ML inference in Future Collider Software

IDT-WG3 topical meeting on running tools for ML inference

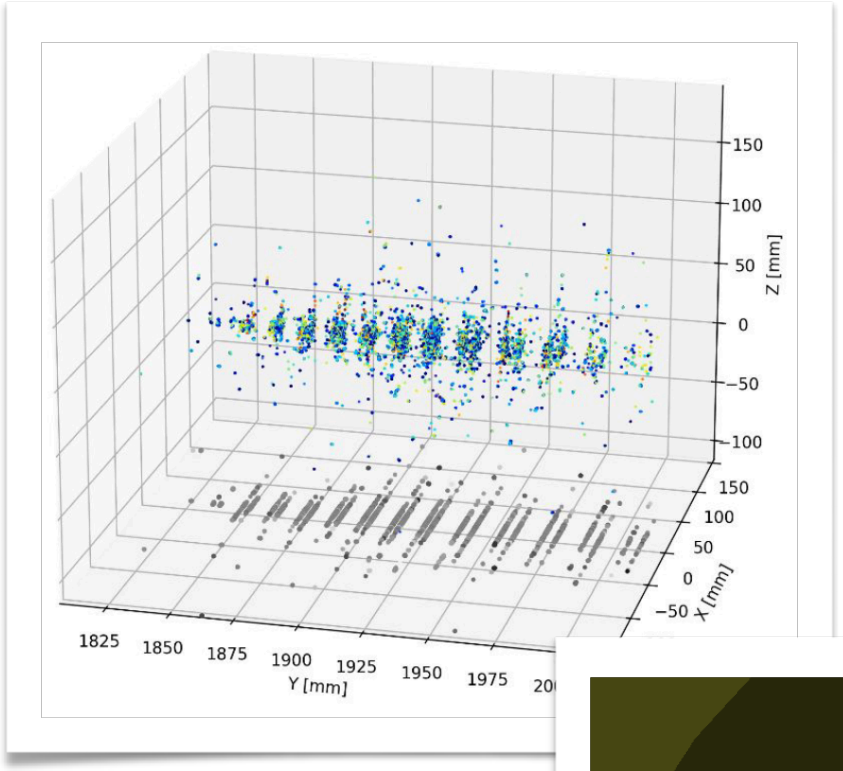
15.05.24

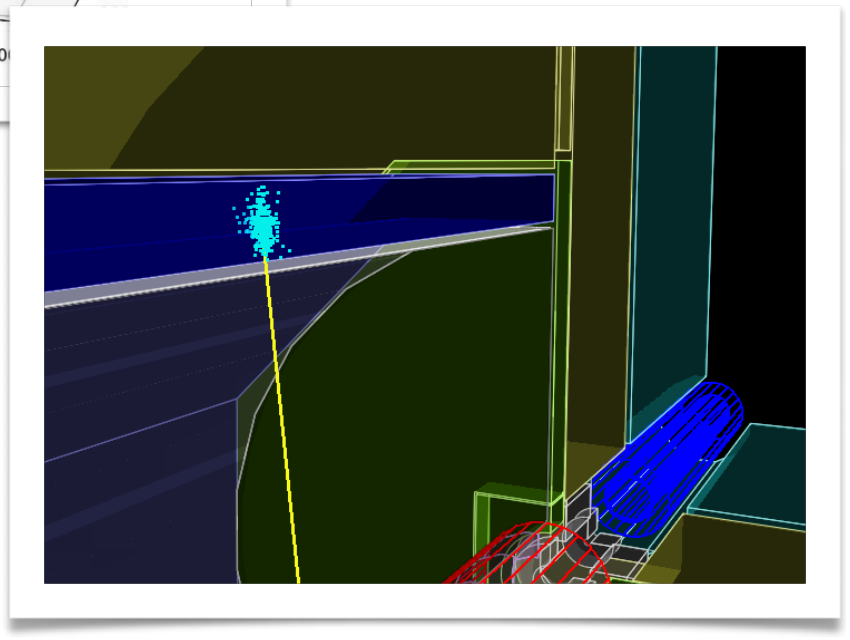
Frank Gaede, DESY

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

Outline

- Introduction
- Generative ML for Simulation
- ML inference for Reconstruction
- Future Developments
- Summary and Outlook



