CPID

Comprehensive PID and its calibration and implementation in the ILD reconstruction chain

Uli Einhaus
3rd ECFA workshop on e^+e^- Higgs, Top & ElectroWeak Factories
09.10.2024

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

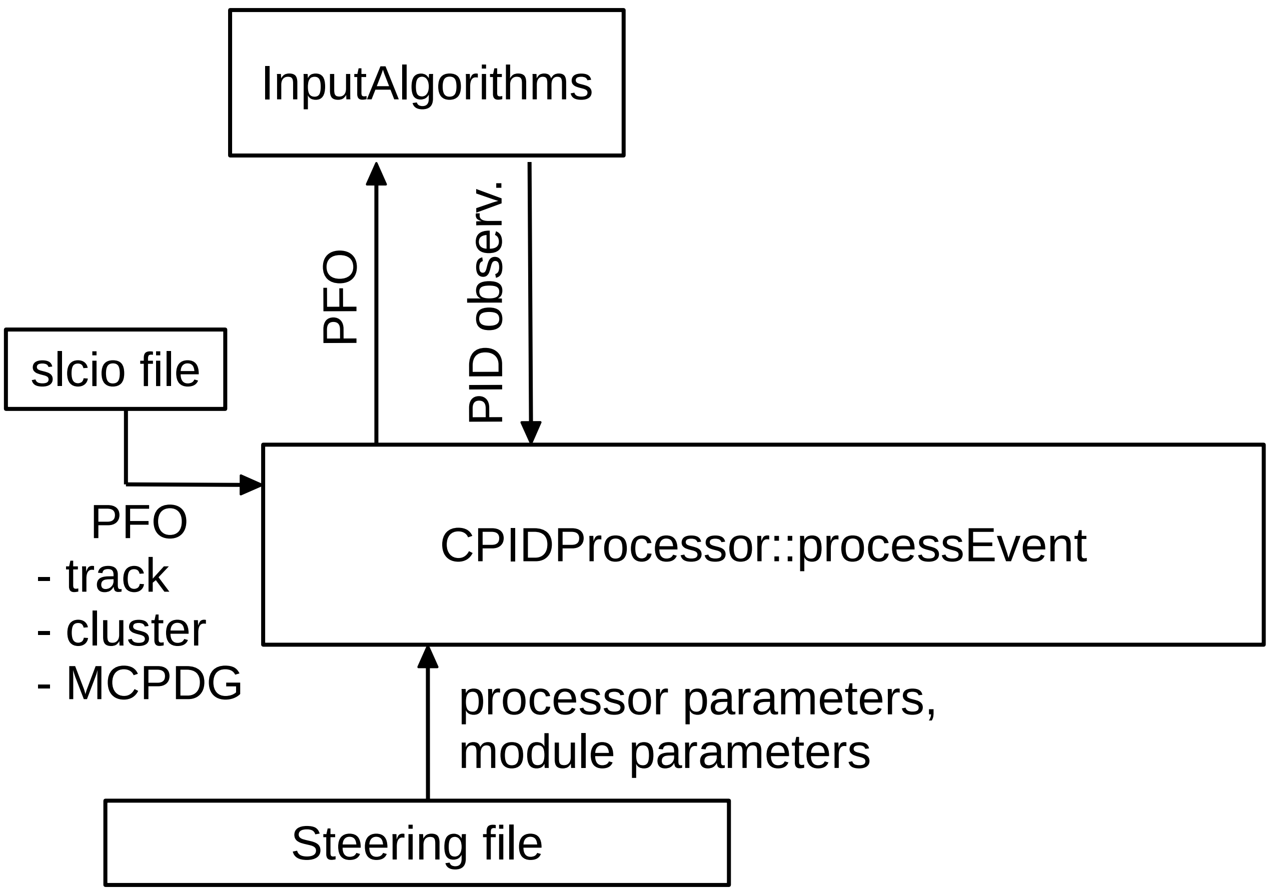
Overview

- Comprehensive Particle Identification (CPID) Processor
- Target: provide platform for future collider detectors to evaluate PID
- Approach: central book-keeping, modules for PID observables as well as training & inference
- Use Particle Flow Objects (PFOs),
- Currently Marlin processor using LCIO, usable in Gaudi via MarlinWrapper, goal is to have native implementation
- CPID is part of MarlinReco in the latest iLCSoft release

- This talk: structure, module overview, PID performance, (how to use (for analysis))

