High Pileup Particle Tracking
with
Learned Clustering

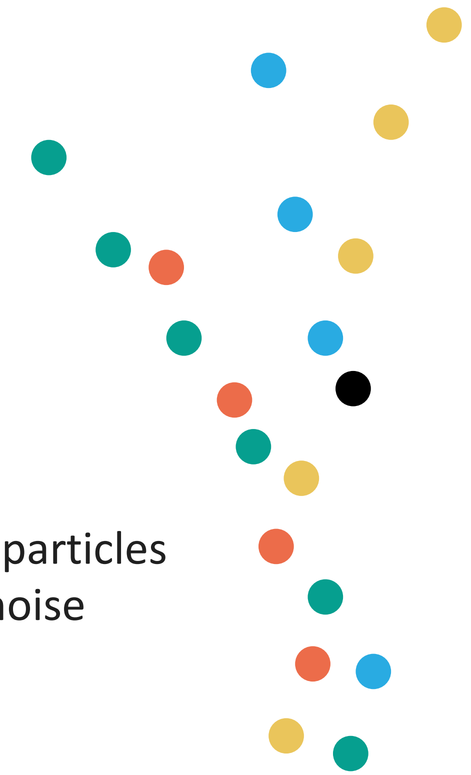

Kilian Lieret
(Princeton)

Gage DeZoort
(Princeton)

2024 fellow: **Aryaman Jeendgar** (BITS Pilani)
2023 summer students: **Jian Park** (Chicago), **Devdoot Chatterjee** (Delhi Tech U)
Transformer exploration: **Siqi Miao** (Georgia Tech), **Pan Li** (Georgia Tech)
Feedback & input: **Javier Duarte** (UCSD), **Savannah Thais** (Columbia)
Liaisons to CMS LST tracking: **Jonathan Guiang** (UCSD), **Philip Chang** (Florida)

One-shot tracking with learned clustering/object condensation

Hits (point cloud)



