TMVA SOFIE
Enhancing the Machine Learning Inference Engine

Sanjiban Sengupta
EP-SFT

Supervisor: Lorenzo Moneta

/sanjibansg /sanjiban-sengupta
● Toolkit for Multivariate Analysis
● Provides a Machine Learning environment for training, testing and evaluation of multivariate methods.
SOFIE

System for
TMVA SOFIE

SOFIE
System for Optimized
SOFIE
System for Optimized Fast
SOFIE
System for Optimized Fast Inference
SOFIE
System for Optimized Fast Inference code Emit
SOFIE

System for Optimized Fast Inference code Emit

inference code, fast to operate, with least dependencies
Motivation
Motivation

- ML ecosystem mostly focuses on model training.
Motivation

- ML ecosystem mostly focuses on model training.
- Inference in Tensorflow & PyTorch
  - supports only their own model
  - usage of C++ environment is difficult
  - heavy dependency
Motivation

- ML ecosystem mostly focuses on model training.
- Inference in Tensorflow & PyTorch
  - supports only their own model
  - usage of C++ environment is difficult
  - heavy dependency
- Inference in ONNX (Open Neural Network Exchange)
  - can use ONNXRuntime by Microsoft
  - large dependency
  - difficult to integrate in HEP applications
    - control of libraries, threads
    - not optimized for single event evaluation
- Intermediate representation following ONNX standards.
• Intermediate representation following ONNX standards.
• Inference code generation with least latency and minimal dependency
- Intermediate representation following ONNX standards.
- Inference code generation with least latency and minimal dependency
- Intermediate representation following ONNX standards.
- Inference code generation with least latency and minimal dependency
- Intermediate representation following ONNX standards.
- Inference code generation with least latency and minimal dependency
- Intermediate representation following ONNX standards.
- Inference code generation with least latency and minimal dependency
RModel

ROperator

ROperator_Gemm  ROperator_Conv  ROperator_Relu

ROperator_RNN  ROperator_concat  ROperator_Selu
Parser for translating an ONNX model to SOFIE's IR

```cpp
using namespace TMVA::Experimental::SOFIE;
RModelParser_ONNX parser;
RModel model = parser.Parse("model.onnx");
```
Parser for translating an ONNX model to SOFIE's IR

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

Parser for translating Keras (.h5) models

```cpp
SOFIE::RModel model = SOFIE::PyKeras::Parse("KerasModel.h5");
```
Parser for translating an ONNX model to SOFIE's IR

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

Parser for translating Keras (.h5) models

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

Inference code generation

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

```cpp
// Code auto generated by TMVA SOFIE

namespace TMVA_SOFIE_Linear_event{

struct Session {

Session(std::string filename ="") {
  if (filename.empty()) filename = "Linear_event.dat";
  std::ifstream f;
  f.open(filename);
  // read weight data file
  .................
}
std::vector<float> infer(float* tensor_input1){
  .................
```
- Extending support of SOFIE Keras parser
Project Objectives

- Extending support of SOFIE Keras parser
- Implement SOFIE Custom operator support
Project Objectives

- Extending support of SOFIE Keras parser
- Implement SOFIE Custom operator support
- Implement support for parsing Graph Neural Networks in SOFIE
- No native support for ONNX translation
- No native support for ONNX translation
- TF2ONNX may convert a Keras .h5 model to ONNX
● No native support for ONNX translation
● TF2ONNX may convert a Keras .h5 model to ONNX
● SOFIE Keras Parser!
  ○ simpler to use
  ○ no need for input spec
  ○ built on latest opset
Algorithm for Parser

1. Load Model
2. Extract Information about Operators
3. Extract weight (initialized tensors)
4. Extract Input tensor info (Name, shape, datatype)
5. Extract Output Tensor Info
6. Return RModel object
## Current Support

<table>
<thead>
<tr>
<th>Keras Layer</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense</td>
<td>Implemented &amp; Integrated</td>
</tr>
<tr>
<td>Permute</td>
<td>Implemented &amp; Integrated</td>
</tr>
<tr>
<td>ReLU, Selu, Sigmoid</td>
<td>Implemented &amp; Integrated</td>
</tr>
<tr>
<td>Batch Normalization</td>
<td>PR Merged</td>
</tr>
<tr>
<td>Convolution (2D)</td>
<td>PR Merged</td>
</tr>
<tr>
<td>Reshape</td>
<td>PR Merged</td>
</tr>
<tr>
<td>Basic Binary Operators: Add, Subtract, Multiply</td>
<td>PR Under Review</td>
</tr>
<tr>
<td>Activations: Softmax, LeakyRelu, Tanh</td>
<td>PR Drafted</td>
</tr>
<tr>
<td>Concat</td>
<td>PR Drafted</td>
</tr>
</tbody>
</table>
• ONNX standards specifies 183 operators currently.
● ONNX standards specifies 183 operators currently.
● Need a custom user operator specification
● ONNX standards specifies 183 operators currently.
● Need a custom user operator specification
  ○ simple to define
ONNX standards specifies 183 operators currently.