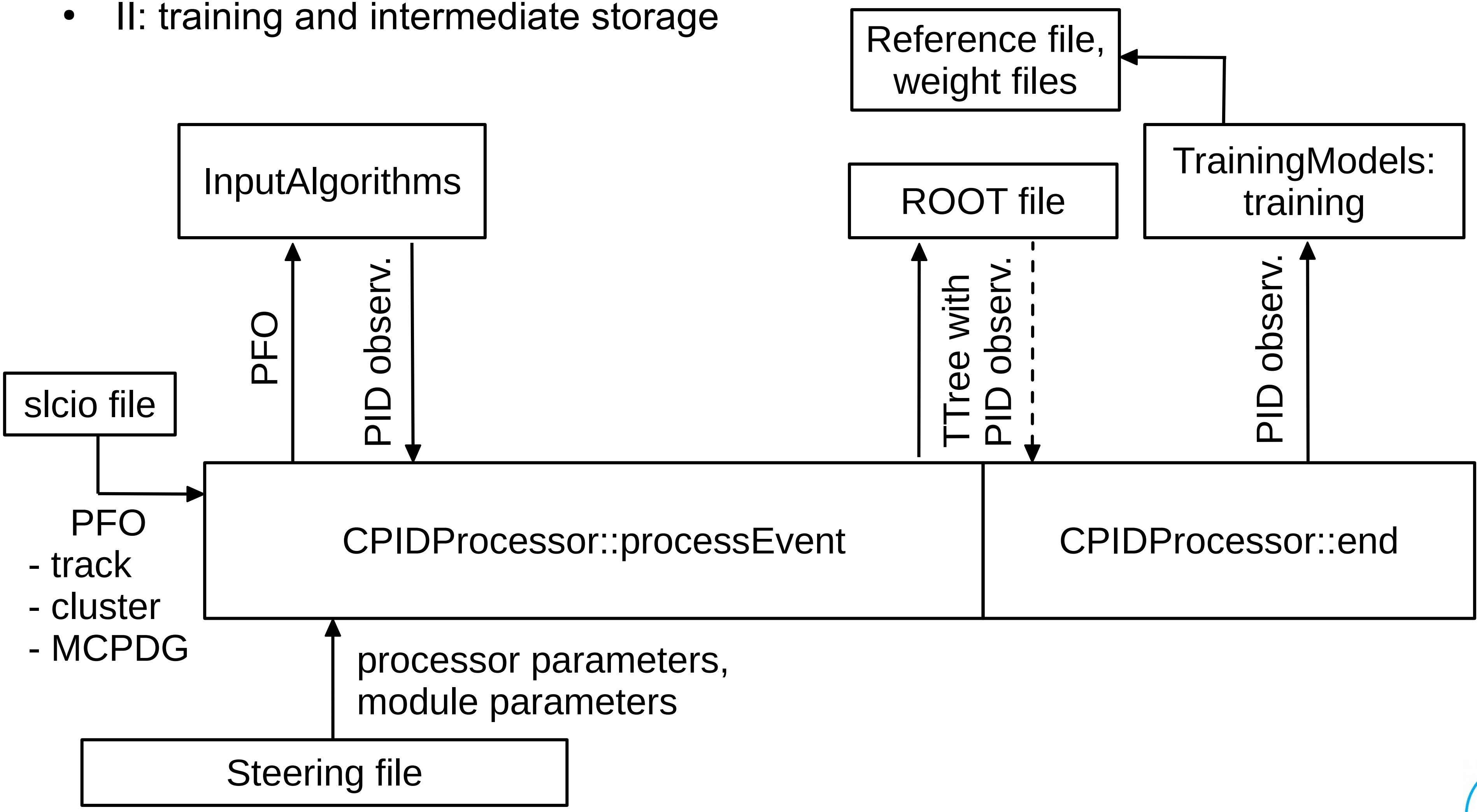
Structure of the CPID workflow

- I: set up and observable extraction



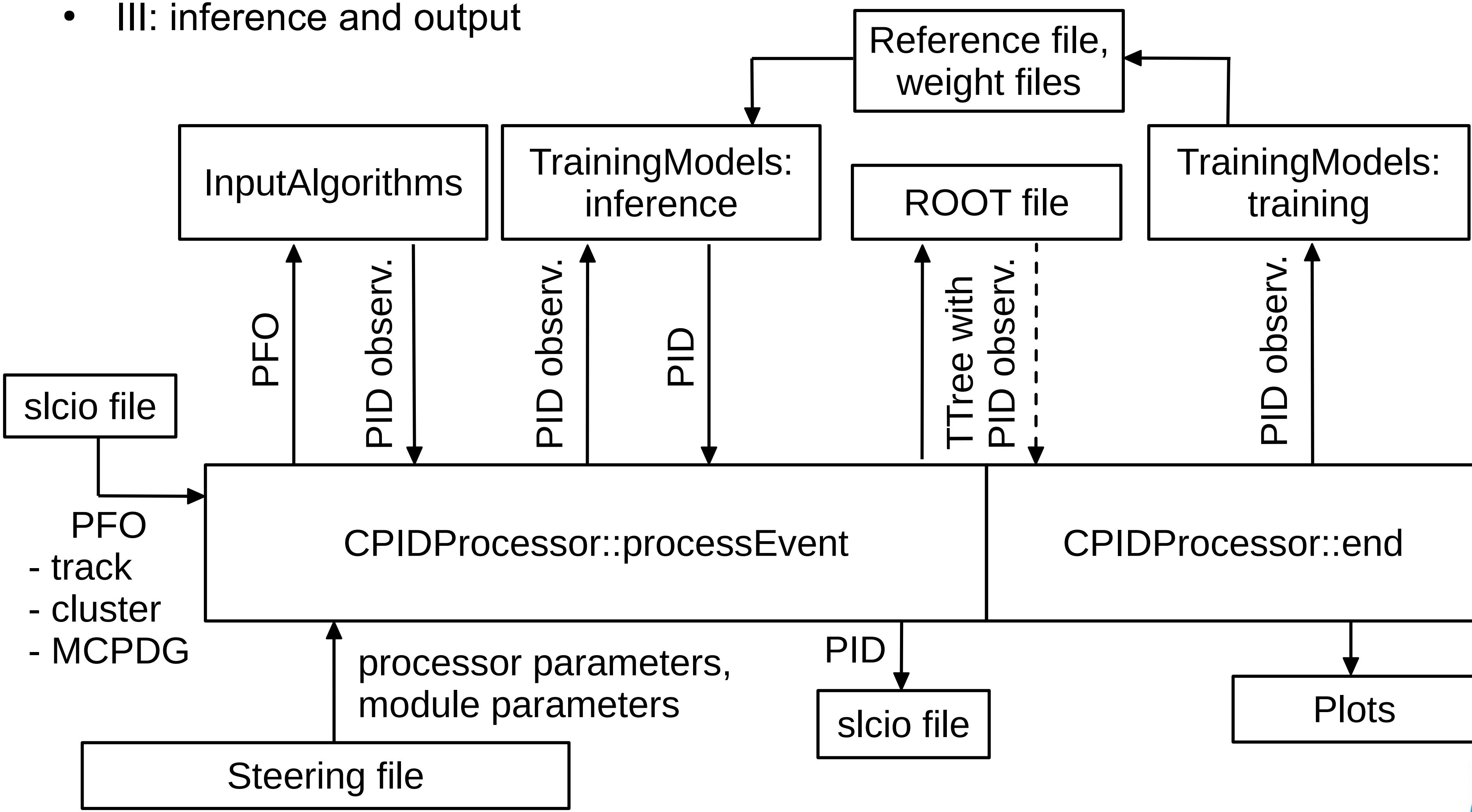
Structure of the CPID workflow

- II: training and intermediate storage



