

01 Motivation: RIFE video frame interpolation

RIFE uses `grid_sample` for image warping to generate intermediate frames in video processing. Supporting this operator efficiently is important for real-world workloads on edge GPUs.



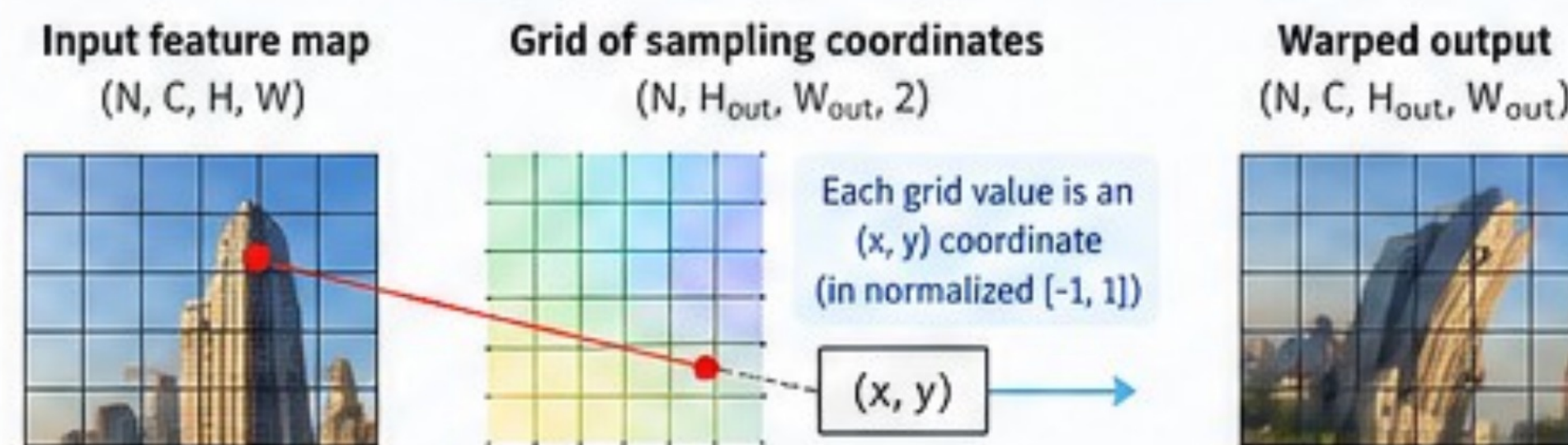
Neural Technology

What is Neural Technology?

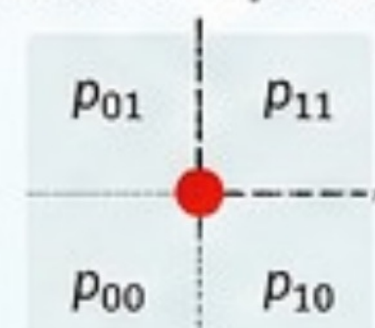
Arm Neural Technology uses machine learning to improve mobile rendering workflows. It includes Neural Frame Rate Upscaling (NFRU), which generates intermediate frames.

02 What is grid_sample?

Samples input features at locations specified by a grid of (x, y) coordinates.



Bilinear interpolation (2D)



$$O(x, y) = w_{00}p_{00} + w_{01}p_{01} + w_{10}p_{10} + w_{11}p_{11}$$

Four neighbouring values contribute according to distance.

03 Why is it difficult to lower to standard TOSA?

Complex operator semantics

- Coordinate normalization**
Grid coordinates in [-1, 1] → input coordinates
- Interpolation mode**
nearest | bilinear | bicubic
- Out-of-bounds behaviour**
zeros | border | reflection
- Corner convention**
align_corners = True | False

Backend-specific concerns

- Tensor layout**
NCHW ↔ NHWC
- Quantization**
quantized grids and feature maps
- Image formats**
mapping to GPU image/sampler
- GPU sampling**
leveraging native sampling hardware



No direct standard-TOSA grid_sample operator

TOSA provides portable tensor primitives, but it does not have an operator that directly captures PyTorch `grid_sample` semantics. Reconstructing the operation would require a large and complex standard-TOSA graph.

