Khronos Standard APIs for Accelerating Vision and Inferencing

Neil Trevett
Khronos President
NVIDIA VP Developer Ecosystems
22nd September 2020
Khronos Connects Software to Silicon

Open interoperability standards to enable software to effectively harness the power of 3D and multiprocessor acceleration

3D graphics, XR, parallel programming, vision acceleration and machine learning

Non-profit, member-driven standards-defining industry consortium

Open to any interested company

All Khronos standards are royalty-free

Well-defined IP Framework protects participant’s intellectual property

Founded in 2000
>150 Members ~ 40% US, 30% Europe, 30% Asia
Khronos Active Initiatives

3D Graphics
- Desktop, Mobile, Web
- Embedded and Safety Critical

3D Assets
- Authoring and Delivery

Portable XR
- Augmented and Virtual Reality

Parallel Computation
- Vision, Inferencing, Machine Learning

Logos:
- Vulkan
- ANARI
- OpenGL
- WebGL
- EGL
- OpenCL
- SYCL
- OpenVX
- NNEF
- SPIR
Increasing industry interest in parallel compute acceleration to combat the ‘End of Moore’s Law’
Embedded Vision and Inferencing Acceleration

- Networks trained on high-end desktop and cloud systems
- Applications link to compiled inferencing code or call vision/inferencing API
- Diverse Embedded Hardware (GPUs, DSPs, FPGAs)

- Neural Network Training
  - Training Data
  - Trained Networks
  - Ingestion
  - Compilation
  - Compiled Code
  - Vision / Inferencing Engine
  - C++ Application Code

- Hardware Acceleration APIs
  - Sensor Data
  - OpenCL
  - OpenVX
  - SYCL
NNEF Neural Network Exchange Format

Before - Training and Inferencing Fragmentation

Every Inferencing Engine needs a custom importer from every Framework

After - NN Training and Inferencing Interoperability

Common optimization and processing tools
## NNEF and ONNX

<table>
<thead>
<tr>
<th>NNEF</th>
<th>ONNX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded Inferencing Import</td>
<td>Training Interchange</td>
</tr>
<tr>
<td>Defined Specification</td>
<td>Open Source Project</td>
</tr>
<tr>
<td>Multi-company Governance at Khronos</td>
<td>Initiated by Facebook &amp; Microsoft</td>
</tr>
<tr>
<td>Stability for hardware deployment</td>
<td>Software stack flexibility</td>
</tr>
</tbody>
</table>

### NNEF V1.0 released in August 2018
After positive industry feedback on Provisional Specification.
Maintenance update issued in September 2019
Extensions to V1.0 released for expanded functionality

### ONNX 1.6 Released in September 2019
Introduced support for Quantization
ONNX Runtime being integrated with GPU inferencing engines such as NVIDIA TensorRT

---

**ONNX and NNEF are Complementary**
ONNX moves quickly to track authoring framework updates
NNEF provides a stable bridge from training into edge inferencing engines

---

**NNEF Working Group Participants**

- AIMOTIVE
- AMD
- arm
- Air Zone
- Cadence
- CEVA
- ETRI
- Huawei
- Intel
- Imagination
- MediaTek
- Member
- Graphics
- NXP
- Qualcomm
- Peakhills
- Group
- Samsung
- Sionext
- Synopsys
- Texas
- Instruments
- ThinkSilicon
- VonSilicon
- Xilinx

---

**ONNX Supporters**

- Alibaba Group
- AMD
- arm
- AWS
- Baidu
- BITMAIN
- CEVA
- Facebook Open Source
- Graphcore
- Habana
- HUAWEI
- IBM
- Idein Inc.
- Intel
- MathWorks
- Microsoft
- Neural Network Libraries
- NVIDIA
- NXP
- Oath
- Preferred Networks
- Qualcomm
- SAS
- Skymizer
- Synopsys
- Tencent
- Unity
NNEF Open Source Tools Ecosystem