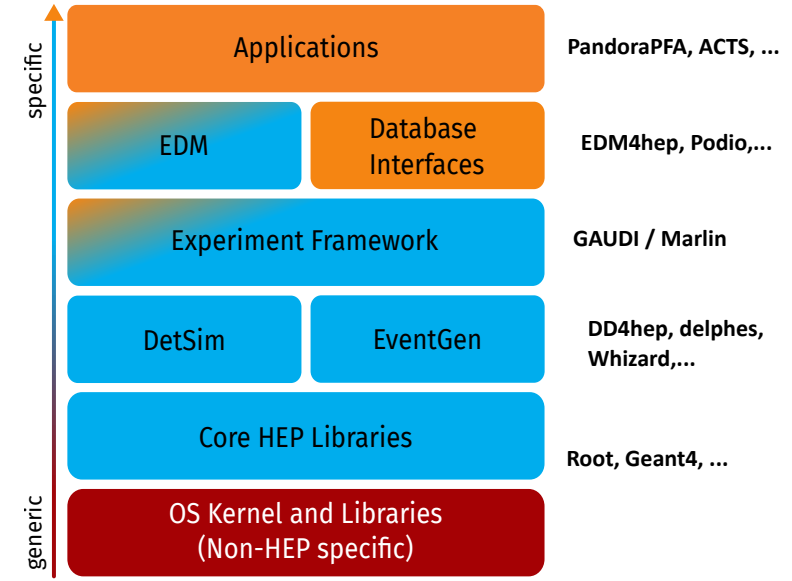


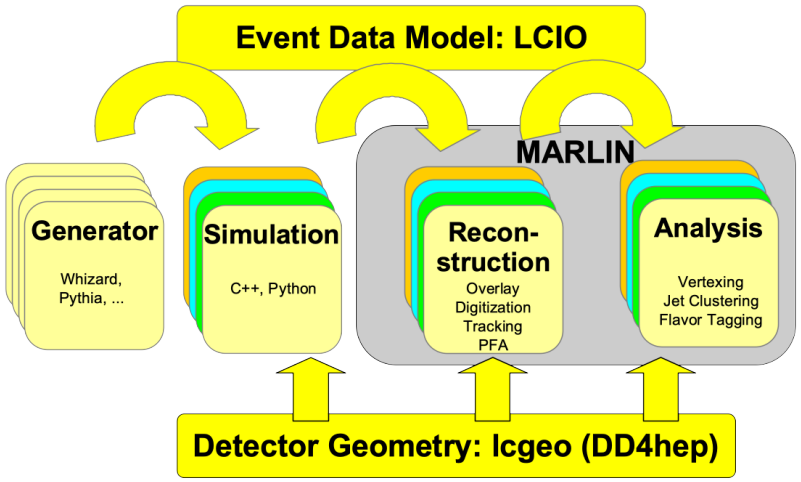
Introduction

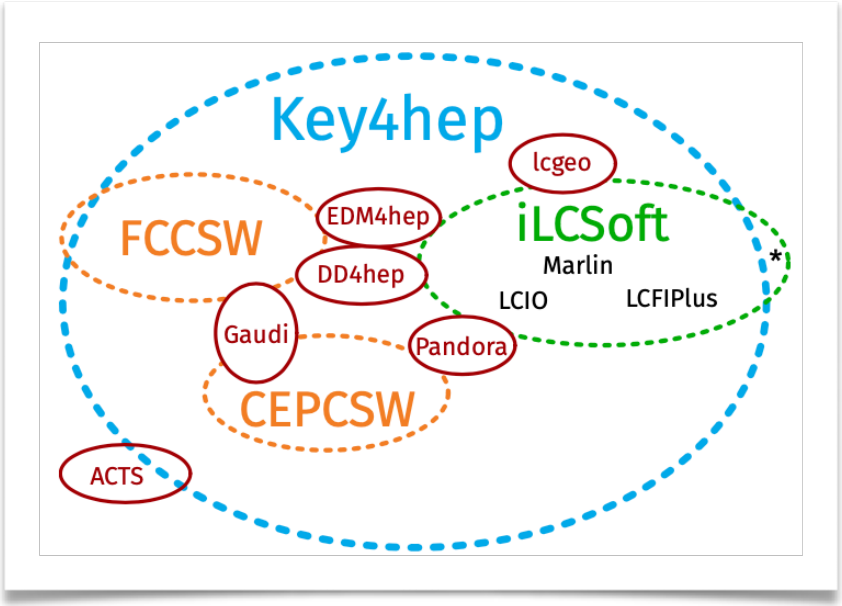
turnkey software stack for all future colliders

- HEP community decided to develop a **common turnkey software stack** – for future collider studies: **Key4hep**
- supported by **HSF** and **CERN EP-R&D** and *AIDAinnova*
- involved communities/contributors:
  - CEPC, CLIC, FCC, EIC, ILC, LUXE, Muon Collider …

- **all future collider ML inference tools should be developed in Key4hep**
  - or iLCSoft as transition for linear collider still ongoing





