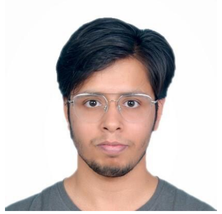
Object Condensation Tracking

Kilian Lieret
(Princeton)

Gage DeZoort
(Princeton)

Core team

Jian Park
(Chicago)



Devdoot Chatterjee
(Delhi Tech U)

Summer fellows

Siqi Miao
(Georgia Tech)

Pan Li
(Georgia Tech)

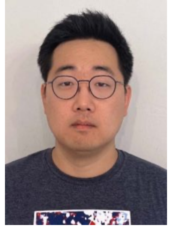
Transformer exploration

Javier Duarte
(UCSD)

Savannah Thais
(Columbia)

Feedback & input

Jonathan Guiang
(UCSD)



Philip Chang
(Florida)

Liaisons to CMS LST tracking

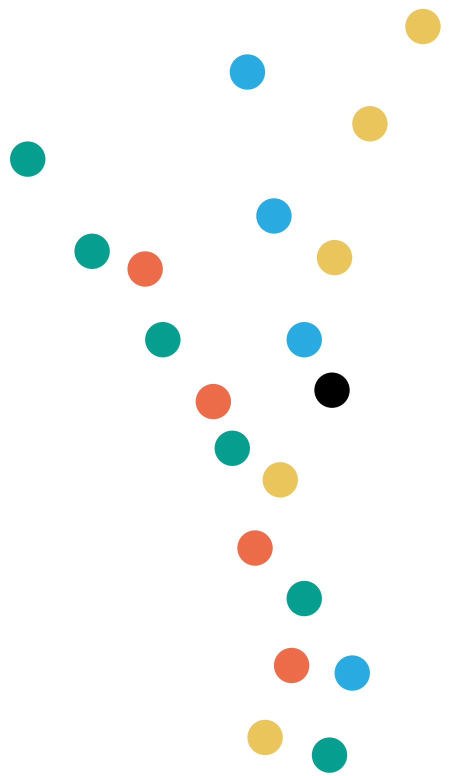
iris hep

PRINCETON UNIVERSITY

Vision: One-shot tracking with learned clustering

Hits

Hit coordinates +
cluster shapes



ML model

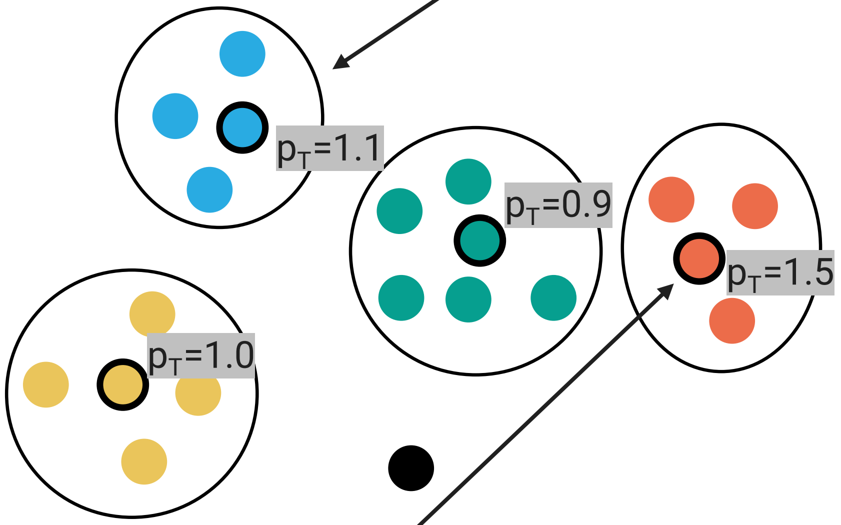
GNN or Transformer

Trained with
Repulsive & **attractive**
loss functions

Learnt latent space

Hits clustered by particle

Circle = reco track

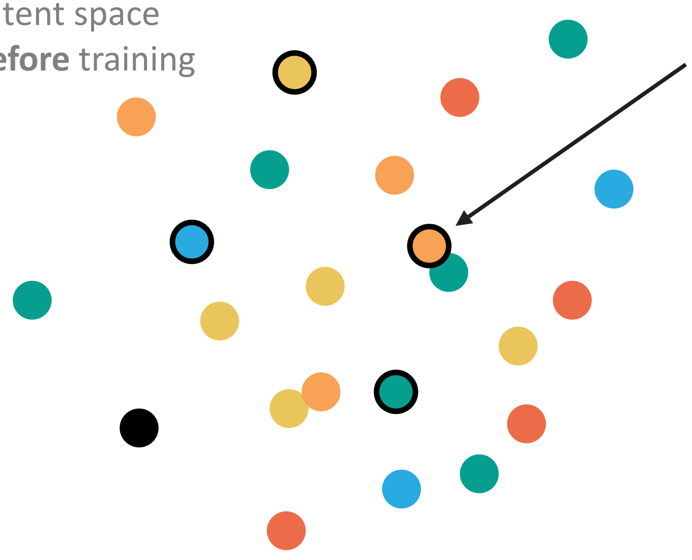


Condensation point =**influencer** in influencer approach
Represents the track, can learn track parameters like p_T (WIP for our approach)

