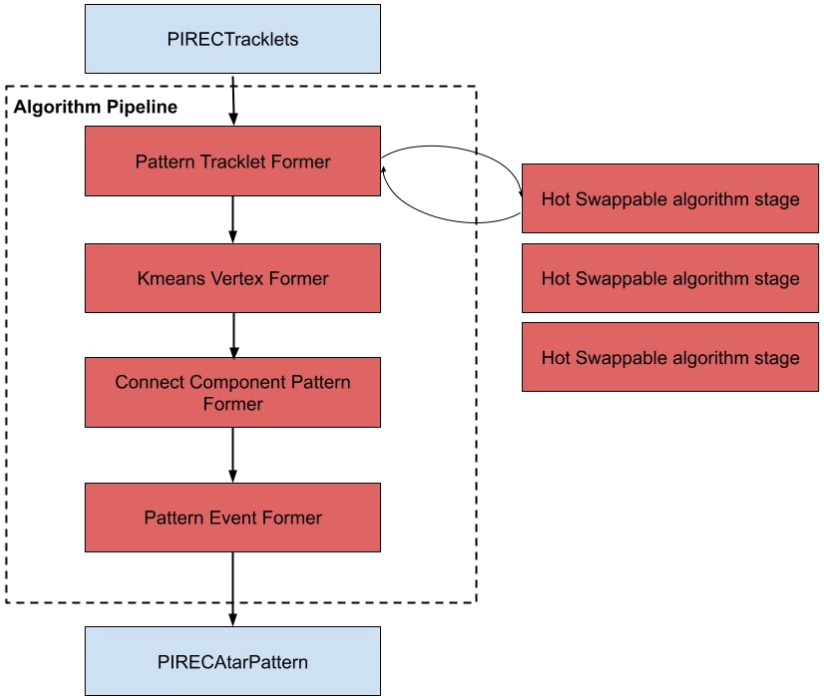
Pattern Finding Implementation

Jack Carlton
University of Kentucky

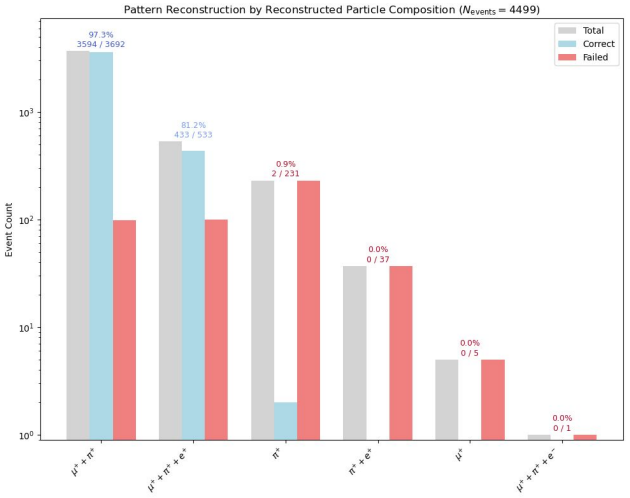
Pattern Finding Framework

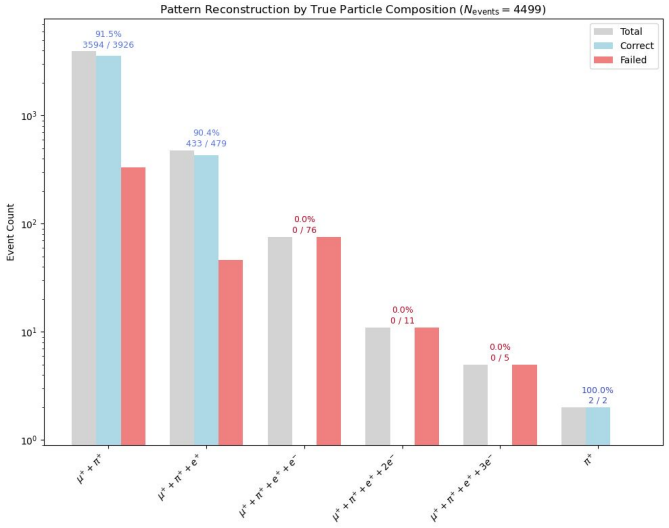
- Stand-alone C++ example
- Working implementation in simulation framework
- Based (loosely) off python pattern finding analysis playground
- Divides pattern finding into abstract algorithm steps
  - Arbitrary number of steps allowed
  - Can construct/edit multiple pipelines for different algorithm implementations
- “Re-invents the wheel”, pretty sure Gaudi framework can do exactly what I built