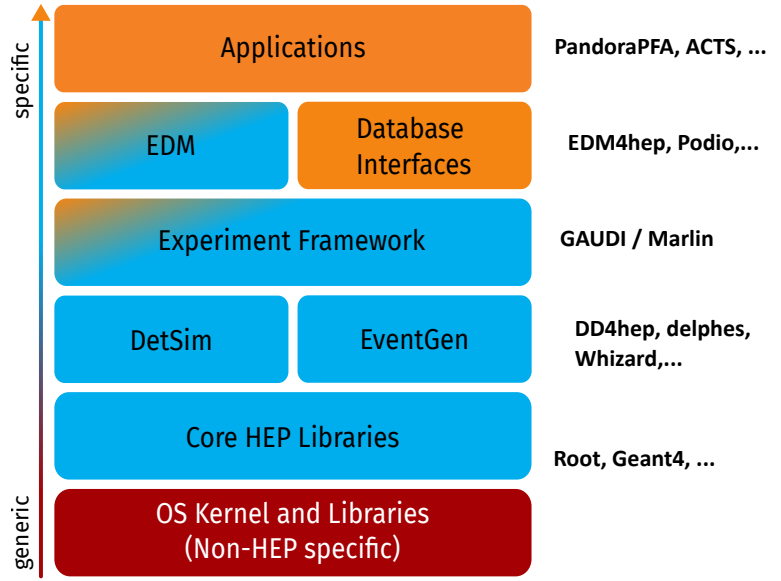


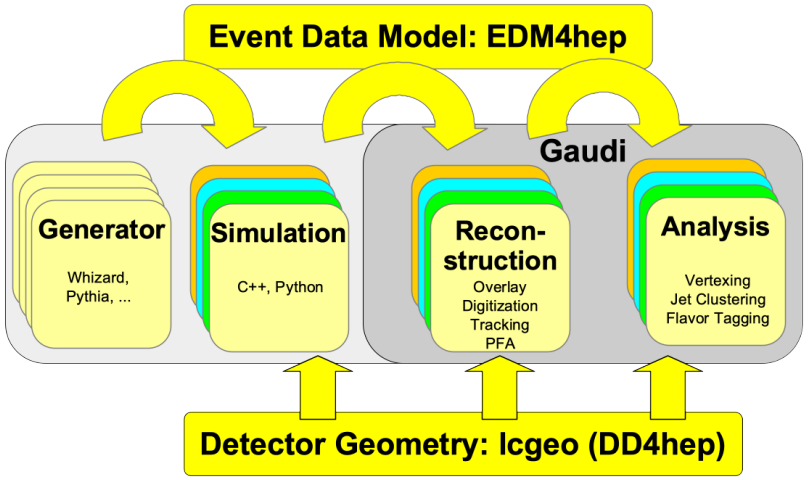
Introduction

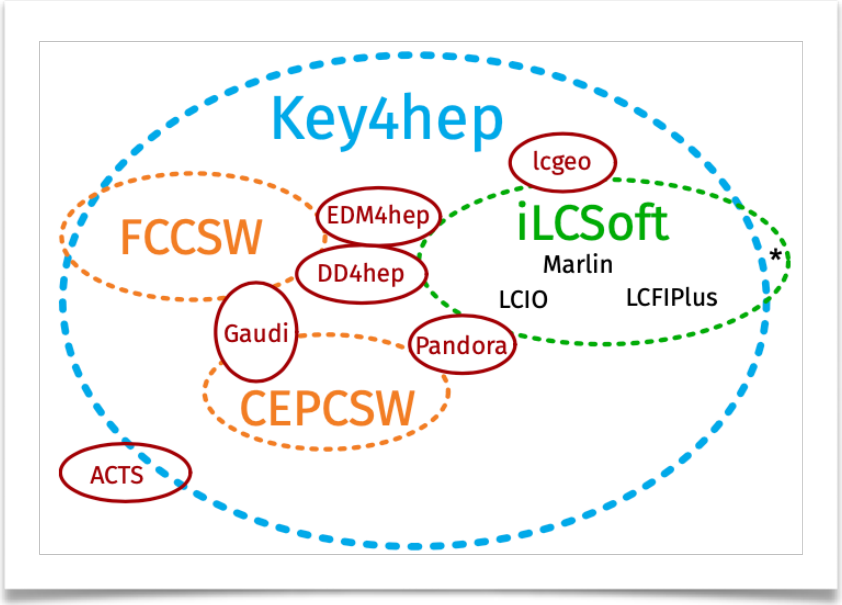
turnkey software stack for all future colliders

- HEP community decided to develop a **common turnkey software stack** – for future collider studies: **Key4hep**
- supported by **HSF** and **CERN EP-R&D** and *AIDAinnova*
- involved communities/contributors:
  - CEPC, CLIC, FCC, EIC, ILC, LUXE, Muon Collider …

- **all future collider ML inference tools should be developed in Key4hep**
  - or iLCSoft as transition for linear collider still ongoing







Introduction

modern (generative) machine learning

- modern machine learning development is almost exclusively done in the **Python** world using the **PyTorch** toolkit and to a much lesser extend TensorFlow/KERAS
  - in HEP traditionally also ROOT-TMVA used (for more classic approaches)
- mostly using **HDF5** file format for training, test and validation data

- to run such new models in standard HEP frameworks such as Key4hep/iLCSoft some **suitable glue code or middleware** is needed -> this talk

- in particular we need C++ inference tools to call the ML model during sim/rec/ana in framework
  - here we focus on
    - **ONNX** - Open Neural Network Exchange (https://onnx.ai/)
    - **libtorch** (https://pytorch.org/cppdocs/)

Existing ML inference tools

in Key4hep - iLCSoft and (briefly) covered today

generative ML in simulation:

- **DDFastShowerML** (DDML)
  - generative ML architectures for fast shower simulation

running inference for (high level) reconstruction:

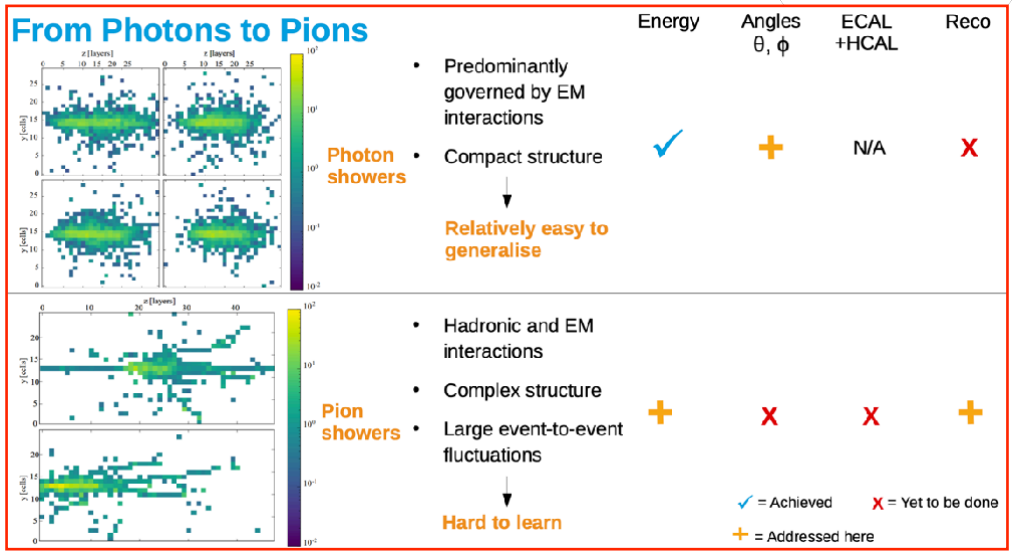
- **CPID**
  - comprehensive particle identification for future colliders

