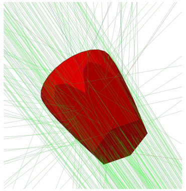
PIONEER ML Based Reconstruction Status

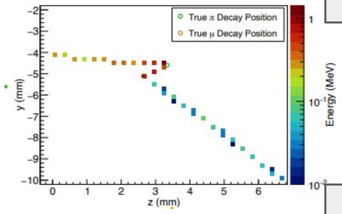
Jack Carlton
University of Kentucky

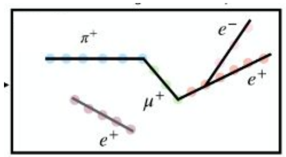
Citations: PIONEER Slides from ACTS Workshop, Patrick Schwendimann
Simulation and Analysis Presentation, Quentin Buat

PIONEER Simulation Framework

- Series of software steps to simulate the PIONEER experiment
  - Work in progress
  - Adapts as we develop our detectors/strategy
- Reconstruction designed to be used on simulated *and* real detector data
  - Current goal: proof of concept
  - Future goal: reconstruction of experimental data
- Most effort at UKy has been on the event reconstruction stage
  - Particular for the ATAR
    - Pattern finding
    - **More recently: AI Reconstruction approach**







Monte Carlo (Geant4 Based)

↓ Truth level physics interaction data in space of detectors

Detector Response (Gaudi Based)

↓ Realistic detector output data

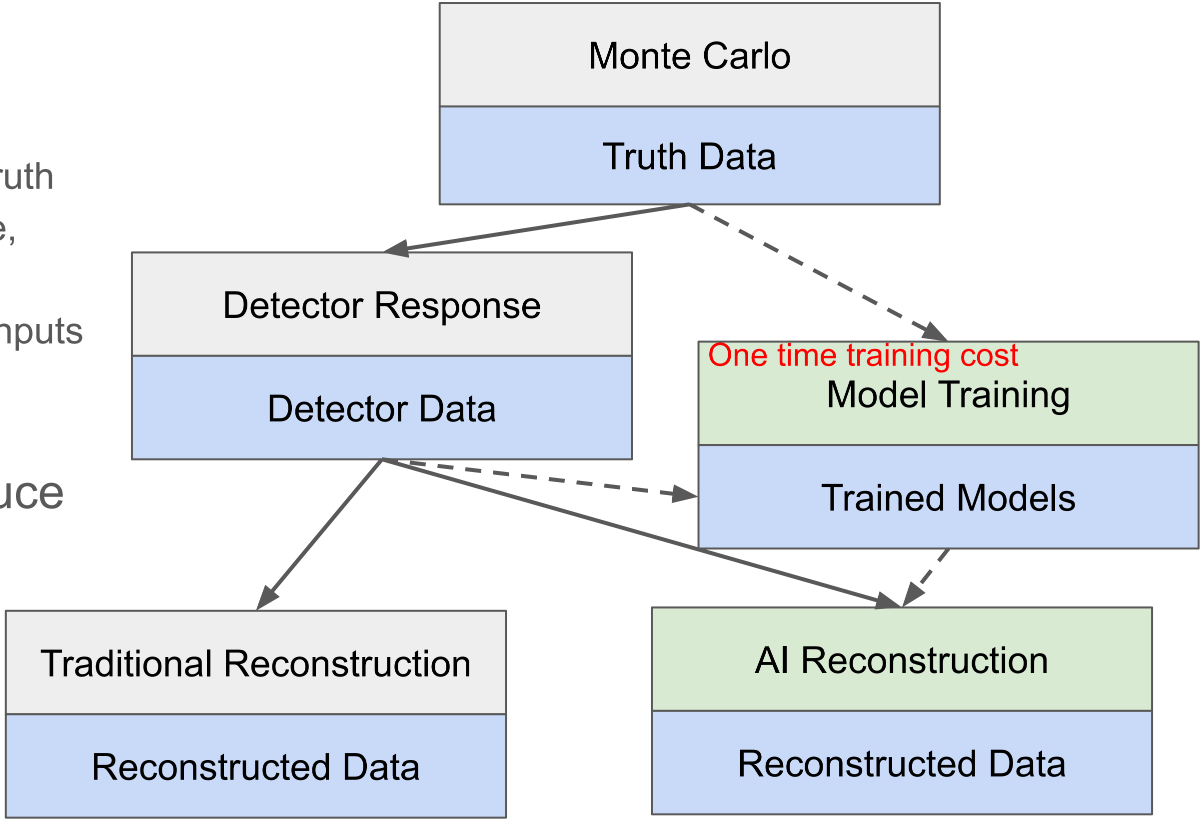
Event Reconstruction (Gaudi Based)

