracking, utilization, continuous diagnostics

Vehicle Telematics »
Real-time GPS, routing, fuel, maintenance, electrification

Site Visibility »
Remote visibility, proactive alerting, on-the-go access

Apps & Driver Workflows »
Messaging, dispatch, documents, ELD

APIs & Integrations »
Turnkey integrations, embedded telematics data

samsara

On-device ML: Safety Event Detection

Some Computer Vision Examples:

- **Forward Collision Warning**
- Lane Departure Warning
- Tailgating
- Outward Obstruction Detection
- and many more

Requires lots of data to train good Edge ML models (we process over 1 million videos a day)

Creating a Good Offline ML Model is just the beginning…

samsara

On-device ML: Scientist vs Coder

Traditional method of edge AI development

- ***Iterations slow and buggy***
- ***Runtime differences difficult to solve***

Machine learning scientist **designs**

Firmware engineer **develops** runtime algorithm through iteration with scientist

Firmware engineer **deploys** and monitors health throughout production

On-device ML: Solve with ML App Framework

<u>Preferred</u> method of edge AI development

- ***Quick iterations with realistic results***
- ***Aligned ownership through production***

Firmware engineer **develops** “general” ML application framework

Machine learning scientist **designs**, **develops**, and **deploys** runtime algorithm

Both monitor health throughout production

Edge CV: Product Lifecycle Considerations

Design

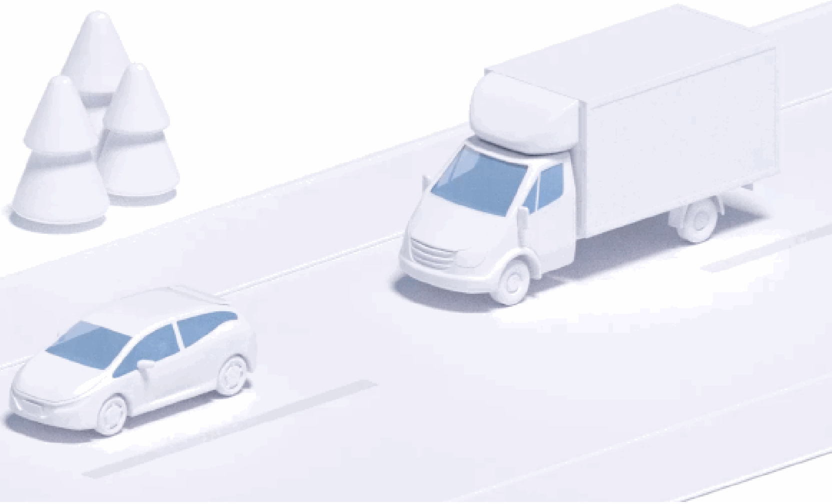
- Cloud-device partitioning (precision vs bandwidth)
- Low-power/high-performance ML accelerators

Development

- Synchronization across various input sensors
- On-device ML application framework (concurrency)

Deployment

- Quick iteration cycles on-device
- Metrics to understand in-field performance



samsara

Edge CV Consideration: Design

ML application framework

- Low-overhead, memory optimized (i.e., not Android)
- Easy reuse across hardware platforms (i.e., not vendor provided)

Utilize available on-device hardware accelerators

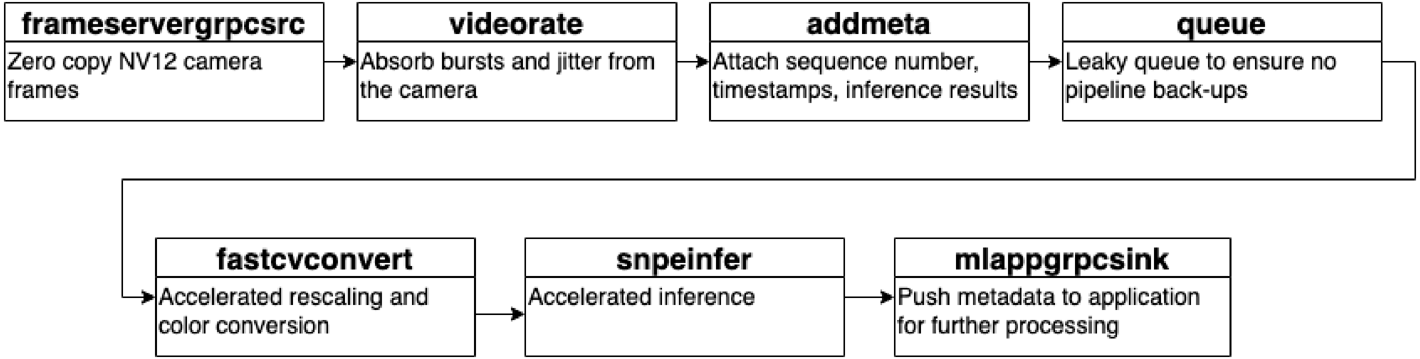
- NPU/GPU/CPU and ML hardware accelerators
- Expansion to other sensor inputs with universal timestamp

Smart use of device resources

- Efficient camera stream handling (e.g., no memcopy)
- Compute contention resolver (multiple ML apps at once)