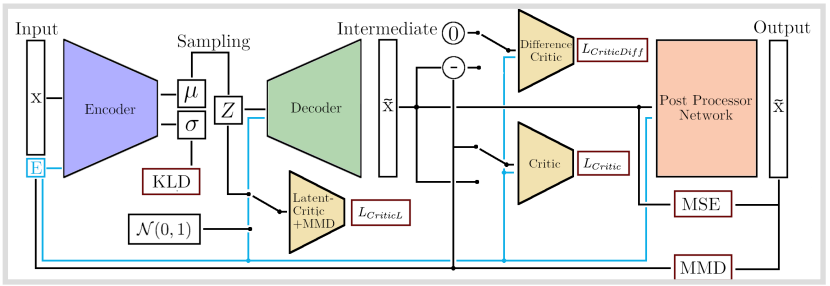
- **MarlinML (FlavourTaggingML)**
  - machine learning flavour tagging for future higgs factories

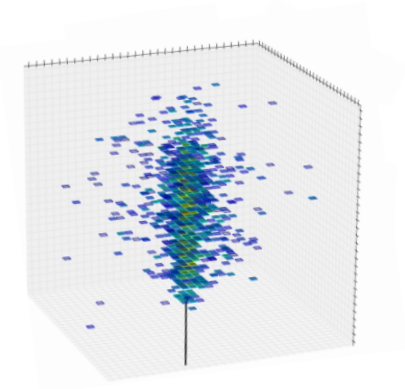
Fast shower simulation

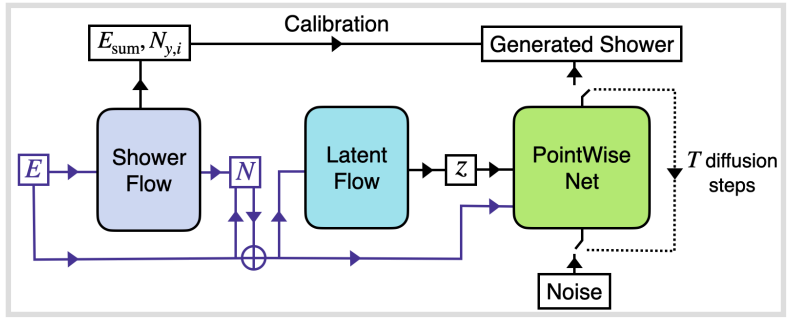
with generative ML

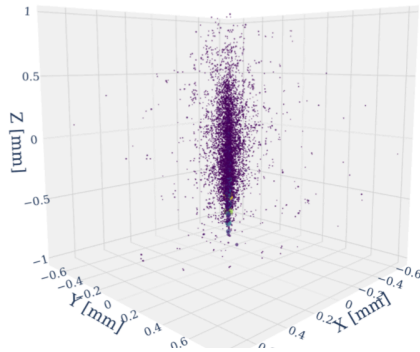
- shower generation with ML offers great potential O(10e3) for faster and more sustainable simulation in HEP
- studying many generative models: (W)GANs, VAE, BIB-AEs, Normalising Flows, Point Cloud Diffusion Models,…
- working on fixed, regular 3d grids or with point clouds of energy depositions
- context of highly granular calorimeters for future colliders ideal context to study this
- and develop a **dedicated common tool** for this…