Need a custom user operator specification
  - simple to define
  - easy to test, debug, and evaluate
- ONNX standards specifies 183 operators currently.
- Need a custom user operator specification
  - simple to define
  - easy to test, debug, and evaluate
  - few overheads and dependencies
Definition
Definition

- Define custom operator with the required attributes
  - Operator name
  - Input tensor names
  - Output tensor names
  - Output tensor shapes
  - Header file name
Definition

- Define custom operator with the required attributes
  - Operator name
  - Input tensor names
  - Output tensor names
  - Output tensor shapes
  - Header file name

- Define Compute function in Header file

- generate.hxx
- weights.dat
- compute.hxx
Definition

- Define custom operator with the required attributes
  - Operator name
  - Input tensor names
  - Output tensor names
  - Output tensor shapes
  - Header file name
- Define Compute function in Header file
High demands of Graph Neural Networks in High energy physics research
● High demands of Graph Neural Networks in High energy physics research
● CMS
High demands of Graph Neural Networks in High energy physics research

CMS
- uses Particlenet; graph neural network supporting graph convolution, i.e. edge convolution and dynamic graph CNN methods
- High demands of Graph Neural Networks in High energy physics research
- CMS
  - uses Particlenet; graph neural network supporting graph convolution, i.e. edge convolution and dynamic graph CNN methods
- LHCb
High demands of Graph Neural Networks in High energy physics research

- CMS
  - uses Particlenet; graph neural network supporting graph convolution, i.e. edge convolution and dynamic graph CNN methods

- LHCb
  - plans to use DeepMind’s Graph Nets library; builds GNN on top of Tensorflow & Sonnet
High demands of Graph Neural Networks in High energy physics research

- CMS
  - uses Particlenet; graph neural network supporting graph convolution, i.e. edge convolution and dynamic graph CNN methods

- LHCb
  - plans to use DeepMind’s Graph Nets library; builds GNN on top of Tensorflow & Sonnet
Current Plans & Implementation
Current Plans & Implementation

- Following the DeepMind’s Graph Nets architecture
Current Plans & Implementation

● Following the DeepMind’s Graph Nets architecture
● Intermediate Representation
  ○ Nodes list
  ○ Edges list
  ○ Sender’s list
  ○ Receiver’s list
  ○ Global values
Current Plans & Implementation

- Following the DeepMind’s Graph Nets architecture
- Intermediate Representation
  - Nodes list
  - Edges list
  - Sender’s list
  - Receiver’s list
  - Global values
Current Plans & Implementation

- Following the DeepMind’s Graph Nets architecture
- Intermediate Representation
  - Nodes list
  - Edges list
  - Sender’s list
  - Receiver’s list
  - Global values
Current plans for implementation

- Following the DeepMind’s Graph Nets architecture
- Intermediate Representation
  - Nodes list
  - Edges list
  - Sender’s list
  - Receiver’s list
  - Global values
- Operating functions
  - Updation functions
  - Aggregation functions
● Finishing implementation & integration of Graph Neural Network support in SOFIE
Future Plan

- Finishing implementation & integration of Graph Neural Network support in SOFIE
- Implementing support for new operators in TMVA SOFIE
● Link to Forked Repository  
  github.com/sanjibansg/root

● Link to SOFIE in current ROOT master  
  github.com/root-project/root/tree/master/tmva/sofie

● Link to TMVA/SOFIE tutorials  
  root.cern.ch/doc/master/group__tutorial__tmva.html

● Link to SOFIE notebooks  
  github.com/lmoneta/tmva-tutorial/tree/master/sofie
Using SOFIE’s Generated code

```cpp
#include "Model.hxx"
// create session class
TMVA_SOFIE_Model::Session s();
//-- event loop

{ // evaluate model: input is an array of type float *
  std::vector<float> result = s.infer(input);
}
```
Using SOFIE’s Generated code

```python
import ROOT
# compile generate SOFIE code using ROOT interpreter
ROOT.gInterpreter.Declare('#include "Model.hxx"')
# create session class
s = ROOT.TMVA_SOFIE_Model.Session()
#-- event loop
....
# evaluate the model, input can be a numpy array of type float32
result = s.infer(input)
```
SOFIE Custom Operator

Definition

● Define custom operator with the required attributes
  ○ Operator name
  ○ Input tensor names
  ○ Output tensor names
  ○ Output tensor shapes
  ○ Header file name
● Define Compute function in Header file

Interface

```cpp
std::unique_ptr<SOFIE::ROperator> op;
op.reset(new SOFIE::ROperator_Custom<float>("Exp", {"denseBiasAdd0"}, {"exp_out"}, {{1,4}}, "exp_compute.hxx");
```
Benchmarking for Keras
Benchmarking for SOFIE
Time per event for different batch size, cache flushed

- Lightweight Trained Neural Network (lwttn)
- ONNX Runtime
- SOFIE (netlib blas)
- SOFIE (OpenBLAS)
Ubuntu 20.04 Intel 5000MHz

Processed Events/sec

SOFIE
ONNXRuntime
LWTNN

DNN Model (5 layers of 200)

Larger = Better
TMVA SOFIE

Ubuntu 20.04 Intel 5000MHz (Batch Size = 1)

SOFIE
ONNXRuntime

Time relative to ONNXRuntime

Smaller = Better

Deep Learning Models

DNN FastSim CNN 2D CNN 3D Resnet RNN LSTM RNN GRU CMS DDB