28th Conference on Computing in High Energy and Nuclear Physics

Bangkok, Thailand

Accelerating ML Inference on heterogeneous architectures using SOFIE and alpaka

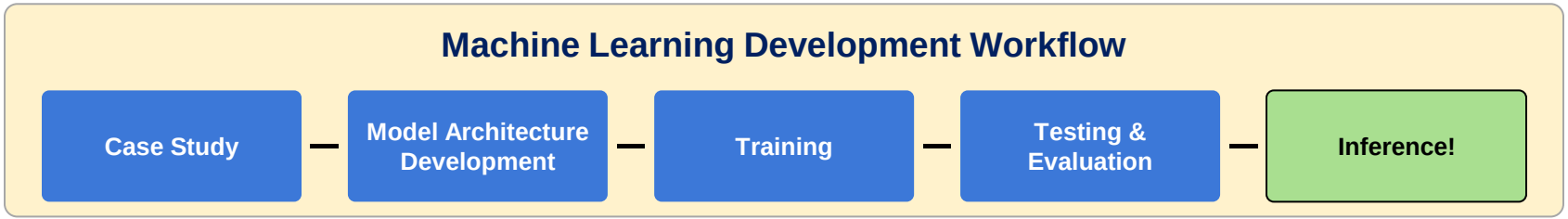
Lorenzo Moneta
CERN EP/SFT
Sanjiban Sengupta
CERN EP/SFT - University of Manchester

NexTGen Next Generation Triggers

28th May 2026

The University of Manchester

Introduction



Inference Engines:

- Loads trained ML models
- Given input data => produces results

The Bottleneck

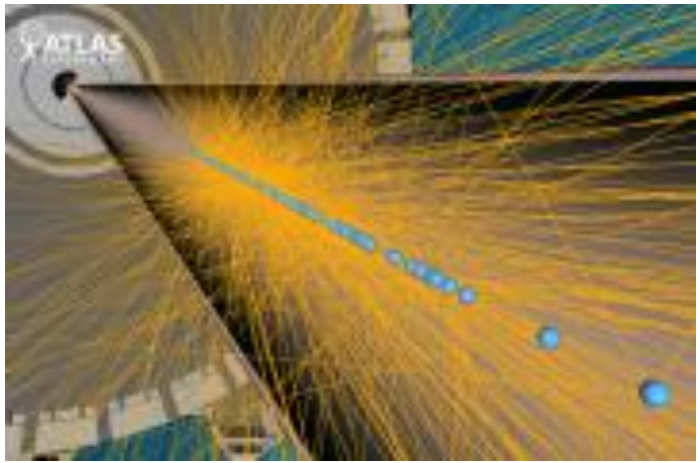
Data rates at HL-LHC are tremendously high

Requires efficient use of heterogeneous architectures!

Integration Challenges

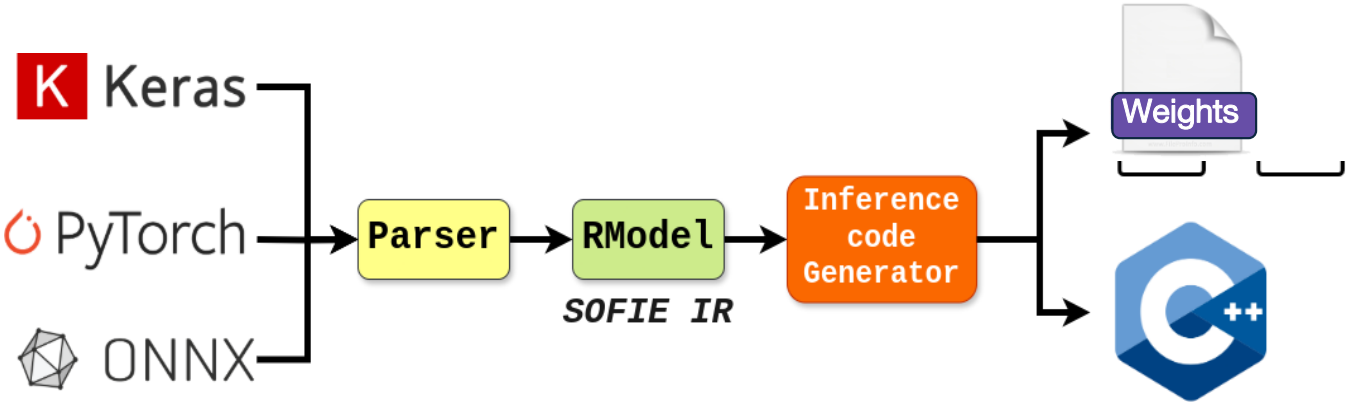
How can we integrate ML into C++ experiments' production?