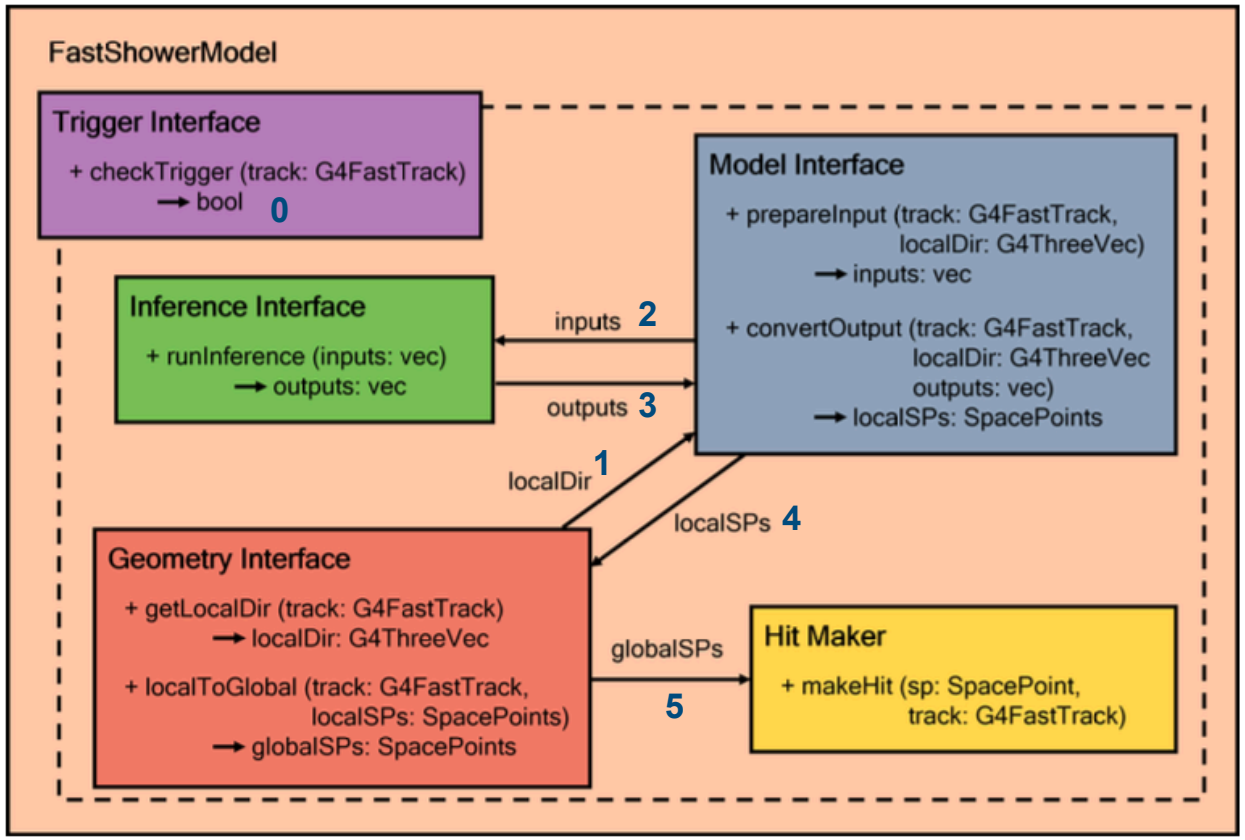




DDFastShowerML

fast simulation with ML inference in DD4hep

- based on FastSim hooks implemented in Geant4/ DDG4
- split different necessary steps into dedicated components that can be - partly - reused:
  - **Trigger**: decide for w/ particles to use ML
  - **Model**: prepare input and output
    - typical for model specific
  - **Inference**: generic inference w/ **ONNX** and **libtorch**
    - common for all models
  - **Geometry**: local to global conversions of cell positions
    - barrel and endcap w/ some reuse between models (uses *ddrec::LayeredCalorimeter*)

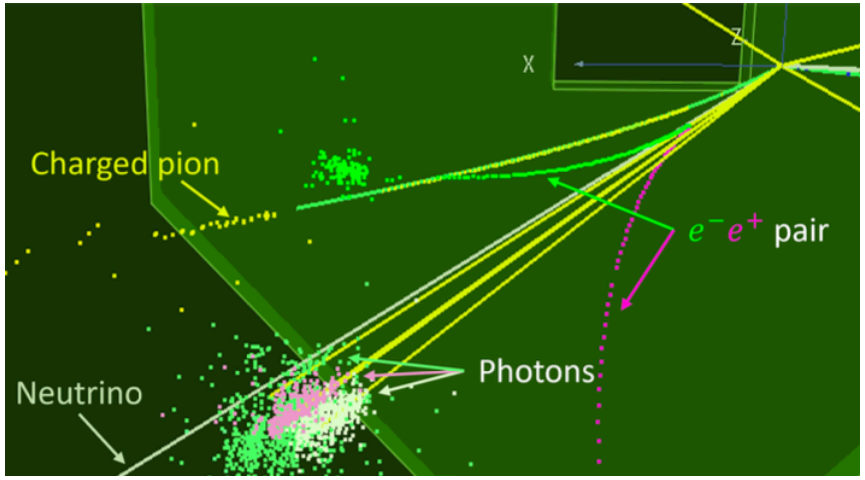


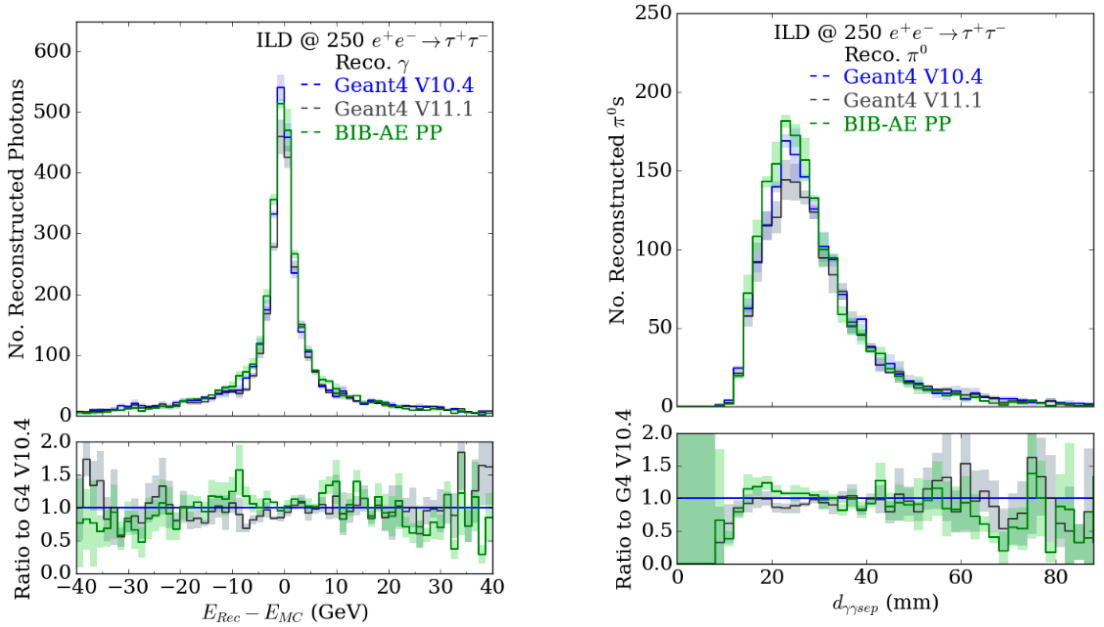
https://gitlab.desy.de/ilcsoft/ddfastshowerml

