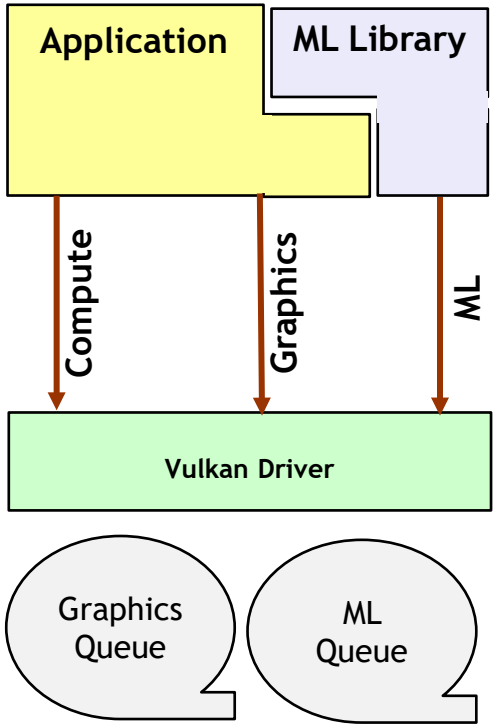
ML Primitives Extension

Jeff Leger, Qualcomm
jleger@qti.qualcomm.com
May 2022

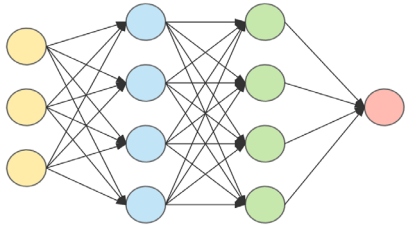
Motivation and Goals

- **Motivation: Efficient dispatch of ML inference workloads**
- **Target Market: Game Engines or Frameworks already using Vulkan for compute or graphics.**
- **Goals**
  - Efficient GPU execution of ML work (optimal layouts, tuned kernels)
  - Fast/easy interop across ML, Graphics, and Compute (zero copies, no stalls)
  - Client-managed cmdBuffers, submits, and synchronization
  - Async dispatch to ML-capable queue family. Could be a separate IP core
  - Alignment to APIs (e.g., DirectML) for portability



Principles

- **Low-level API for accelerating ML (“metacommands”)**
  - Backend target for ML Frameworks or Game Engines
  - VkPipeline objects represent ML Ops.
  - ML workloads recorded into VK cmdBuffer
- **Expose a core set of commonly used ML Ops**
  - 16 Ops exposed via “fixed-function” VkPipelines
    - IHVs to provide “hand-tuned” kerels that may outperform equivalent SPIR-V
  - Additional Ops can be added via compute shader
  - Rely on external definition of ML Ops and semantics (NNEF or ONNX).



Primary

Convolution	Leaky_relu
Deconvolutoin	Softmax
Sigmoid	Maxpool
Relu	Avg_pool
Prelu	Gemm
Tanh	Fully_connected

Normalization

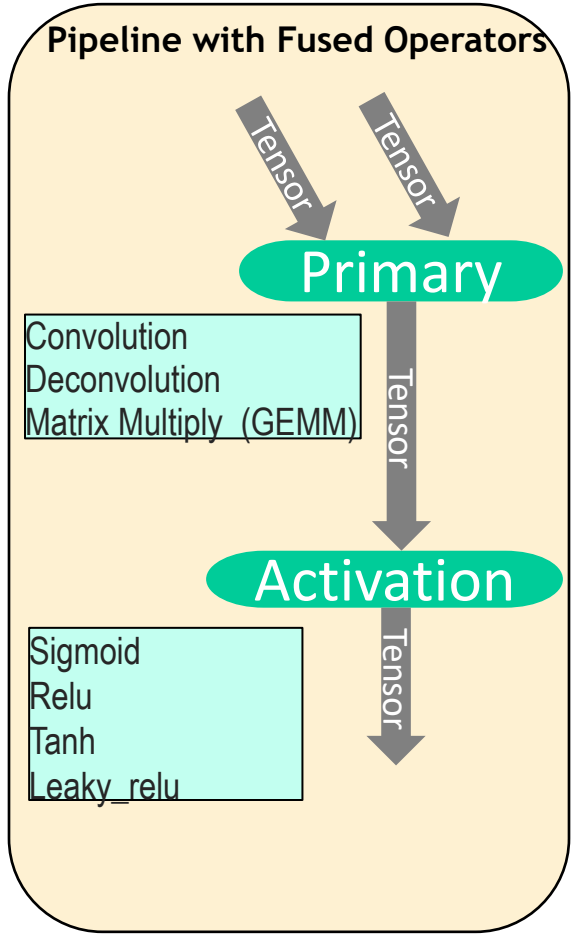
Batch_normalization
Local_response_normalization