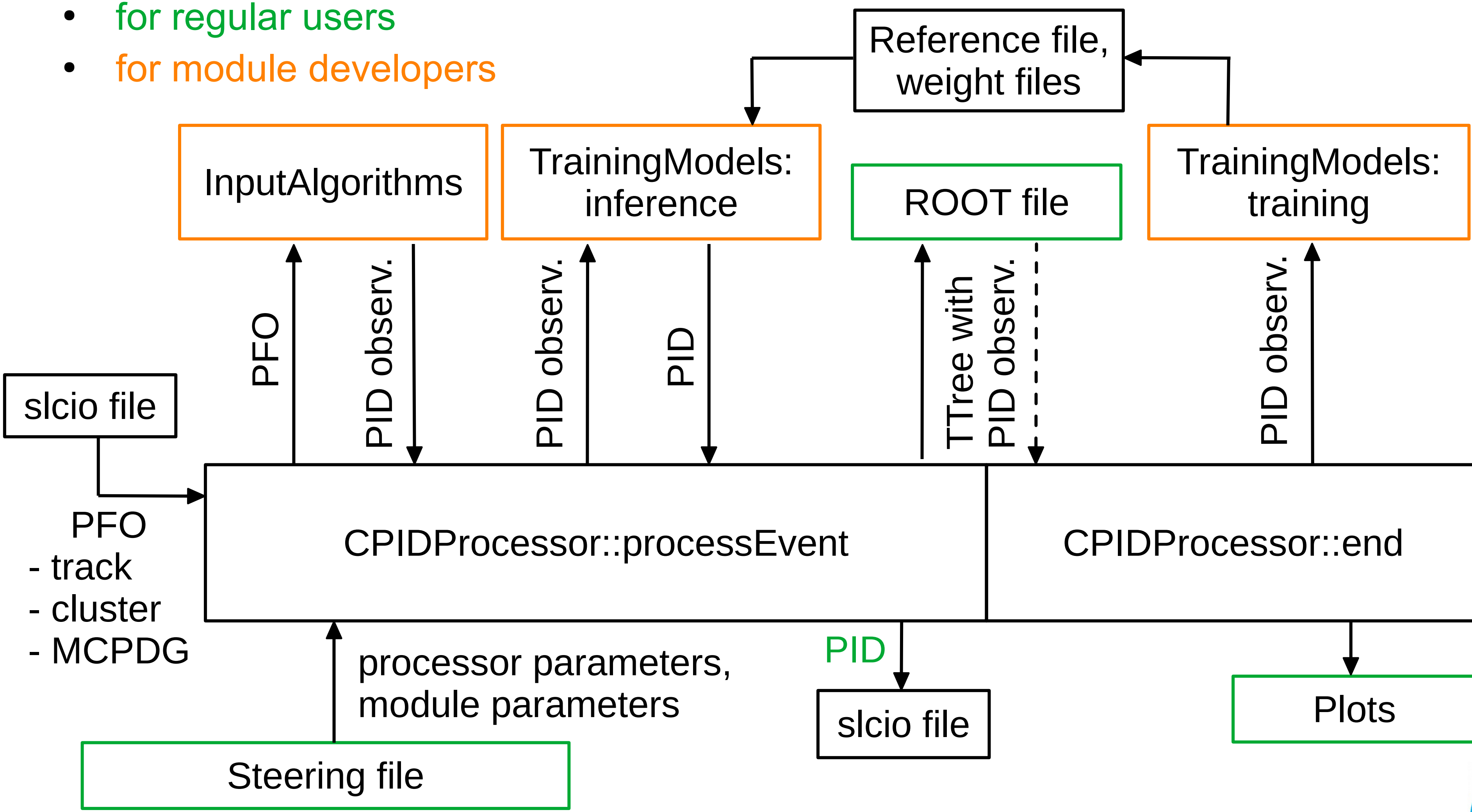
Structure of the CPID workflow

- III: inference and output



Structure of the CPID workflow

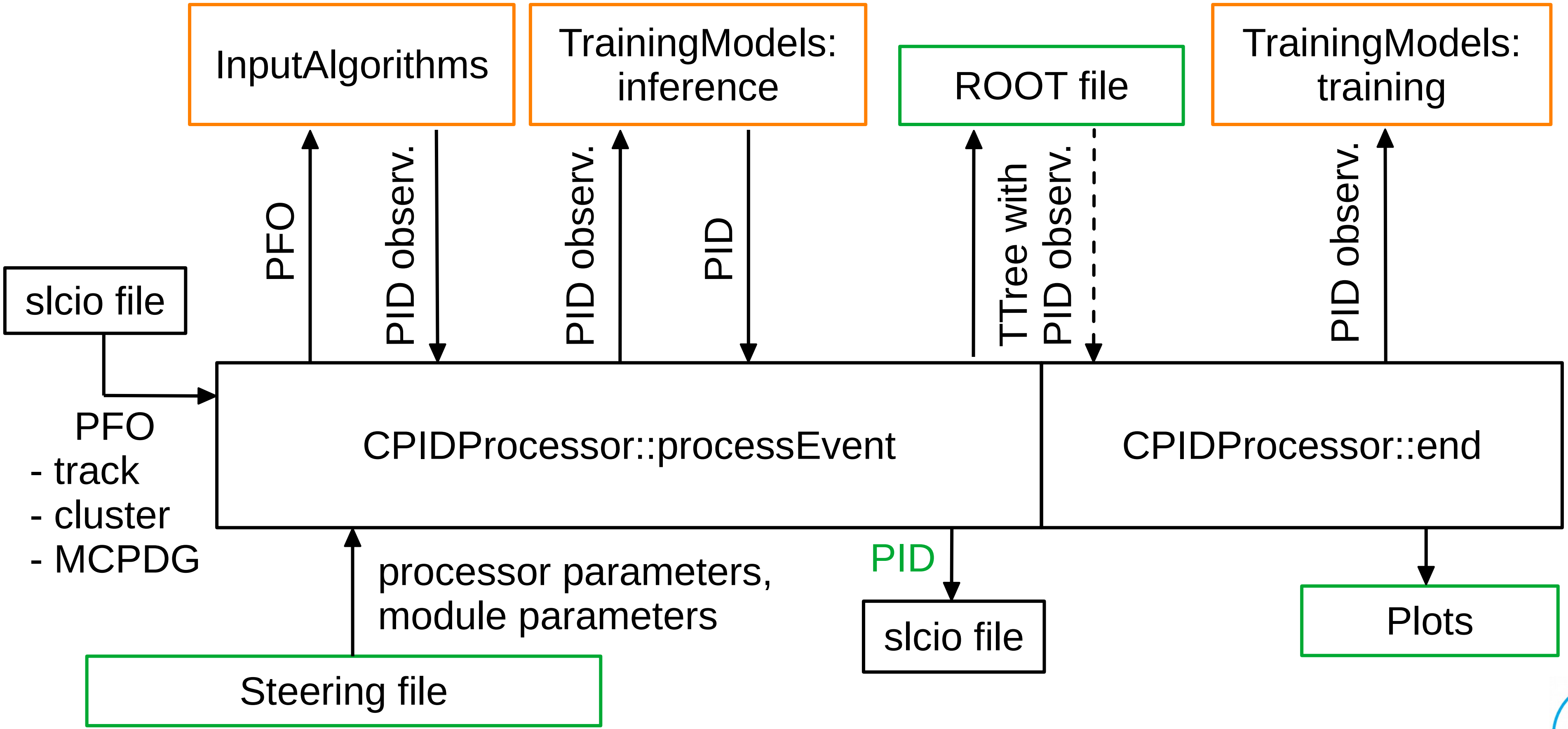
- for regular users
- for module developers



Structure of the CPID workflow