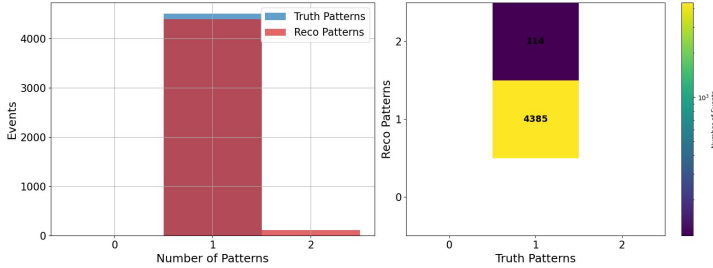


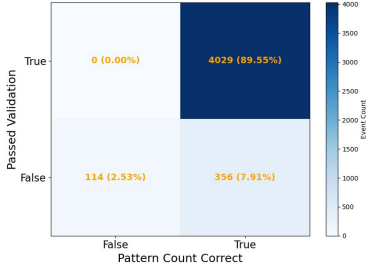
Example Reconstruction Accuracy

- Results purely from simulation framework
  - No more python pattern finding stage
  - Similar results to python pattern finding (expected)









Total Events: 4499

Passed Validation: 4029

Failed Validation: 470

Performance: 89.55%

Another Pattern Finding Approach

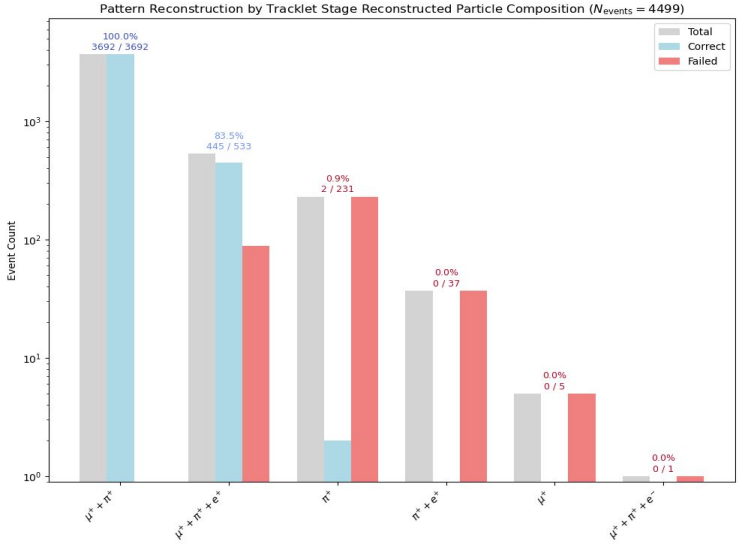
- Define “expected” vertex types
  - Ex: pi->e, pi->mu
- Each tracklet gets “scored” based on what vertex it belongs to
  - Ex. pi tracklet looks for e or mu tracklet to connect to, whichever is closer is scored higher
- Biased
  - Only finds vertices it’s “told” to look for
- Scoring comparisons need to be tuned
  - Inherently subjective

