ROOT Machine Learning Ecosystem

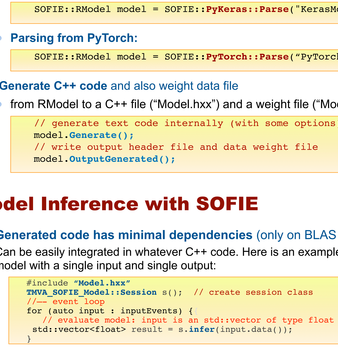
Lorenzo Moneta, Sitong An, Sanjiban Sengupta, Ahmet Hamdan, Neel Shah, Harshal Shende, Sanchi Mittal, Omar Zapata

Machine Learning in ROOT

- Shifting focus in ROOT from training to model evaluation and to interoperability
- Several Python tools exist for training ML models (e.g. Tensorflow, PyTorch, scikit-learn)
  - Missing fast, efficient and easy-to-use tools to deploy these models in production
- Developed **new interfaces for model inference** (evaluation) that can be used easily in external software or integrated into analysis tools such as ROOT RDataFrame
  - **SOFIE** for deep learning models
  - **RBDT** for fast Boosted Decision Tree
- Generate C++ code from trained models for fast evaluation

SOFIE: New Inference for DL models

- SOFIE: **inference engine that generates C++ code** from an input trained model.
- The input parsed model is in **ONNX format** (a standard for ML and supported by several tools)
  - but supported also native Keras or PyTorch formats.
  -



- The generated C++ code **can be easily integrated** into any code where model evaluation is used:
  - analysis code based on RDataFrame (a special interface is provided to easy this integration);
  - reconstruction or simulation code where ML models could be used.
- The code has **minimal dependencies**, only on BLAS for matrix computation.
- The code can be compiled on the fly using the CLING JIT system: **can be used at run-time.**
- Several operators are supported, but the code is modular and is **easy to add custom operators** (user-defined).

Code Parsing and code generation

- Parsing: **from ONNX** to an intermediate representation: **SOFIE::Model**

```
using namespace TMVA::Experimental;
SOFIE::RModelParser_ONNX parser;
SOFIE::RModel model = parser.Parse("Model.onnx");
```

- **Parsing from Keras:**

```
SOFIE::RModel model = SOFIE::PyKeras::Parse("KerasModel.h5");
```

- **Parsing from PyTorch:**

```
SOFIE::RModel model = SOFIE::PyTorch::Parse("PyTorchModel.pt");
```

- **Generate C++ code** and also weight data file
  - from RModel to a C++ file ("Model.hxx") and a weight file ("Model.dat")

```
// generate text code internally (with some options)
model.Generate();
// write output header file and data weight file
model.OutputGenerated();
```

Model Inference with SOFIE

- **Generated code has minimal dependencies** (only on BLAS libraries)
- Can be easily integrated in whatever C++ code. Here is an example of using it with a model with a single input and single output:

```
#include "Model.hxx"
TMVA_SOFIE_Model::Session s();   // create session class
//--- event loop
for (auto input : inputEvents) {
   // evaluate model: input is an std::vector of type float *
   std::vector<float> result = s.infer(input.data());
}
```

- Model parsing and generation can also be done at run time using CLING and instead of including the model header file one can just do in C++ or Python

```
// compile generate SOFIE code using ROOT interpreter
gInterpreter->Declare('#include "Model.hxx"')
```

Model Inference: RDF Integration

- Model evaluation can be easily performed with RDataFrame using the adapter class **SofieFunctor**

```
auto h1 = df.DefineSlot("DNN_Value",
SofieFunctor<7,TMVA_SOFIE_higgs_model_dense::Session>(nslots),
{"m_jj", "m_jjj", "m_lv","m_jlv","m_bb","m_wbb","m_wwbb"})
.Histo1D("DNN_Value");
```

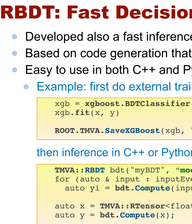
- See the full example tutorial code in C++ or in Python