↓ Physical events reconstructed from detector data

Reconstructed Data

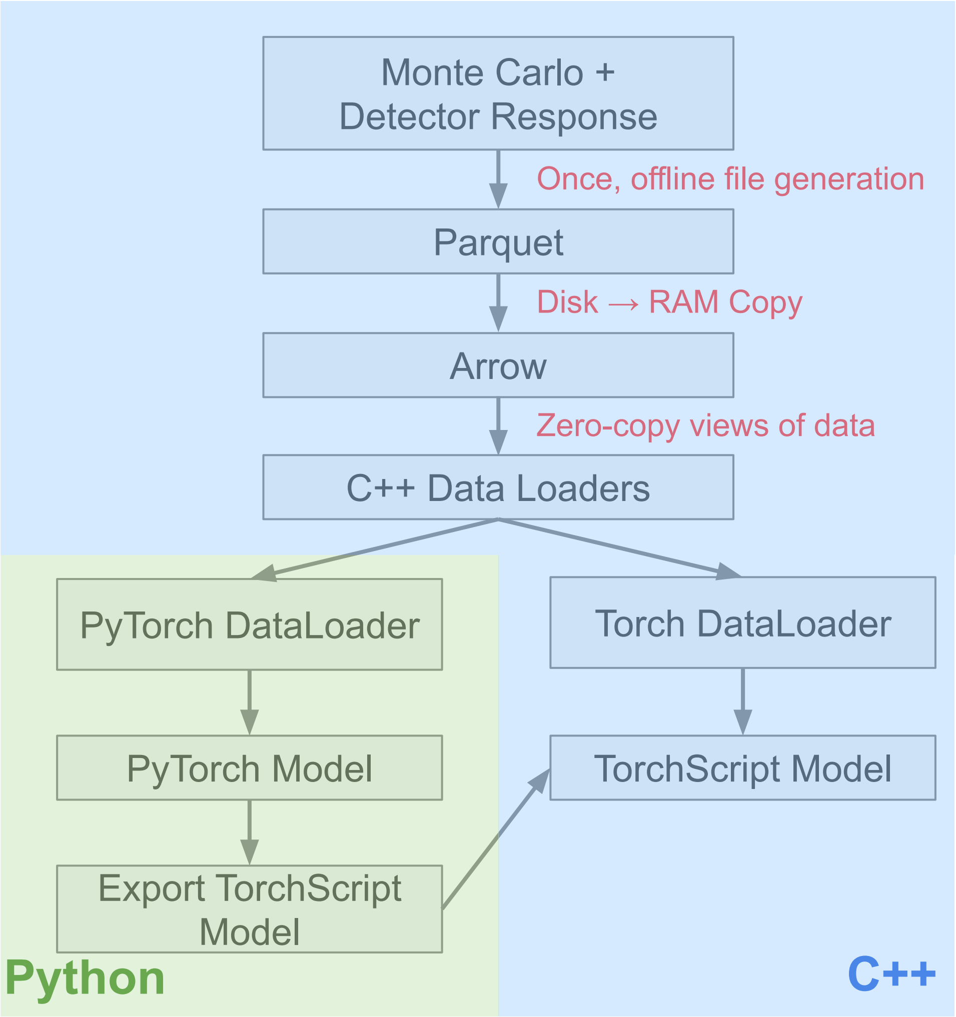
Why Reconstruction Can be an AI task

- Simulation provides all needed information
  - Geant4 simulation gives truth targets (ex. Positron angle, true pion stop)
  - Detector response gives inputs (ex. ATAR strip hit 5D information)
- The simulation can produce large quantities of data needed for training