Some Libraries exist for C++ inference (libTorch, ONNXRuntime) but might require complex dependencies



ML Inference through code generation

- SOFIE - System for Optimized Fast Inference code Emit
- Tool for optimized ML Inference - developed at CERN
  - Parses a trained model in ONNX, Keras or PyTorch format to its IR (based on ONNX standard)
  - Generates C++ code for inference functions (only dependent on BLAS)
  - Supports a large number of ONNX operators
    - Able to parse complex models including Transformers and GNNs based
    - Supports dynamic input shapes



SOFIE Capabilities

Complex Model Support

Extended support on CPU for state-of-the-art architectures:

- **Transformers:** ParT, MLPF, ATLAS GNN2
- **Diffusion models**: CaloDiT-2 used for fast sim
- **GNN**: Particle Net, ATLAS GNN1, GNN tracking for ATLAS and CMS (HLT)
- **Mamba** (State Space Models)

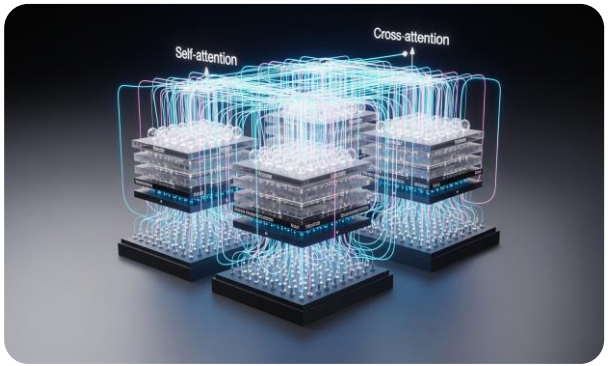
Analysis Ecosystem Integration

ROOT RDataFrame Integration:

- Easy deployment in analysis loops
- High-performance column-wise inference

RooFit & Automatic Differentiation:

- Native integration using AD for NSBI
  - See Jonas Rembser's presentation



Integration of ML Inference in Computing workflows

SOFIE emits standalone code **only dependent on BLAS** and fully under user control.

It ensures maximal efficiency and minimal friction when moving from complex research models to high-throughput experiment software stacks.

CPU Optimizations

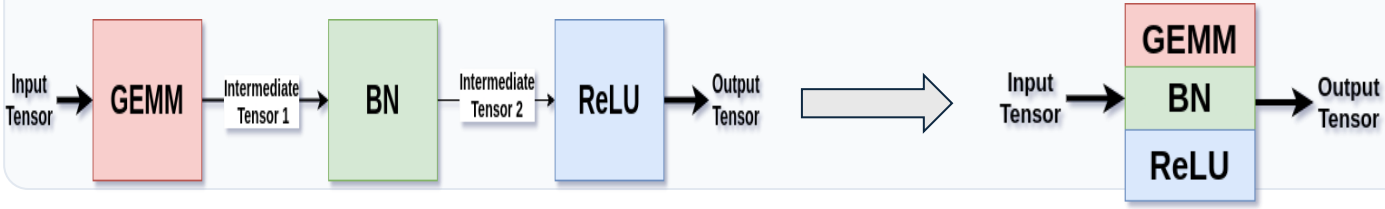
Performance & Memory Optimization for CPU Inference

Memory Optimization

- Memory reuse: Implemented via Greedy Interval Scheduling
  -> Substantial reduction in memory usage
- On-the-fly tensor broadcasting
- Constant tensor elimination and operator fusion

Performance Gains

- Multi-operator Fusion
- Operator elimination
- Operator-level optimizations
  - E.g. use of vdt for Softmax



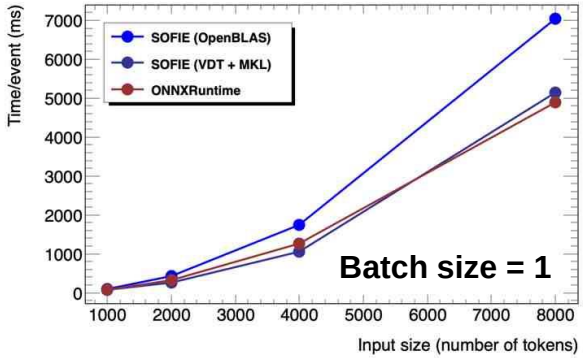
Benchmarking on CPU

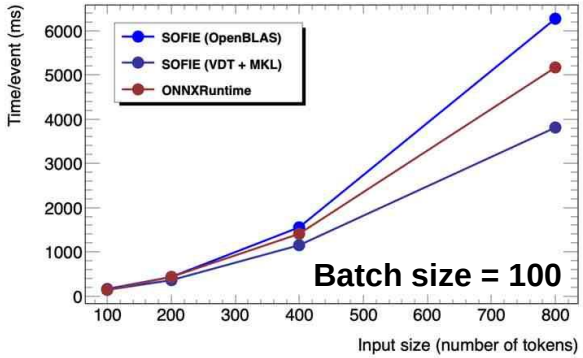
Benchmarking for a simple transformer model (8 Heads Attention Layer)