Copy

Concat
Reshape

Key Features

- **New resource type, the vkTensor**
  - 4D/5D data storage with flexible layouts (NCHW, NHCW, NHWC)
  - Adds `VK_TENSOR_TILING_OPTIMAL` for efficient layouts
  - Adds **vkTensorView** for HW tensor descriptors
  - Adds Tensor copies and barriers.
- **Fit into existing VK API where possible:**
  - memory allocation, barriers, pipelines, descriptors, command buffers, and queues.
- **Support Fused (primary+activation) ML pipelines**



Tensor access in Compute Pipelines

- **Shader built-ins enable compute pipelines to load/store Tensor resources.**
  - Related GLSL and SPIR-V extensions add tensorLoad() and tensorStore().
  - Tensor type/format is known to compile-time, but tensor dimensions/strides/layout not known.

```
#extension GL_EXT_ML_primitives: enable

layout (r32f, set=0, binding=0) readonly  uniform tensor4D inTensor;
layout (r32f, set=0, binding=1) writeonly uniform tensor4D outTensor;

 void main () {

    uvec4 coords = uvec4(0);
    float f = tensorLoad(inTensor, coords);
    tensorStore(outTensor, coords, f);


    return;
}
```

Creating Tensor Resouces

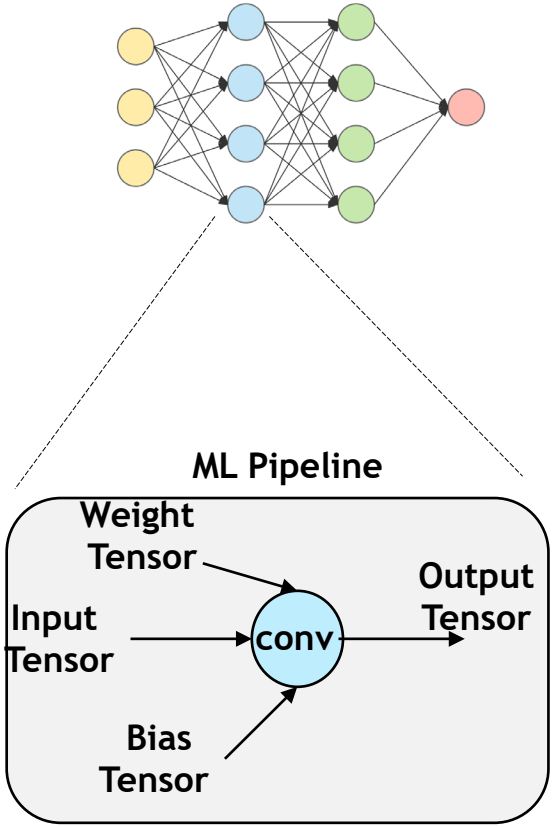
- **vkCreateTensor, vkDestroyTensor**
  - Create and Destroy vkTensor resources
  - 4D (NCHW) / 5D (NCDHW) tensor types. LINEAR/OPTIMAL tiling.
  - Single-component formats, R16_SFLOAT and R32_SFLOAT required
- **vkCreateTensorView, vkDestroyTensorView,**
  - Create/Destroy a views of a Tensor
- **vkGetTensorMemoryRequirements**
  - Implementation-controlled size for backing tensor memory
- **vkBindTensorMemory**
  - binds backing memory for vkTensor
- **vkCmdCopyTensor**
  - Copy data between two tensors

All following existing API patterns for other resource types.