No time resolution of points
⇨ Everything everywhere all at once

Object condensation: Training losses

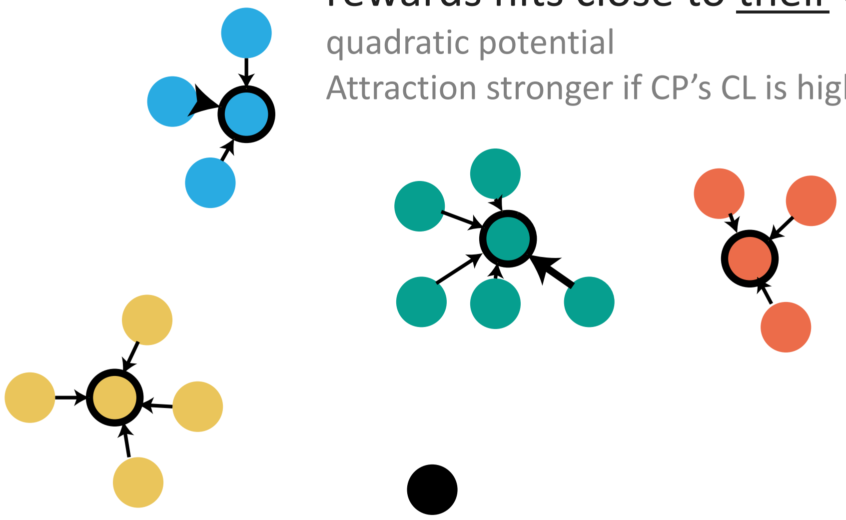
Latent space
before training



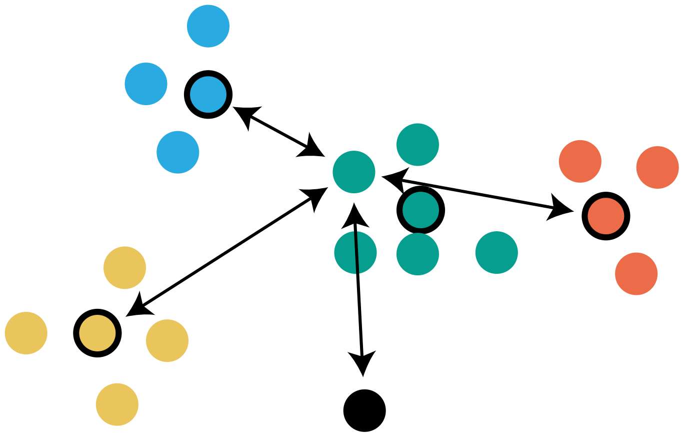
GNN predicts **condensation likelihoods (CL)** for every hit.
Hit with max CL for particle* is **condensation point (CP)**

*during inference: for cluster

Attractive loss function
rewards hits close to <u>their</u> CP
quadratic potential
Attraction stronger if CP's CL is high



Repulsive loss function
penalizes hits close to <u>other</u> CP
hinge loss: no more repulsion after certain distance
repulsion stronger for strong CP CLs



Background loss function
noise hits should have low CL

Loss functions implemented from Kieseler 2020 (2002.03605)

Detail: Multi-objective optimization

- OC comes with a lot of different loss functions (attractive, repulsive, background, track parameters)

- We currently use **linear scalarization**, i.e.,

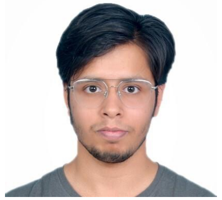
$$\mathcal{L} = \mathcal{L}_{\mathrm{attr.}} + w_{\mathrm{rep.}}\mathcal{L}_{\mathrm{rep.}} + w_{\mathrm{bkg.}}\mathcal{L}_{\mathrm{bkg}} + w_{\mathrm{param.}}\mathcal{L}_{\mathrm{param}}$$

- Tried a different method over the summer: **Modified Differential Multiplier Method (MDMM),** minimizing primary loss function relative to others subject to constraints