DDFastShowerML

example usage: tau-pair events in ILD

- with DDFastShowerML can include generative ML in **standard full simulation** to replace part of the showers with generative fast sim
- prerequisites:
  - need to be able to create ONNX or torch binary files:
    - typically have the whole ML network architecture running in one large **PyTorch** script
  - have a **DD4hep detector model** (with DDRec extensions)
- example: e+e- -> tau+ tau- -> XXX + pi0 -> gamma gamma at 250 GeV in ILD using a BIB-AE model conditioned on two angles





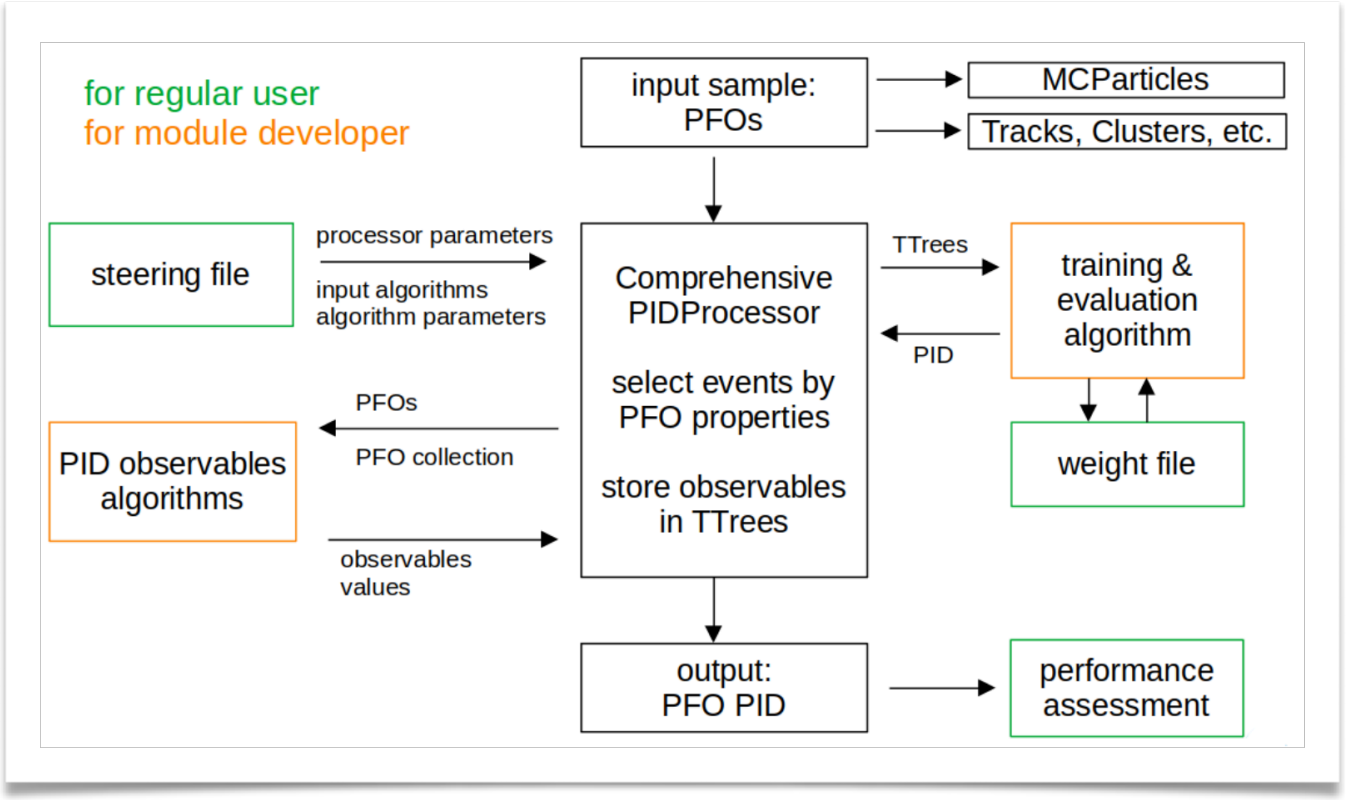
taus decaying in ILD detector w/ ML photons (PhD theses Peter McKeown)

Particle ID with ML

CPID: Comprehensive Particle identification Framework for Future Colliders

U.Einhaus

- implemented in LCIO and Marlin
  - see: MarlinReco/Analysis/PIDTools
- using **ROOT::TTree** as data exchange format
  - provides direct interface to **ROOT-TMVA**
  - easy conversion to **HDF5** (if needed)
- current implementation uses **TMVA** only
- could maybe extended to use **libtorch/ONNX** …