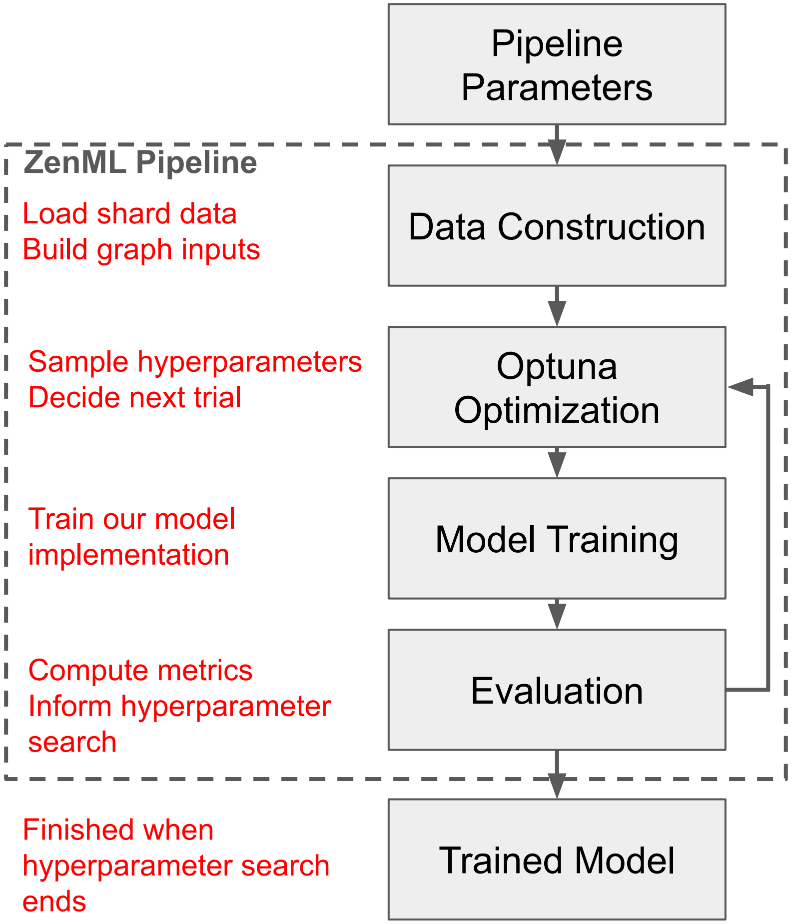
Current Status: Data Flow

- Models ultimately will run in the simulation framework, options:
  - **TorchScripts run in C++**
    - Least friction with current development efforts
  - Gaudi python stages
    - Likely too slow
  - Pybinds running model
    - Likely too slow
- Training must remains in Python
  - ML ecosystem is much more mature in python
- Logic should be shared until division is necessary
  - Natural division is right before creating torch objects



Current Status: Training

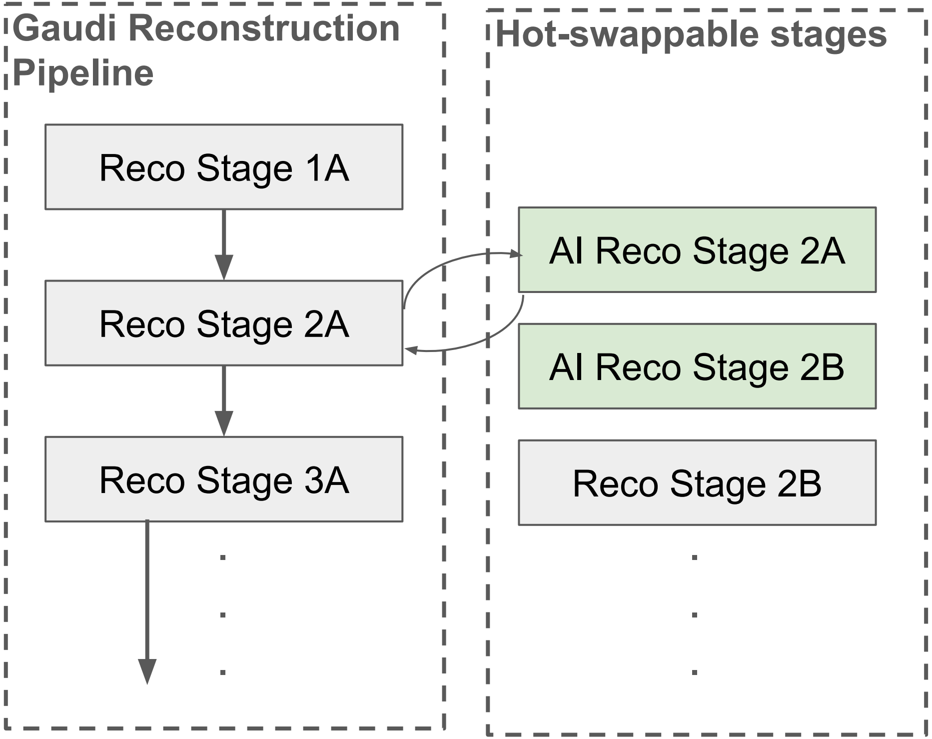
- PyTorch
  - Provides a standard base abstraction classes for many model types (ex. `nn.Module`)
  - Enforces a consistent model interface (ex. `forward` method)
- ZenML
  - Encodes pipelines as composable, declarative units and orchestrates execution
  - Allows pipelines to grow by adding or reordering steps; easy to add new pipelines
  - Manages pipeline state, artifacts, and execution metadata outside user code
- Optuna
  - Isolates hyperparameter search code
  - Enables experimentation without modifying core implementations
  - Really a package for black box searching, by default uses Tree-Structured Parzen Estimator (TPE)