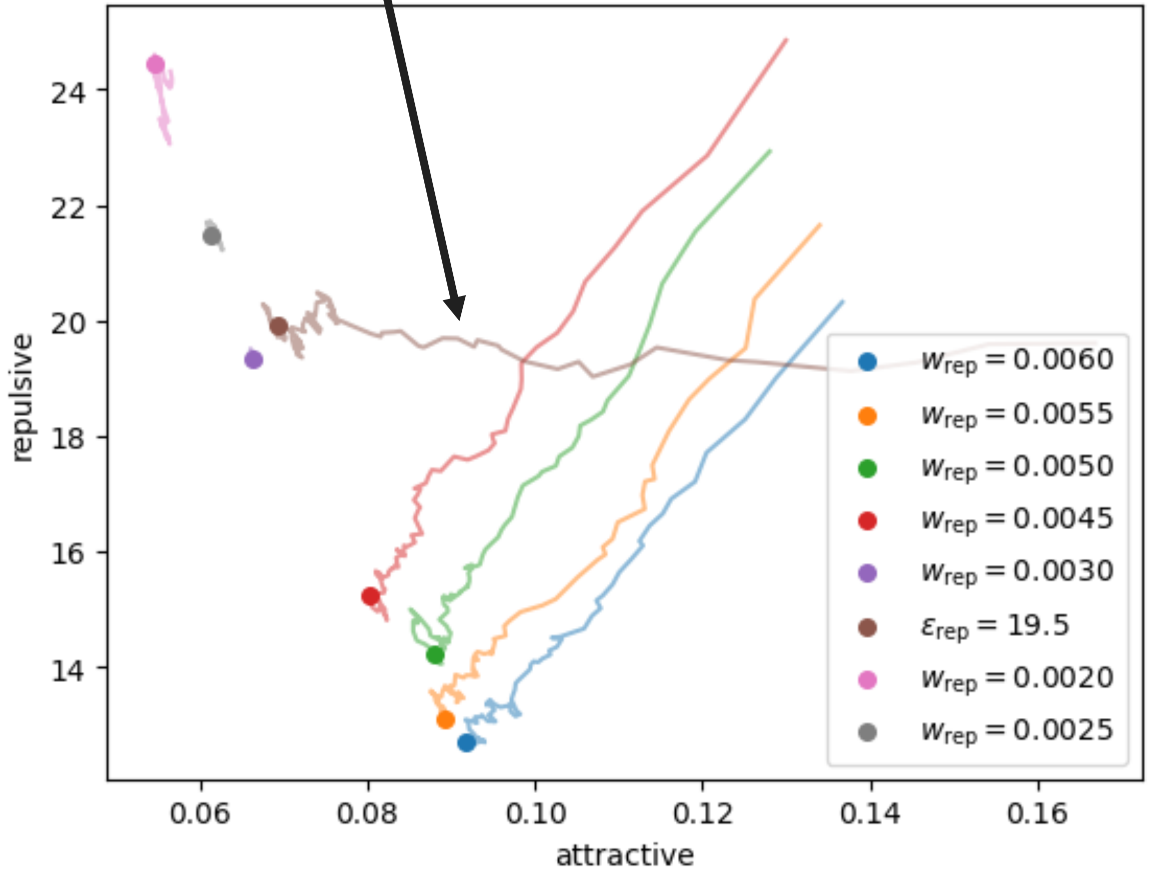
- Confirmed that linear scalarization converges nicely along a **convex pareto front** and generally gives same results as MDMM (and MDMM is more complex and comes with additional hyperparameters)

- Bottom line: Might take another look at MDMM once we zoom in on track param. prediction, currently overkill