Colors = particles
Black = noise

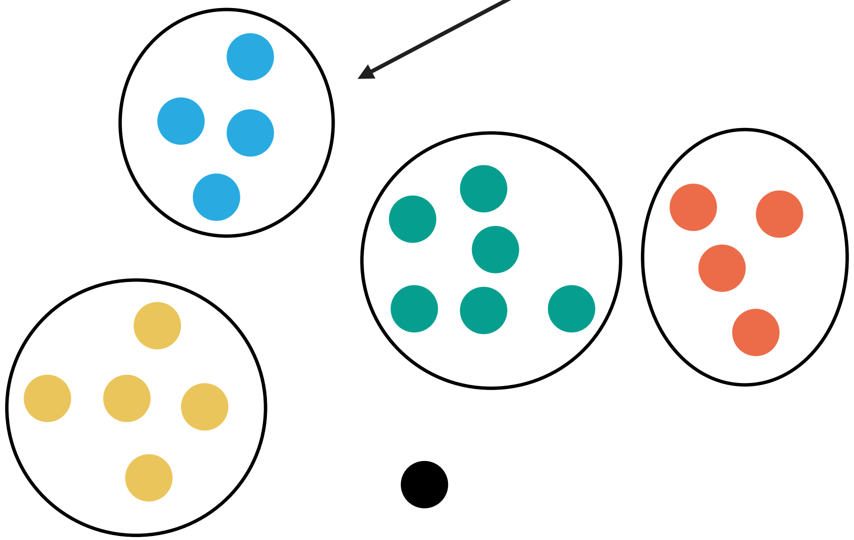
ML model

GNN, Transformer

Trained with
Repulsive & **attractive**
loss functions

Learnt latent space
Hits clustered by particle

Circle = reco track

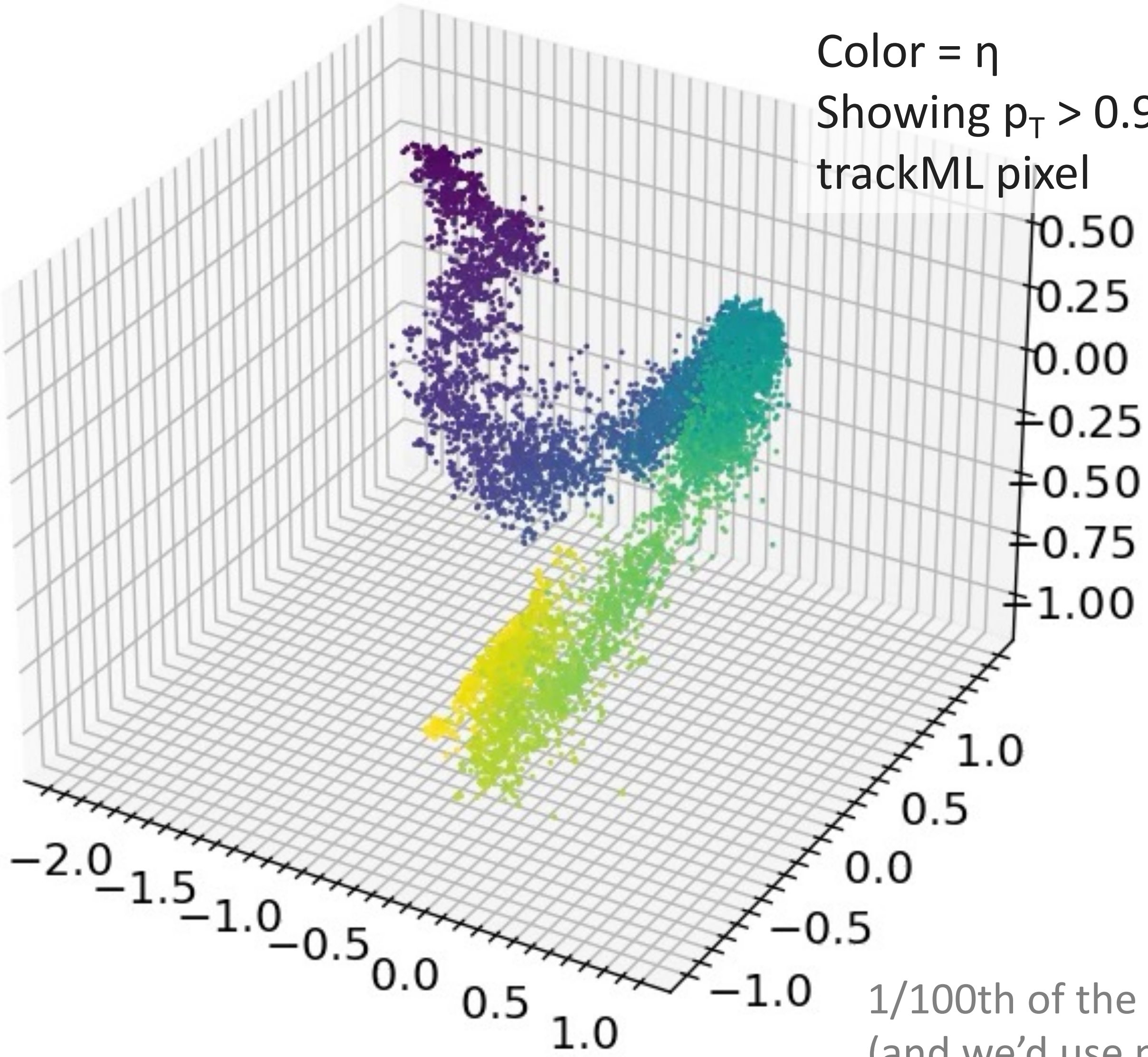


Hit features: coordinates + cluster shapes

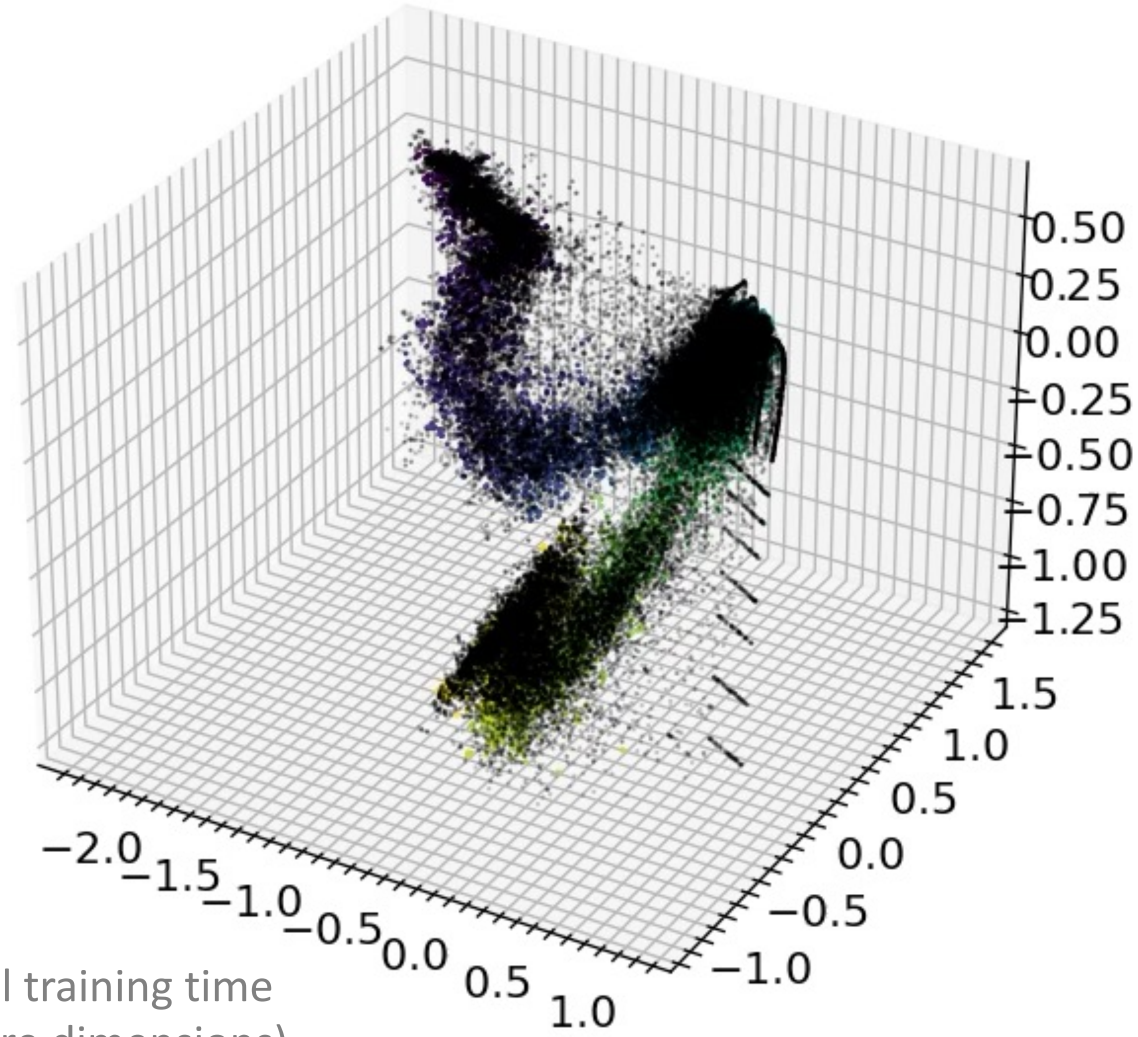
Training learned clustering

If you can’t see the video, download the ppt file

Color = η
Showing $p_T > 0.9$
trackML pixel



With noise and $p_T < 0.9$ GeV hits in black



1/100th of the full training time
(and we’d use more dimensions)

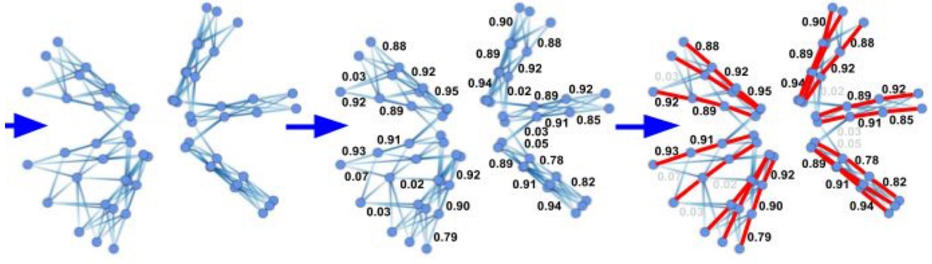
The high-pileup tracking zoo

Examples of classical algorithms (potentially with ML support)

- **Combinatorial Kalman Filter:** Extrapolate & refine “seeded” tracks
- **LST tracking:** Iteratively combine track segments (see talk by M. Vourliotis)

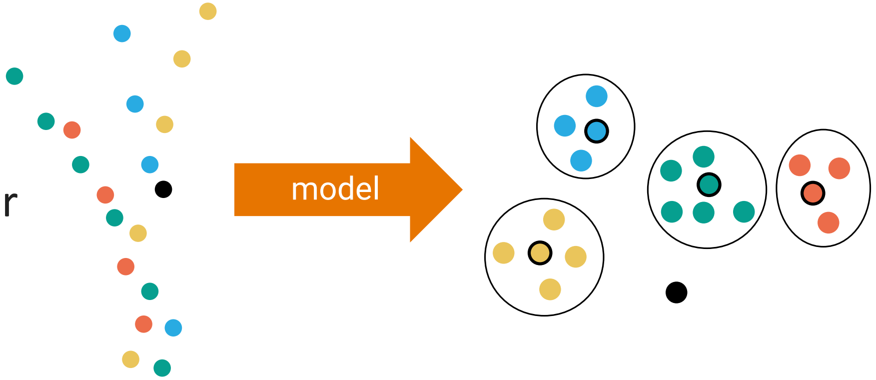
Example of ML algorithms

- **ExaTrx:**
  - Build graph out of initial hits → **edge classification (EC)**
    → tracks emerge as **connected components of final graph** (simplified)
  - Graph is used for both message passing and as track representation
  - Any incorrectly pruned edge cannot be restored
  - See D. Murnane’s talk tomorrow



(a) *Graph Construction* (b) *Edge Classification* (c) *Track Construction*

- **Learned Clustering (this talk):** Tracks emerge as clusters in a latent space
  - If using GNN: Similar graph building as ExaTrkx, but graph is used *only* for message passing: tracks are rendered on node-space
  - **Recursive Graph Attention Network:** Model with iterative graph construction; See talk by J. Chan today
- **Influencer (Murnane):** Hits gravitate to influencers representing tracks (CTD talk)