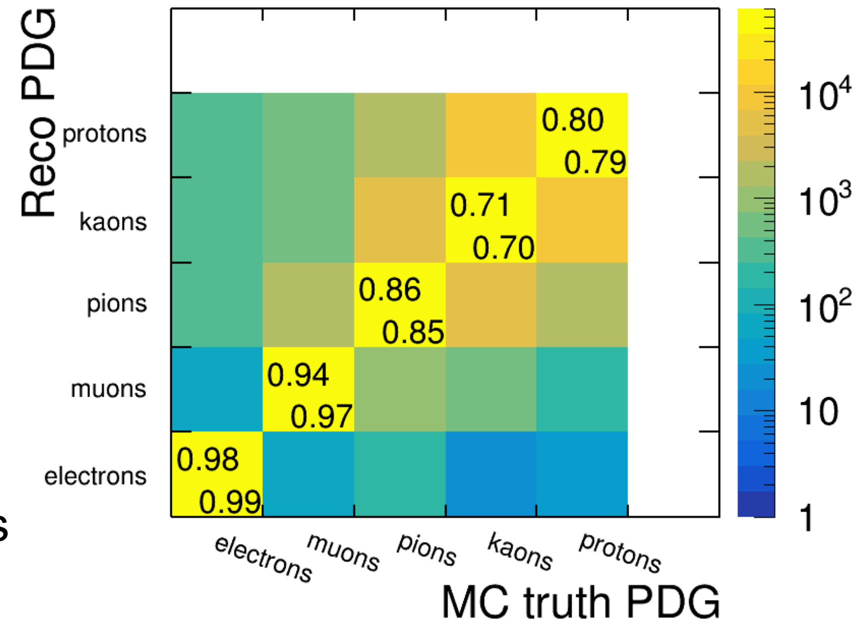
- for regular users
- for module developers

Dynamic loading of modules means module developers don’t need to touch the actual processor (analogous to Marlin processors and actual Marlin)



Typical performance plot

- Confusion matrix of reconstructed vs. MC PID for the 5 detector-stable charged particles (electrons, muons, pions, kaons, protons)
- Numbers on diagonal are efficiency/purity of that element, i.e. correctly identified PID
- Note: colour is log scale