Trained with MDMM, constraining repulsive loss to < 20 and minimizing attr. loss



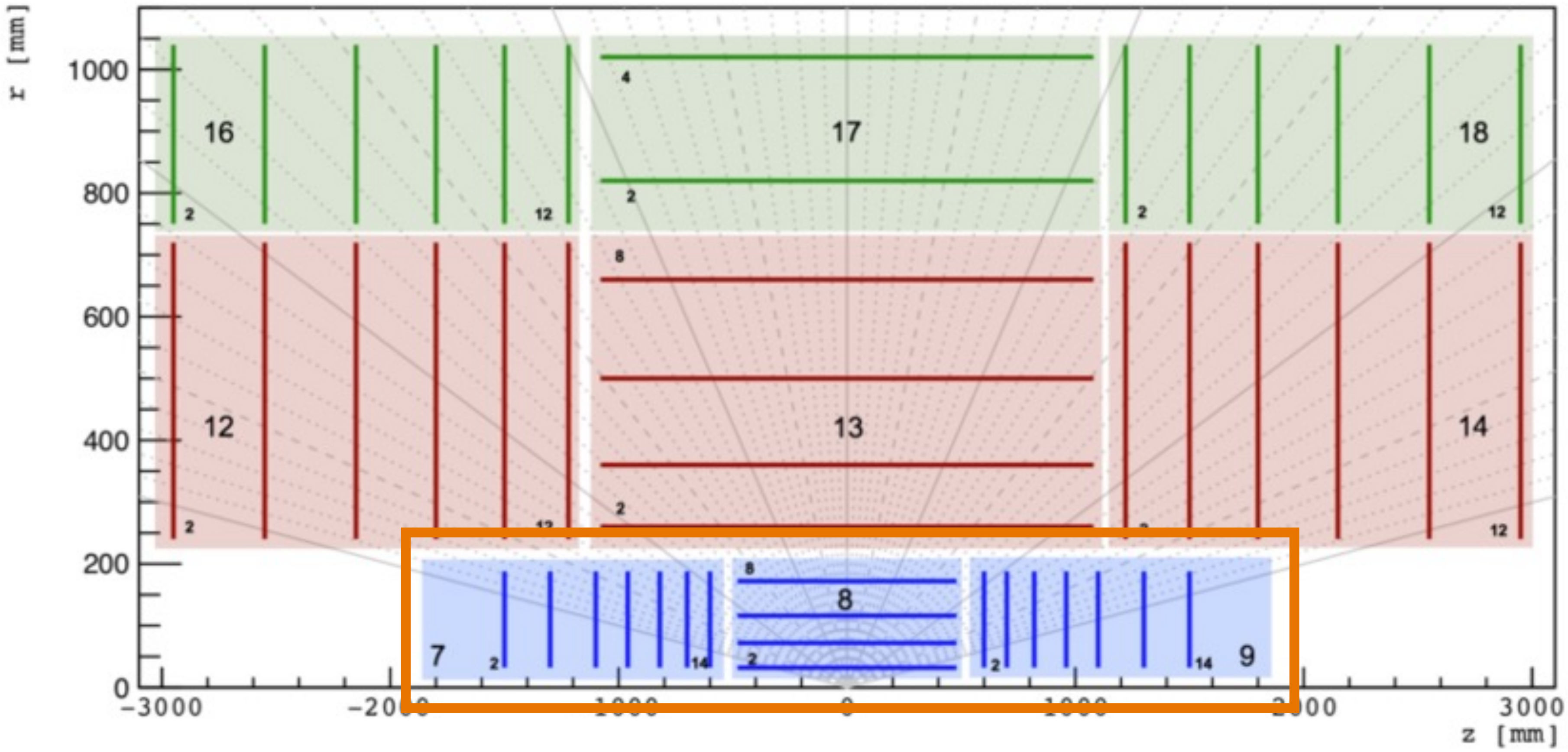
Devdoot Chatterjee
(Delhi Tech U)



[Original MDMM paper]
[Very nice blog post series]