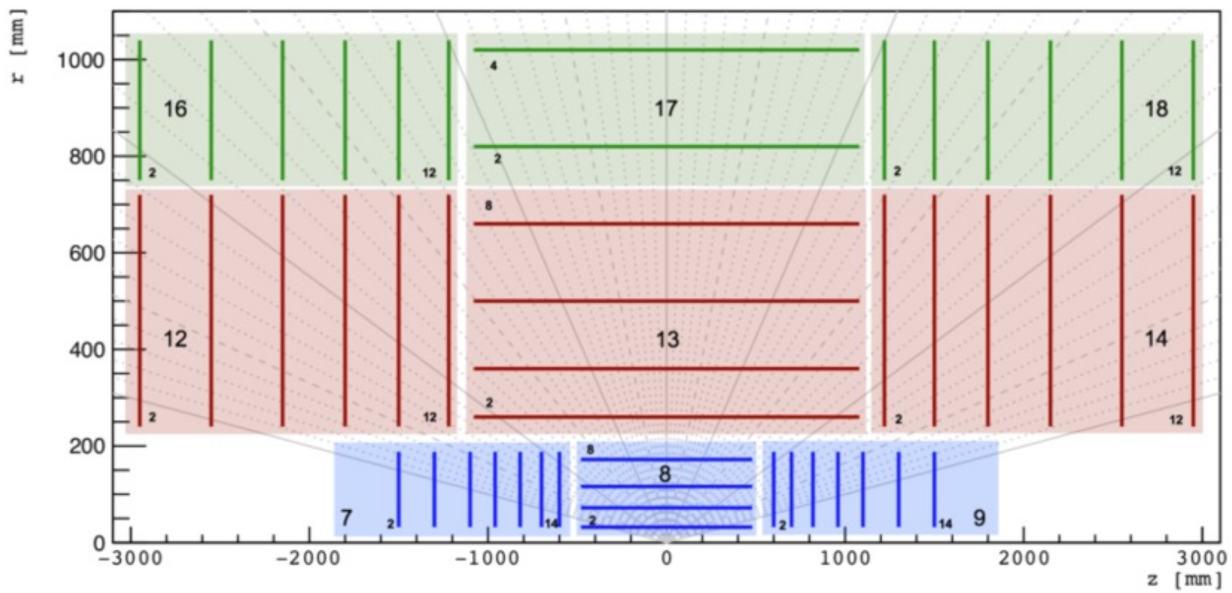
Object condensation tracking on trackML: Old & new

All results have so far been evaluated on the trackML dataset

- CTD22: **Proof of concept** on truth-cut pixel detector data
- CHEP23: **First results** on pixel layers (geometric graph construction) ([2309.16754](2309.16754))
- CTD23: Improved results using **learned graph construction** ([2312.03823](2312.03823))

This talk:

1. Simplified and improved **loss functions**
2. **Improved results on pixel detector**
3. First results on **full detector** (pixel + strip layers)
4. **Ongoing work** & shoutouts

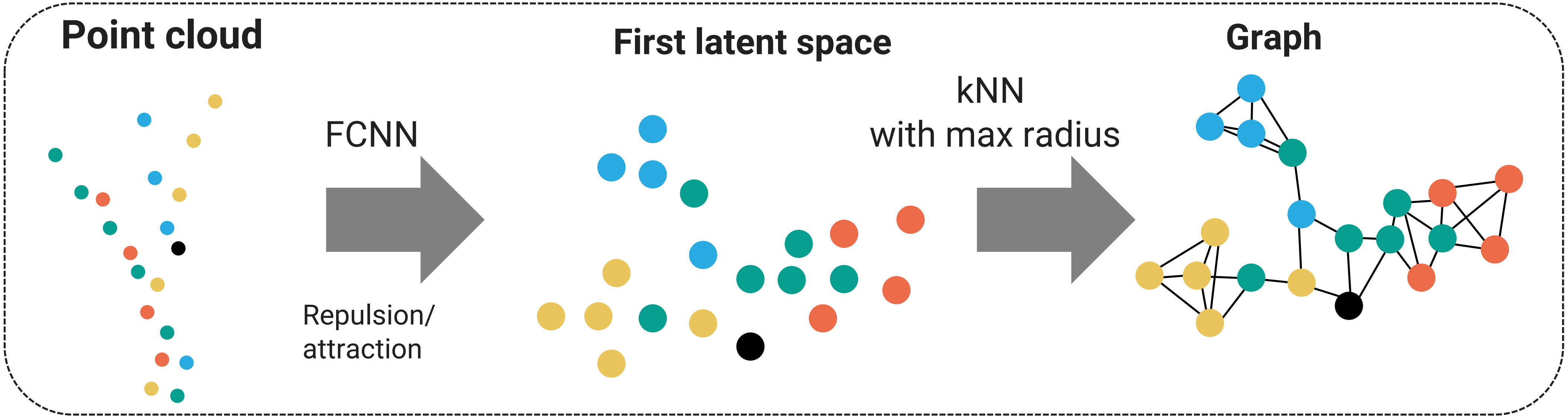


The trackML detector

GNN Pipeline

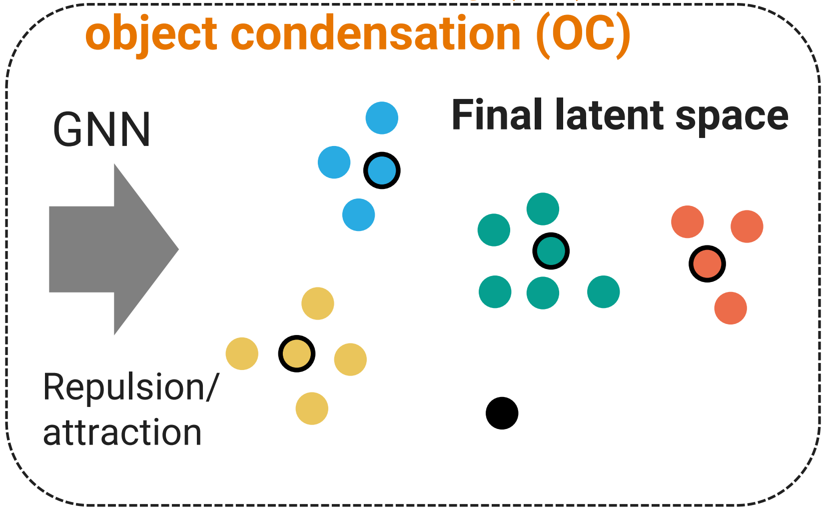
Graph construction (GC) with learned clustering

GC heavily inspired by ExaTrkx

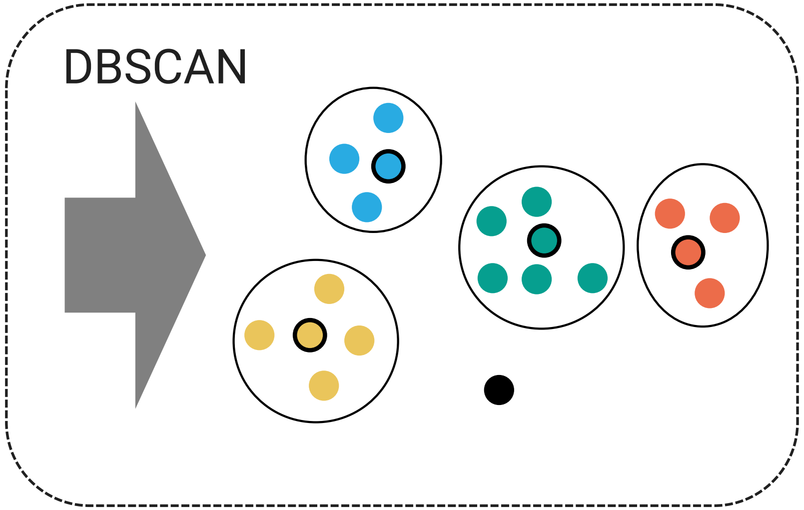


Major difference: Currently also training to build edges that skip detector layers

Learned clustering (LC)/ object condensation (OC)



Collect clusters



- Quality of GC is crucial for GNN performance
- Higher k → more edges → better performance, slower runtime

Pipeline described in detail in [2312.03823]