Tests run on Intel Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz

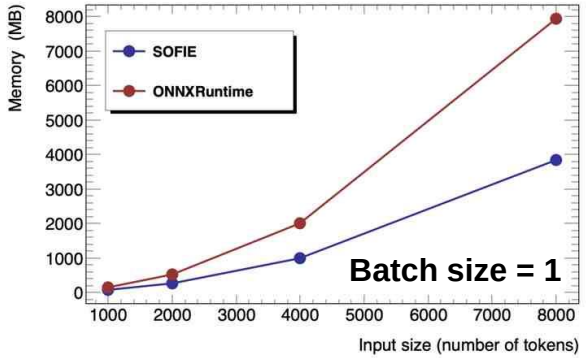
1 single thread

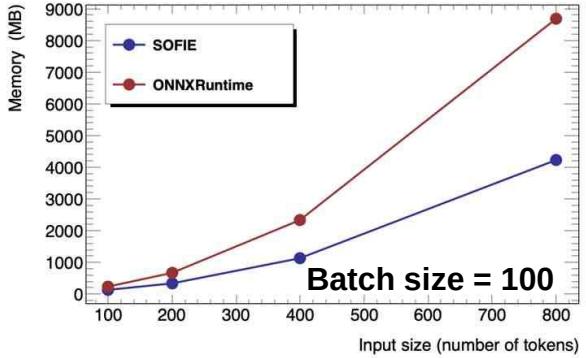
Latency





Memory consumption





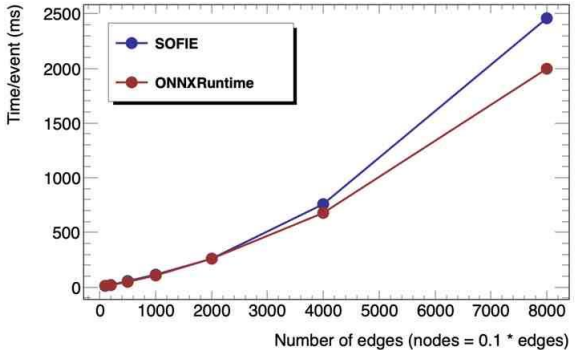
Benchmarking on CPU

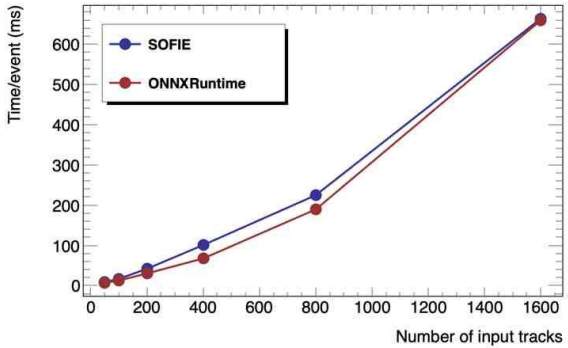
Benchmarking ATLAS and CMS Models

Tests run on Intel Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz

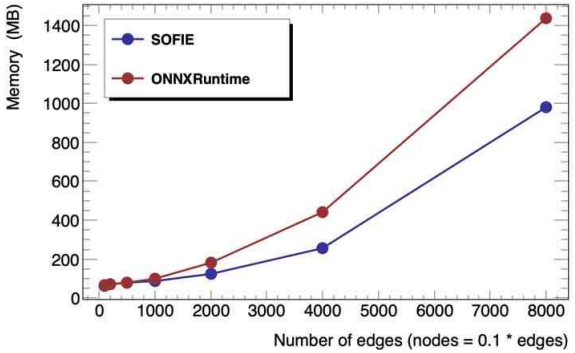
1 single thread

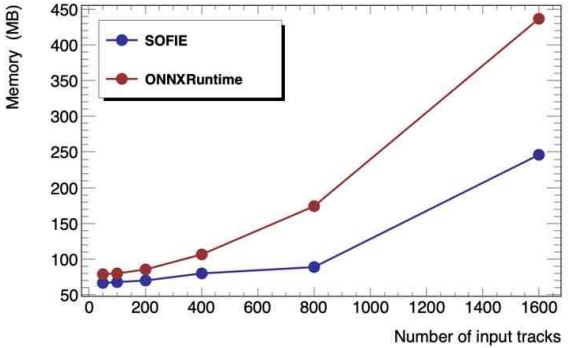
Latency





Memory consumption





CMS Jet Tagger ParticleNet

ATLAS Jet Tagger GNN2

Significant Memory Reduction in SOFIE

Benchmarking SOFIE on CPU

Benchmarking ATLAS model for track reconstruction at HL-LHC in Event Filter

(in collaboration with NGT 2.4)

- **Tested within ACTS:** Integrated directly into the full tracking chain.
- **Data Sample:** 20 events from the EF-tracking ttbar PUP200 sample
- **Single-event Inference:** Real-time performance evaluation.

Inference backend	SOFIE [s]	ONNX [s]
Graph construction	12.8	12.8
GNN 128/fp32	8.8	19.5
GNN 32/fp32	2.2	4.1
Track building	0.02	0.02