Dataset

All results shown use the **pixel layers** of the **trackML dataset**



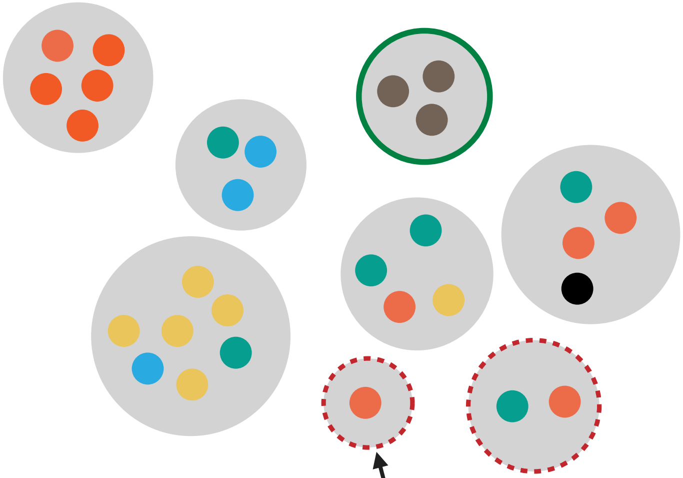
trackML dataset generated by ACTS

Input features: **Hit coordinates** + **cluster shapes**

Metrics

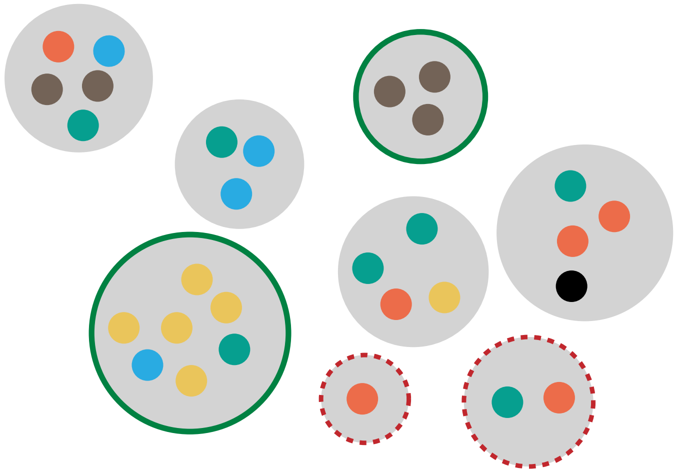
Perfect

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



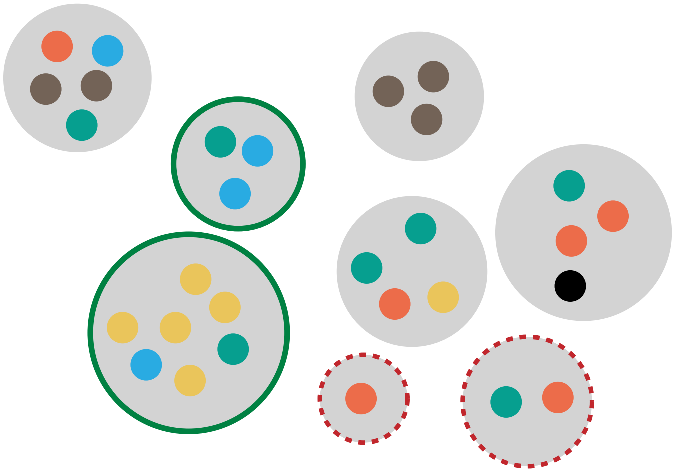
LHC

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

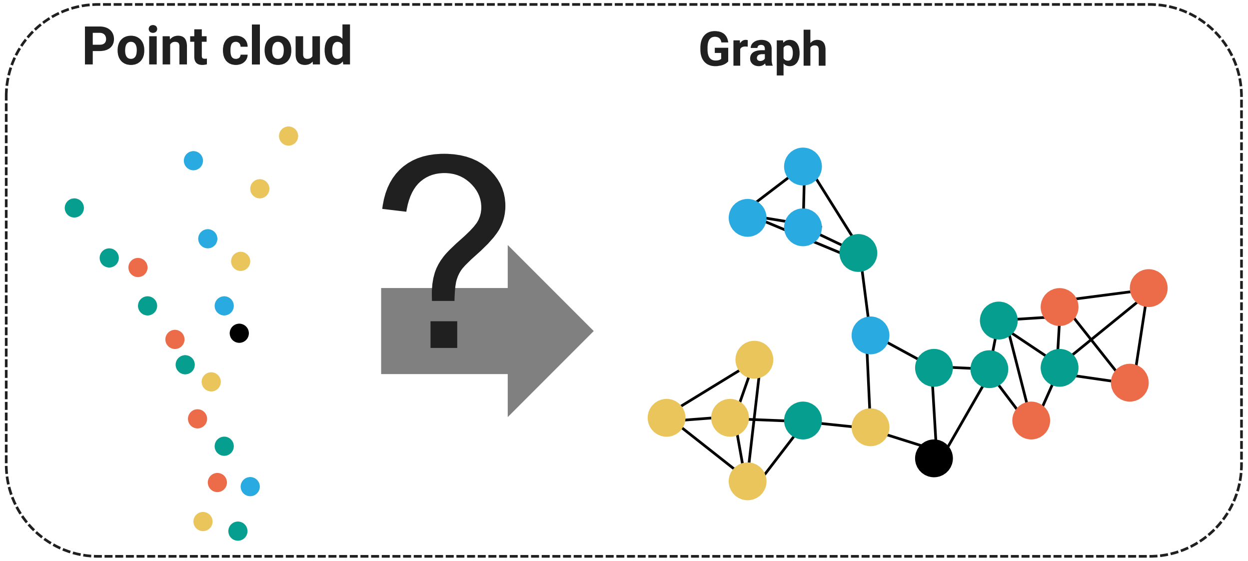
DM efficiency = **2/5**
DM fakes = **4/5**

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

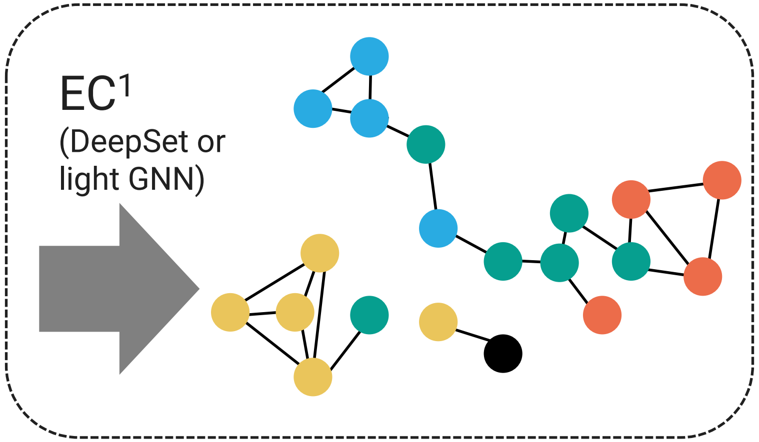
Reconstructable: >= 3 hits

General GNN pipeline

STAGE 1: Graph construction (GC)