Simpler loss functions

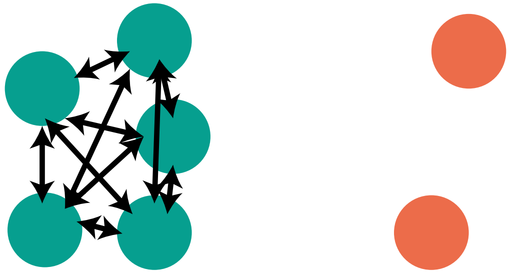
Our implementation is described in detail in 2312.03823

“Standard” learned clustering (LC) loss

Attractive loss function

quadratic potential $\|x_i - x_j\|^2$

Only hits of hinterest

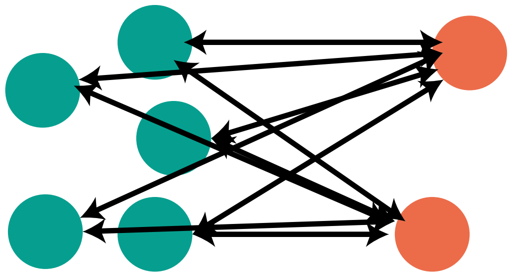


Repulsive loss function

quadratic hinge loss

At least one hit of interest

$\max(0, 1 - \|x_i - x_j\|^2)$



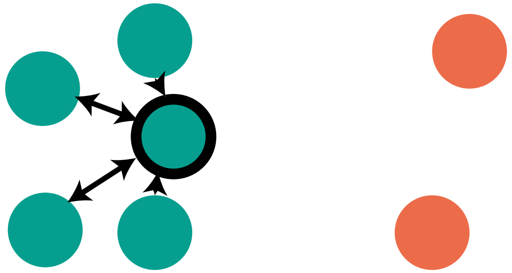
Previously only used for GC, now also for main model.

Combination & normalization scheme of loss functions (in particular rep.) matters

Object Condensation loss

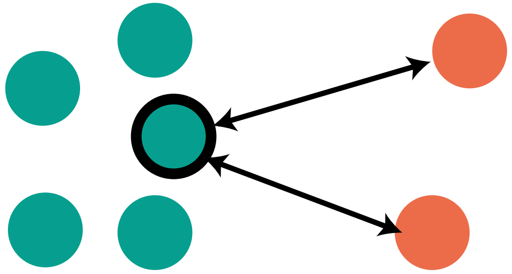
(Kieseler 2020): Learn a **condensation likelihood (CL)** for all hits; hit with highest CL per particle is **condensation point (CP)**

Very successful loss for calorimetry: CP can be used to infer properties of shower



Similar but only relative to CP

Attraction stronger if CP’s CL is high



Similar but only relative to CPs

repulsion stronger for strong CP CLs

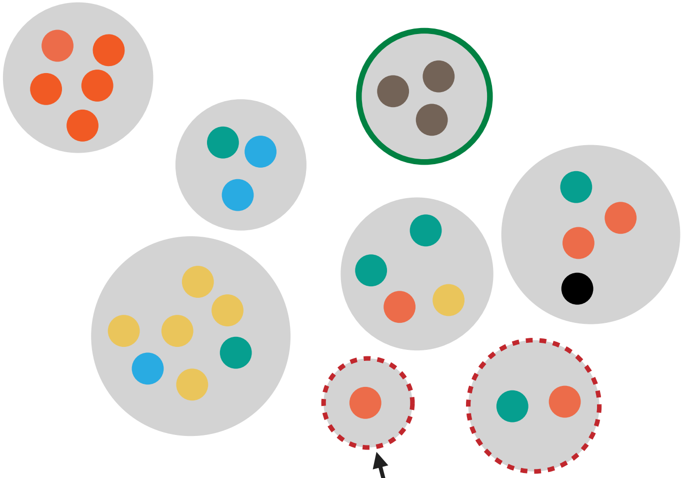
Previously used for main model (comparable results)

- Hard to implement memory & GPU efficiently
- Additional hyperparameters, loss functions, complexity

Metrics

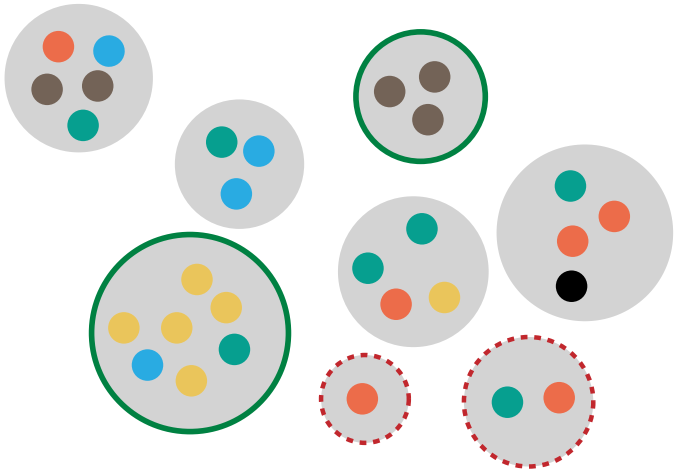
Perfect

Cluster contains only hits from one particle
and
no hits outside of cluster



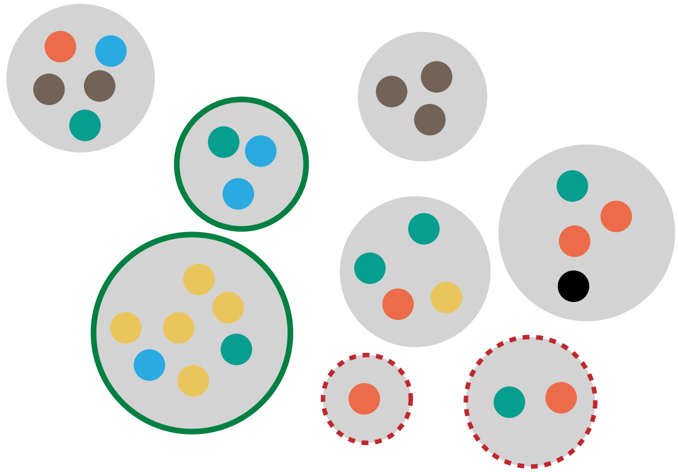
LHC

Cluster contains >= 75% hits from one particle



Double Majority (DM)

Cluster contains >= 50% hits from one particle
and
This particle has < 50% of its hits outside