See ATLAS contribution

Prototype code: github.com/sanjibansg/sofie-atlas-tracking

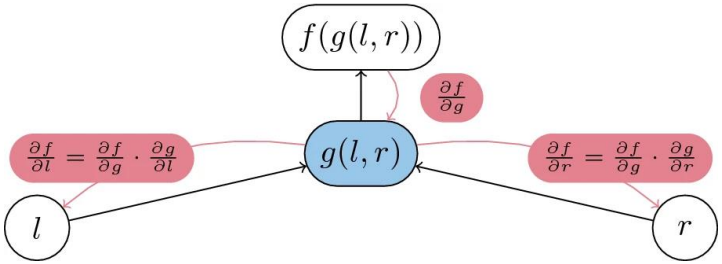
AD with SOFIE

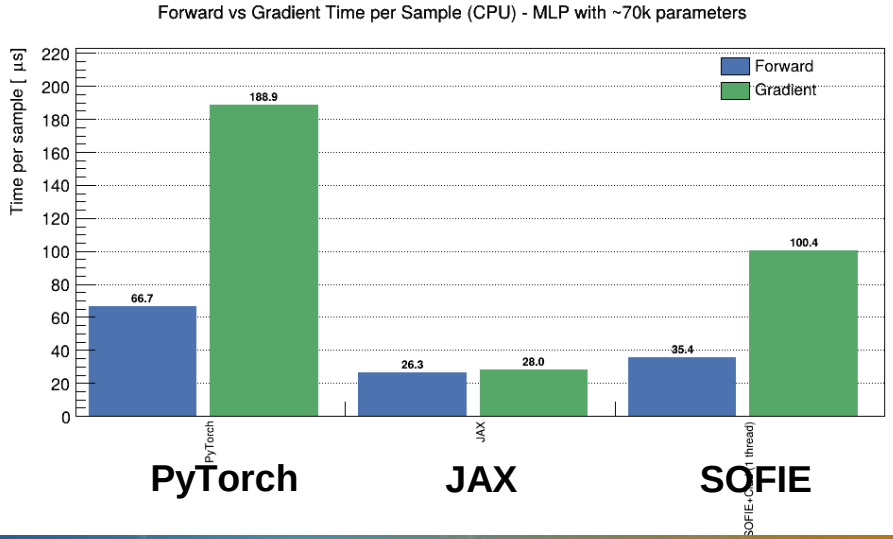
SOFIE is also capable of performing Automatic Differentiation using Clad

Code generated is understood by Clad

See presentation from Jonas Rembser

Benchmark Forward and Backward (Gradient) pass
Compare with PyTorch and JAX





Inference on Heterogeneous Architectures

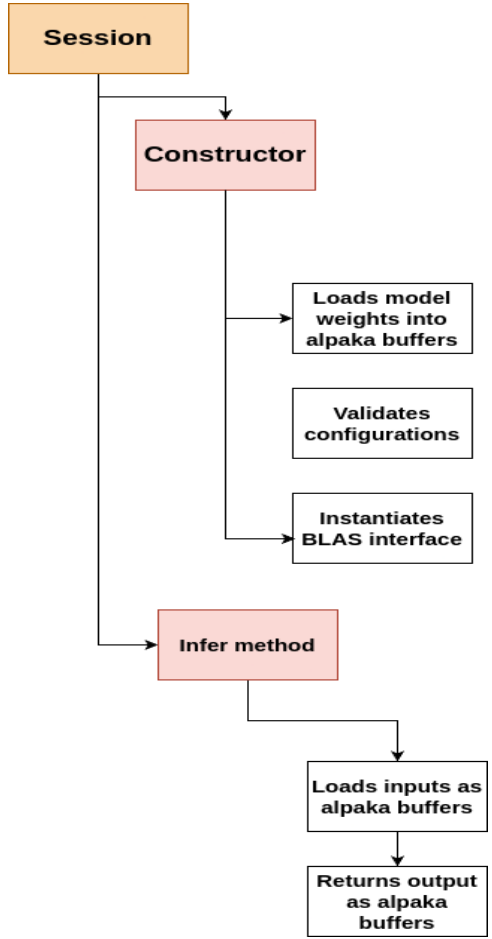
Integration with alpaka (v2.2)

- Accepts buffers as inputs
- Returns buffers as outputs
- No intermediate layout transformation required

Abstract Inference

- User has the control of running it on **Intel, NVIDIA, or AMD GPUs**

For alpaka see Andrea's presentation



sofieBLAS

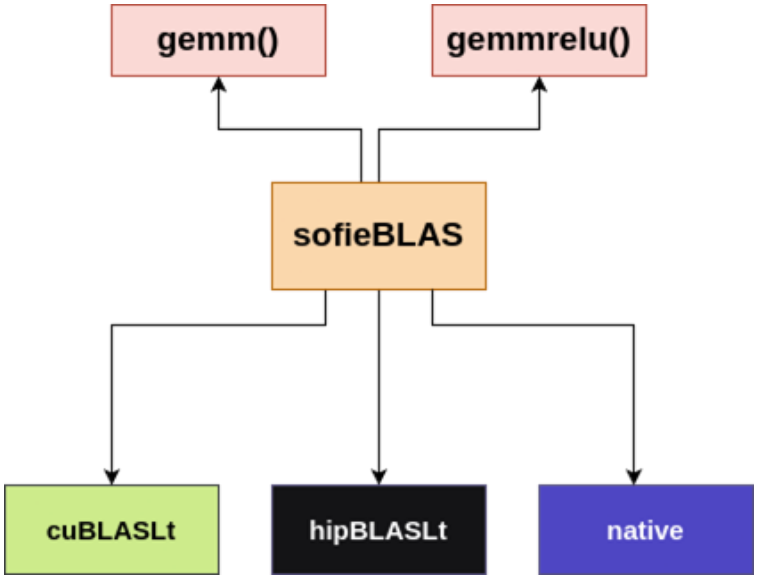
Unified Interface

Common C++ API over multiple BLAS backends.