https://arxiv.org/pdf/2307.15635

Flavour Tagging

With Deep Neural Networks

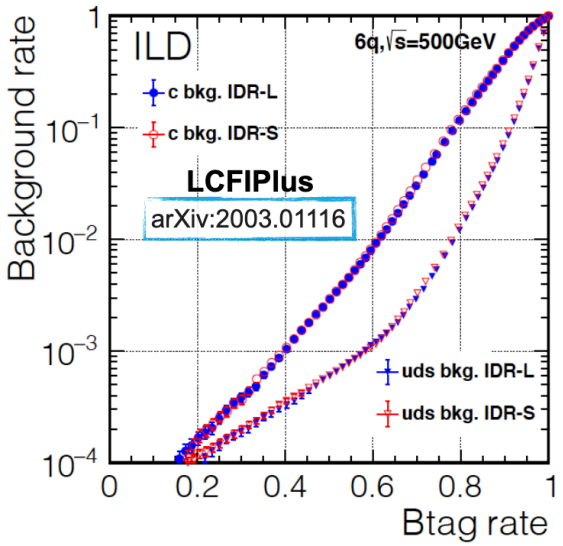
M.Meyer

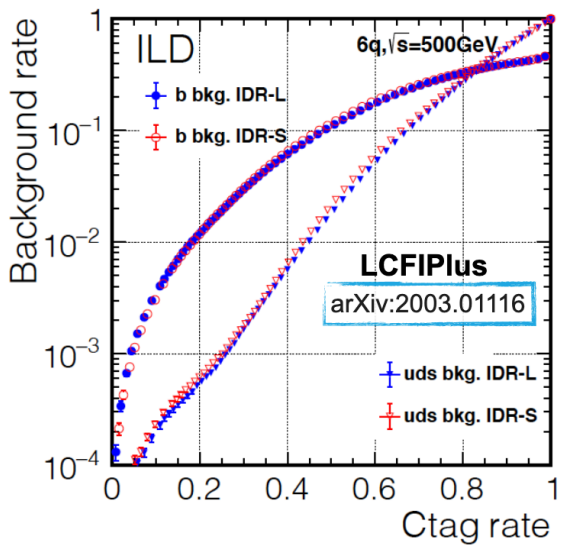
- LCFIPlus flavour tagging workhorse for ILC studies for ~20 years
- recent progress in deep ML applied to LHC suggests significant performance increases possible
- study application of CMS DeepJet and ParticleNet to ILD (in iLCSoft)

- **current standard** for heavy flavour tagging at ILD: **LCFIPlus**

arXiv:1506.08371,
https://github.com/lcfiplus/LCFIPlus

- based on TMVA (BDTs)





➡ Can the **heavy flavour tagging** be **improved** by replacing the BDTs used in LCFIPlus **with (deep) NNs**?

Flavour Tagging

With Deep Neural Networks

M.Meyer

DeepJet: input features

global variables

p^{jet}, p_T^{jet},
$N_{charged\ jet\ const.}$, $N_{neutral\ jet\ const.}$, N_{SV}
additional global variables from LCFIPlus

21 input features

charged jet constituents

p^{track}/p^{jet}, p_T^{track} (rel. jet), $\vec{p}^{track} \cdot \vec{p}^{jet}/p^{jet}$
ΔR(track, jet)
impact parameter & significances
track reconstructed in PV?
lepton related variables
pid variables
χ2/ndf

19 input features

neutral jet constituents

$p^{neutral\ const.}$, $p^{neutral\ const.}/p^{jet}$
ΔR(jet, neutral const.)
is photon?
$E_{HCAL}/E_{HCAL+ECAL}$

5 input features

secondary vertices

m_{SV}
$N_{tracks\ in\ SV}$
ΔR(SV, jet)
E_{SV}/E_{jet}, E_{SV}
cos(flight direction$_{SV}$, $\vec{p}_{SV}$)
3D IP and significance
χ2, ndf

10 input features

ParticleNet: input features

jet constituents: coordinates

Δη, ΔΦ

jet constituents: features

Δη, ΔΦ
$\log(p_T)$, $\log(E)$, $\log(p_T/p_T^{jet})$, $\log(E/E^{jet})$, $\vec{p}^{track} \cdot \vec{p}^{jet}/p^{jet}$
ΔR
q
isElectron, isMuon, isChargedHadron, isNeutralHadron, isPhoton
impact parameter & significances
track used in PV?
lepton related variables
pid variables
$E_{HCAL}/E_{HCAL+ECAL}$
χ2/ndf

28 input features

secondary vertices: coordinates

Δη, ΔΦ

secondary vertices: features

Δη, ΔΦ
$\log(p_T)$, E_{SV}/E_{jet}, E_{SV}
η
m_{SV}
$N_{tracks\ in\ SV}$
χ2/ndf
impact parameters &significances
cos(flight direction$_{SV}$, $\vec{p}_{SV}$)

14 input features

2 SVs & all jet constituents considered, no ordering of inputs

- dedicated Marlin processor to prepare input variables from event (jet constituents):
  - VariablesForDeepMLFlavorTagger
- writes ROOT::Three file for training and assigns parameters (as PIDParameters) to PFOs for inference