Clusters with < 3 hits or non-reconstructable majority particle are discarded

#reconstructable particles

Perfect efficiency = **1/5**
Perfect fakes = **5/5**

#clusters with >= 3 hits & majority particle reconstructable

LHC efficiency = **2/5**
LHC fakes = **4/6**

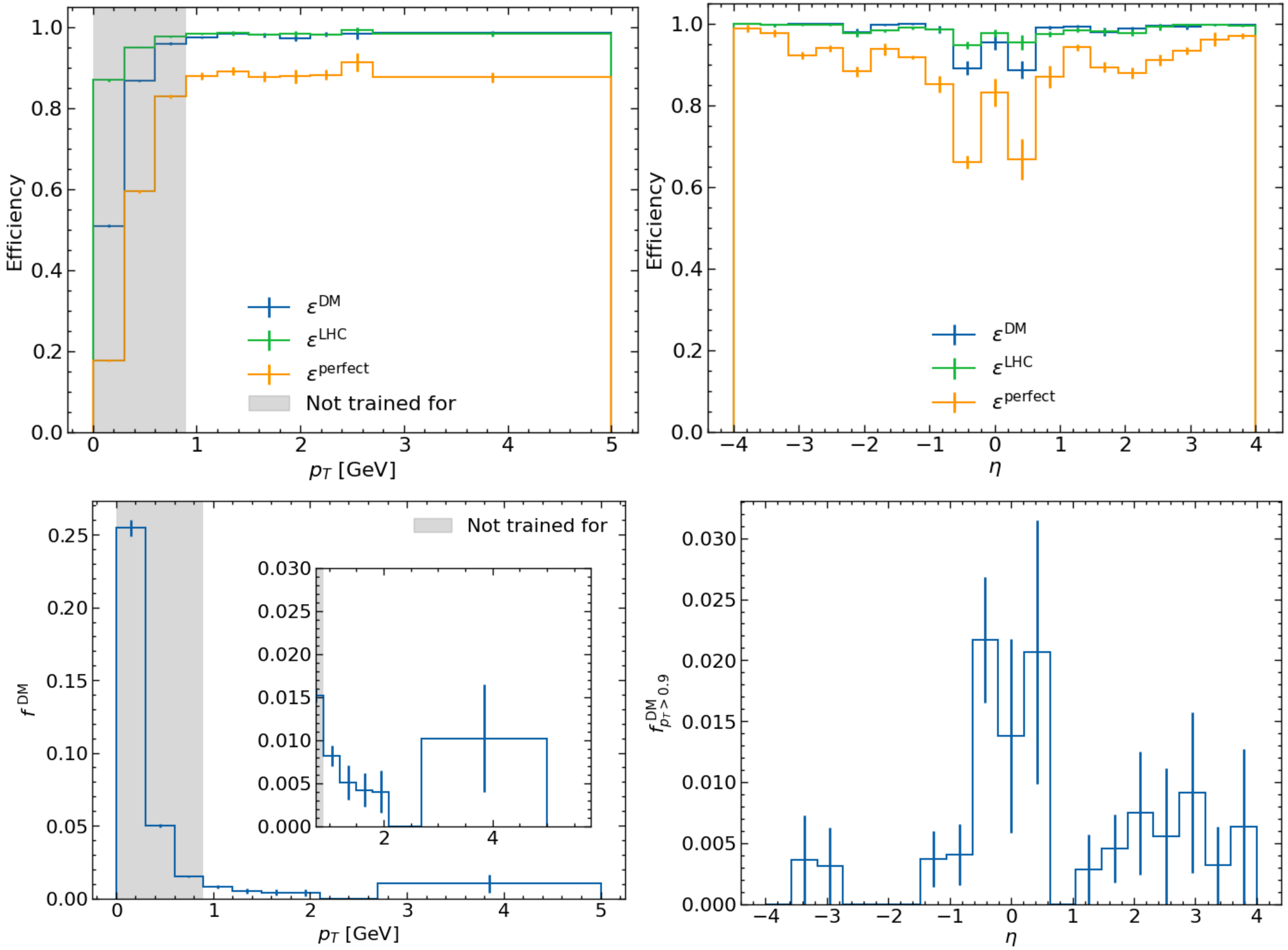
#reconstructable particles

DM efficiency = **2/5**
DM fakes = **4/5**

We also evaluate these **metrics at p_T thresholds**: p_T cut is applied to majority particle of cluster or particle (this is not a truth cut on the data, but simply a efficiency vs p_T study)

Reconstructable: >= 3 hits

Latest results on pixel detector



Model:

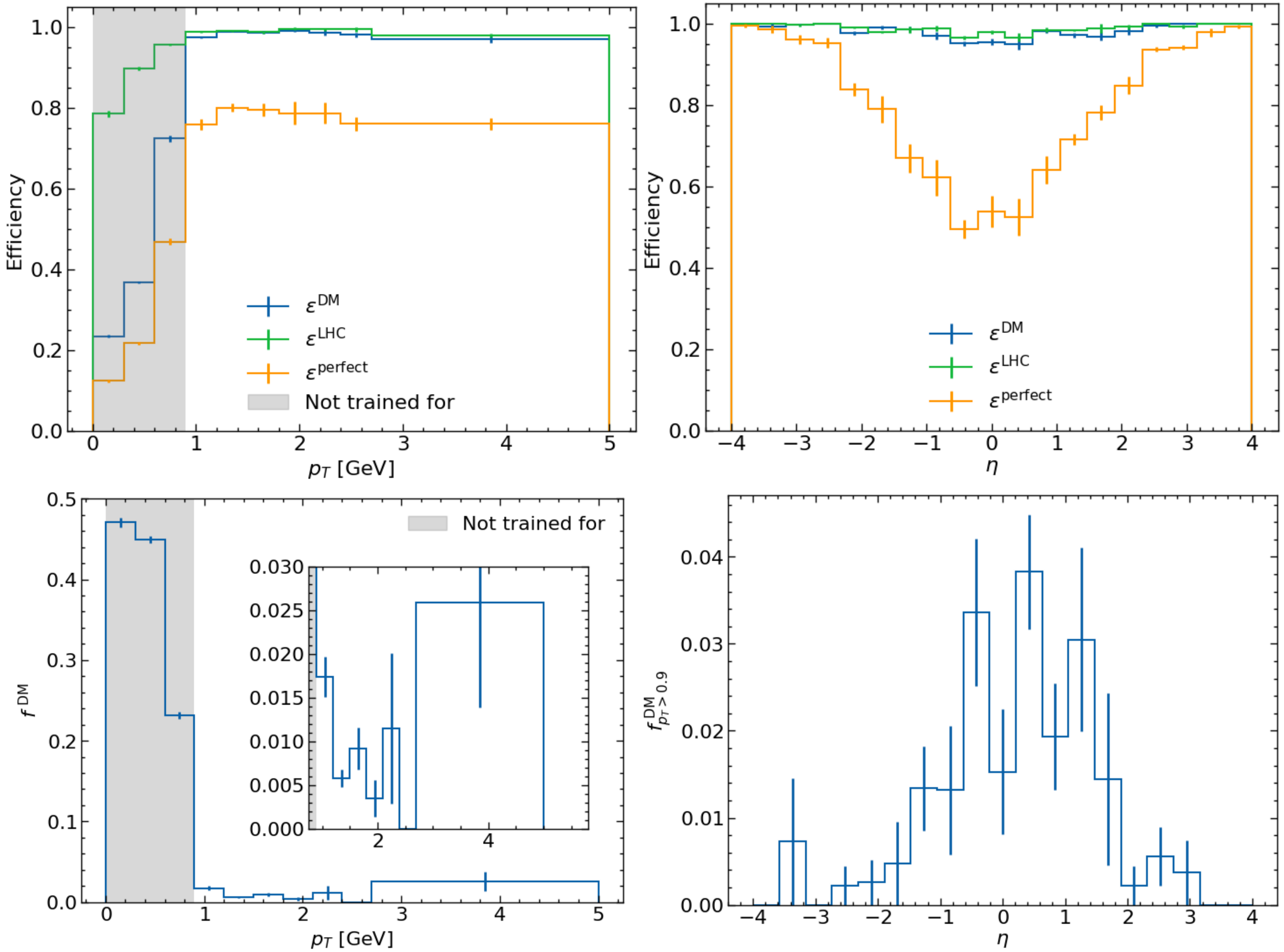
- 2.2M parameters (but no attempt of minimizing was made so far) in 4 layers of interaction networks (1612.00222)
- GC kNN k=17

Performance for p_T > 0.9 GeV

- **DM: 97.7%**
- **LHC: 98.2%**
- **Perfect: 88.1%**
- **Fake DM: 0.6%**

Training time ~30h (GC) + 60h (OC) on A100

First full detector results



Model:

- 2.6M parameters (but no attempt of minimizing was made so far) in 4 layers of interaction networks (1612.00222)
- GC kNN *k*=30

