XDC
2025

Machine learning with NIR

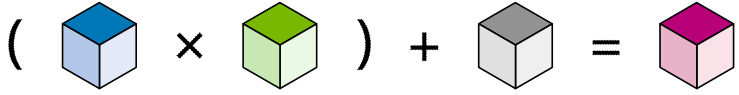
Tomeu Vizoso – tomeu@tomeuvizoso.net
Philipp Zabel – p.zabel@pengutronix.de

Agenda Part 1

- Difference between shader and NN operation graph
- ffma and conv2d
- Tensor operation replacements in etnaviv
  - Fusing non-linear activation functions
  - int8_t on uint8_t hardware
  - Addition using conv2d
  - Strided conv2d
  - Per-axis quantization

Shader vs neural network 1/2

- Shader
  - Basic unit: 32-bit float scalar
  - Arithmetic operations, load&store, flow control
  - Representative operation: fused multiplication and addition



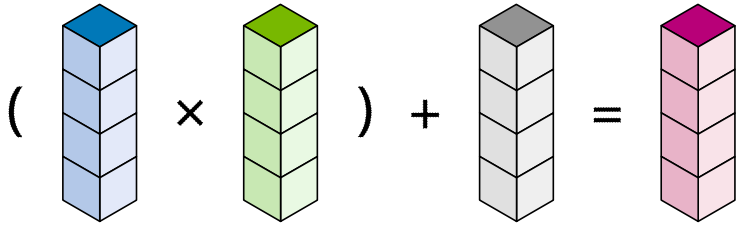
Neural network

  - Basic unit: tensor, quantization
  - Convolutions, non-linear activation, element-wise arithmetic
  - Saturated arithmetic
  - Representative op: Conv2D

Shader vs neural network 2/2

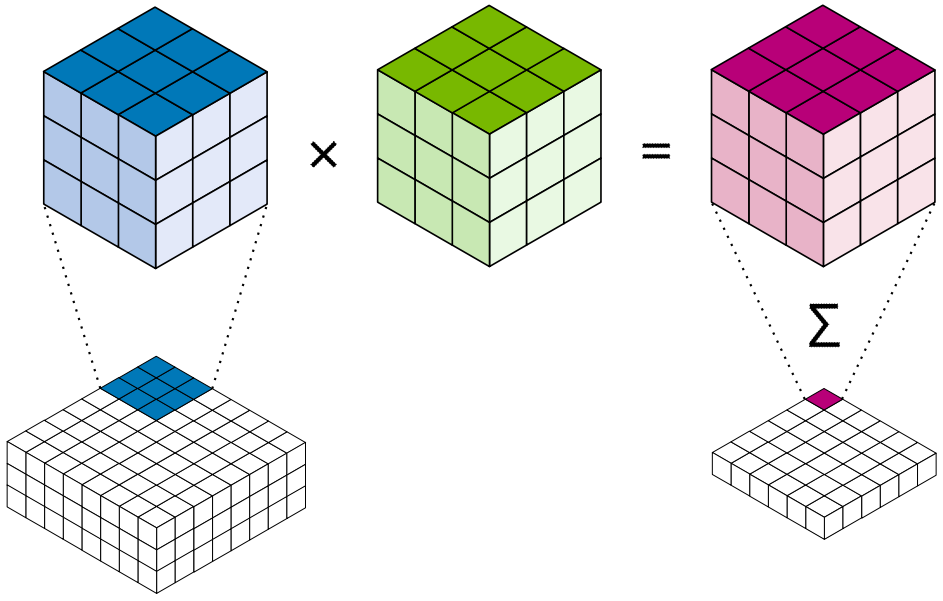
- Shader
  - Basic unit: 32-bit float scalar or 4-vector
  - Arithmetic operations, load&store, flow control
  - Representative operation: fused multiplication and addition

Neural network



  - Basic unit: tensor, quantization
  - Convolutions, non-linear activation, element-wise arithmetic
  - Saturated arithmetic
  - Representative op: Conv2D

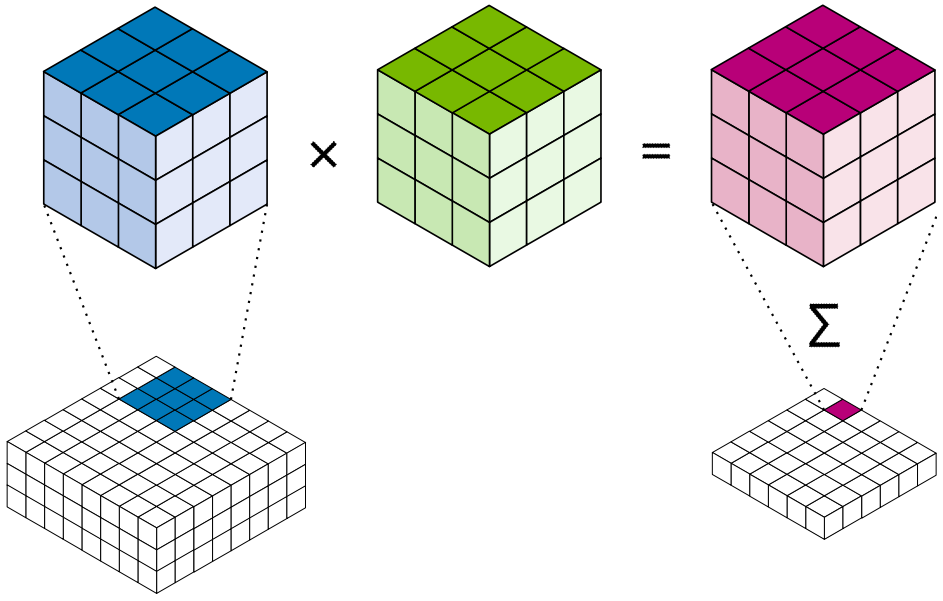
Conv2D 1/6



- Tensor input
- 8-bit quantization [1]
  $r = S * (q - z)$
- Weight tensor constant, compressed
- $w_x * w_y * w_z$ muls, Σ together around each 2D input pos