→ Built framework using GStreamer elements and abstracted compute interfaces

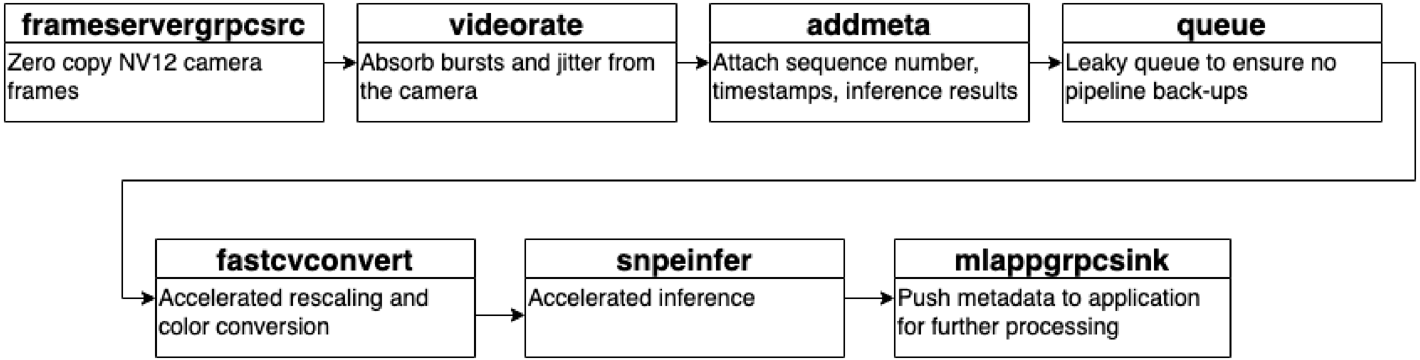
Design Detail: GStreamer Pros and Cons



Pros:

- Leverage open-source community
- Vendor agnostic—allows consistent development experience across devices
- Easily extensible functionality with in-house custom “elements”
- Element abstraction provides clean separation between firmware and ML engineers

Design Detail: GStreamer Pros and Cons

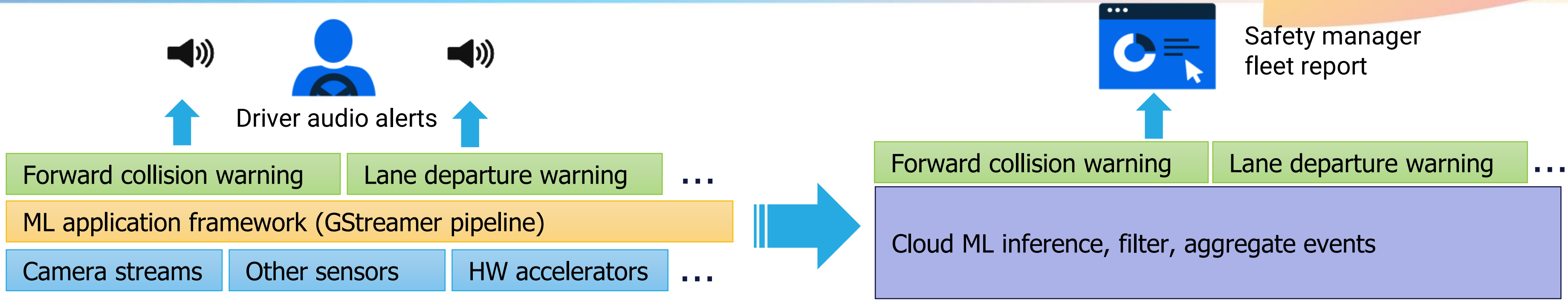


Cons:

- Steep learning curve for firmware developers
- Originally multi-media focused, lacked easy and efficient way to adopt neural network models and pipelines and support for latest ML accelerators
- NNstreamer: one example to alleviate the above — was difficult to customize to our use-cases, and difficult to debug. Ended up writing our own elements

samsara

Edge CV Consideration: Development



Framework written in low-level C/C++

- Optimized for hardware peculiarities
- Synchronization across various input sensors

```
snpe_calculate_shape(input_tensor.shape, input_tensor.encoding->getElementSize(), input_strides,
                     input_data_size);
_in_buf = zdl::SNPE::SNPEFactory::getUserBufferFactory().createUserBuffer(
    input_tensor.data, input_data_size, input_strides, input_tensor.encoding);
_input_buffer_map.add(input_tensor.name.c_str(), _in_buf.get());
_input_shape_map.add(input_tensor.name.c_str(), zdl::DlSystem::TensorShape(input_tensor.shape));

for (TensorInfo output_tensor : output_tensors) {
  std::vector<size_t> output_strides;
  size_t output_data_size = 0;
  snpe_calculate_shape(output_tensor.shape, output_tensor.encoding->getElementSize(),
                       output_strides, output_data_size);
  std::unique_ptr<zdl::DlSystem::IUserBuffer> out_buf =
      zdl::SNPE::SNPEFactory::getUserBufferFactory().createUserBuffer(
          output_tensor.data, output_data_size, output_strides, output_tensor.encoding);
  _output_buffer_map.add(output_tensor.name.c_str(), out_buf.get());
```

