Current and Planned Standards for Computer Vision and Machine Learning

Neil Trevett
Khronos President
NVIDIA VP Developer Ecosystems
ntrevett@nvidia.com  |  @neilt3d
Khronos is an open, non-profit, member-driven industry consortium developing royalty-free standards to harness the power of silicon acceleration for demanding graphics rendering and computationally intensive applications such as 3D Graphics, Virtual Reality, Augmented Reality, Vision Processing and Machine Learning.
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

Guidelines for creating APIs to streamline system safety certification

Exploratory Groups
- Heterogeneous Communications between offload compute devices
- Making High-Level Languages more effective at acceleration offload
- Rendering for scientific visualization and data analytics

This work is licensed under a Creative Commons Attribution 4.0 International License
Khronos Compute Acceleration Standards

Increasing industry interest in parallel heterogeneous compute acceleration to combat the ‘End of Moore’s Law’

**Higher-level APIs**
Streamlined programming and performance portability

**Lower-level APIs**
Direct Hardware Control

- **Vulkan:** GPU Rendering + Compute Acceleration
- **SPIR:** Intermediate Representation (IR) supporting Parallel Execution and Graphics
- **SYCL:** Single source C++ Programming with Compute Acceleration
- **OpenVX:** Graph-based Vision and Inferencing Acceleration
- **NNEF:** Trained Neural Network Exchange Format

**Intermediate Representation (IR) supporting Parallel Execution and Graphics**

**CPU**

**GPU**

**FPGA**

**DSP**

**Custom Hardware**

This work is licensed under a Creative Commons Attribution 4.0 International License
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

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

Accelerated code passed into device OpenCL compilers

Complex ML frameworks can be directly compiled and accelerated

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

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

ML Frameworks

CPU

OpenCL

CPU

GPU

FPGA

DSP

Custom Hardware

TensorFlow
SYCL Implementations

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

Multiple Backend Support in Development
SYCL beginning to be supported on multiple low-level APIs in addition to OpenCL e.g. Vulkan, ROCm and CUDA
http://sycl.tech

Intel’s ‘One API’
Based on SYCL using LLVM/clang

ComputeCpp
Codeplay’s SYCL 1.2.1 on multiple hardware

triSYCL
Open-source test-bed

HipSYCL
SYCL 1.2.1 implementation over NVIDIA CUDA and AMD HIP/ROCm

Intel clang (Intel SYCL)
Codeplay’s ComputeCpp
triSYCL
HipSYCL
sycl-gtx
OpenCL 1.2

Does not support non-standard macros

Any CPU
Any CPU
Any CPU
Any CPU

OpenCL + SPIR-V
OpenCL + SPIR-V
OpenCL + SPIR-V

Intel CPUs
Intel GPUs
AMD GPUs (depending on driver stack)
ARM Mali
Renesas R-Car

NVIDIA GPUs
PTX devices

(with OpenMP)

ROCm
CUDA
**SPIR-V Ecosystem**

- **MSL**
- **HLSL**
- **GLSL**
- **SPIRV-Cross**
- **SPIR-V (Dis)Assembler**
- **SPIRV Validator**
- **SPIRV-opt | SPIRV-remap**

**Optimization Tools**

- **GLSL**
- **HLSL**
- **glslang**
- **DXC**

- **OpenCL C Front-end**
- **OpenCL C++ Front-end**
- **SYCL for ISO C++ Front-end**

**SPIR-V**
- Khronos-defined cross-API IR
- Native graphics and parallel compute support
- Easily parsed/extended 32-bit stream
- Data object/control flow retained for effective code generation/translation

**Environment spec for each target API used to drive compilation**

**3rd Party-hosted Open Source Projects**

- **Khronos-hosted Open Source Projects**

**C++ for OpenCL in clang Front-end**

**SYCL**
- Khronos cooperating closely with clang/LLVM Community

**https://github.com/KhronosGroup/SPIRV-Tools**
Neural Network Exchange Formats

Before - Training and Inferencing Fragmentation

Training Framework 1
Training Framework 2
Training Framework 3

Inference Engine 1
Inference Engine 2
Inference Engine 3

Every Inferencing Engine needs a custom importer from every Framework

After - NN Training and Inferencing Interoperability

Training Framework 1
Training Framework 2
Training Framework 3

Inference Engine 1
Inference Engine 2
Inference Engine 3

Neural Network Exchange Format

Common Optimization and processing tools

Two Neural Network Exchange Format Initiatives

<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>

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 and ONNX Industry Support

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

NNEF Working Group Participants

ONNX Supporters
NNEF 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

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

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

Compound operations captured by exporting graph Python script
OpenVX Cross-Vendor 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 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
OpenVX Accelerated Custom Nodes

**OpenVX/OpenCL Interop**
- Provisional Extension
- Enables custom OpenCL acceleration to be invoked from OpenVX User Kernels
- Memory objects can be mapped or copied

**Kernel/Graph Import**
- Provisional Extension
- Defines container for executable or IR code
- Enables arbitrary code to be inserted as an OpenVX Node in a graph

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

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

Fully asynchronous host-device operations during data exchange.
OpenVX 1.3 Released October 2019

Functionality Consolidation into Core 1.3
Neural Net Extension, NNEF Kernel Import, Safety Critical etc.