Simplified Example Pipeline for Training Models

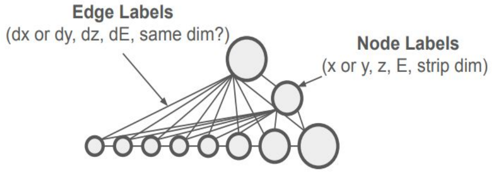
Task: Running Models in Simulation Framework

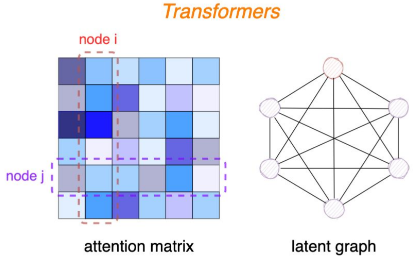
- **Goal:** Implement ML reco stages that are “hot-swappable” with traditional reco stages
  - Ideal scenario for benchmarking performance of ML models in terms of physics goals
- **Problem:** Models need to operate on batched data for performance, Gaudi framework designed for event by event reconstruction
  - How to optimally interface traditional stage → ML stage → traditional stage is unclear
  - ML stage speed performances must be on par with traditional reco speed performances

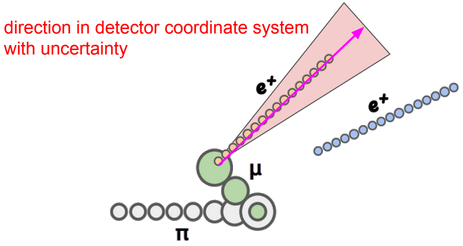


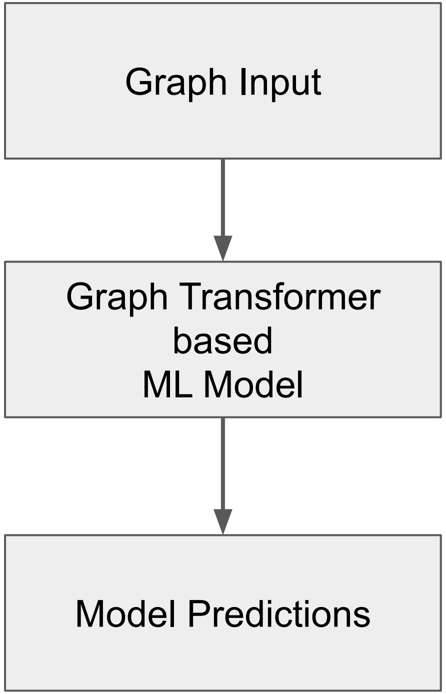
Task: Optimizing Accuracy of Models

- Use graph transformers for every task currently
  - These are among the most expressive AI models
    - Particularly effective for high-context, relational tasks (i.e. the ATAR reconstruction)
  - Computation expensive
    - Are they necessary for every task?
    - Can we achieve similar accuracy on some tasks with simpler models?
- Are there more ways we can give the model “hints” at relevant features to improve accuracy?
  - Similarly can we reduce computational expense by removing unneeded information?