NNEF Model Zoo
Now available on GitHub. Useful for checking that ingested NNEF produces acceptable results on target system

NNEF adopts a rigorous approach to design lifecycle
Especially important for safety-critical or mission-critical applications in automotive, industrial and infrastructure markets

NNEF open source projects hosted on Khronos NNEF GitHub repository under Apache 2.0
https://github.com/KhronosGroup/NNEF-Tools

ONNX Import/Export
Syntax Parser and Validator
OpenVX Ingestion and Execution
TensorFlow and TensorFlow Lite Import/Export
Caffe and Caffe2 Import/Export

Compound operations captured by exporting graph Python script
SYCL Single Source C++ Parallel Programming

- SYCL-BLAS, SYCL-DNN, SYCL-Eigen, SYCL Parallel STL
- C++ Libraries
- Standard C++ Application Code
- ML Frameworks
- C++ Template Libraries
- C++ Template Libraries
- C++ Template Libraries

SYCL Compiler for OpenCL

- CPU Compiler
- CPU

- SYCL Compiler for OpenCL
- GPU
- FPGA
- DSP
- AI/Tensor HW
- Custom Hardware

SYCL is ideal for accelerating larger C++-based engines and applications with performance portability

- Complex ML frameworks can be directly compiled and accelerated

- C++ templates and lambda functions separate host & accelerated device code

- C++ Kernel Fusion can give better performance on complex apps and libs than hand-coding

- Accelerated code passed into device OpenCL compilers

- OpenCL

- TensorFlow
SYCL Implementations

SYCL, OpenCL and SPIR-V, as open industry standards, enable flexible integration and deployment of multiple acceleration technologies.

SYCL Source Code

SYCL enables Khronos to influence ISO C++ to (eventually) support heterogeneous compute.

Multiple Backends in Development
SYCL beginning to be supported on multiple low-level APIs in addition to OpenCL e.g. ROCm and CUDA
For more information: http://sycl.tech
OpenVX Cross-Vendor Vision and Inferencing

OpenVX
High-level graph-based abstraction for portable, efficient vision processing
Graph can contain vision processing and NN nodes - enables global optimizations
Optimized OpenVX drivers created, optimized and shipped by processor vendors
Implementable on almost any hardware or processor with performance portability
Run-time graph execution need very little host CPU interaction

Performance comparable to hand-optimized, non-portable code
Real, complex applications on real, complex hardware
Much lower development effort than hand-optimized code

NNEF Translator converts NNEF representation into OpenVX Node Graphs

OpenVX Graph

Native Camera Control → Vision Node → CNN Nodes → Vision Node → Downstream Application Processing

Hardware Implementations
OpenVX 1.3 Released October 2019

Functionality Consolidation into Core
- Neural Net Extension, NNEF Kernel Import, Safety Critical etc.

Open Source Conformance Test Suite
https://github.com/KhronosGroup/OpenVX-cts/tree/openvx_1.3

OpenCL Interop
- Custom accelerated Nodes

Deployment Flexibility through Feature Sets
- Conformant Implementations ship one or more complete feature sets
  - Enables market-focused Implementations
    - Baseline Graph Infrastructure (enables other Feature Sets)
    - Default Vision Functions
    - Enhanced Vision Functions (introduced in OpenVX 1.2)
    - Neural Network Inferencing (including tensor objects)
    - NNEF Kernel import (including tensor objects)
    - Binary Images
    - Safety Critical (reduced features for easier safety certification)


OpenVX user-kernels can access command queue and cl_mem objects to asynchronously schedule OpenCL kernel execution

OpenVX data objects
OpenCL Command Queue
OpenVX/OpenCL Interop

Application

Fully asynchronous host-device operations during data exchange

Copy or export cl_mem buffers into OpenVX data objects
Map or copy OpenVX data objects into cl_mem buffers

cl_mem buffers
Open Source OpenVX & Samples

Fully Conformant Open Source OpenVX 1.3 for Raspberry Pi
https://github.com/KhronosGroup/OpenVX-sample-impl/tree/openvx_1.3
Raspberry Pi 3 and 4 Model B with Raspbian OS
Memory access optimization via tiling/chaining
Highly optimized kernels on multimedia instruction set
Automatic parallelization for multicore CPUs and GPUs
Automatic merging of common kernel sequences

“Raspberry Pi is excited to bring the Khronos OpenVX 1.3 API to our line of single-board computers. Many of the most exciting commercial and hobbyist applications of our products involve computer vision, and we hope that the availability of OpenVX will help lower barriers to entry for newcomers to the field.”