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)


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

Open Source OpenVX Tutorial and Code Samples
https://github.com/rgiduthuri/openvx_tutorial

Open Source OpenVX 1.3 for Raspberry Pi
Raspberry Pi 3 Model B with Raspbian OS
- Automatic optimization of memory access patterns via tiling and chaining
- Highly optimized kernels leveraging multimedia instruction set
- Automatic parallelization for multicore CPUs and GPUs
- Automatic merging of common kernel sequences

https://github.com/KhronosGroup/OpenVX-sample-impl/tree/openvx_1.3
## OpenVX and OpenCV are Complementary

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Community driven open source library</th>
<th>Callable API implemented, optimized and shipped by hardware vendors</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scope</strong></td>
<td>100s of imaging and vision functions Multiple camera APIs/interfaces</td>
<td>Tight focus on dozens of core hardware accelerated functions plus extensions and accelerated custom nodes. Uses external camera drivers</td>
</tr>
<tr>
<td><strong>Conformance</strong></td>
<td>Extensive OpenCV Test Suite but no formal Adopters program</td>
<td>Implementations must pass Khronos Conformance Test Suite to use trademark</td>
</tr>
<tr>
<td><strong>IP Protection</strong></td>
<td>None. Source code licensed under BSD. Some modules require royalties/licensing</td>
<td>Protected under Khronos IP Framework - Khronos members agree not to assert patents against API when used in Conformant implementations</td>
</tr>
<tr>
<td><strong>Acceleration</strong></td>
<td>OpenCV 3.0 Transparent API (or T-API) enables function offload to OpenCL devices</td>
<td>Implementation free to use any underlying API such as OpenCL. Can use OpenCL for Custom Nodes</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td>OpenCV 4.0 G-API graph model for some filters, arithmetic/binary operations, and well-defined geometrical transformations</td>
<td>Graph-based execution of all Nodes. Optimizable computation and data transfer</td>
</tr>
<tr>
<td><strong>Inferencing</strong></td>
<td>Deep Neural Network module to construct networks from layers for forward pass computations only. Import from ONNX, TensorFlow, Torch, Caffe</td>
<td>Neural Network layers and operations represented directly in the OpenVX Graph. NNEF direct import, ONNX through NNEF convertor</td>
</tr>
</tbody>
</table>
OpenVX Adoption for Inferencing

Rise in use of OpenVX for Inferencing

- Availability of NNEF Import and NN Extension
- Opportunity for silicon vendors to optimize inferencing solutions under a standard API
- Increasing deployment on embedded systems (in addition to HPC, Cloud, PCs, Mobile)

Trend will be accelerated by OpenVX 1.3’s deployment flexibility
# Primary Machine Learning Compilers

<table>
<thead>
<tr>
<th>Import Formats</th>
<th>Caffe, Keras, MXNet, ONNX</th>
<th>TensorFlow Graph, MXNet, Paddle, Keras, ONNX</th>
<th>PyTorch, ONNX</th>
<th>TensorFlow Graph, PyTorch</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 / LLVM, TPU IR, XLA IR</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 TensorFlow Lite / NNAPI (inc. HW accel)</td>
</tr>
</tbody>
</table>

![Diagram of Machine Learning Compilers](image)

This work is licensed under a Creative Commons Attribution 4.0 International License
# ML Compiler 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**

## 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

## Embedded NN Compilers

- **CEVA Deep Neural Network (CDNN)**
- **Cadence Xtensa Neural Network Compiler (XNNC)**

## 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 (combined primitive commands) like DirectML
OpenCL - Low-level Parallel Programming

- Low-level programming of heterogeneous parallel compute resources
  - One code tree can be executed on CPUs, GPUs, DSPs and FPGA ...