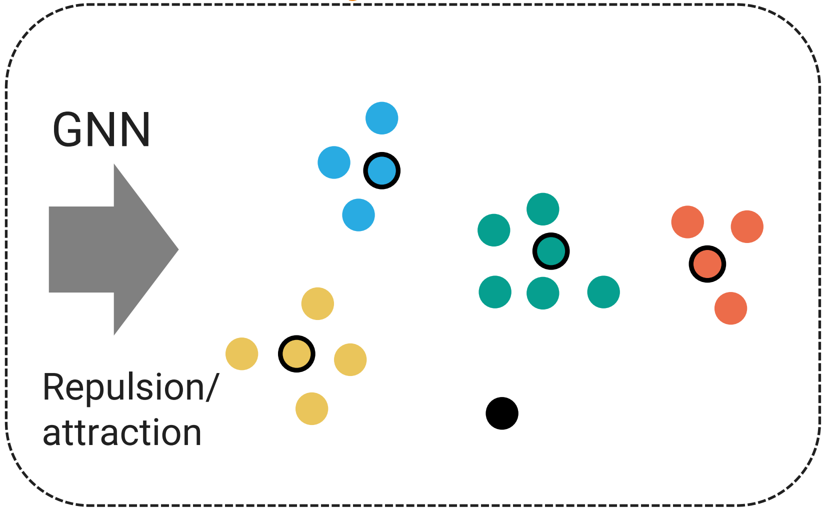


(Optional STAGE 1a: Graph refinement)



[1] Edge Classifier

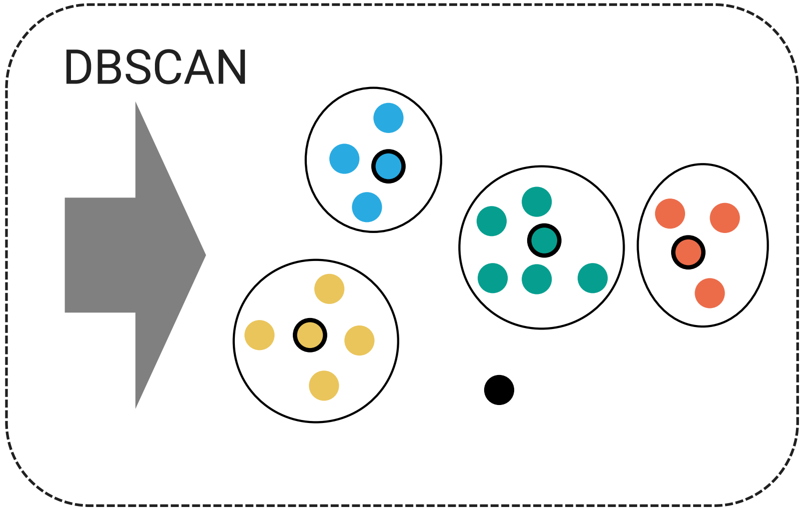
STAGE 2: Object condensation



(Optional STAGE 2a…)