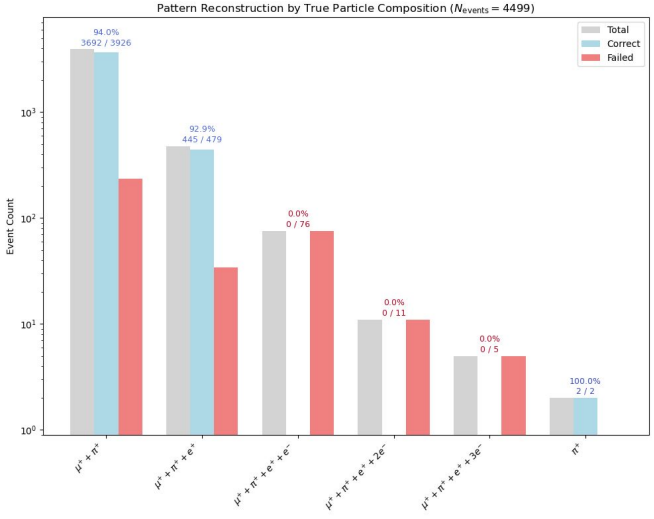
```python
# vertex_types.py

from algorithms.vertex.vertex_types.vertex_type import VertexType
from algorithms.vertex.vertex_types.scoring.distance_scorer import DistanceScorer


class PionMuonVertex(VertexType):
    def __init__(self):
        super().__init__(
            id="pi+_to_mu+",
            input_particles={211},  # pi+
            output_particles={-13},  # mu+
            scorer=DistanceScorer()
        )


class PionPositronVertex(VertexType):
    def __init__(self):
        super().__init__(
            id="pi+_to_e+",
            input_particles={211},  # pi+
            output_particles={-11},  # e+
            scorer=DistanceScorer()
        )


class MuonPositronVertex(VertexType):
    def __init__(self):
        super().__init__(
            id="mu+_to_e+",
            input_particles={-13},  # mu+
            output_particles={-11},  # e+
            scorer=DistanceScorer()
        )
```

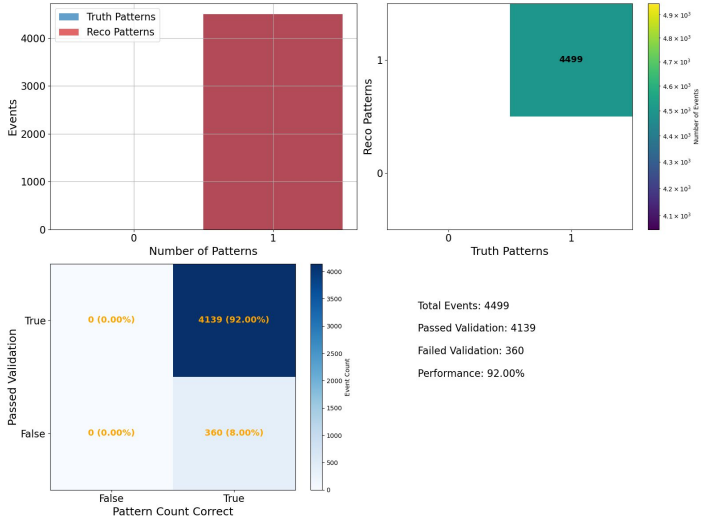
Example list of vertex types

Example Reconstruction Accuracy (New Approach)

- Performs better
  - Reverse engineered, so it “cheats”
  - Inaccuracy solely due to Tracklet Finding inaccuracy







Potential Issues with New Approach

Some remaining puzzles if we choose to go this route:

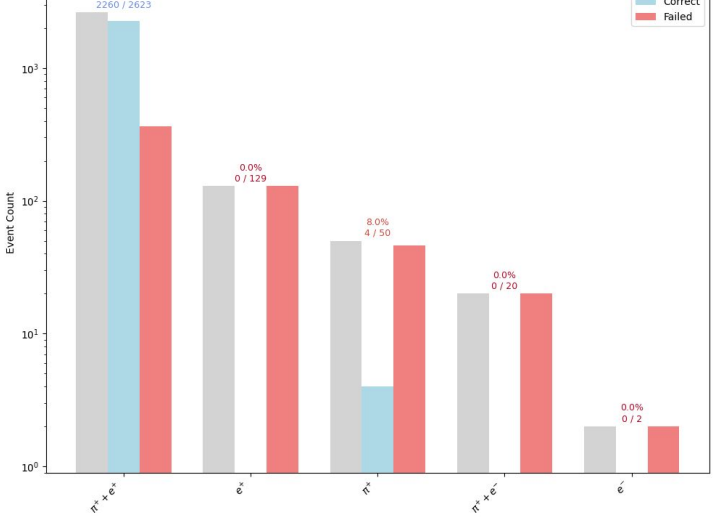
- What additional vertex types to program in?