- OpenCL C or C++ language to write kernel programs to execute on any compute device
  - Platform Layer API - to query, select and initialize compute devices
  - Runtime API - to build and execute kernels programs on multiple devices

- The programmer gets to control:
  - What programs execute on what device
  - Where data is stored in various speed and size memories in the system
  - When programs are run, and what operations are dependent on earlier operations
OpenCL is Widely Deployed and Used

Hardware Implementations

Desktop Creative Apps

Linear Algebra Libraries

Parallel Computation Languages

Math and Physics Libraries

Vision and Imaging Libraries

Machine Learning Inferencing Compilers

Machine Learning Libraries
OpenCL Evolution

**OpenCL Extensions**
- Scratch-Pad Memory Management for DSPs
- Vulkan / OpenCL Interop
- Extended Subgroups etc.

**Expanding Language Ecosystem**
- Tighter LLVM integration and cooperation
- Open source C++ for OpenCL Kernel Language
- SPIR-V 1.4 ingestion for compiler efficiency
- SPIR-V Extended debug info

**Improving Software Ecosystem**
- Tool, libraries, ICD Loader
- Regular Maintenance Updates
- Spec clarifications, formatting and bug fixes
  
  [https://www.khronos.org/registry/OpenCL/](https://www.khronos.org/registry/OpenCL/)

**Integration of Extensions plus New Core functionality**

**Repeat Cycle for next Core Specification**

**Focus for OpenCL Next is ‘Deployment Flexibility’**
- Flexible Profile so mobile and embedded vendors can ship customer-targeted functionality and be officially conformant
- Raise the bar for cross-vendor functionality on desktop and in cloud

**Target 2020 ‘OpenCL Next’**
Deploying OpenCL C Over Vulkan

- **Clspv** - Google’s experimental compiler for OpenCL C to Vulkan SPIR-V
  - Open source - tracks top-of-tree LLVM and clang, not a fork
- **Adobe Premiere Rush** has 200K lines of OpenCL C kernel code
  - Professional-quality, cross-platform video capture and editing system
  - Now shipping on Android on Vulkan

![Diagram showing the process of deploying OpenCL C over Vulkan](https://github.com/google/clspv)

This work is licensed under a Creative Commons Attribution 4.0 International License
Need for New Camera Control API Standard?

• Khronos suspended work on OpenKCam standard several years ago
  - Mobile market went proprietary - but embedded market has different needs

• OpenKCAM was aiming at advanced control of ISP and camera with cross-platform portability
  - Generate sophisticated image stream for advanced imaging & vision apps
  - Portable access to growing sensor diversity: e.g. depth sensors and sensor arrays
  - Cross sensor synch: e.g. synch of multiple camera and MEMS sensors
  - Advanced, high-frequency per-frame burst control of camera/sensor: e.g. ROI
  - Multiple input, output re-circulating streams with RAW, Bayer or YUV Processing

Emerging potential liaison opportunity for cooperation - maybe over OpenKCam and GeniCam?
Work together to integrate next generation camera control and acceleration APIs?

Kronos
European Machine Vision Association
MESA / X.org
Embedded Vision Alliance

Image Signal Processor (ISP)

Defines control of Sensor, Color Filter Array
Lens, Flash, Focus, Aperture

Auto Exposure (AE)
Auto White Balance (AWB)
Auto Focus (AF)

Stream of images for downstream processing
Khronos Ecosystem Engagement

- **Khronos Forums and Slack Channels**: Open to all!
  - [https://community.khronos.org/](https://community.khronos.org/)
  - [www.khr.io/slack](http://www.khr.io/slack)

- **Contribute to open source specs, CTS, tools and ecosystem**: Spec fixes and suggestions made under the Khronos IP Framework. Open source contributions under repo’s CLA - typically Apache 2.0
  - [https://github.com/KhronosGroup](https://github.com/KhronosGroup)

- **Advisory Panels**

- **Working Groups**
  - Invited Advisors under the Khronos NDA and IP Framework can comment and contribute to requirements and draft specifications
    - [https://www.khronos.org/advisors/](https://www.khronos.org/advisors/)

- **Khronos members under Khronos NDA and IP Framework participate and vote in working group meetings. Starts at $3.5K/yr**
  - [https://www.khronos.org/members/](https://www.khronos.org/members/)
Thank You and Resources

- Khronos is creating cutting-edge royalty-free open standards
  - For 3D, compute, vision, inferencing acceleration
- These slides and information on Khronos Standards
  - www.khronos.org
- Any company is welcome to join Khronos
  - https://www.khronos.org/members/
- Dedicated developer resources
  - Khronos Developer Forum: https://community.khronos.org/
  - Khronos Developer Slack Channel: www.khr.io/slack
- We are happy to help answer any questions!
  - Neil Trevett, Khronos President: ntrevett@nvidia.com, @neilt3d
  - Khronos Developer Relations, Kris Rose: kris@khronos.org, @kristoferrose