[1] https://ai.google.dev/edge/litert/models/quantization_spec

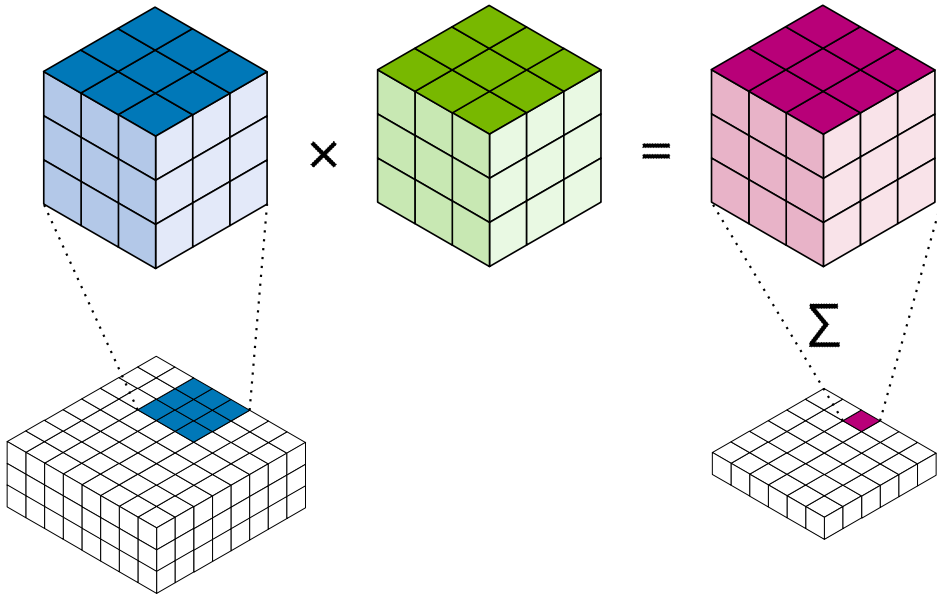
Conv2D 2/6



- Tensor input
- 8-bit quantization [1]
  $r = S * (q - z)$
- Weight tensor constant, compressed
- $w_x * w_y * w_z$ muls, Σ together around each 2D input pos

[1] https://ai.google.dev/edge/litert/models/quantization_spec

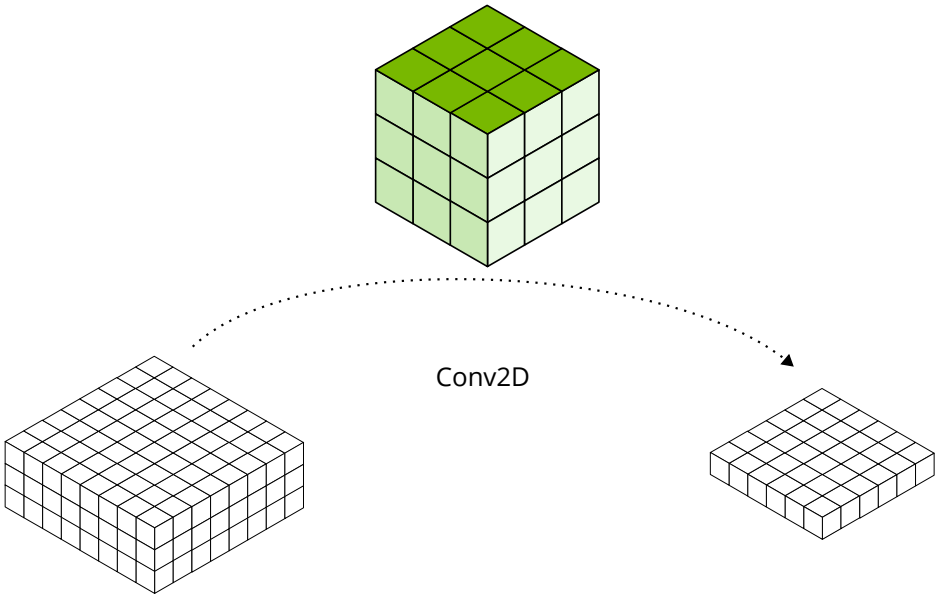
Conv2D 3/6



- Tensor input
- 8-bit quantization [1]
  $r = S * (q - z)$
- Weight tensor constant, compressed
- $w_x * w_y * w_z$ muls, Σ together around each 2D input pos