Creating ML Pipelines

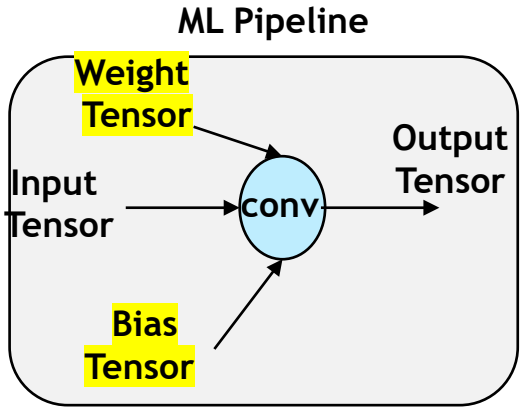
- **`vkCreateMLPipelines`**
  - Creates one or more ML Pipelines
  - Each ML Pipeline implements a single operation (i.e., conv, reshape, prelu, etc.)
  - Each tensor inputs / outputs is fully described
    - Tensor size/shape, formats, bindings, etc.

- **ML Pipelines can optionally read/write to VkImages/VkBuffers as if they are VkTensors.**
  - For use-cases where ML naturally interops with buffer/image resources from graphics.
  - 2D VkImage treated as a 4D Tensor, were (N=1, C = “VkFormat component count”.



ML Pipeline Interfaces

- **`ML Ops`** may define **`Static`** Tensor inputs
  - Typically, constant across model invocations. Must be **`TILING_LINEAR`**.
  - Used for **`Weight`** and **`Bias`** inputs to **`Convolution`**.
- **`ML Ops`** may require supplemental storage buffers
  - **`Scratch buffer`** for Op-private transient data
  - **`Constant buffer`** for HW-optimized static data
  - Same as *temporary* and *persistent* resources in DirectML.
- **`vkCmdUpdateMLConstantBuffer`** reads static tensors and may write to the constant buffer in a HW-optimized layout.
- **`vkGetMLPipelineMemoryRequirements`** gets required supplemental buffer size(s)



ML Pipeline Creation Structs (1 of 2)

- `VkMLPrimitiveIdEXT` enum identifies the ML operation
- Set/binding for supplemental resources (scratch+constants)
- `pPrimCreateInfo` points to an Op-specific creation structure (next slide)

```
typedef struct VkPipelineMLCreateInfoEXT {
    VkStructureType                 sType;
    VkPipelineCreateFlags           flags;
    const void*                     pPrimCreateInfo;
    VkMLPrimitiveIdEXT              primitiveID;
    int32_t                         primitiveVersion;
    VkMLIntermediatePrecisionEXT    precision;
    uint32_t                        constantSet;
    uint32_t                        constantBinding;
    uint32_t                        scratchSet;
    uint32_t                        scratchBinding;
    VkPipelineLayout                layout;
    VkPipeline                      basePipelineHandle;
    int32_t                         basePipelineIndex;
} VkPipelineMLCreateInfoEXT;
```

No application shader is provided. The implementation provides a HW-optimized kernel.