Performance for p_T > 0.9 GeV

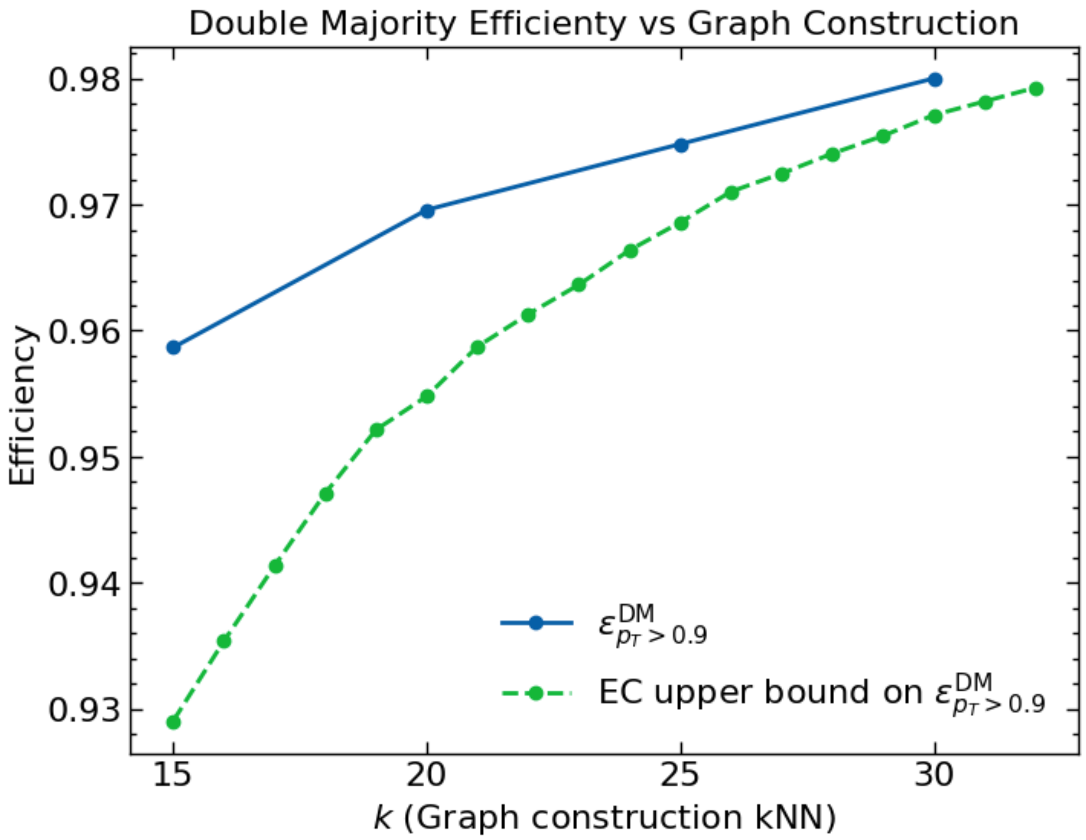
- **DM: 97.9%**
- **LHC: 98.7%**
- **Perfect: 77.3%**
- **Fake DM: 1.3%**

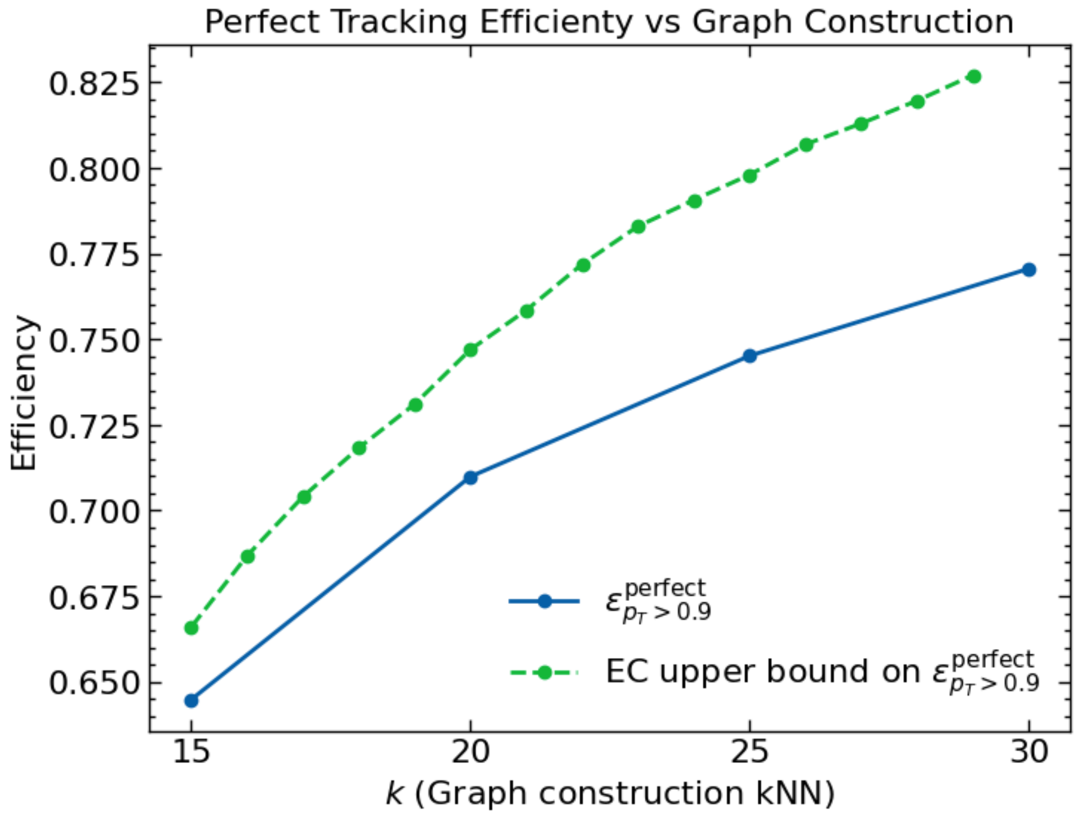
Training time ~30h (GC) + 50h (OC) on A100

First full detector results

Given our GC, LC outperforms any (!) EC pipeline.
We can ”join” tracks that are not well connected during GC!
Our performance is probably limited by GC right now.
→ Possible avenue: (Partially) remove skip-connections in GC in favor of higher *k*

However, we can still do better for *perfectly* reconstructing tracks (getting perfectly homogeneous clusters and not missing a single hit) – though that’s a very high bar, anyway





EC = Edge classifier. More details on EC upper bound: 2312.03823. Number of edges ~ k * 100.000

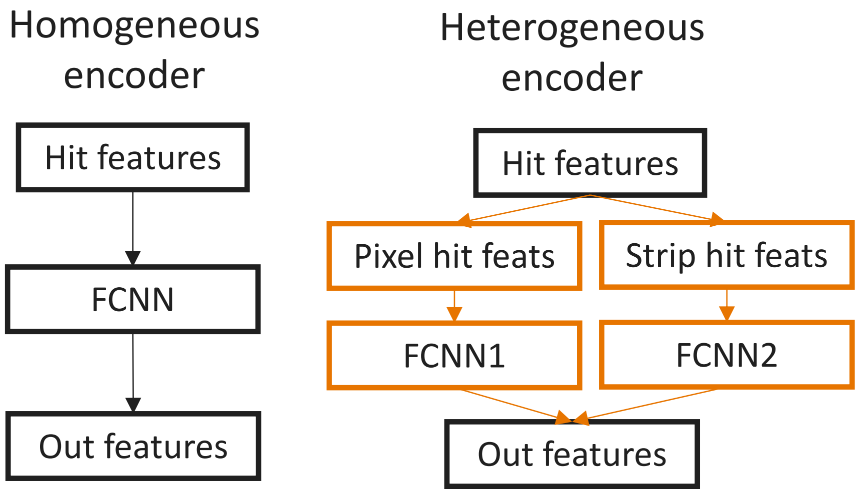
Other ongoing efforts

- Early **noise filtering**: Can we remove noise before we even build a graph?
  → Less nodes/edges, cleaner graphs
- However, false positives are very bad → add uncertainties to classification with **conformal scores** → only remove points if we're certain