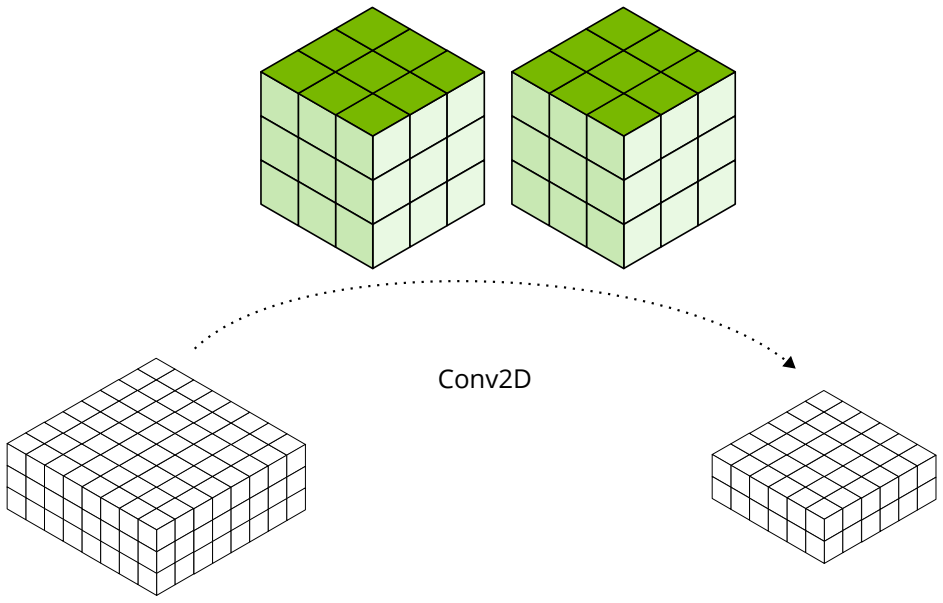
[1] https://ai.google.dev/edge/litert/models/quantization_spec

Conv2D 4/6



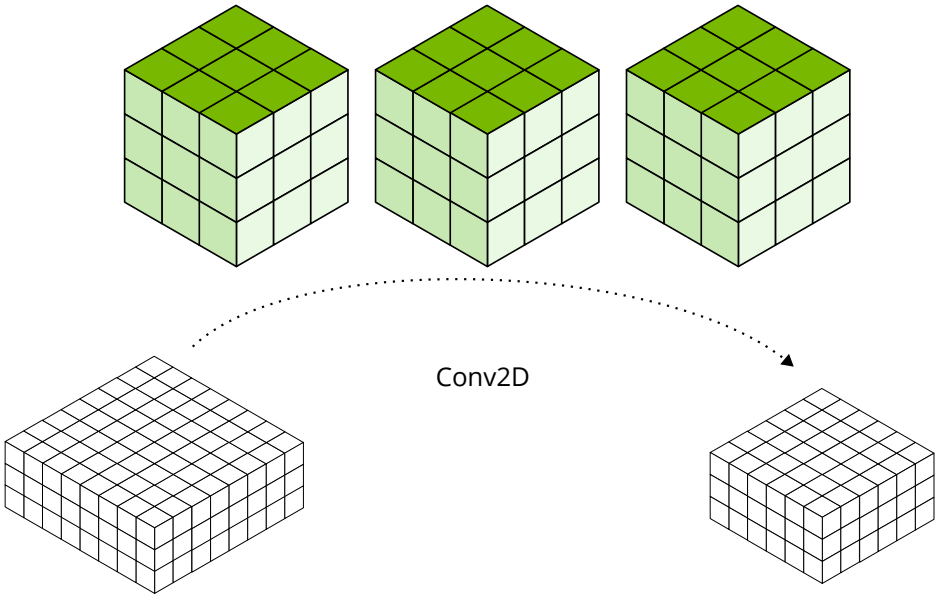
- 4th weight dimension for multiple output channels

Conv2D 5/6



- 4th weight dimension for multiple output channels
- allows work to be split over multiple cores

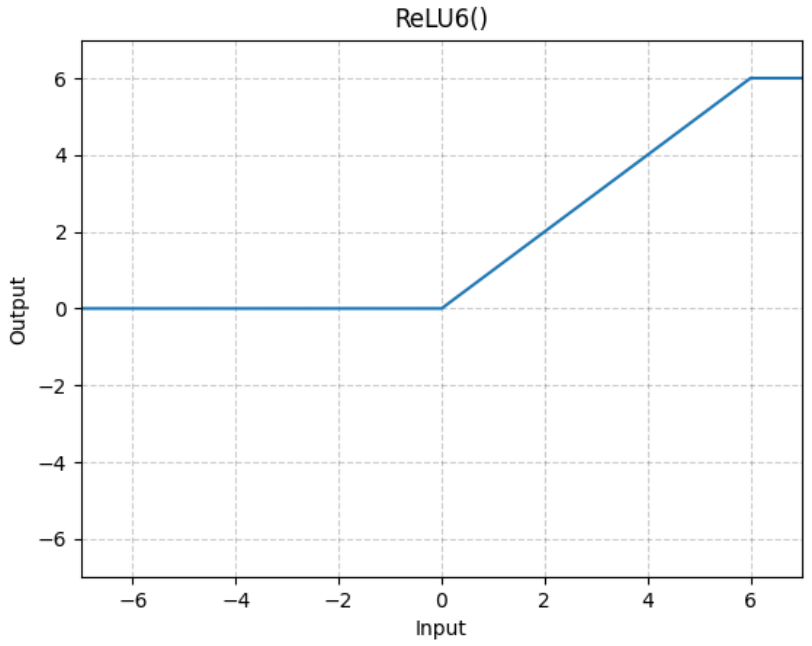
Conv2D 6/6



- 4th weight dimension for multiple output channels
- allows work to be split over multiple cores

Fusing activation

- Some non-linear activation functions can be fused into preceding operations
- Hardware supports fused convolution + ReLU
- ReLU6 is a no-op if output tensor quantization limits are within [0...6]