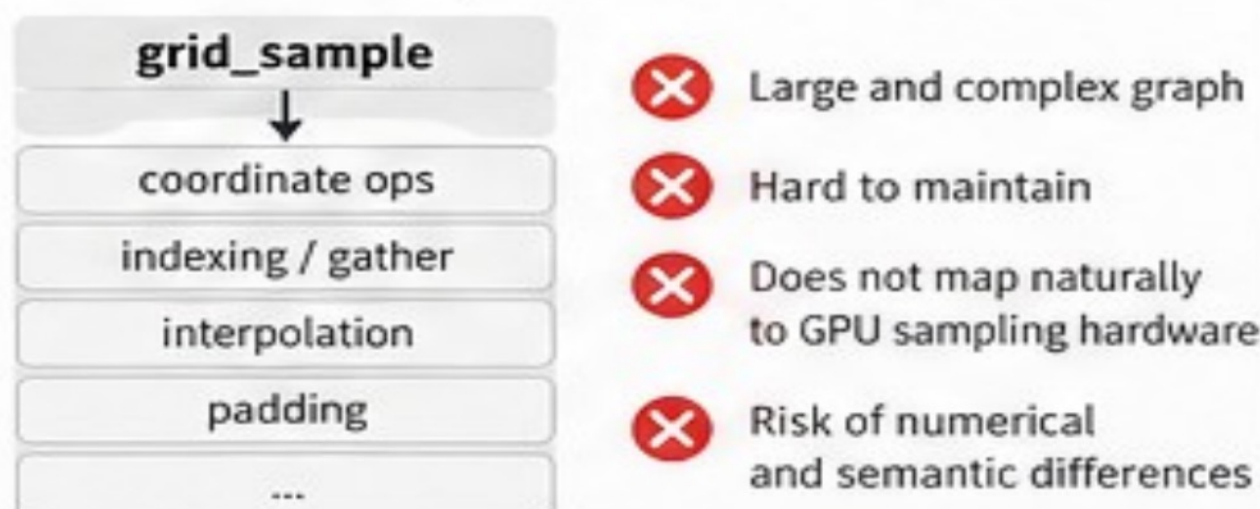
04 Design decision: preserve semantics with tosa.CUSTOM

Reusable beyond grid_sample

A reusable path for PyTorch operators outside standard TOSA to backend-native GPU capabilities.

Decompose to standard-TOSA graph

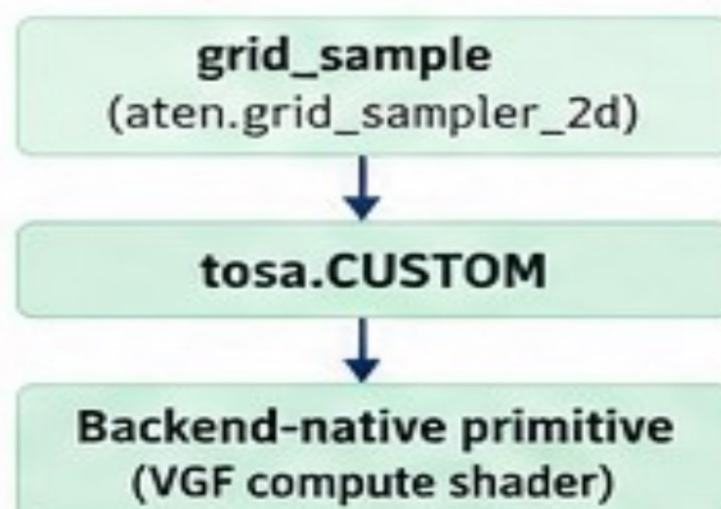
Represents `grid_sample` using many lower-level TOSA operations.