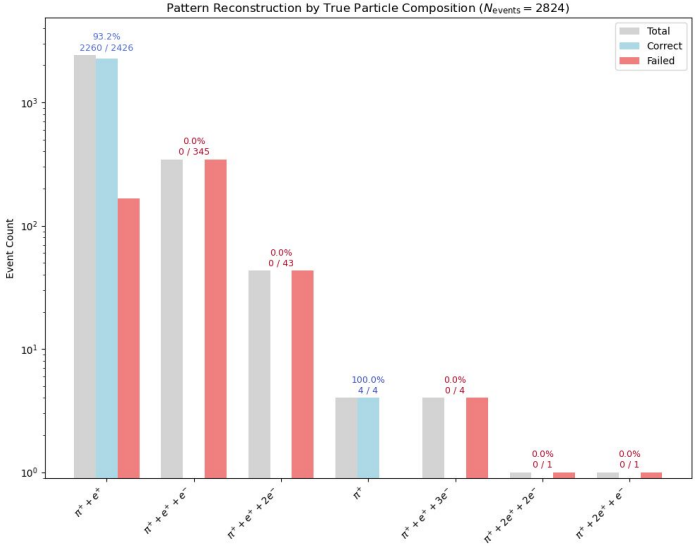
- How to tune the "scoring" system so it outputs the correct results?

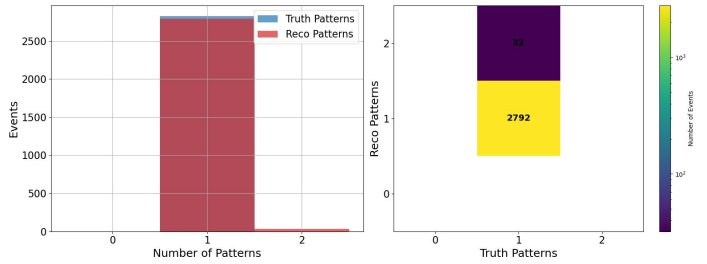
- Event mixing has the issue where tracklets could be incorrectly attached to other tracklets. The details of this are left ot be figured out by the scoring system.

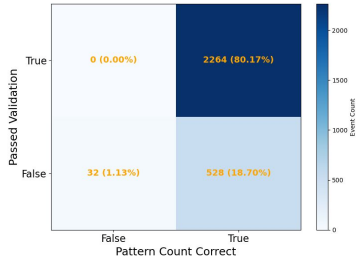
Auxiliary Slides

Pi -> e (simulation truth vs simulation reco)