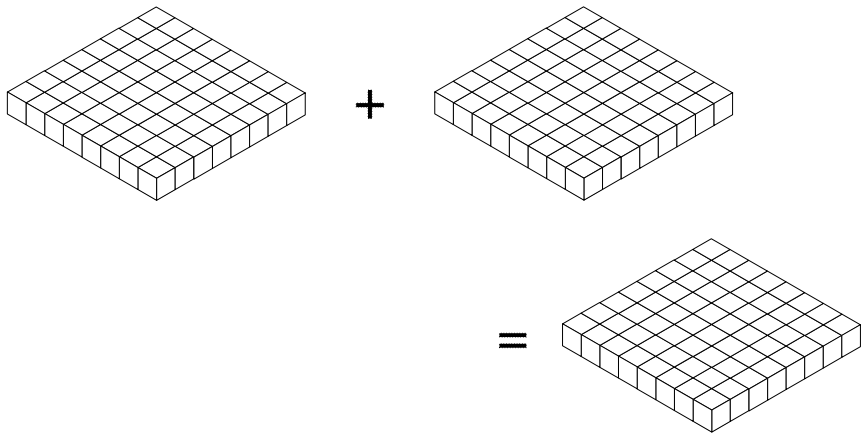


https://docs.pytorch.org/docs/stable/generated/torch.nn.ReLU6.html

int8_t on uint8_t-only hardware

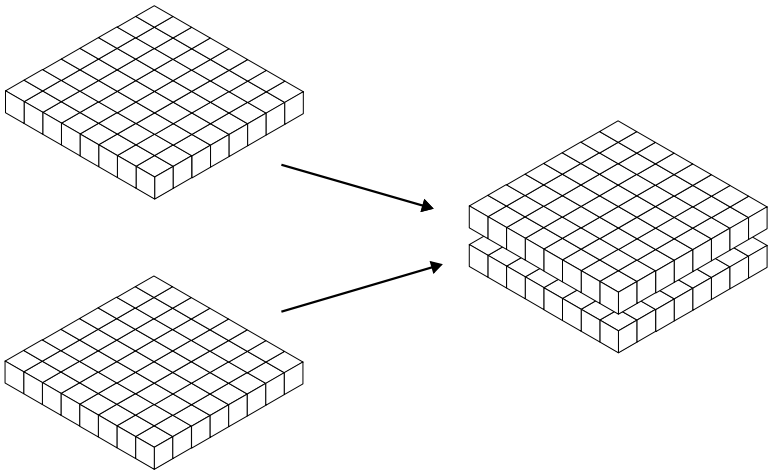
- Model specifies int8_t quantization
- Hardware only supports uint8_t
- Add 128 to all inputs, to move from [-128..127] to [0..255], subtract 128 from all output tensors
- Adapt tensor zero points and do all operations on uint8_t tensors

Addition using conv2d 1/3



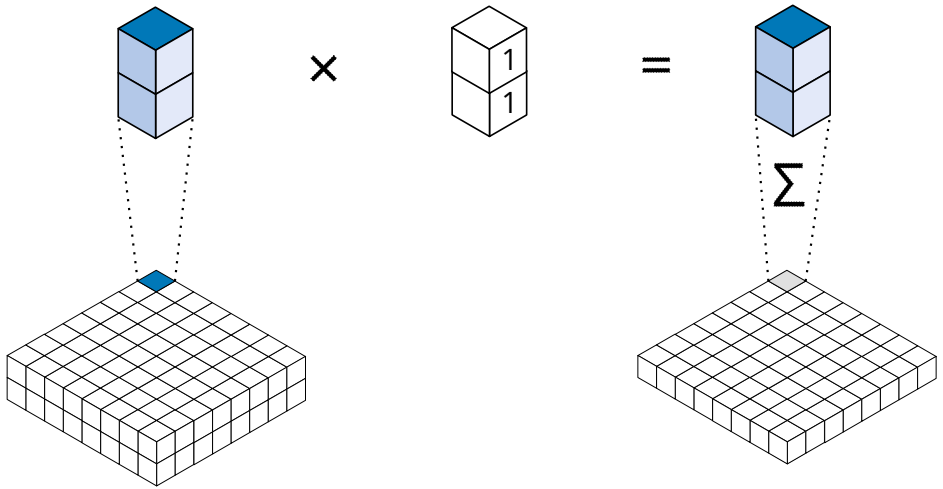
- No fixed function tensor-tensor addition
- Shader fallback slow

Addition using conv2d 2/3



- No fixed function tensor-tensor addition
- Shader fallback slow
- Concatenate input tensors

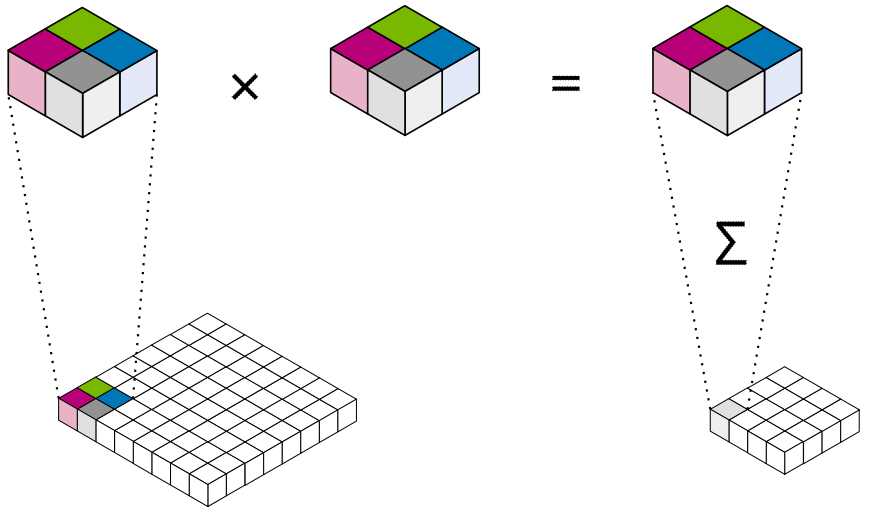
Addition using conv2d 3/3



- No fixed function tensor-tensor addition
- Shader fallback slow
- Concatenate input tensors
- Use conv2d with 2 channels and kernel containing 1.0s