Eben Upton
Chief Executive Raspberry Pi Trading

Open Source OpenVX Tutorial and Code Samples
https://github.com/rgiduthuri/openvx_tutorial
https://github.com/KhronosGroup/openvx-samples
OpenCL is Widely Deployed and Used

Accelerated Implementations

OpenCL - Low-level Parallel Programming

Programming and Runtime Framework for Application Acceleration
Offload compute-intensive kernels onto parallel heterogeneous processors
 CPUs, GPUs, DSPs, FPGAs, Tensor Processors
 OpenCL C or C++ kernel languages

Platform Layer API
Query, select and initialize compute devices

Runtime API
Build and execute kernels programs on multiple devices

Explicit Application Control
 Which programs execute on what device
 Where data is stored in memories in the system
 When programs are run, and what operations are dependent on earlier operations

Complements GPU-only APIs
Simpler programming model
Relatively lightweight run-time
More language flexibility, e.g. pointers
Rigorously defined numeric precision
OpenCL 3.0

Increased Ecosystem Flexibility
All functionality beyond OpenCL 1.2 queryable plus macros for optional OpenCL C language features
New extensions that become widely adopted will be integrated into new OpenCL core specifications

OpenCL C++ for OpenCL
Open source C++ for OpenCL front end compiler combines OpenCL C and C++17 replacing OpenCL C++ language specification

Unified Specification
All versions of OpenCL in one specification for easier maintenance, evolution and accessibility
Source on Khronos GitHub for community feedback, functionality requests and bug fixes

Moving Applications to OpenCL 3.0
OpenCL 1.2 applications - no change
OpenCL 2.X applications - no code changes if all used functionality is present
Queries recommended for future portability

C++ for OpenCL
Supported by Clang and uses the LLVM compiler infrastructure
OpenCL C code is valid and fully compatible
Supports most C++17 features
Generates SPIR-V kernels
Google Ports TensorFlow Lite to OpenCL

Even Faster Mobile GPU Inference with OpenCL

August 17, 2020

Even Faster Mobile GPU Inference with OpenCL

OpenCL providing ~2x inferencing speedup over OpenGL ES acceleration

TensorFlow Lite uses OpenGL ES as a backup if OpenCL not available...

...but most mobile GPU vendors provide an OpenCL drivers - even if not exposed directly to Android developers

OpenCL is increasingly used as acceleration target for higher-level framework and compilers

Improvements over the OpenGL Backend

OpenCL is an API designed for rendering vector graphics. Compute shaders were added with OpenGL ES 3.1, but its backward-compatible API design decisions were limiting us...

Figure 1: Buck of effects are powered by our opencl backend.

Figure 2: Inference latency of MobiNet 1.3 on select Android devices with OpenCL.

Figure 3: Inference latency of SSD MobileNet v2 (large) on select Android devices with OpenCL.
# Primary Machine Learning Compilers

<table>
<thead>
<tr>
<th>Import Formats</th>
<th>Caffe, Keras, MXNet, ONNX</th>
<th>TensorFlow Graph, MXNet, PaddlePaddle, Keras, ONNX</th>
<th>PyTorch, ONNX</th>
<th>TensorFlow Graph, PyTorch, ONNX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Front-end / IR</td>
<td>NNVM / Relay IR</td>
<td>nGraph / Stripe IR</td>
<td>Glow Core / Glow IR</td>
<td>XLA HLO</td>
</tr>
<tr>
<td>Output</td>
<td>OpenCL, LLVM, CUDA, Metal</td>
<td>OpenCL, LLVM, CUDA</td>
<td>OpenCL LLVM</td>
<td>LLVM, TPU IR, XLA IR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tensorflow Lite / NNAPI</td>
<td>(inc. HW accel)</td>
</tr>
</tbody>
</table>