Heterogeneous Support

- **CPU:** OpenBLAS, MKL
- **GPU:** cuBLASlt

*Thanks to **Andrea Bocci (CMS)** for the initial development support of sofieBLAS.*



sofieBLAS Backend Integration Workflow

UI for Heterogeneous Inference

Same code can be called for inference on different architectures just by changing this device tag

```
// Instantiate session to load weights
SOFIE_Linear_16::Session<alpaka::TagGpuCudaRt> session;


// Host device setup
alpaka::PlatformCpu hostPlatform{};
auto host = alpaka::getDevByIdx(hostPlatform, 0u);


// Select GPU device + queue
alpaka::PlatformCudaRt platform{};
alpaka::DevCudaRt device = alpaka::getDevByIdx(platform, 0u);
alpaka::Queue<...> queue{device};


// Allocate buffers & transfer data
auto A_d = alpaka::allocBuf<float, Idx>(device, ...);
alpaka::memcpy(queue, A_d, A);


// Inference execution
auto result = session.infer(A_d);
```

Current Support for GPU Heterogeneous Inference

Matrix Operations
GeMM - General Matrix Multiplication

Activation Functions
ReLU, Leaky ReLU, Sigmoid, Softmax

Basic Arithmetic
Add, Multiply, Subtract, Divide

Basic Unary
Negate, Reciprocal, Log, Sin, Cos, Sqrt

Data Movement
Gather, GatherND, Split, Tile, Transpose, ScatterElements

Tensor Manipulation
Reshape, Squeeze, Unsqueeze, Flatten, Expand

Metadata
Shape, Constant of Shape

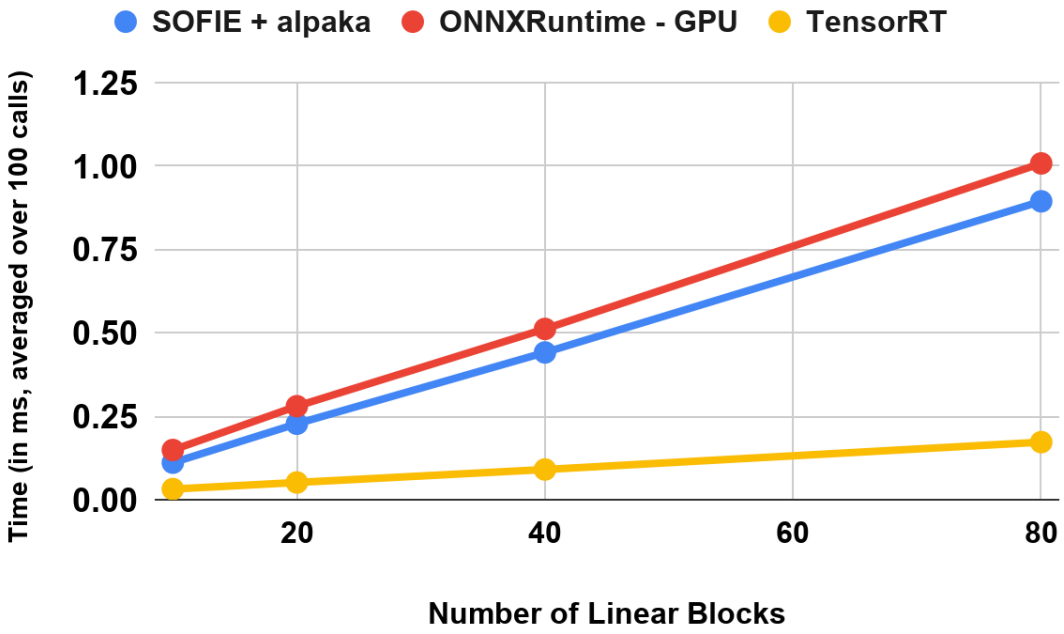
Type Management
Cast

Supporting now simple GNN models (e.g model from CMS planned for HLT tracking) and transformers (flash attention not yet implemented)

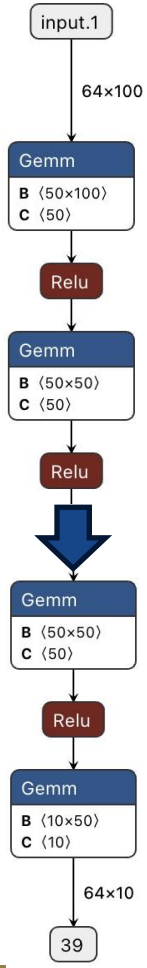
Benchmarking SOFIE on GPU (Linear Model)

