Hardware acceleration for Machine Learning and Computer Vision through Khronos open standard APIs
# Workshop agenda

<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:05 AM</td>
<td>Khronos/OpenCL/SPIR-V (Neil Trevett, NVIDIA) – 35 minutes</td>
</tr>
<tr>
<td>9:40 AM</td>
<td>OpenCL/SYCL (Andrew Richards, Codeplay) – 20 minutes</td>
</tr>
<tr>
<td>10:00 AM</td>
<td>Break – 15 minutes</td>
</tr>
<tr>
<td>10:15 AM</td>
<td>SYCL presentation (Konstantin S. Bobrovsky, Intel) – 45 minutes</td>
</tr>
<tr>
<td>11:00 AM</td>
<td>OpenVX presentations (Frank Brill, Cadence, Niclas Danielsson and Mikael Pendse, Axis – 1 hour)</td>
</tr>
<tr>
<td>12:00 PM</td>
<td>Lunch – 1 hour</td>
</tr>
<tr>
<td>1:00 PM</td>
<td>OpenVX presentation (Mike Schmit, AMD) – 30 minutes</td>
</tr>
<tr>
<td>1:30 PM</td>
<td>NNEF presentation (Gergely Debreczeni, AImotive) – 30 min</td>
</tr>
<tr>
<td>2:00 PM</td>
<td>NNEF hands-on (Gergely Debreczeni, AImotive) – 30 min</td>
</tr>
<tr>
<td>2:30 PM</td>
<td>OpenVX hands-on part 1 (Rajy Rawther and Kiriti Nagesh Gowda, AMD) – 30 minutes</td>
</tr>
<tr>
<td>3:00 PM</td>
<td>Break – 15 minutes</td>
</tr>
<tr>
<td>3:15 PM</td>
<td>OpenVX hands-on part 2 (Rajy Rawther and Kiriti Nagesh Gowda, AMD) – 2 hours</td>
</tr>
<tr>
<td>5:00 PM</td>
<td>Finish</td>
</tr>
</tbody>
</table>
Portable performance via the OpenVX™ computer vision library

Frank Brill
Cadence Design Systems
May 2019
OpenVX Extensions

• **Neural Network**: run inference as part of a graph
  • Layers are represented as OpenVX nodes

• **Classification**: detect and recognize objects in an image based on a set of features
  • Import a cascade detector/classifier model trained offline
  • Classify objects based on a set of input features

• **Pipelining**: increase hardware utilization and throughput
  • Provide a way of pipelining, streaming, and batch processing
  • Multiple initiations of a graph with different inputs and outputs

• **OpenCL Interop**: interop between OpenVX and OpenCL application & user-kernels

• **Import/Export**: provide a way of exporting and importing pre-verified graphs & objects

• **Import Kernel**: import pre-compiled vendor binary (e.g., pre-compiled NN as a kernel)
- Wide range of vision hardware architectures
- OpenVX provides a high-level Graph-based abstraction
  - Enables Graph-level optimizations!
  - Can be implemented on almost any hardware or processor!
- Portable, Efficient Vision Processing!
OpenVX Efficiency through Graphs

**Graph Scheduling**
- Split the graph execution across the whole system: CPU / GPU / dedicated HW
- Faster execution or lower power consumption

**Memory Management**
- Reuse pre-allocated memory for multiple intermediate data
- Less allocation overhead, more memory for other applications

**Kernel Fusion**
- Replace a sub-graph with a single faster node
- Better memory locality, less kernel launch overhead

**Data Tiling**
- Execute a sub-graph at tile granularity instead of image granularity
- Better use of data cache and local memory
OpenVX Extensions

- **Neural Network**: run inference as part of a graph
  - Layers are represented as OpenVX nodes

- **Classification**: detect and recognize objects in an image based on a set of features
  - Import a cascade detector/classifier model trained offline
  - Classify objects based on a set of input features

- **Pipelining**: increase hardware utilization and throughput
  - Provide a way of pipelining, streaming, and batch processing
  - Multiple initiations of a graph with different inputs and outputs

- **OpenCL Interop**: interop between OpenVX and OpenCL application & user-kernels

- **Import/Export**: provide a way of exporting and importing pre-verified graphs & objects

- **Import Kernel**: import pre-compiled vendor binary (e.g., pre-compiled NN as a kernel)
OpenVX delivers portable performance

- Application code **portable** across a broad range of hardware platforms
- **Performance** comparable to hand-optimized, non-portable code
  - Real, complex applications on real, complex hardware
  - Much lower development effort than hand-optimized
- **Integrate** neural-network and pre/post processing to **optimize** globally
OpenVX Roadmap and Resources
OpenVX Roadmap

• OpenVX 1.3 expected to be approved for release in July
  • Enhanced neural-network support
    • NNEF import with conformance tests
  • Feature sets to enable compliance for diverse application spaces
    • Classical computer vision / image processing
    • Neural networks via OpenVX extension nodes or NNEF import
    • Binary (one bit) image processing
  • Merge safety-critical features into single main specification

• Open-source implementation on Raspberry Pi in development
  • MulticoreWare, mostly ARM NEON via ARM Compute Library (ACL)
  • OpenCL / VC4CL: https://github.com/doe300/VC4CL
  • Target September
OpenVX and NNEF resources

- OpenVX Overview: [https://www.khronos.org/openvx](https://www.khronos.org/openvx)

- OpenVX Specifications: current, previous, and extensions
  - [https://www.khronos.org/registry/OpenVX](https://www.khronos.org/registry/OpenVX)

- OpenVX implementations, tutorials, reference guides, etc.
  - [https://www.khronos.org/openvx/resources](https://www.khronos.org/openvx/resources)

- NNEF Specification: [https://www.khronos.org/registry/NNEF](https://www.khronos.org/registry/NNEF)

- Embedded Vision Summit Workshop
  “Hardware acceleration for Machine Learning and Computer Vision through Khronos open standard APIs”
  Thursday, May 23, 2019 from 9:00am-5:00pm