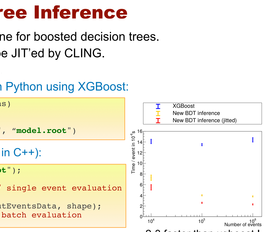
SOFIE Operator Support

Perceptron: Gemm	Implemented and integrated (ROOT 6.26)
Activations: Relu, Selu, Sigmoid, Softmax, LeakyRelu	Implemented and integrated (ROOT 6.26)
Convolution (1D, 2D and 3D)	Implemented and integrated (ROOT 6.26)
Recurrent: RNN, GRU, LSTM	Implemented and integrated (ROOT 6.26)
BatchNormalization	Implemented and integrated (ROOT 6.26)
Pooling: MaxPool, AveragePool, GlobalAverage	Implemented and integrated (ROOT 6.26)
Deconvolution (1D,2D,3D)	Implemented and integrated in master (for ROOT 6.28)
Layer operations: Neg, Exp, Sqrt, Reciprocal (Unary op.), Add, Sum, Mul, Div (Binary op.), Reshape, Flatten, Transpose, Squeeze, Unsqueeze, Slice, Concat, Identity, Reduce,	Implemented and integrated in master (for ROOT 6.28)
Custom operator	Implemented and integrated in master (for ROOT 6.28)
InstanceNorm, LayerNormalization	Implemented but to be integrated (PR #8885, #11595)
Gather (for embedding)	Planned for next release
GNN (Message Passing GNN based on Deep Mind graph_nets	Implemented but to be integrated (PR #11208)
Next to support: e.g. DGCNN, Normalising flows,….?	Depending on user needs

SOFIE Benchmarks

- Comparison of **SOFIE** inference with **ONNXRuntime** (from Microsoft) and **LWTNN** (ATLAS)
  - 2-3 faster than ONNXRuntime for DNN with batch size=1
    - e.g. using RDF interface for a DNN with 5 layers of 200x200 nodes:
      - SOFIE: **310K evts/s**, ONNXRuntime: **120K evt/s**, LWTNN: **120K evts/s**
  - 20% faster for RNN operators
  - slightly slower for CNN (20% for 2D) but further optimisations still possible





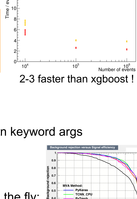
RBDT: Fast Decision Tree Inference

- Developed also a fast inference engine for boosted decision trees.
- Based on code generation that can be JIT'ed by CLING.
- Easy to use in both C++ and Python.
  - Example: first do external training in Python using XGBoost:

```
xgb = xgboost.BDTClassifier(options)
xgb.fit(x, y)

ROOT.TMVA.SaveXGBoost(xgb, "myBDT", "model.root")
```

then inference in C++ or Python (e.g. in C++):

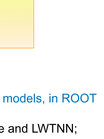
```
TMVA::RBDT bdt("myBDT", "model.root");
for (auto & input : inputEvents)
   auto yi = bdt.Compute(input); // single event evaluation

auto x = TMVA::RTensor<float>(inputEventsData, shape);
auto y = bdt.Compute(x);       // batch evaluation
```



2-3 faster than xgboost !

Interoperability with Python

- **Pythonization of TMVA interfaces:** from string API to python keyword args
- Training of Python external ML tools within TMVA workflow
  - **interfaces to scikit-learn, Keras and PyTorch**
- **Generic data loader for ML workflows:**
  - generator doing batching and shuffling from ROOT files on the fly;
  - allows for efficient training of datasets larger than RAM sizes;
  - direct feeding of data from disk to GPU. Here is a possible example:




```
df = ROOT.RDataFrame("Events", "http://file.root")
generator = TMVA.BatchGenerator(df, columns, batchSize)
for step in gradientSteps:
    x = generator()
    model.fit(x)
```

Summary

- Released of SOFIE, a fast and easy-to-use inference engine for ML models, in ROOT 6.26 with a lot of enhancements coming in 6.28
  - Good performance compared to existing packages ONNXRuntime and LWTNN; further optimisations are still possible
  - Integrated with other ROOT tools to evaluate models in user analysis: RDataFrame
  - On-going developments according to user needs (GNN, CUDA support, etc..)
- Working overall on better interoperability of TMVA with other ML ecosystems
  - Batch generator for training efficiently from ROOT I/O and RDF to external Python tools

References

- SOFIE GitHub in current ROOT master.
- SOFIE notebooks and tutorials.
- Summer student report on new GNN support.
- ICHEP 2022 presentation about SOFIE.
- Benchmark tests of SOFIE (PR on rootbench github).

Acknowledgements

S. An. gratefully acknowledges the support of the Marie Skłodowska-Curie Innovative Training Network Fellowship of the European Commission Horizon 2020 Programme, under contract number 765710 INSIGHTS.
S. Sengupta and A. Hamdan are 2022 CERN summer students and 2021 Google Summer of Code. Neel Shah, Sanchi Mittal and Harshal Shende are 2022 Google Summer of Code students.