ML Pipeline Creation Structs (2 of 2)

```
typedef struct VkMLPipelineLeakyReluCreateInfoEXT {
    VkStructureType                    sType;
    const VkMLResourceBindingStateEXT* pX;
    const VkMLResourceBindingStateEXT* pY;
    float                              pAlpha; }
VkMLPipelineNnefPreluCreateInfoEXT;
```

```
typedef struct VkMLResourceBindingStateEXT {
    VkStructureType                sType;
    VkMLBindingTypeEXT             bindingType;
    const VkTensorDescriptionEXT*  pTensorDesc;
    uint32_t                       set;
    uint32_t                       binding;
} VkMLResourceBindingStateEXT;
```

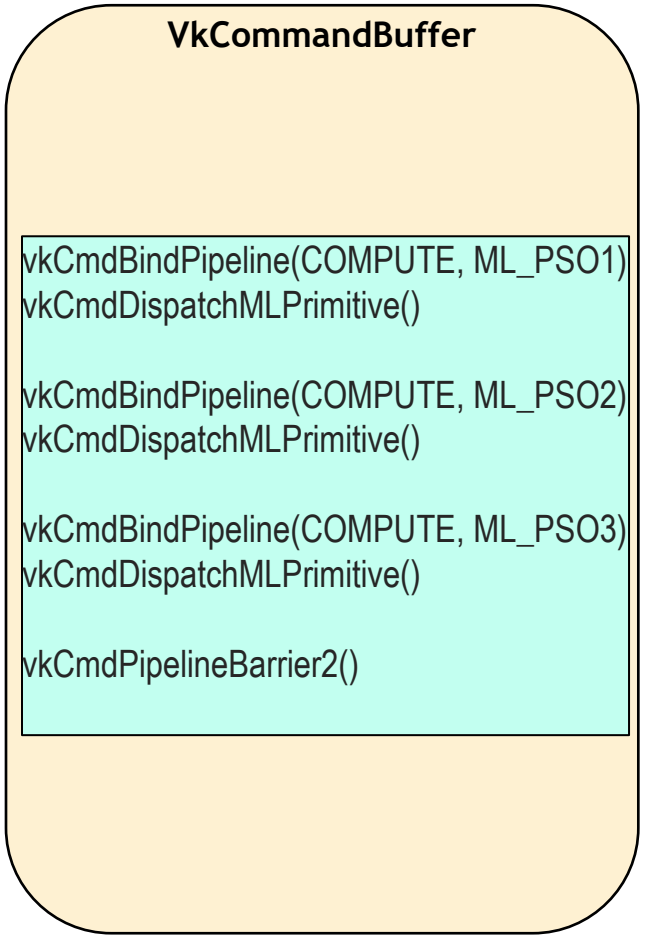
```
typedef struct VkTensorDescriptionEXT {
    VkStructureType       sType;
    VkTensorTypeEXT       type;
    VkTensorTilingEXT     tiling;
    VkFormat              format;
    uint32_t              dimensionCount;
    const uint32_t*       pDimensions;
    const uint32_t*       pStrides;
    VkTensorUsageFlagsEXT usage;
} VkTensorDescriptionEXT;
```

The create struct fully describes each of the input and output tensor (e.g., format, dimensions, strides, set/binding, etc), allowing creation of a fully-specialized kernel.

Dispatching ML work

- **vkCmdDispatchMLPrimitive(vkCommandBuffer c)**
  - Records a dispatch of the currently bound ML Pipeline.
  - Iterates over elements in the output Tensor
  - reads from input Tensors/buffers/images as needed.

- **vkCmdPipelineBarrier2KHR**
  - Synchronization of ML workloads is accomplished via extension structs
  - VkDependencyInfoTensorBarriersEXT describes tensor resource barriers



Status and plans

- **“EXT_ML_primitives” proposed extension spec is available**
  - Currently drafted as a cross-vendor extension
  - Internally reviewed by Khronos members. Feedback incorporated.
- **QCOM has a beta implementation**
  - Future exposure in public Adreno drivers will depend on partner feedback / interest
  - QCOM also has similar vendor extension for OpenCL, shipping in Adreno drivers.
- **Today’s Call to Action:**
  - Looking for feedback from ISVs and/or framework owners

Thank you

KHRONOS GROUP