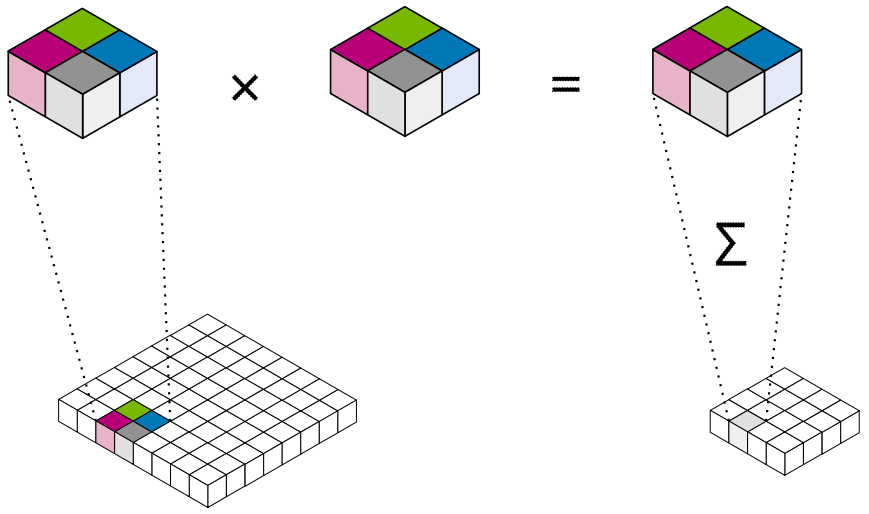
Strided conv2d 1/6

- e.g. stride = 2
- No hardware support



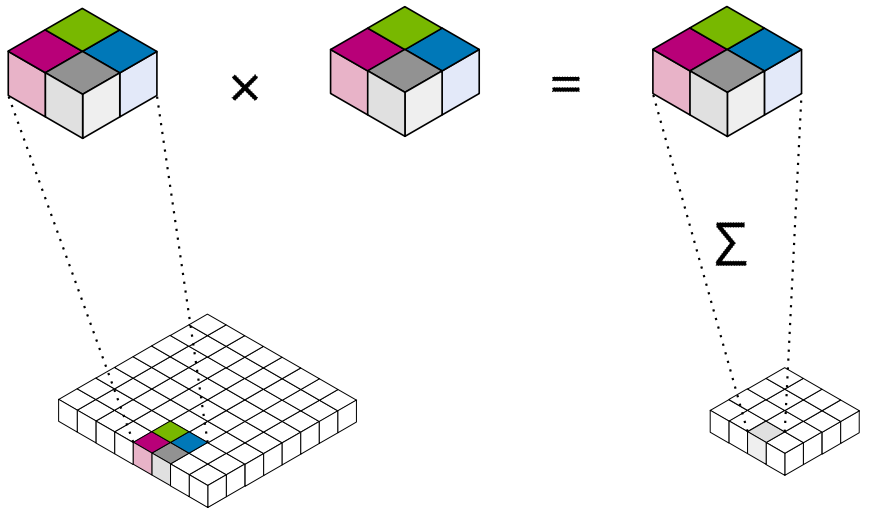
Strided conv2d 2/6

- e.g. stride = 2
- No hardware support



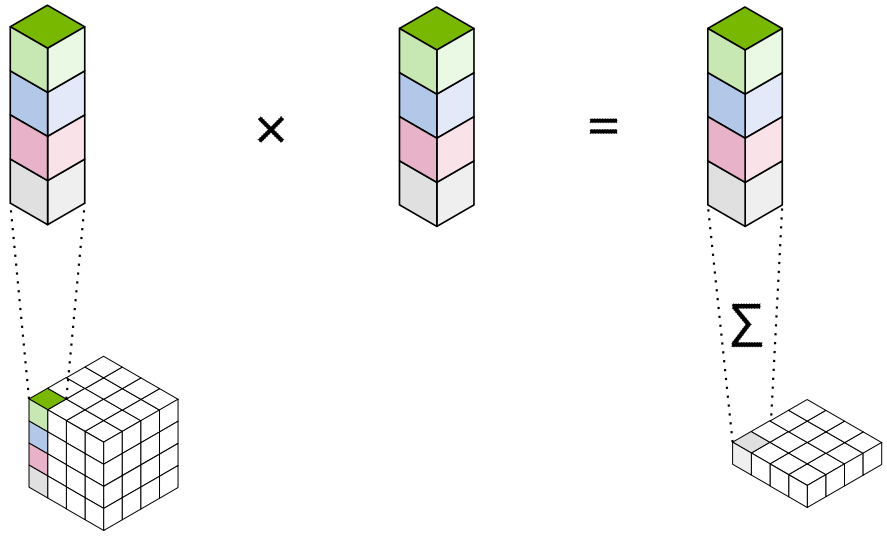
Strided conv2d 3/6

- e.g. stride = 2
- No hardware support



Strided conv2d 4/6

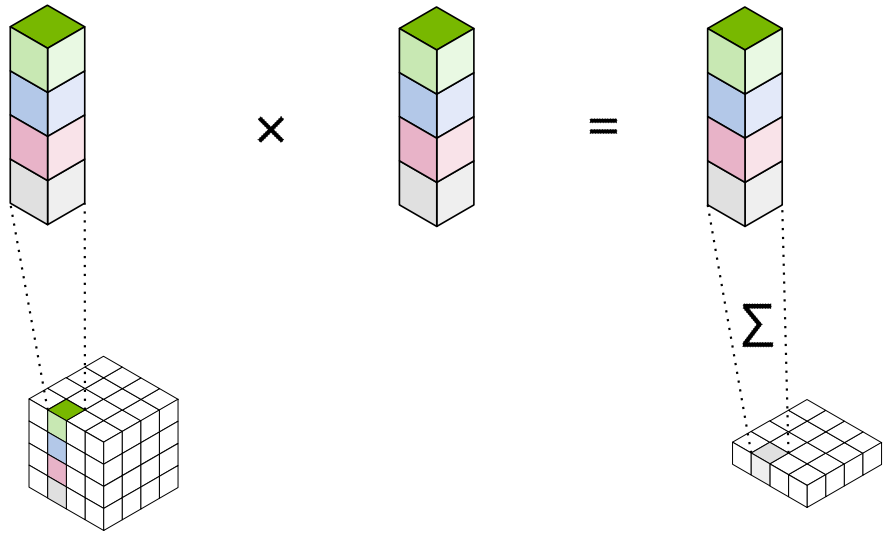
- e.g. stride = 2
- No hardware support
- Reorder input and weight tensor [1]



[1] „Take it in your stride: Do we need striding in CNNs?“ - https://arxiv.org/abs/1712.02502

Strided conv2d 5/6

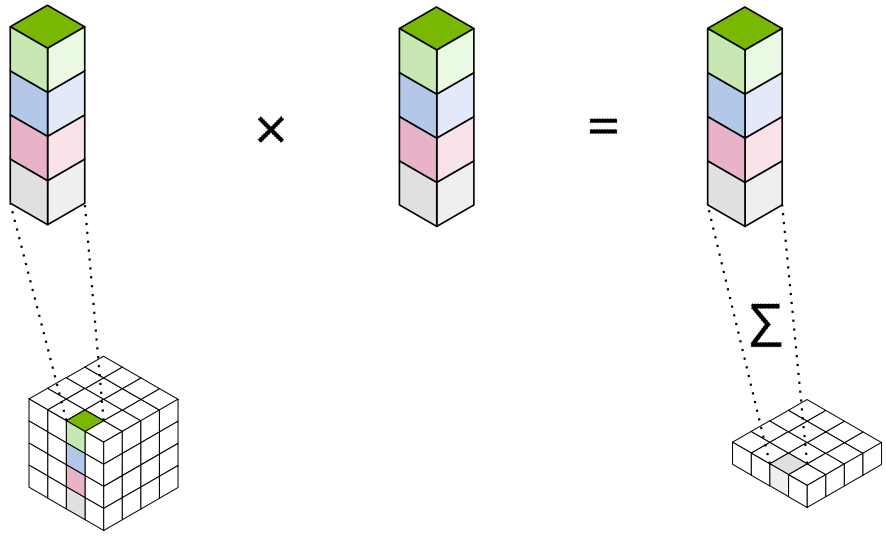
- e.g. stride = 2