Aryaman Jeendgar
(BITS Pilani)

Exploring **heterogeneous GNNs** and other tricks to deal with differences between pixel/strip detector

- Results from this talk use *exactly* the same model for pixel-only and for full-detector
- Preliminary studies with heterogeneous node encoders for pixel/strip showed no significant improvement
- Might also heavily depend on dataset



Completely different architecture: **efficient sparse transformers**

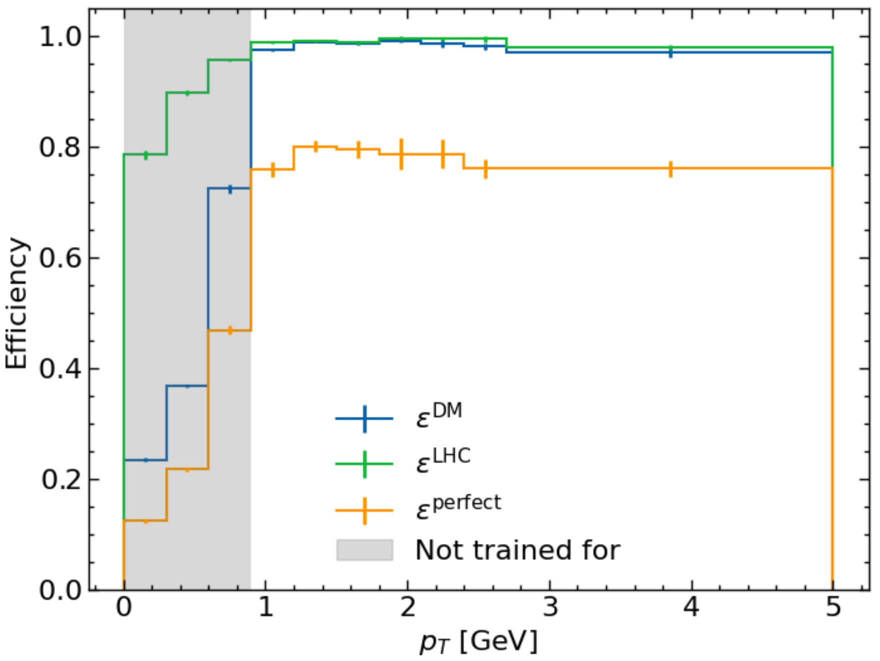
- Transformers can be faster and more GPU-efficient than GNNs
- Clustering metrics look good; currently working on evaluating tracking metrics (double majority eff. etc.)
- Read the paper: 2402.12535

Siqi Miao
(Georgia Tech)

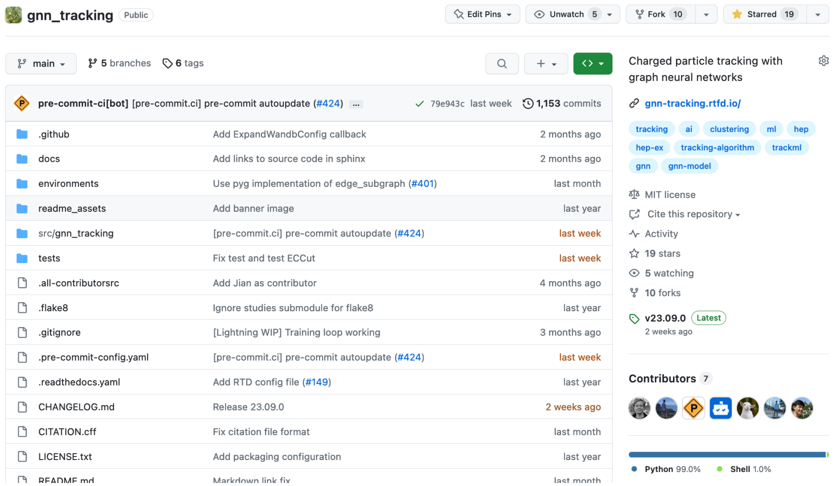
Pan Li
(Georgia Tech)

Summary & Outlook

- **Learned clustering/object condensation:**
  - Possible architecture for one-shot tracking with ML
  - Tracks are reconstructed as clusters of hits in a latent space
- Compared to previous work, we use a **simpler and more GPU efficient loss function** to train the GNN (same as the one used for GC; relatively standard embedding loss)
- Improved results on **pixel-only** trackML challenge: **97.7% DM, 88.1% perfect** (counting only p_T > 0.9 tracks)
- First results on **full detector** trackML challenge: **97.9% DM, 77.3% perfect** (counting only p_T > 0.9 tracks; expecting to still improve on these results significantly)
- **No p_T truth cuts** as in most proof-of-concept studies
- **Given our GC, we outperform any EC-based pipeline**: We can "join broken tracks" → Our GC is probably still lagging behind, though
- Challenges ahead:
  - Optimize & measure speed of reconstruction
  - Apply pipelines to simulations for real detectors (e.g., CMS phase 2)