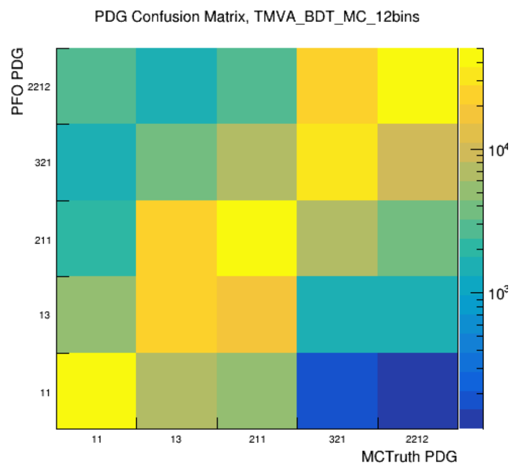
- Use single particle samples with identical numbers of particles per species, flat in log(p) and isotropic
- Split momentum range of 1 - 100 GeV into 12 momentum bin with separate multiclass BDT each (to ease momentum dependence of PID observables)
- CPID Output: BDT score for each species hypothesis, for plot put in bin with highest score

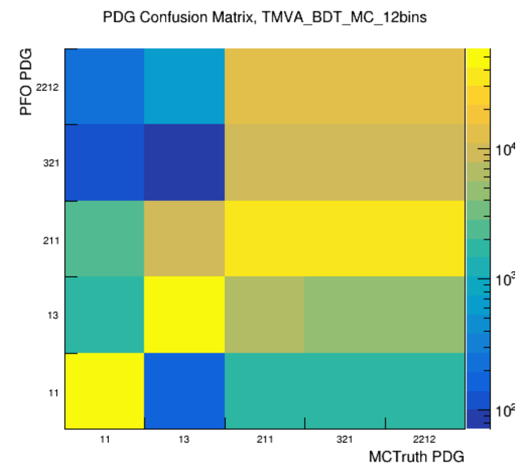
Training setup for ILD Std Reco

- Run CPID training on ILD to provide default weight files for current ~best PID
- This config is intended for the 250 GeV MC production of 2020
- To be run on DST files → included in v02-02-03-p01 miniDST.xml
- Use available observables:
  - dE/dx, resolution of about 4.5%
  - Pandora PID output based on cluster info (e vs. μ vs. pi)
  - LeptonID by Leonhard, largely re-assessment of cluster info + some dE/dx, some improvements over Pandora PID
    (for now, LeptonID needs output from the LikelihoodPIDProcessor)
  - Time of Flight, unfortunately the ‘old’ version of 2020, not Bohdan’s latest & greatest, assumes 50 ps timing resolution for 10 first hits of a PFO in the ECal

Individual observable performances

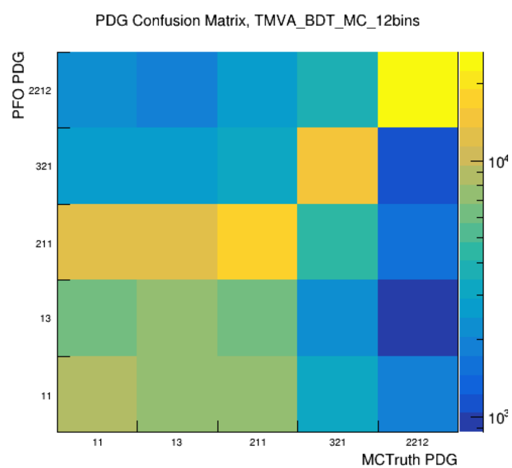
dE/dx

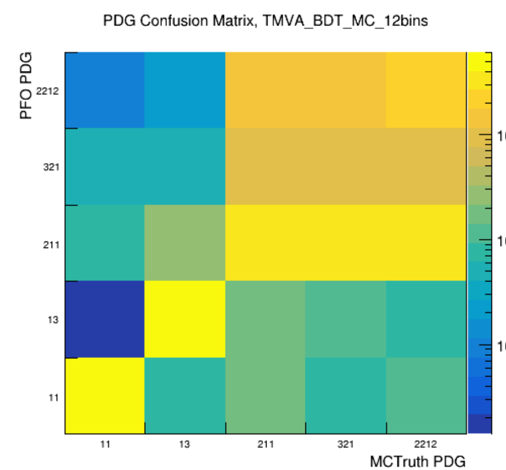




Pandora

TOF
(only from
1 - 10 GeV)