Benchmarking Linear models - **Preliminary results**

Tested on NVIDIA RTX 2070



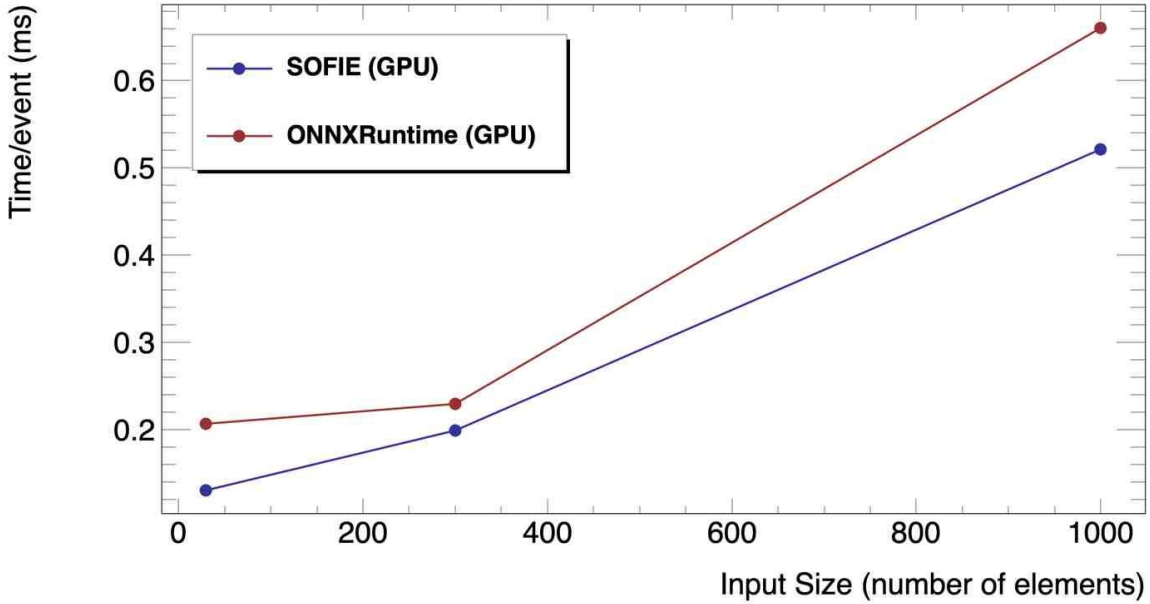
10 MLP layers



Benchmarking SOFIE on GPU (Transformer Model)

Benchmarking a Simple Transformer models (1 Layer)

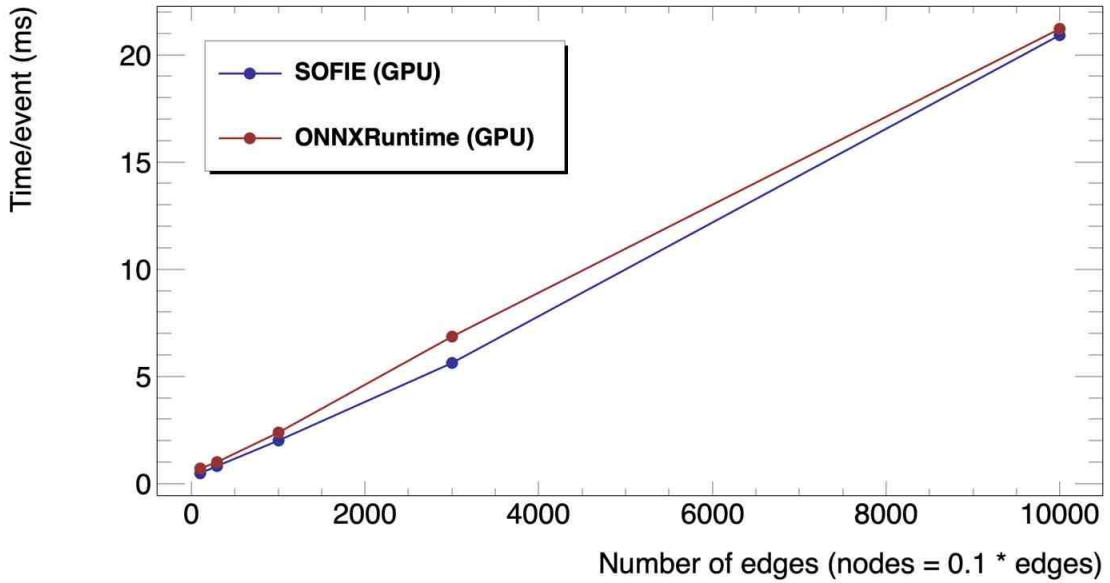
Tested on NVIDIA RTX 2070



Benchmarking SOFIE on GPU (GNN Model)

Benchmarking CMS GNN Model for HLT reconstruction - **Preliminary results**

Tested on NVIDIA RTX 2070



Summary

CPU Inference

- Validated using several complex models :
  - GNN (ParticleNet and ATLAS GNNTk)
  - Transformers based for tagging (ParT, ATLAS GNN2)
  - Particle flow (MLPF for CMS)
  - Diffusion transformer (CaloDiT-2)
- Benchmark on latency and memory usage
  - SOFIE shows similar runtime vs ONNX
  - Strong dependence on which BLAS implementation is used
  - **Significant reduction in memory usage** (~2)

GPU Heterogeneous Inference

- Support basics operators
  - GeMM matrix operations
  - Activation functions and some advanced tensor manipulations
- Also supports simple GNN and transformers
- Benchmarked on NVidia GPU
  - Competitive results with respect to ONNX runtime GPU
  - **GPU Optimization not yet done** (e.g. implement flash attention)

SOFIE - Future directions

CPU Support

Further optimization of CPU and memory Inference.

Extend Operator Support according to user requests, major operators already implemented

Interoperability

Integration with RooFit for NSBI and AD

Interoperability with ROOT RDataFrame for analysis

Interoperability of SOFIE with hls4ml for trigger emulation of ML inference

Integration with PQuantML for pruned models

GPU Support

Extend support for inference through SOFIE with alpaka on Heterogeneous architectures.

GPU code optimization (runtime and memory)

Add support for streams

Introduce support for quantisation (float8 and float16)

Collaboration

Collaboration with NGT and LHC experiments for optimizing ML Inference pipelines using SOFIE

Integrate SOFIE in a Unified ML interface
(see Sanjiban's presentation on Yukti)

Thank you !