ML app written as GStreamer elements

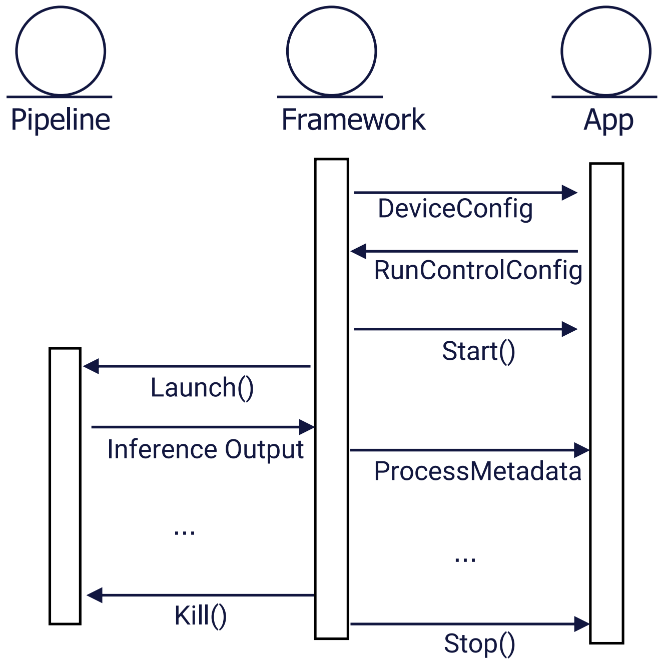
- Easy to implement and debug
- Clear way to compare and tune vs cloud

```
gst-launch-1.0 frameservergrpcsrc ! fcvconvert ! snpeinfer ! grpcsink
```

samsara

Development Detail: Application Framework

- Eventually have multiple concurrent apps contending for same inference hardware
- Simple framework to manage MLApp lifecycle and schedule GStreamer pipeline
- Handles configuration changes such as thermal throttling
- Handles MLApp and model updates



Edge CV Consideration: Deployment

Debugging lessons learned for quick ML developer iteration cycles w/o knowing firmware

- Always maintain struct with auxiliary details (bounding boxes, confidence levels, configs)
- Build local livestream with model detections for ML developer to see real-time
- Video replay on actual device for regressions and improvements

Deployment tracking

- Versioning: not only firmware version, but now app, model, configs
- Separate cohort model updates for A/B testing
- Metrics to understand device performance in the field
    - System level: fps, CPU, memory, inference latency, frame drops
    - ML performance: safety event review, cloud inference correlation, random field sampling

Conclusions: Samsara’s ML App Framework

Design

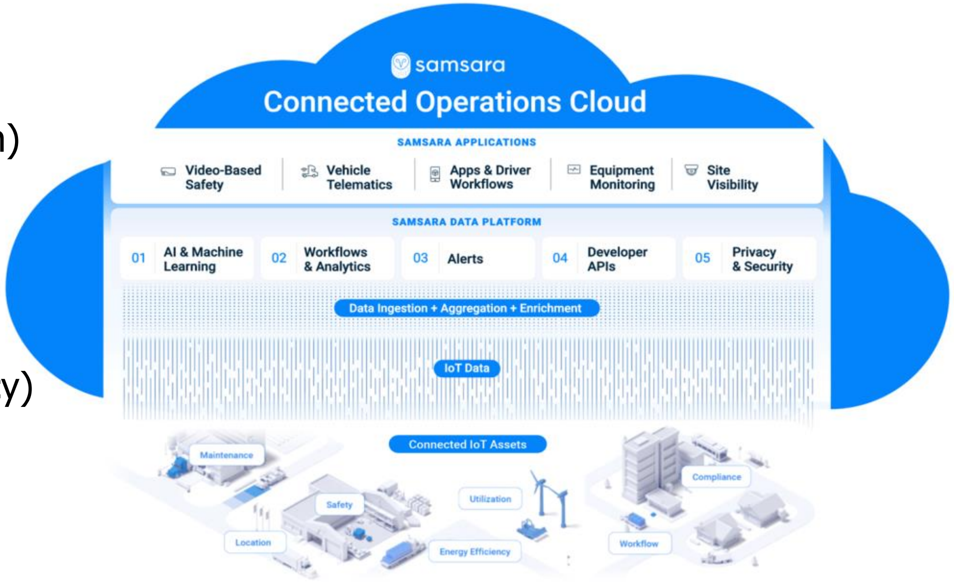
- Cloud-device partitioning (precision vs bandwidth)
- Low-power/high-performance ML accelerators

Development

- Synchronization across various input sensors
- On-device ML application framework (concurrency)

Deployment

- Quick iteration cycles on-device
- Metrics to understand in-field performance



samsara

samsara

Samsara Links

Samsara

We are hiring!
https://www.samsara.com/company/careers/roles/

Our blog
https://www.samsara.com/blog/

GStreamer open-source multimedia framework
https://gstreamer.freedesktop.org/