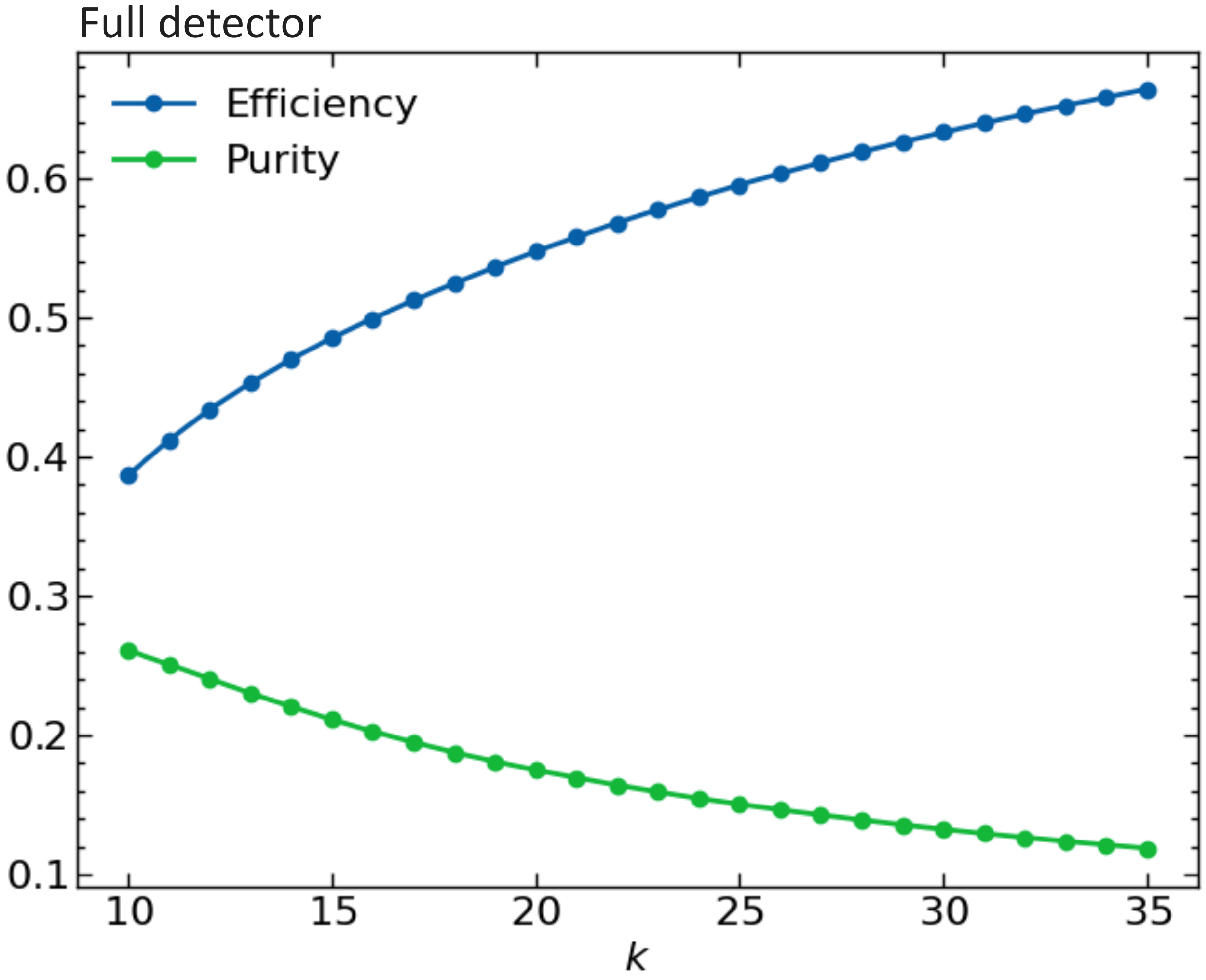


Completely public & documented on GitHub!
https://github.com/gnn-tracking/gnn_tracking

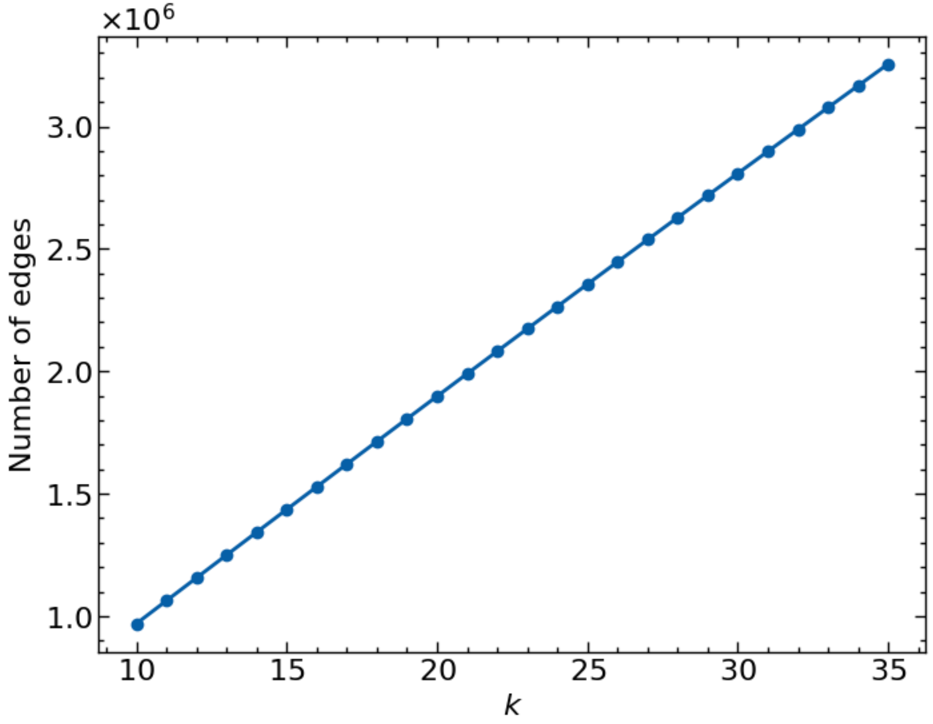


Backup

More on GC



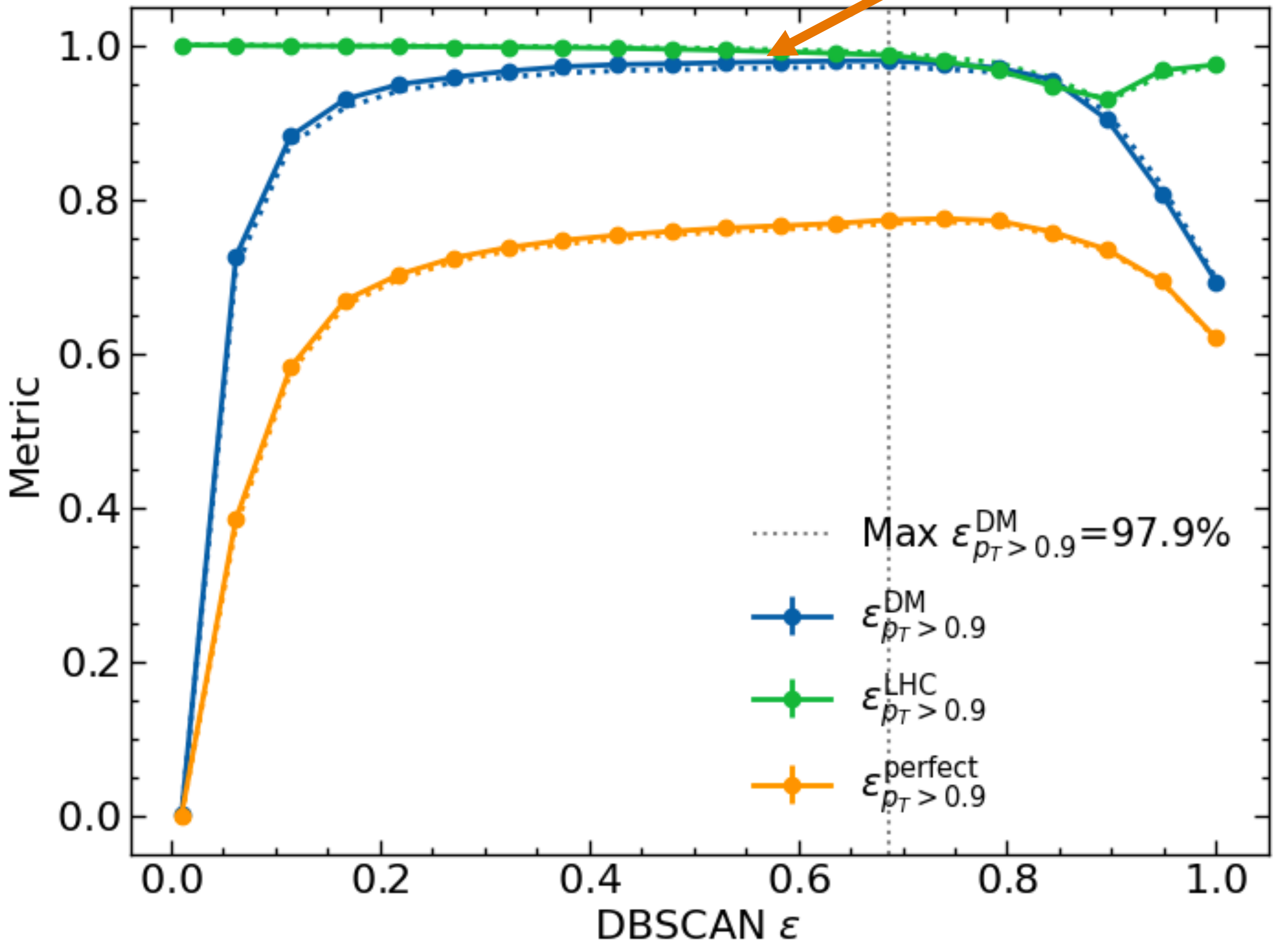
Important: Efficiency is with respect to *all* possible edges, *including skip-layer edges* (that's a lot of edges!)



DBSCAN

Nice broad plateau showing well defined clusters.

Full detector



Solid lines: DBSCAN k=3, dashed lines: k=4.