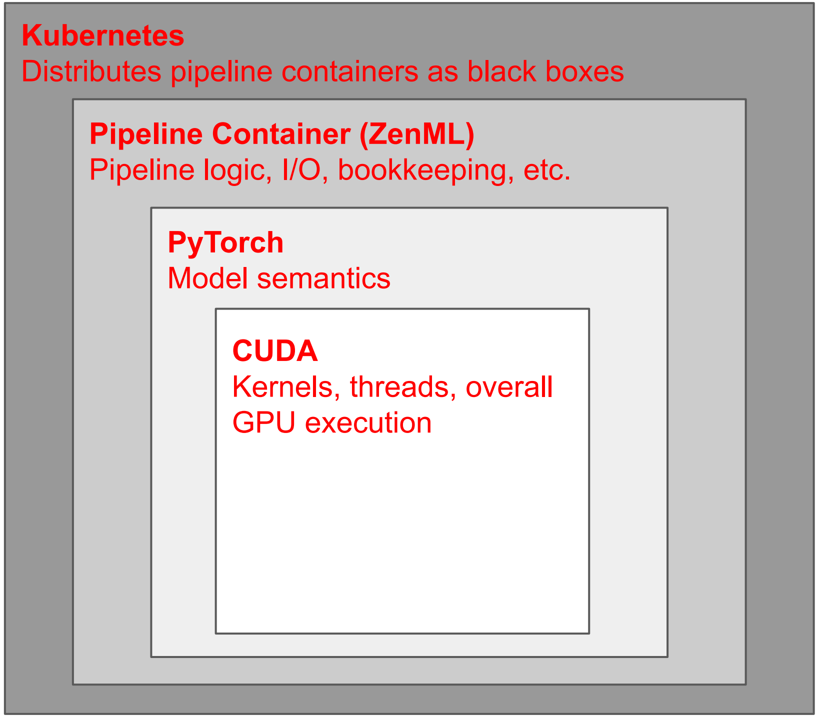




Mental model that every ML model in our pipeline currently follows

Task: Training on Multiple Compute Nodes

- **Goal:** be able to run training on arbitrary sized computing clusters
- **Idea:** Use Kubernetes to schedule Dockerized training pipelines orchestrated with ZenML
  - Designed the codebase to support it, but haven’t deployed the Kubernetes backend yet
    - What will be in each pod? How many resources for each pod?
  - Are other technologies useful for this task?
    - [PyTorch DDP?]



Simplified “Scope” Of Technologies

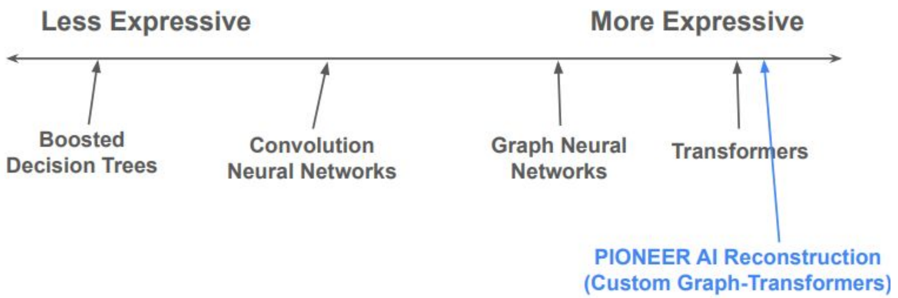
Outer Technologies Manage Inner Technologies

Auxiliary Slides

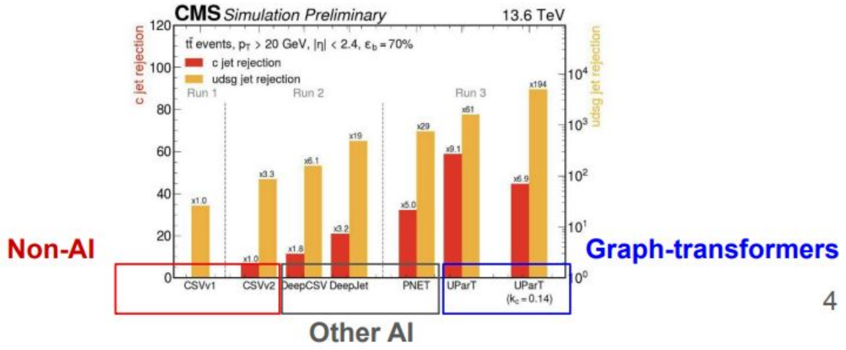
Why use Graph Transformers?

- These are among the most expressive modern ML models
    - Can solve a wider class of problems than most models
- [Fairly easy to construct with torch geometric]
- Attention based transformers allow models to learn how event information is related, as opposed to having to be more explicitly “told” in a traditional GNN approach

AI-based reconstruction:



- The UParT clustering model, built on a graph-transformer architecture, is the most successful CMS jet reconstruction algorithm to date (10.22323/1.476.0992)



Repositories

- PIONEER simulation (private, need access)
  - You can follow these instructions to get started in a docker container
  - DetReponse and shared branches with ML dataset generation code
    - Caveat: the docker container does not yet have Apache Arrow installed
      - This is needed for creating parquet files for training/running ML models
    - I typically create a static container then install Apache Arrow myself
- pioneerML (public)
  - WIP
    - use branch refactor/parquet-dataset-boundary for now
  - You can build a docker image with `./scripts/docker/build.sh`
  - You can run the built docker image with `./scripts/docker/run.sh --static --gpu -p 8888:8888`
  - Much of the codebase is “out of whack” right now, but it in principle contains tools to create data loaders, training pipelines, inference tests, etc.