# Efficient Data Movement for Machine Learning Inference in Heterogeneous CMS Software

Michalski L. [1] Zeh C. [2] Beltrame L. [3] Valsecchi D. [4] Pantaleo F. [5] Cano E. [5]

On behalf of the CMS Collaboration

[1]Wroclaw University of Science and Technology [2]Technical University Vienna [3]Polytechnic of Milan

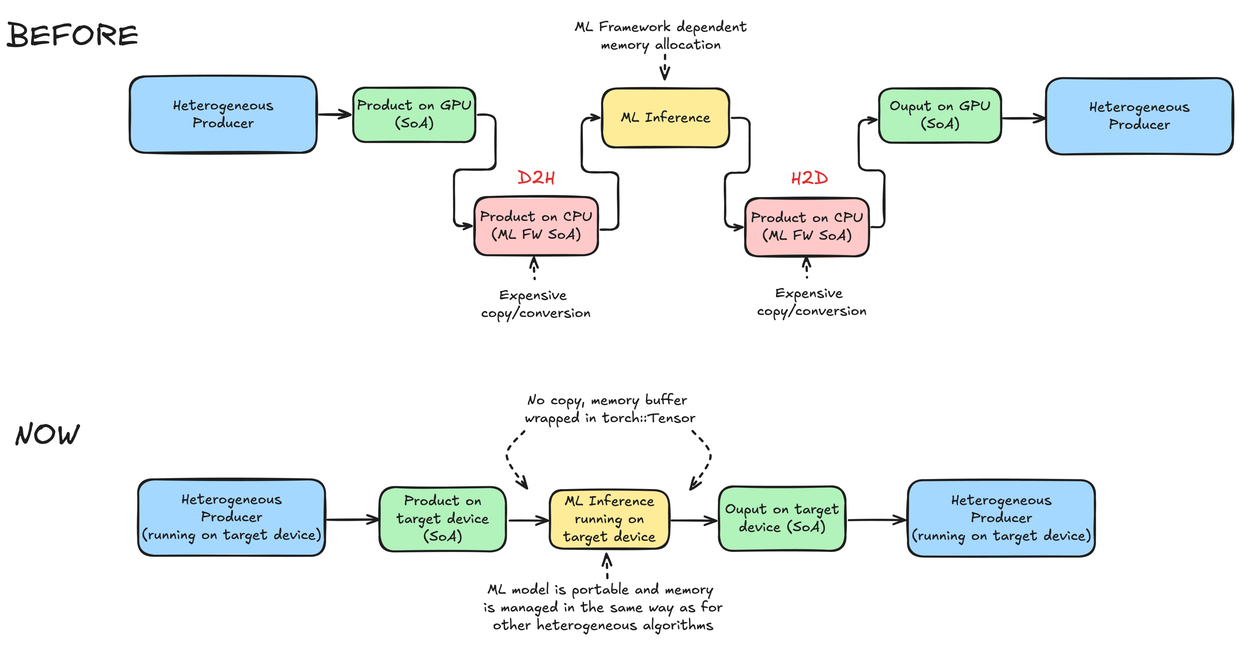
[4]ETH Zurich [5]European Organization for Nuclear Research (CERN)

## Abstract

Efficient machine learning in GPU-based workflows requires overcoming the challenge of unnecessary data movement between GPU memory and tensor formats. To address this, we introduce an interface within CMSSW that seamlessly converts Structure of Arrays (SoA) data directly into PyTorch tensors without requiring explicit data transfer. This approach computes necessary strides for various data types, allowing direct access to GPU memory. By using metadata to define tensor layouts, we eliminate redundant copying and accelerate model execution. Integrated with the Alpaka library for heterogeneous environments, this solution enhances GPU efficiency while simplifying integration with machine learning models.

## Motivation

- Adoption of efficient memory structures like SoAs and PortableCollections.
- Lack of a streamlined interface for ML model inference in Alpaka-based workflows.
- ONNX requires multiple copies and tensor conversions, leading to inefficiency.
- TensorFlow has limited GPU support for SoA usage.
- Users rely on suboptimal memory management, such as tensor conversions.
- Opportunity to directly interact with memory using `torch::from_blob()` (not readily available in TensorFlow/ONNX).




## Heterogeneous CMSSW Inference

Integrating PyTorch into CMSSW prioritizes tight control over threading, device streams, and resource management to ensure safe and scalable ML inference.

- Thread-Safe: PyTorch's internal threading is disabled.
- Scoped Scheduling: all execution is bound to CMSSW's per-event threads, ensuring no unmanaged parallelism.
- Stream-Aware Execution: ML operations run on CMSSW-managed CUDA/HIP streams using queue-based resource control
- Scalable Design: Efficient for both large models and lightweight preprocessing steps.