Diagram: [Diagram of machine learning compilers and formats]
### ML Compiler Steps

<table>
<thead>
<tr>
<th>Import Formats</th>
<th>Front-end / IR</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caffe, Keras, MXNet, ONNX</td>
<td>NNVM / Relay IR</td>
<td>OpenCL, LLVM, CUDA, Metal</td>
</tr>
<tr>
<td>TensorFlow Graph, MXNet, PaddlePaddle, Keras, ONNX</td>
<td>nGraph / Stripe IR</td>
<td>OpenCL, LLVM, CUDA</td>
</tr>
<tr>
<td>PyTorch, ONNX</td>
<td>Glow Core / Glow IR</td>
<td>OpenCL LLVM</td>
</tr>
<tr>
<td>TensorFlow Graph, PyTorch, ONNX</td>
<td>XLA HLO</td>
<td>LLVM, TPU IR, XLA IR, TensorFlow Lite / NNAPI (inc. HW accel)</td>
</tr>
</tbody>
</table>

#### Consistent Steps

1. Import Trained Network Description
2. Apply graph-level optimizations e.g. node fusion, node lowering and memory tiling
3. Decompose to primitive instructions and emit programs for accelerated run-times

**Fast progress but still area of intense research**

If compiler optimizations are effective - hardware accelerator APIs can stay ‘simple’ and won’t need complex metacommands (e.g. combined primitive commands like DirectML)
Google MLIR and IREE Compilers

MLIR
Multi-level Intermediate Representation
Format and library of compiler utilities that sits between the trained model representation and low-level compilers/executors that generate hardware-specific code

IREE
Intermediate Representation
Execution Environment
Lowers and optimizes ML models for real-time accelerated inferencing on mobile/edge heterogeneous hardware
Contains scheduling logic to communicate data dependencies to low-level parallel pipelined hardware/APIs like Vulkan, and execution logic to encode dense computation in the form of hardware/API-specific binaries like SPIR-V

IREE is a research project today. Google is working with Khronos working groups to explore how SPIR-V code can provide effective inferencing acceleration on APIs such as Vulkan through SPIR-V
SPIR-V enables a rich ecosystem of languages and compilers to target low-level APIs such as Vulkan and OpenCL, including deployment flexibility: e.g. running OpenCL C kernels on Vulkan.
Khronos membership is open to any company
Influence the design and direction of key open standards that will drive your business
Accelerate time-to-market with early access to specification drafts
Provide industry thought leadership and gain insights into industry trends and directions
Benefit from Adopter discounts

www.khronos.org/members/
ntrevett@nvidia.com  | @neilt3d
Resources

- Khronos Website and home page for all Khronos Standards
  - https://www.khronos.org/

- OpenCL Resources and C++ for OpenCL documentation
  - https://www.khronos.org/opencl/resources
  - https://github.com/KhronosGroup/Khronosdotorg/blob/master/api/opencl/assets/CXX_for_OpenCL.pdf

- OpenVX Tutorial, Samples and Sample Implementation
  - https://github.com/rgiduthuri/openvx_tutorial
  - https://github.com/KhronosGroup/openvx-samples
  - https://github.com/KhronosGroup/OpenVX-sample-impl/tree/openvx_1.3

- NNEF Tools
  - https://github.com/KhronosGroup/NNEF-Tools

- SYCL Resources
  - http://sycl.tech

- SPIR-V User Guide
  - https://github.com/KhronosGroup/SPIRV-Guide

- MLIR Blog

- IREE GitHub Repository
  - https://google.github.io/iree/