Objectives of Vulkan Machine Learning (ML)

• Enable native Vulkan application to use ML with low latency and overhead
  - Avoid interop, or to embed very large third-party frameworks (python)

• There are many common blocks in Neural Nets:
  - Matrix multiplication
  - Convolution
  - Tanh/Sigmoid/ReLU
  - ...

• But the eco system is actually diverse and new mathematical approaches are introduced every week:
  - Locality sensitive hashing, binary weights, sparse matrices, ...

• Vulkan should not limit itself to current architectures
  - Luckily, Vulkan already has a compute shader abstraction!
Inferencing Acceleration

Networks trained on high-end desktop and cloud systems

Applications link to compiled inferencing code or dynamically generate it

Accélération Hardware (GPUs)
Data type extensions

• By default, the 32 bit floating precision is used for both training and inferencing, which are basically just running a computational graph:
  - Training runs a forward pass, and often times a backward pass to propagate back the gradient
  - Inferencing is just about doing the forward pass

• But both can be done in lower precision types for faster compute time and reduced data storage
  - The following extensions are available in vulkan 1.2:
    - VK_KHR_shader_float16_int8
    - VK_KHR_8bit_storage / VK_KHR_16bit_storage
  - 8 bit integers data types are used for quantized Neural Nets
  - FP16 data types can be used for faster math with gradient rescaling in training
  - Upcoming: SPV_KHR_integer_dot_product
    - Take advantage of fast integer math without implicit compiler peephole optimization for quantized neural networks
Improved Compute Shader

- New extensions are devised to improve efficiency
  - Upcoming:
    - VK_KHR_workgroup_memory_explicit_layout
    - Allow more efficient data loading into shared memory for further use with efficient matrix multiplication operations.
    - VK_EXT_ML_primitives
    - Exposes basic primitives used in the main stream Neural Nets as optimized building blocks
  - Available:
    - VK_NV_cooperative_matrix (NVIDIA)
    - Exposes high throughput matrix/vector multiplication hardware units
    - Typically used be convolution / matmul layer in fp16 formats
## Primary Machine Learning Compilers

<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>SPIR-V</td>
</tr>
<tr>
<td>TensorFlow Graph, PyTorch, ONNX</td>
<td>XLA HLO MLIR</td>
<td>SPIR-V / vulkan API</td>
</tr>
</tbody>
</table>

### Logical Architecture Components

![Logical Architecture Diagram]
Google MLIR and IREE Compilers

Trained Models → Optimizes and Lowers for Acceleration → Generate Hardware Specific Binaries

- TF 2.0
- JAX
- PyTorch
- ONNX

(Gray boxes do not yet exist)

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.
## Third Party frameworks

<table>
<thead>
<tr>
<th></th>
<th>Alibaba</th>
<th>Tencent</th>
<th>Ax inc.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MNN</strong></td>
<td><img src="MNN.png" alt="MNN" /></td>
<td><img src="NCNN.png" alt="NCNN" /></td>
<td><img src="Ailia.png" alt="Ailia" /></td>
</tr>
<tr>
<td><strong>Tencent</strong></td>
<td><img src="NCNN.png" alt="NCNN" /></td>
<td><img src="Ailia.png" alt="Ailia" /></td>
<td></td>
</tr>
<tr>
<td><strong>Ax inc.</strong></td>
<td><img src="Ailia.png" alt="Ailia" /></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **MNN**
  - Lightweight frameworks for inferencing and training for mobile platforms
  - [https://github.com/alibaba/MNN](https://github.com/alibaba/MNN)

- **NCNN**
  - Cross platform optimized inferencing framework
  - [https://github.com/Tencent/ncnn](https://github.com/Tencent/ncnn)

- **Ailia**
  - SDK for optimized running of many popular neural networks on multiple platform
  - [https://axinc.jp/en/](https://axinc.jp/en/)
## Non Neural Net Machine Learning

<table>
<thead>
<tr>
<th>Jülich</th>
<th>Ethical ML</th>
<th>UNITY</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Jülich Forschungszentrum" /></td>
<td><img src="image" alt="Ethical ML" /></td>
<td><img src="image" alt="UNITY" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>VkFFT</strong></th>
<th><strong>Kompute</strong></th>
<th><strong>ML agents</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerated fast fourier transform library, which can be used for image resampling, image registration, signal processing ...</td>
<td>General purpose GPU compute framework cross graphics card</td>
<td>Open source project that enables simulation for training intelligent agents via the Unity engine</td>
</tr>
</tbody>
</table>
Call To Action

- The Vulkan Machine Learning Subgroup is welcoming feedback:
  - What functionality would you like to see in Vulkan to accelerate your ML needs?
    - Neural Nets
    - Other algorithms:
      - random forest
      - logistic regression
      - K-nearest neighbor
      - T-SNE
      - ...
  - Do you have existing product using Vulkan for ML:
    - Do you want to get in touch with hardware vendors?
    - Do you have suggestion on how to improve your pain points?

- Contact us at: pboudier@nvidia.com
- Help us and join the Vulkan Advisory Panel, or become a Khronos member