Zero-overhead ML training from Python with ROOT

in an ATLAS Open Data analysis

Martin Føll[1,2], Danilo Piparo[1], Vincenzo Eduardo Padulano[1]
Farid Ould-Saada[2], Eirik Gramstad[2], James Catmore[2]

[1]CERN EP-SFT, [2]University of Oslo

PyHEP 2025, October 28

EP-SFT
Software Frameworks and Tools

UNIVERSITY OF OSLO

The ROOT ML DataLoader in action

Python only
Available in ROOT 6.38
See Silia's talk about
ROOT's Newest Pythonizations

Inputs:

- Load ROOT file(s) into RDataFrame:
  - RNTuple, TTree, transformations of data
- Training features and target columns, ...

Outputs:

- Iterator over training and validation batches
- PyTorch/TensorFlow tensors

Code for generating bathes in PyTorch*:
(works on both CPU and GPU)

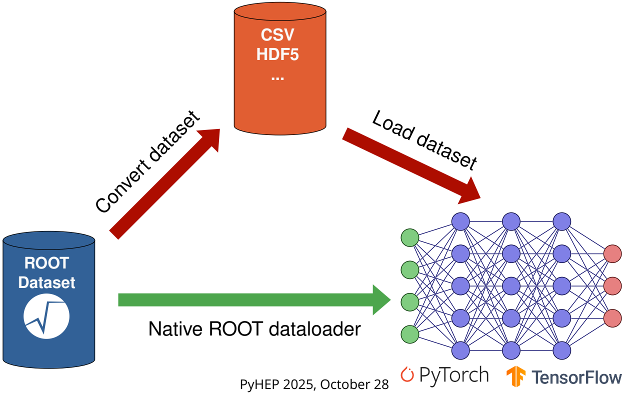
```
import ROOT

# TTree or RNTuple
df = ROOT.RDataFrame("tree", "dataset.root")

# generate training and validation batches
train, validation =
ROOT.CreatePyTorchGenerators(df, batch_size,
                             chunk_size, block_size,
                             columns, target,
                             validation_split)

# define PyTorch model...

# loop over training batches
for x_train, y_train in train:
  outputs = model(x_train)
```

*API will evolve, your feedback is welcome!

Motivation

Zero-overhead ML training:

- Create batches directly from ROOT files with the flexibility of RDataFrame:
  - RNTuple/TTree, remote reads, other input types
- No file conversion, duplication and double bookkeeping
- Reduce the number of custom dataloaders
- Representation compatible with PyTorch, TensorFlow, ...



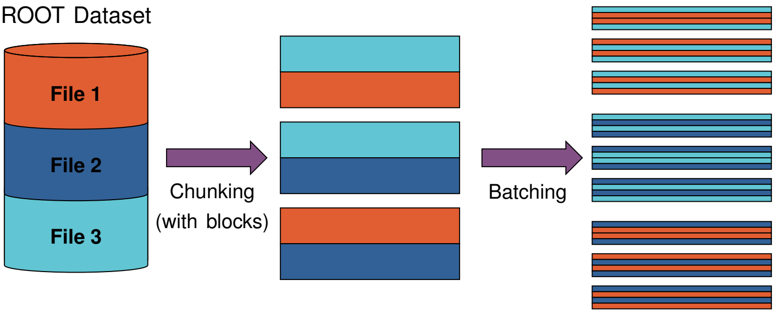
What if your dataset does not fit in memory?

Enable training with datasets that do not fit in memory:

- Divide data into chunks, one chunk loaded into memory at a time
- Further split into batches for the training

Shuffling strategy:

- Goal: Avoid training bias through shuffling
- Load blocks of consecutive entries into chunks for optimized I/O



ATLAS Open Data Analysis

Training a Parametrized Neural Network (PNN) [1] for binary classification of:

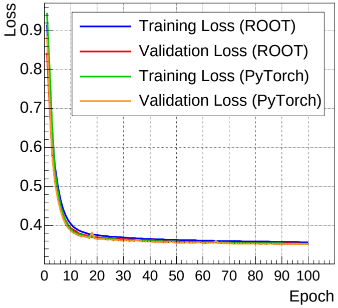
- Standard Model and Supersymmetry *pp* collisions into a $\mu^+\mu^- + E_T^{miss}$ final state

Same model output from the training for both:

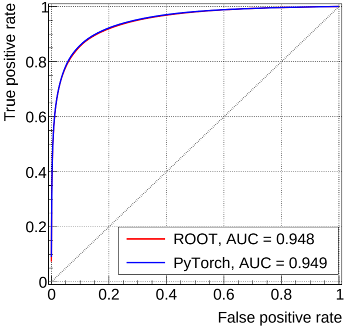
- Loading batches from CSV file with PyTorch DataLoader
- Loading batches from ROOT file with ROOT DataLoader