Questions?

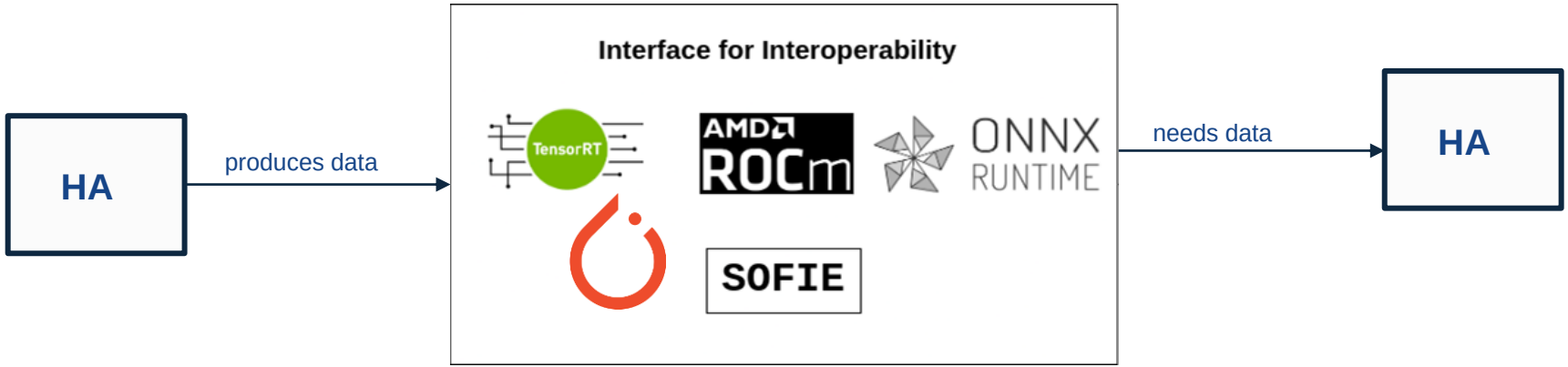
References for SOFIE:

- ROOT version (CPU): https://github.com/root-project/root/tree/master/tmva/sofie
- GPU version (standalone): https://github.com/ML4EP/SOFIE

Backup Slides

NGT 1.7 Fast ML Inference

Efficient interfaces to Machine Learning inference engines to minimize data movements and execution latencies



*** HA: Heterogeneous Architecture**

ML Inference using SOFIE

- SOFIE - System for Optimized Fast Inference code Emit
- Tool for optimized ML Inference - developed at CERN, that
  - Parses a trained model in ONNX, Keras or PyTorch format to its IR (based on ONNX standard)
  - Generates inference code in the form of C++ functions (only dependent on BLAS)
  - Supports several ONNX operators, capable of inferring Transformers, GNNs from Deepmind Graphnets.

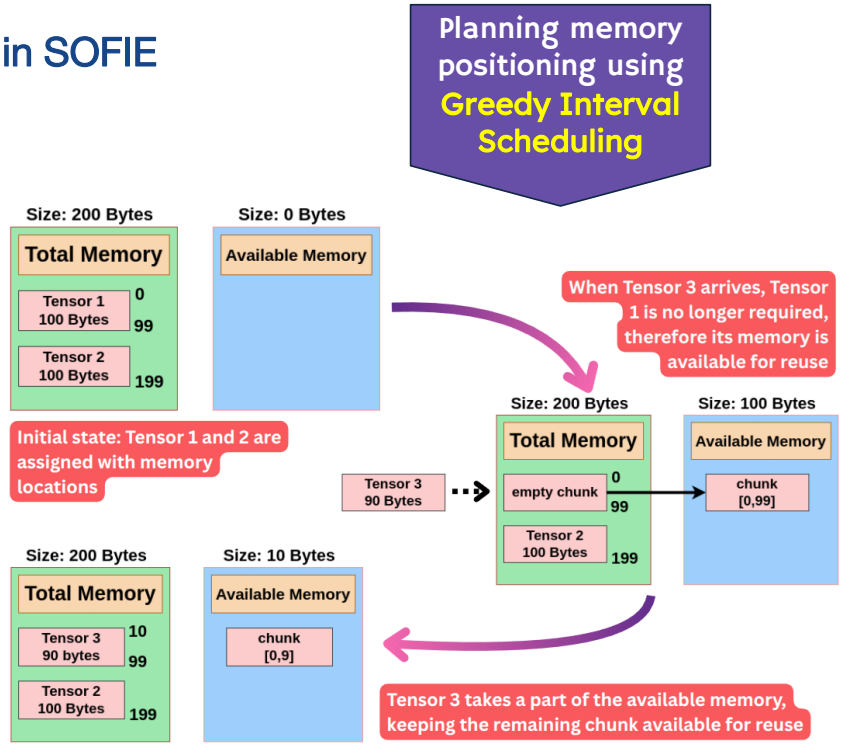
```
#include "SOFIE/RModel.hxx"
#include "SOFIE/RModelParser_ONNX.hxx"

SOFIE::RModelParser_ONNX parser;
SOFIE::RModel model = parser.Parse("Linear_16.onnx");

model.Generate();
model.OutputGenerated();
```