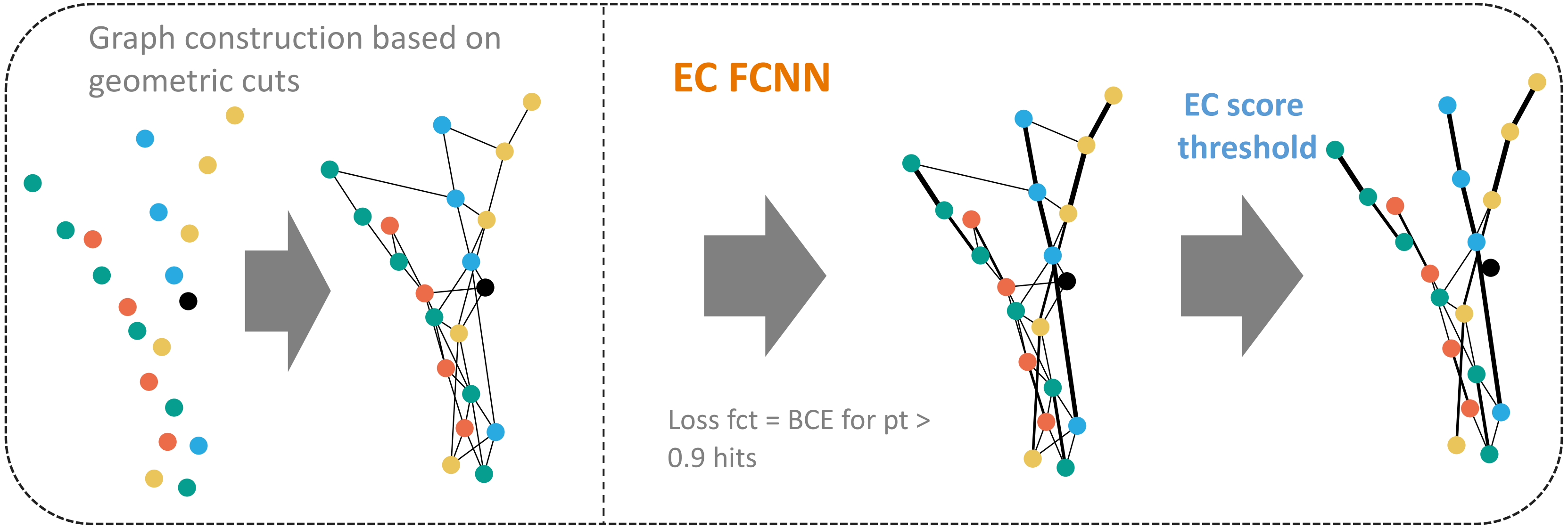
1. Build **new edges** based on OC latent space
2. Run **OC GNN** on new graph for even better OC latent space