## Model Compilation for Low-Latency Inference

To enable deep learning in high-energy physics workflows, CMSSW integrates both *ahead-of-time* (AOT) and *just-in-time* (JIT) compilation strategies.

### *just-in-time*

- Compiles models dynamically at runtime.
- Ideal for flexible development and quick iteration.
- Supports on-the-fly execution without build-time preparation.

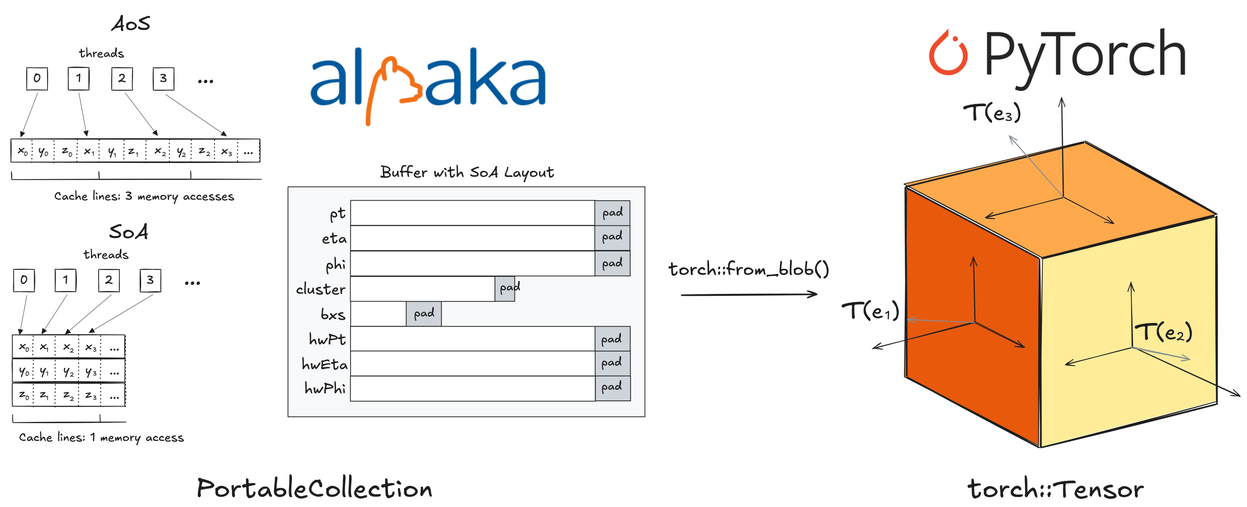
### *ahead-of-time*

- Uses `TorchInductor` and `Triton` to compile models into optimized binaries for specific hardware (e.g., NVIDIA GPUs).
- Produces shared library, loaded at runtime using standard C++ interfaces

## Copy-Efficient Memory Buffer Manipulation

Optimizing data movement is key to high-performance ML on heterogeneous hardware. CMSSW introduces a flexible interface for converting structured physics data into PyTorch tensors without copying.

- SoALayout: Structure-of-Arrays format for fast, parallel access on GPUs.
- PortableCollections: Hardware-agnostic data structures with auto-alignment.
- Alpaka: Write-once kernels compiled for multiple backends (CUDA, ROCm, CPU).
- Zero-Copy Tensors: Direct conversion to PyTorch tensors, no data movement.




## Inference Code

```
using namespace cms::torch::alpaka;

// structs
PortableCollection<SoAInputs, Device> inputs_device(batch_size, alpaka_device);
PortableCollection<SoAOutputs, Device> outputs_device(batch_size, alpaka_device);

// instantiate model
auto model = Model<CompilationType::kJustInTime>(m_path);
model.to(queue);

// metadata for automatic tensor conversion
auto input_records = inputs_device.view().records();
auto output_records = outputs_device.view().records();
SoAMetadata<SoAInputs> inputs_metadata(batch_size);
inputs_metadata.append_block("features", input_records.x(), input_records.y(),
 ↪  input_records.z());
SoAMetadata<SoAOutputs> outputs_metadata(batch_size);
outputs_metadata.append_block("preds", output_records.m(), output_records.n());
ModelMetadata<SoAInputs, SoAOutputs> metadata(inputs_metadata, outputs_metadata);

// inference
model.forward(metadata);
```

## More Information

Presentations

Pull Request

Christine Zeh
christine.zeh@cern.ch
CERN EP Department

Lukasz Michalski
lukasz.artur.michalski@cern.ch
CERN EP Department

## Partners

IML2025: 7th IML Workshop on Machine Learning at the LHC, 19-23 May 2025, CERN, Geneva, Switzerland