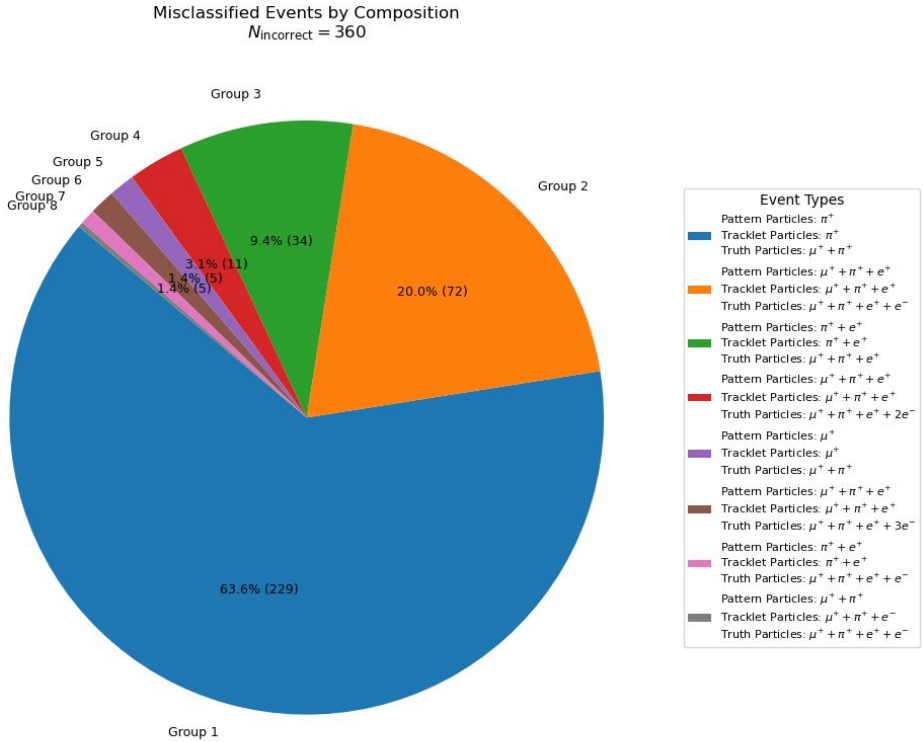
Total Events: 2824

Passed Validation: 2264

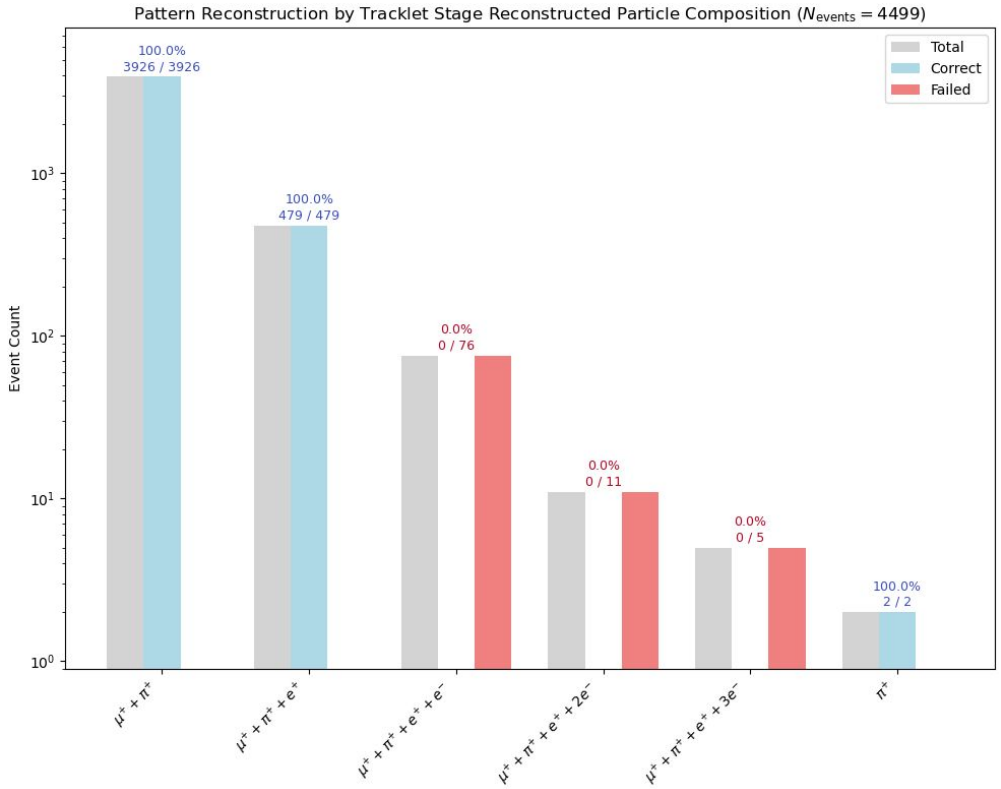
Failed Validation: 560

Performance: 80.17%

New Approach Misclassified events; caused by tracklet stage inaccuracies



New approach pi->mu->e using truth tracklets



New approach pi->e using truth tracklets