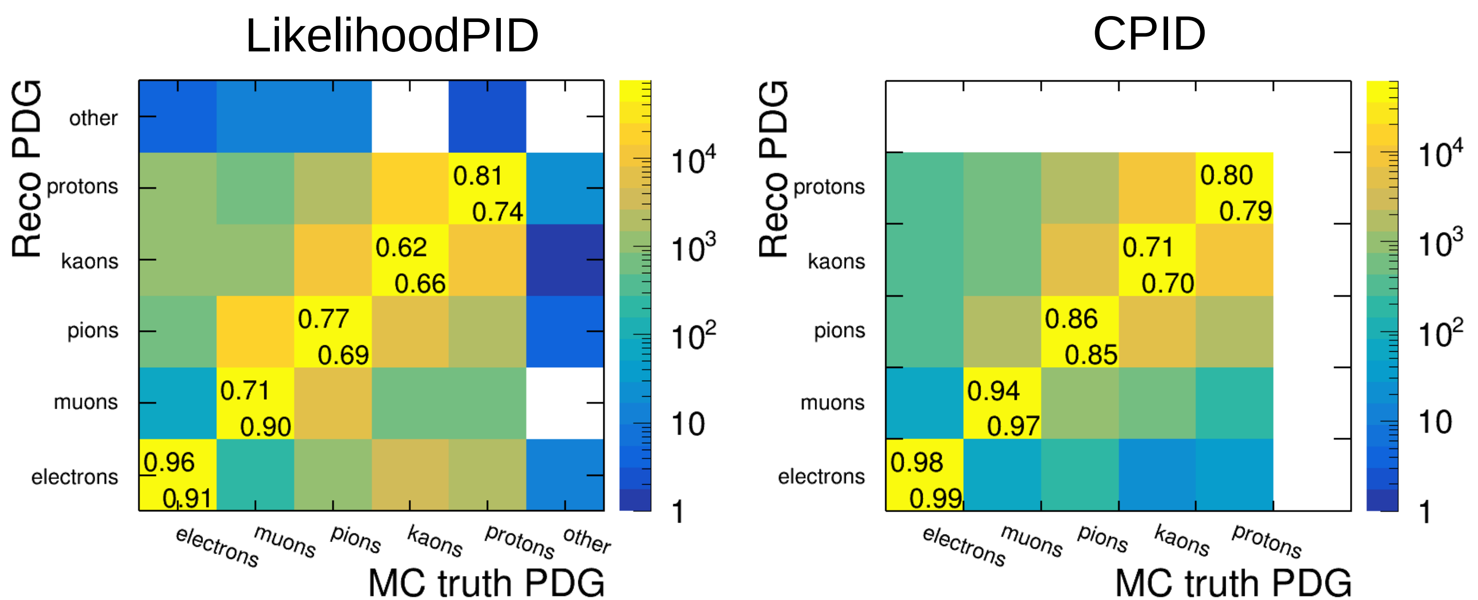




LeptonID

CPID calibration for 250 GeV

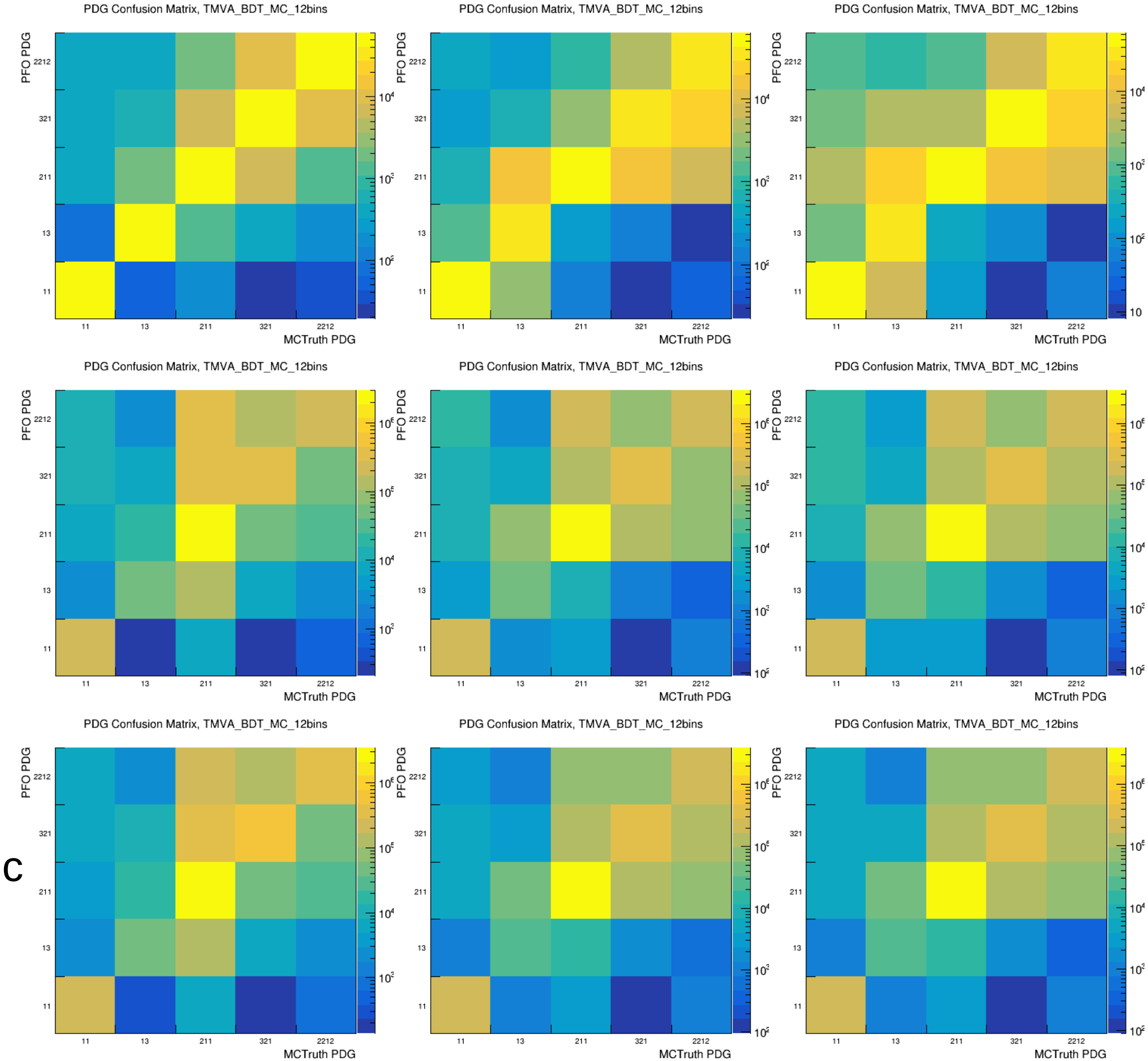
- Result below shows that it does everywhere better than the current LikelihoodPID, in particular thanks to the additional TOF
- Check if training is universal or sample-specific
  → train on single particles, 2f-Z-hadronic, 4f-WW-hadronic, infer full cross-over
- Results on next slide: confusion matrix of confusion matrices



training on: single particles 2f-Z-hadronic 4f-WW-hadronic

inference to:

- single particles
- 2f-Z-hadronic
- 4f-WW-hadronic



CPID calibration result & implementation

- Conclusion: training is sufficiently independent from sample to provide a default set of weights
- Differences largely due to relative abundance of species in the samples, can be corrected for by chosing different working point via BDT scores

- Calibration part of ILDConfig, on set trained on single particles as default and one trained on 2f-Z-hadronic as alternative
- Implemented as part of the standard reconstruction
- Recently added: weight files for 100 MeV < p < 1 GeV
- Possibility: provide weight files for a conservative vs. an ambitious ILD? (dE/dx 4.5% vs. 3.5%, TOF 100 ps vs. 50 ps for first 10 ECal layers)

Summary

- Comprehensive PID is up and running