- No hardware support
- Reorder input and weight tensor [1]



[1] „Take it in your stride: Do we need striding in CNNs?“ - https://arxiv.org/abs/1712.02502

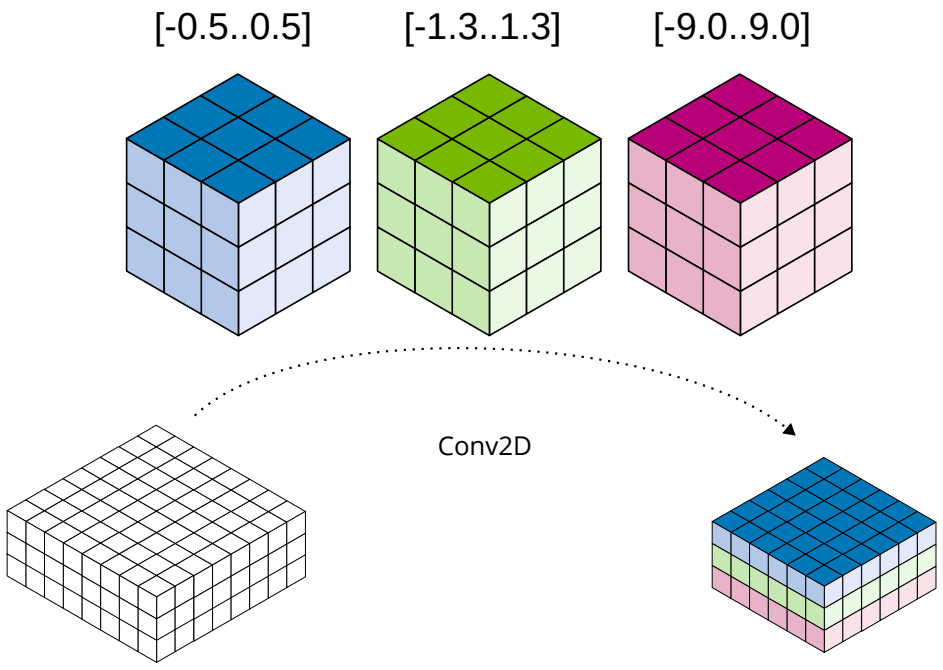
Strided conv2d 6/6

- e.g. stride = 2
- No hardware support
- Reorder input and weight tensor [1]



[1] „Take it in your stride: Do we need striding in CNNs?“ - https://arxiv.org/abs/1712.02502

Per-axis quantization



- Weight tensor quantization per output channel [1]
- Split into multiple conv2d with a single output channel each
- Concatenate output tensors
- Overhead scales with number of channels
- Reduces parallelism in etnaviv