- ❌ Large and complex graph
- ❌ Hard to maintain
- ❌ Does not map naturally to GPU sampling hardware
- ❌ Risk of numerical and semantic differences

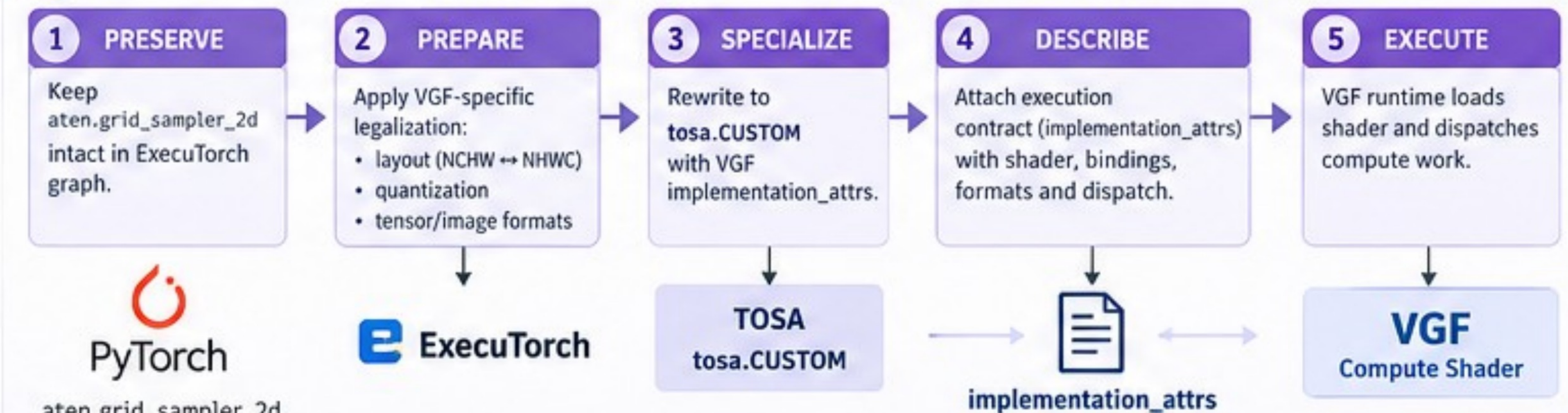
Preserve as a custom operator

Keep `grid_sample` as a semantic operation and map to a backend-native primitive.



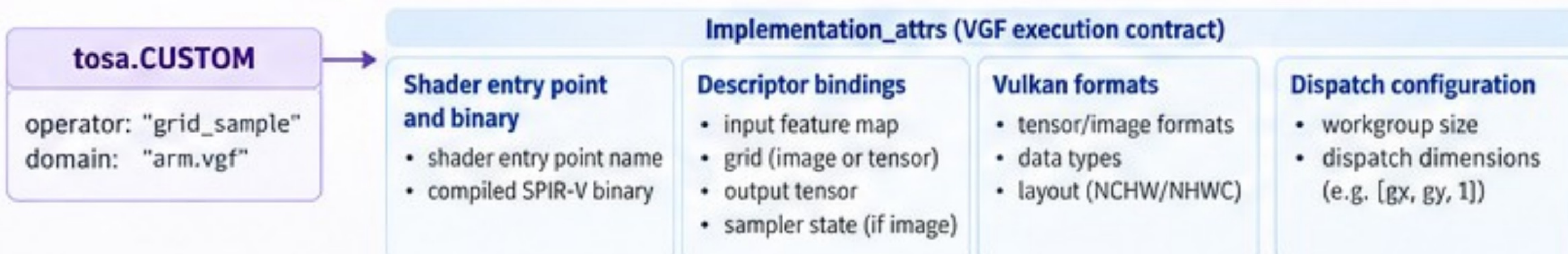
- ✅ Preserves PyTorch semantics
- ✅ Clean graph boundary in ExecuTorch/TOSA
- ✅ Maps naturally to GPU sampling capabilities
- ✅ Efficient implementation on VGF

05 How it works: from PyTorch to VGF execution



06 What does tosa.CUSTOM carry?

`tosa.CUSTOM` is not just an unsupported-op marker — it carries a complete VGF execution contract.