- Dedicated training done to provide default weight files for ILD, made sure they are reasonably independent from training sample

- For ILD now part of standard reconstruction chain, used in follow-up steps, e.g. flavour taggers
- Open:
  - ‘ambitious’ ILD training
  - implementation in native key4hep
  - application to CLD (with ARC) / IDEA (dN/dx)

How to access CPID output?

- CPID output is stored as PID info in the PFOs
- The PID algorithm name is the name of the CPID training model chosen in the Marlin steering file, in the example case: **TMVA_BDT_MC_12bins**
- The best PDG, i.e. the one with the highest BDT score, is returned by

```
_PIDMethod = "TMVA_BDT_MC_12bins";
PIDHan = new PIDHandler(PFO_collection);
PDG = PIDHan->getParticleID(PFO,PIDHan->getAlgorithmID(_PIDMethod)).getPDG();
```

- The individual BDT scores are stored as parameters in the PID info, with the names constructed from the PDG numbers defined as signal PDGs extended by "-ness", so: **11-ness, 13-ness, 211-ness, 321-ness** and **2212-ness**
- To return the BDT score for the electron hypothesis:

```
Para = PIDHan->getParticleID(PFO,PIDHan->getAlgorithmID(_PIDMethod)).getParameters();
score = Para[PIDHan->getParameterIndex(PIDHan->getAlgorithmID(_PIDMethod),"11-ness")];
```