[1] https://ai.google.dev/edge/litert/models/quantization_spec

Current situation: Gallium

```
struct pipe_tensor {
   struct pipe_resource *resource;
   unsigned index;
   unsigned dims[4];
   float scale;
   float *scales;
   int zero_point;
   int *zero_points;
   bool is_signed;
};
```

XDC
2025

Current situation: Gallium (2)

```
enum pipe_ml_operation_type {
   PIPE_ML_OPERATION_TYPE_ADD,
   PIPE_ML_OPERATION_TYPE_CONVOLUTION,
   PIPE_ML_OPERATION_TYPE_POOLING,
   PIPE_ML_OPERATION_TYPE_CONCATENATION,
   PIPE_ML_OPERATION_TYPE_SPLIT,
   PIPE_ML_OPERATION_TYPE_PAD,
   PIPE_ML_OPERATION_TYPE_FULLY_CONNECTED,
   PIPE_ML_OPERATION_TYPE_RESHAPE,
   PIPE_ML_OPERATION_TYPE_RELU,
   PIPE_ML_OPERATION_TYPE_ABSOLUTE,
   PIPE_ML_OPERATION_TYPE_LOGISTIC,
   PIPE_ML_OPERATION_TYPE_SUBTRACT,
   PIPE_ML_OPERATION_TYPE_TRANSPOSE,
};
```

XDC 2025

Current situation: Gallium (3)

```
struct pipe_ml_operation
{
   enum pipe_ml_operation_type type;

   struct pipe_tensor **input_tensors;
   unsigned input_count;

   struct pipe_tensor **output_tensors;
   unsigned output_count;

   union {
      struct {
         struct pipe_tensor *weight_tensor;
         struct pipe_tensor *bias_tensor;
         unsigned stride_x;
         unsigned stride_y;
         bool padding_same;
         bool pointwise;
         bool depthwise;
         bool relu;
         unsigned dilation_width_factor;
         unsigned dilation_height_factor;
      } conv;
      struct {
         unsigned stride_x;
         unsigned stride_y;
         unsigned filter_width;
         unsigned filter_height;
         bool padding_same;
      } pooling;
      ....
```

Current situation: Drivers

```c
#define MAX_TENSORS 10
struct etna_operation {
   struct list_head link;

   enum etna_job_type type;
   enum etna_ml_tp_type tp_type;

   bool addition;
   bool depthwise;
   bool pointwise;
   bool fully_connected;
   bool
   bool padding_same;
   bool relu;

   unsigned stride;
```

bool etna_operation::padding_same

```c
   unsigned input_tensors[MAX_TENSORS];
   unsigned input_count;
   unsigned input_tensor_sizes[MAX_TENSORS];

   unsigned input_width;
   unsigned input_height;
   unsigned input_channels;
   uint8_t input_zero_point;
   float input_scale;

   unsigned output_tensors[MAX_TENSORS];
   unsigned output_count;
   unsigned output_tensor_sizes[MAX_TENSORS];
```

Current situation: Drivers (2)

```
struct etna_vip_instruction {
   enum etna_job_type type;
   enum etna_ml_tp_type tp_type;

   struct etna_bo *configs[MAX_CONFIG_BOS];
   struct etna_bo *coefficients;
   struct etna_bo *pwl_lut;
   struct pipe_resource *input;
   unsigned input_offset;
   struct pipe_resource *output;
   unsigned output_offset;

   struct etna_bo *kernel;
};
```

XDC 2025

Proposal: Use NIR as the base for these IRs

- RFC: Add tensor operations to NIR
- https://gitlab.freedesktop.org/mesa/mesa/-/merge_requests/35441/

XDC
2025

Tomeu Vizoso <tomeu@tomeuvizoso.net>

Tensor GLSL type (src/compiler)

- New GLSL_TYPE_TENSOR in enum glsl_base_type
- New description for this type:

```
#define GLSL_MAX_TENSOR_RANK 4

struct glsl_tensor_description {
   enum glsl_base_type element_type;
   uint8_t rank; // Number of dimensions (e.g., 2 for matrix, 4 for a 4D tensor)
   uint32_t dims[GLSL_MAX_TENSOR_RANK]; // Array of dimension sizes (e.g., {D0, D1, D2, D3})
   int zero_point;
   float scale;
   bool channels_last;
};
```

XDC 2025

How intrinsics use the tensor type

- Each tensor used in a graph is represented by a nir_variable with a glsl_base_type of `GLSL_TYPE_TENSOR`
- Intrinsics such as `nir_conv2d` operate on references to these variables
- The `glsl_tensor_description` struct contains all information that defines the type of the tensor. We can deref the `nir_def`, then get the tensor definition with `glsl_get_tensor_description(deref->type)`

XDC 2025

Intrinsics

```
intrinsic(name='conv2d',
          src_comp=[-1, -1, -1, -1],  # Srcs: InputTensorHandle, KernelTensorHandle,
          dest_comp=-1,               # No SSA destination; output is via writing to
          indices=[STRIDE_X, STRIDE_Y, PADDING, DILATION_X, DILATION_Y, DEPTHWISE],
         )

intrinsic(name='transpose',
          src_comp=[-1, -1],  # Srcs: InputTensorHandle, OutputTensorHandle
          dest_comp=-1,       # No SSA destination; output is via writing to OutputTe
          indices=[PERMUTATION],
         )
```

XDC
2025

Program generation: convolutions

```
TfLiteConvParams* params = (TfLiteConvParams*)node->builtin_data;
nir_conv2d(b, input_src, weights_src, bias_src, output_src,
   .stride_x = params->stride_width,
   .stride_y = params->stride_height,
   .dilation_x = params->dilation_width_factor,
   .dilation_y = params->dilation_height_factor,
   .padding = padding_to_nir(params->padding),
   .depthwise = false,
   );
```