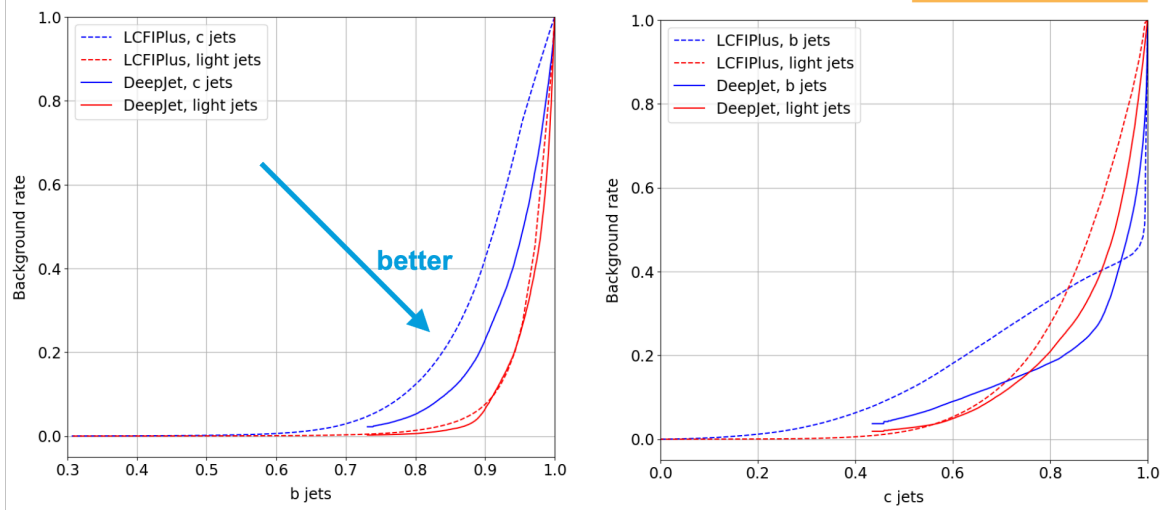
Flavour Tagging

With Deep Neural Networks

M.Meyer

DeepJet: ROC curves - comparison to LCFIPlus

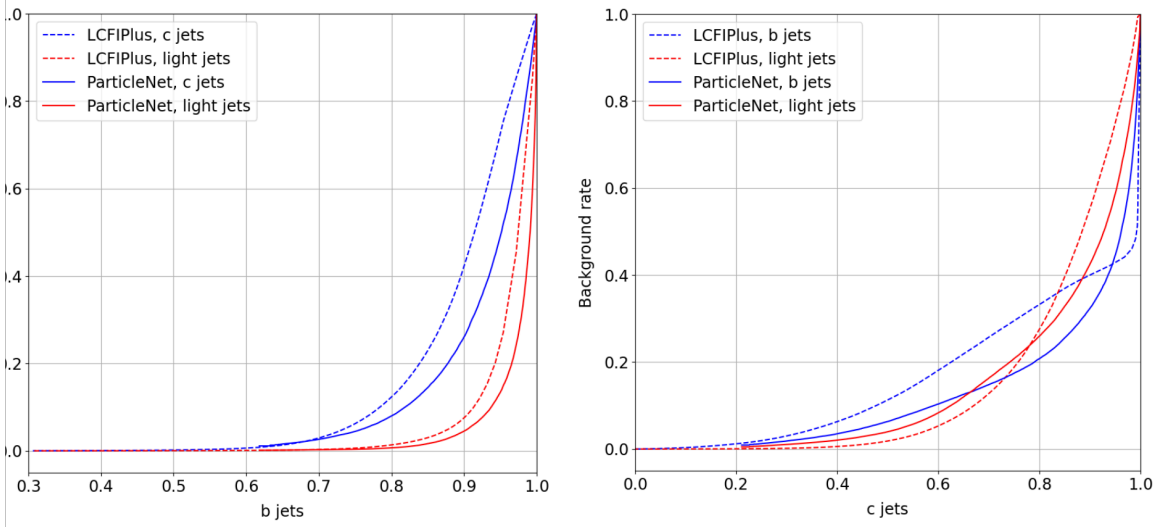
validation data



better performance of DeepJet training **over large parts of the b & c tagging efficiencies** w.r.t default LCFIPlus used in ILD

ParticleNet: ROC curves - comparison to LCFIPlus

validation data



better performance than LCFIPlus **over large parts of the b and c tagging efficiencies**

one of the first trainings with this architecture, **a lot of possibilities for optimization** (architecture, hyperparameters, features, over-training in c-jet category…)

- both deep deep learning methods achieve better performance than LCFIPlus

Flavour Tagging

M.Meyer

With Deep Neural Networks - Packages and Modules

VariablesForDeepMLFlavorTagger
determines and stores the input features via the PIDHandler

training

FlavourTaggingML

Python scripts for defining the ML models (PyTorch) and performance evaluation scripts

inference

DeepJetTaggerProcessor/ ParticleNetProcessor
create the input torch tensors, do the pre-processing and the inference

BTaggingVariables
reads out the results of the inference and stores the output in a tree and in histograms

- all packages available at https://gitlab.desy.de/ilcsoft
- serve as prototype example implementation and could be used in ILD standard reconstruction
- currently no manpower to continue working on this …

Future Developments

and related open questions

- MarlinML with the JetTagger examples is still somewhat work in progress
  - however, serves as demonstrator and prototype for how ML inference can be done in Marlin for reconstruction and analysis
- need to also include other ongoing ML activities in Marlin, e.g.:
  - Particle Transformer flavour tagging and PFA w/ DNN (see talk T. Suehara 2nd ECFA H.EW ws)
- Open Questions:
  - can some code be reused as in DDML ?
    - generic ML inference (libtorch, ONXX, …) -> probably straight forward
    - preparation of input variables (training/test/validation/inference data) -> probably quite some work needed to make this generic - maybe better to quickly write dedicated code ?
  - how can this be (re)used in Key4hep with Gaudi and EDM4hep ?
    - currently via MarlinWrapper
    - eventually need to create a **k4MLInference** package -> need EDM4hep prod. version

Summary and Outlook

- being able to run ML inference in Key4hep/iLCSoft absolutely mandatory
- some prototypes and example implementations exist:
  - DDFastShowerML, MarlinML (JetTagger), CPID
- work in progress towards:
  - more generic tools - how many different applications can be combined in one package ?
  - proper Key4hep examples using EDM4hep and Gaudi
  - manpower limited …
- your contribution and thoughts are welcome !