From ACAT 2025

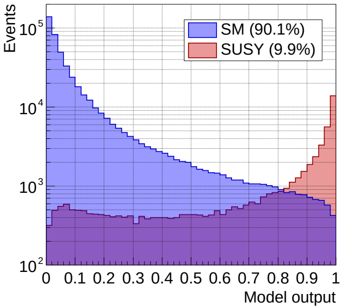
Loss



ROC

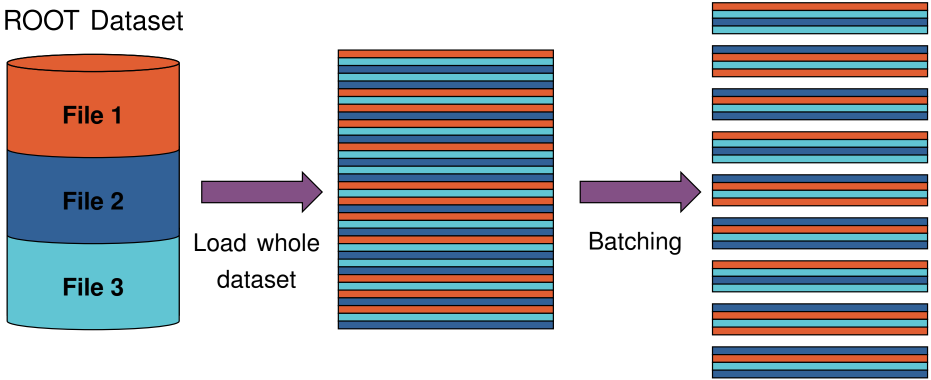


Outputs from PNN



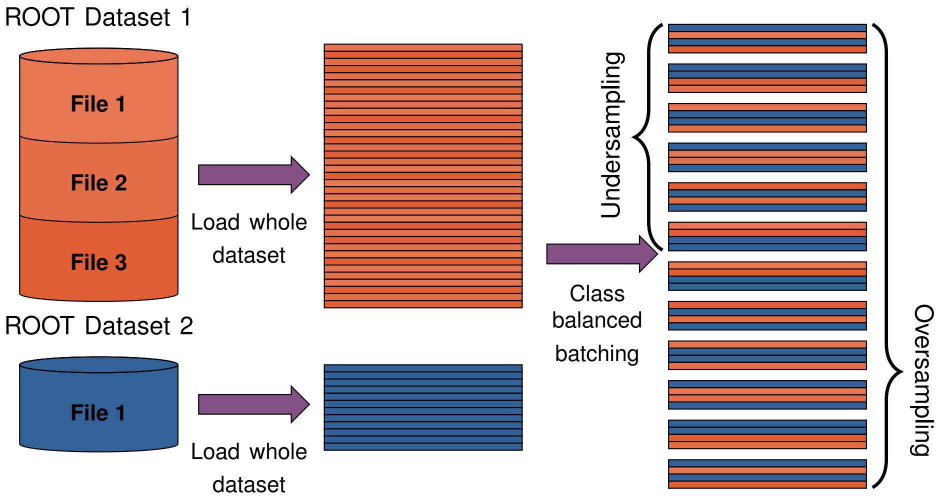
What is next?

Implement simplest case: Load the whole dataset in memory



Batch balancing from datasets with class imbalance

Example: Want batches with 50/50 from **Class 1** and **Class 2**



Summary:

- Can do zero-overhead ML training with ROOT taking advantage of RDataFrame:
  - RNTuple, TTree, transformations of data
- Enable training with datasets that do not fit in memory
- Demonstrated in an ATLAS Open Data analysis that the shuffling strategy provides good accuracy for the ML training

Outlook:

- Implement batching when the whole dataset is loaded in memory and for batch balancing from datasets with class imbalance
- Further I/O optimization are needed for performant large scale dataloading: align chunks to cluster boundaries
- Further evaluation of the dataloading in more diverse usecases

[1] P. Baldi et al. In: *Eur. Phys. J. C* **76**.5 (2016), p. 235. DOI: 10.1140/epjc/s10052-016-4099-4.

[2] ATLAS collaboration (2024). ATLAS Open Data. In: *CERN Open Data Portal* (2024). DOI: 10.7483/OPENDATA.ATLAS.9HK7.P5SI.

[3] J. Schaarschmidt et al. In: *EPJ Web of Conf.* **295** (2024), p. 06017. DOI: 10.1051/epjconf/202429506017.

[4] ATLAS collaboration. In: *JHEP* **06** (2023), p. 031. DOI: 10.1007/JHEP06(2023)031.

Backup

Training a PNN with ATLAS Open Data (1/2)

Input data:

- ATLAS Open Data for research [2] stored in the PHYSLITE format [3]
- Standard Model and Supersymmetry pp collisions into a $\mu^+\mu^- + E_T^{miss}$ final state
- SUSY process [4] $pp \rightarrow \tilde{\chi}_1^+\tilde{\chi}_1^- \rightarrow W^+W^-\tilde{\chi}_1^0\tilde{\chi}_1^0 \rightarrow \mu^+\nu_\mu\mu^-\bar{\nu}_\mu\tilde{\chi}_1^0\tilde{\chi}_1^0$
  - many different chargino, neutralino mass pairs $(\tilde{\chi}_1^\pm, \tilde{\chi}_1^0)$

Training features:

- Binary classification of SM (label 0) and SUSY (label 1)
- Kinematics features: $(p_T, \eta, \phi)_{\mu^+}$, $(p_T, \eta, \phi)_{\mu^-}$, $m_{\mu\mu}, m_T, E_T^{miss}, m_{T2}, n_{jets}$
- SUSY model parameters: masses of $\tilde{\chi}_1^\pm$ and $\tilde{\chi}_1^0$

Parameterized neural network (PNN) [1] - ML model:

- 3 hidden layers, each with 60 nodes
- Extend kinematics features ($\bar{x}$) with SUSY masses ($\bar{\theta}$)
- Enable training of only one network with many SUSY mass pairs