XDC
2025

Tomeu Vizoso <tomeu@tomeuvizoso.net>

Program generation: tensor additions

```
nir_def *input_0_src = tensor_to_nir(tf_context, b, "input_feature_map_0", node->input
nir_def *input_1_src = tensor_to_nir(tf_context, b, "input_feature_map_1", node->input
nir_def *output_src = tensor_to_nir(tf_context, b, "output_feature_map", node->outputs

nir_tensor_add(b, input_0_src, input_1_src, output_src);
```

XDC
2025

Tomeu Vizoso <tomeu@tomeuvizoso.net>

Compiler passes: ensure NCHW

```
bool
etna_ensure_nchw(nir_shader *psubgraph, struct etna_ml_subgraph *subgraph)
{
   bool progress = false;

   nir_foreach_function_impl(impl, psubgraph) {
      nir_builder b = nir_builder_create(impl);
      bool func_progress = false;

      nir_foreach_block(block, impl) {
         nir_foreach_instr(instr, block) {
            if (instr->type == nir_instr_type_intrinsic) {
               nir_intrinsic_instr *intr = nir_instr_as_intrinsic(instr);
               switch(intr->intrinsic) {
                  case nir_intrinsic_conv2d: {
                     bool input_channels_last = etna_ml_get_tensor_desc(intr->src[0].ssa)->char
                     bool output_channels_last = etna_ml_get_tensor_desc(intr->src[3].ssa)->cha

                     if (input_channels_last) {
                        /* The tensor is in channel-last, but the hardware needs channel-first
                        add_transpose(subgraph, &b, intr, 0, true);
                        func_progress = true;
                     }
```

XDC 2025

Compiler passes: ensure NCHW (2)

```
static void
add_transpose(struct etna_ml_subgraph *subgraph, nir_builder *b, nir_intrinsic_instr *intr,
{
   b->cursor = nir_before_instr(&intr->instr);
   nir_def *interm_tensor = create_tensor(subgraph, b, intr->src[src_idx].ssa, false);
   if (last_to_first) {
      b->cursor = nir_before_instr(&intr->instr);
      nir_transpose(b, intr->src[src_idx].ssa, interm_tensor, .permutation = {0, 3, 1, 2});
   } else {
      b->cursor = nir_after_instr(&intr->instr);
      nir_transpose(b, interm_tensor, intr->src[src_idx].ssa, .permutation = {0, 2, 3, 1});
   }
   nir_src_rewrite(&intr->src[src_idx], interm_tensor);
}
```

XDC
2025

Compiler passes: remove dead transposes

```
bool
etna_dead_transposes(nir_shader *psubgraph, struct etna_ml_subgraph *subgraph)
{
   bool progress = false;

   nir_function_impl *impl = nir_shader_get_entrypoint(psubgraph);
   for_each_transpose_safe(impl) {
      nir_intrinsic_instr *intr = nir_instr_as_intrinsic(instr);
      struct glsl_permutation perm = nir_intrinsic_permutation(intr);
      unsigned output_idx = tensor_idx_from_src(instr, 1);
      bool can_be_removed = true;
      bool first_to_last = perm.dims[0] == 0 &&
                           perm.dims[1] == 2 &&
                           perm.dims[2] == 3 &&
                           perm.dims[3] == 1;
      struct util_dynarray candidates = {0};
      util_dynarray_init(&candidates, NULL);
      for_each_consumer(impl, output_idx) {
         if (!check_dead_transposes(impl, instr, first_to_last, &candidates)) {
            can_be_removed = false;
            break;
```

Compiler passes: remove dead transposes (2)

```
      if (can_be_removed && util_dynarray_num_elements(&candidates, nir_instr *)) {
         nir_def_rewrite_uses(intr->src[1].ssa, intr->src[0].ssa);
         nir_instr_remove(&intr->instr);

         util_dynarray_foreach(&candidates, nir_instr *, instr) {
            nir_intrinsic_instr *transpose = nir_instr_as_intrinsic(*instr);
            nir_def_rewrite_uses(transpose->src[1].ssa, transpose->src[0].ssa);
            nir_instr_remove(&transpose->instr);
         }
      }
      util_dynarray_fini(&candidates);
   }

   nir_progress(progress, impl, nir_metadata_none);

   return progress;
}
```

XDC 2025

Compiler passes: lower strided convolution

- A particular NPU might not support strided convolutions natively
- It has been proved that a strided convolution can be lowered to a normal one without any loss in accuracy
- There may be performance implications, but the results should be the same
- We can provide a HW-independent lowering pass that replaces all strided convolutions with regular ones, for drivers to use as needed
- See the implementation in the merge request

XDC 2025

Tomeu Vizoso <tomeu@tomeuvizoso.net>

Why doing all this with NIR?

- We need infrastructure to create several IRs:
  - one to represent the operations that ML frameworks give to us, which we can use for lowering to hardware-specific IRs but also for common optimizations and lowering passes,
  - at least one per driver to lower to something closer to the HW, for performing further optimization passes and in some cases to lower further, and
  - some drivers will need one more IR before hardware instructions can be generated.

XDC 2025

Why doing all this with NIR? (2)

- If we didn’t have NIR yet, we would need to write it anyway (and maintain it!)
- Vulkan drivers are probably going to need at some point to implement extensions with operations on tensors
- Seems like NIR can be extended in this direction with little disruption?

XDC 2025

Questions

- Or you can also ask in #ml-mainline at OFTC
- Or feel free to ask Philipp or Tomeu any time after this talk

XDC
2025