07 Results & validation

End-to-end correctness

PyTorch eager → ExecuTorch → TOSA CUSTOM → VGF runtime → compare output

✓ graph rewrite ✓ payload ✓ artifact ✓ runtime numerics

End-to-end RIFE inference latency

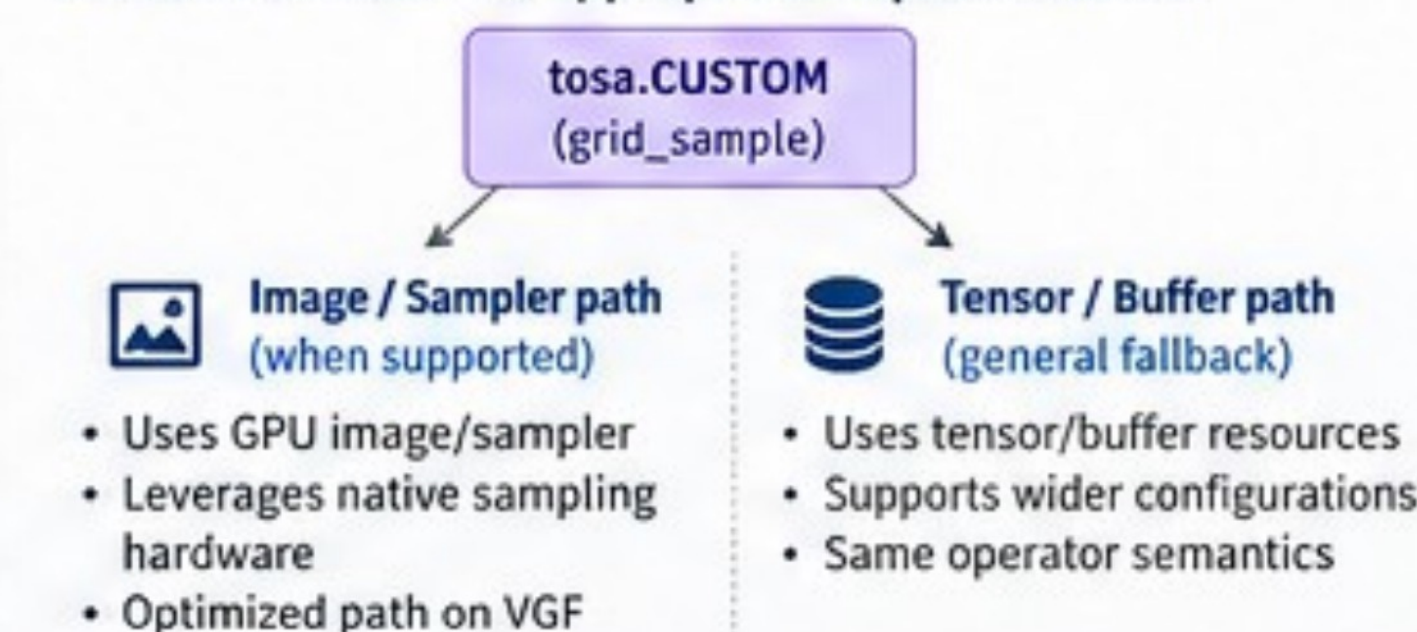
Decomposed TOSA ops: ~117 ms

tosa.CUSTOM shader: ~35 ms

Representative end-to-end result; exact uplift depends on model, image resolution and number of `grid_sample` calls.

OVER 70%
lower inference latency

Backend chooses the appropriate implementation



VGF compute shader execution (image/sampler path)