ML Inference using SOFIE - CPU

- Memory Reuse in SOFIE

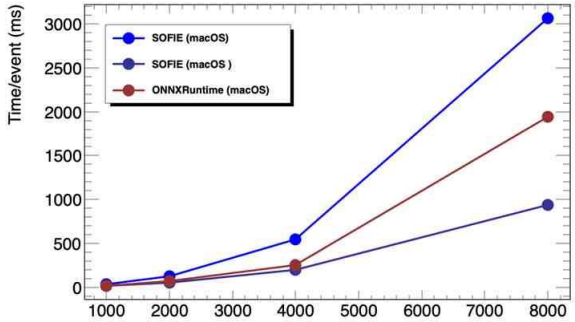


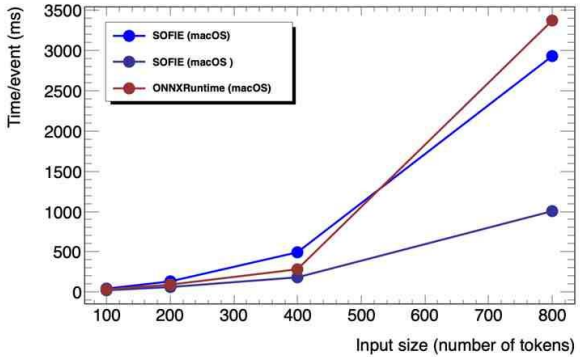
Benchmarking on CPU (MacOS)

Benchmarking for a simple transformer model
(8 Heads Attention Layer)

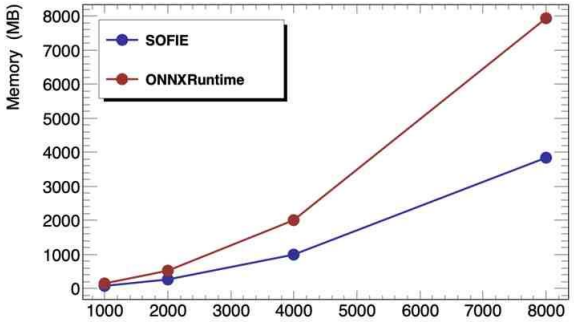
Tests run on
MacOS M1 Max
Using native BLAS
1 single thread

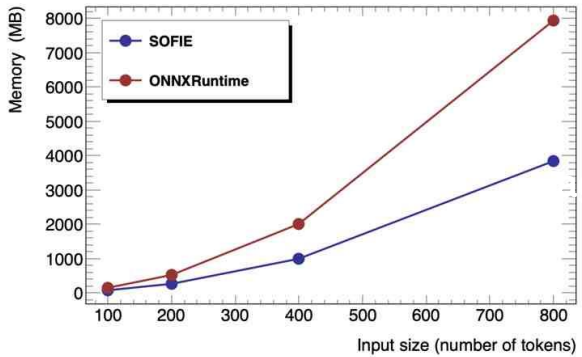
Latency





Memory consumption





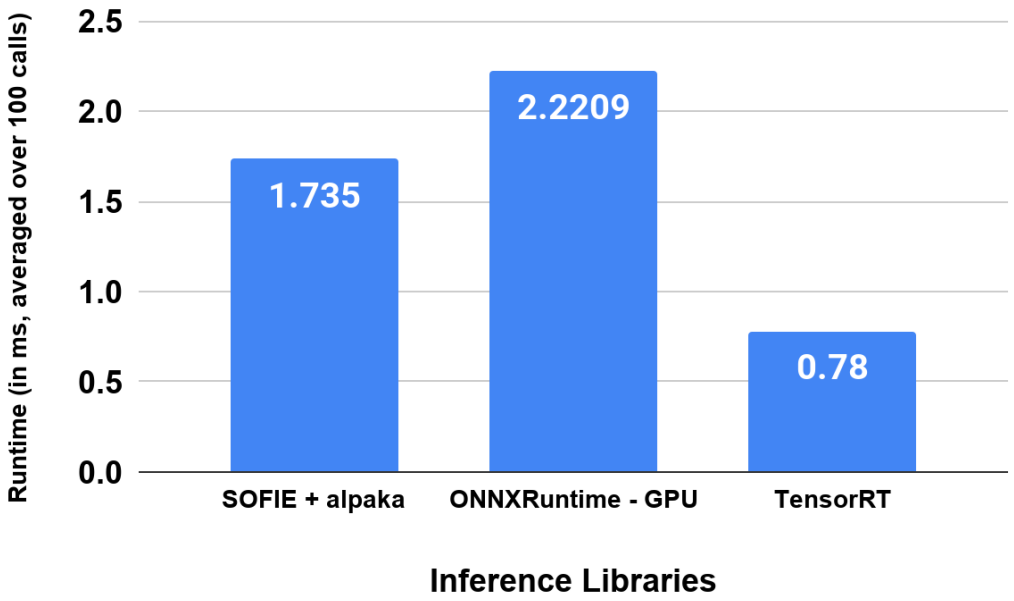
Batch Size = 1

Batch Size = 100

Benchmarking SOFIE on GPU (GNN Model)

Benchmarking CMS GNN Model for HLT reconstruction - **Preliminary results**

Tested on NVIDIA RTX 2070



Input features:

X:
`float32[3370,29]`

edge_features:
`float32[24126, 5]`

edge_index:
`int64[24126, 2]`