STAGE 3: Collect clusters

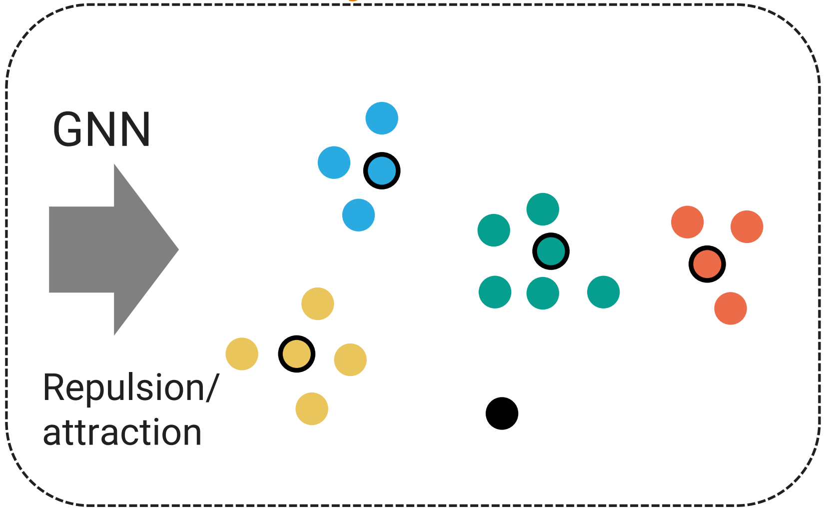


Pipeline 1.1 (@CHEP proceedings): Geometric GC + EC FCNN + OC GNN

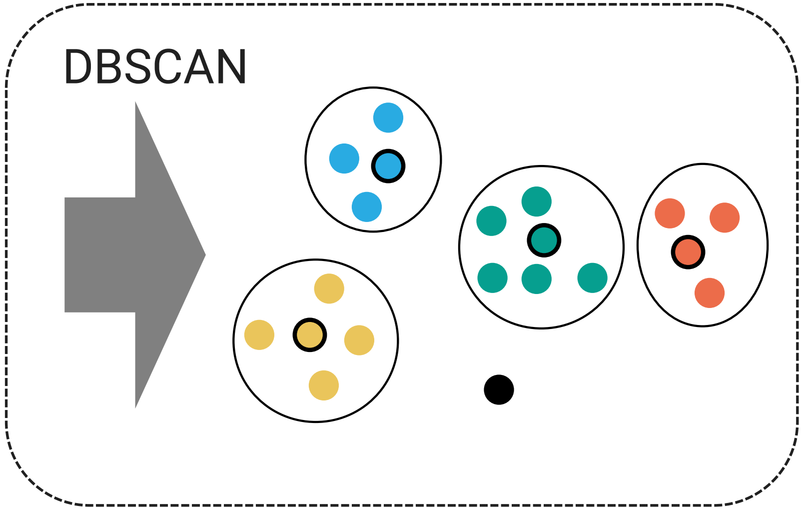
STAGE 1 + 1a: GC + EC



STAGE 2: Object condensation



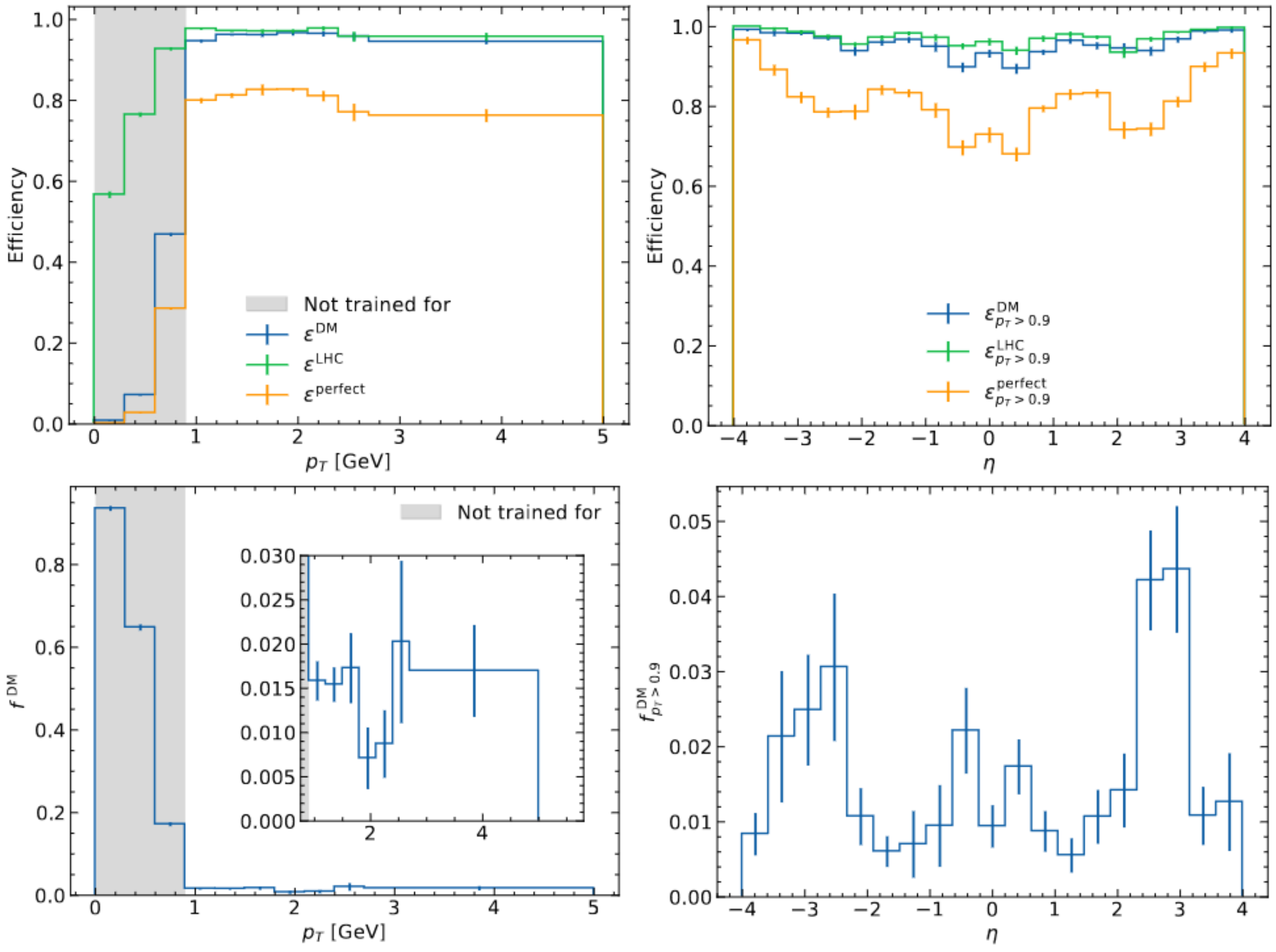
STAGE 3: Collect clusters



- Significantly improved since CHEP presentation: No EC GNN needed anymore
- **Can combine geometric constraints with EC in inference** ⇨ Much faster inference
- Purity of GC + EC: 68%, 90k edges
- OC: interaction networks with residual connections (5 layers, 192 node/edge dim)

arXiv: 2309.16754

Pipeline 1.1 (@CHEP proceedings): Geometric GC + EC MLP + OC GNN



Model:

- EC: 270k parameters
- OC: 1.9M parameters

Performance for pT > 0.9 GeV:

- **DM: 95%**
- **LHC: 97%**
- **Perfect: 80%**
- **Fake DM: 1.7%**

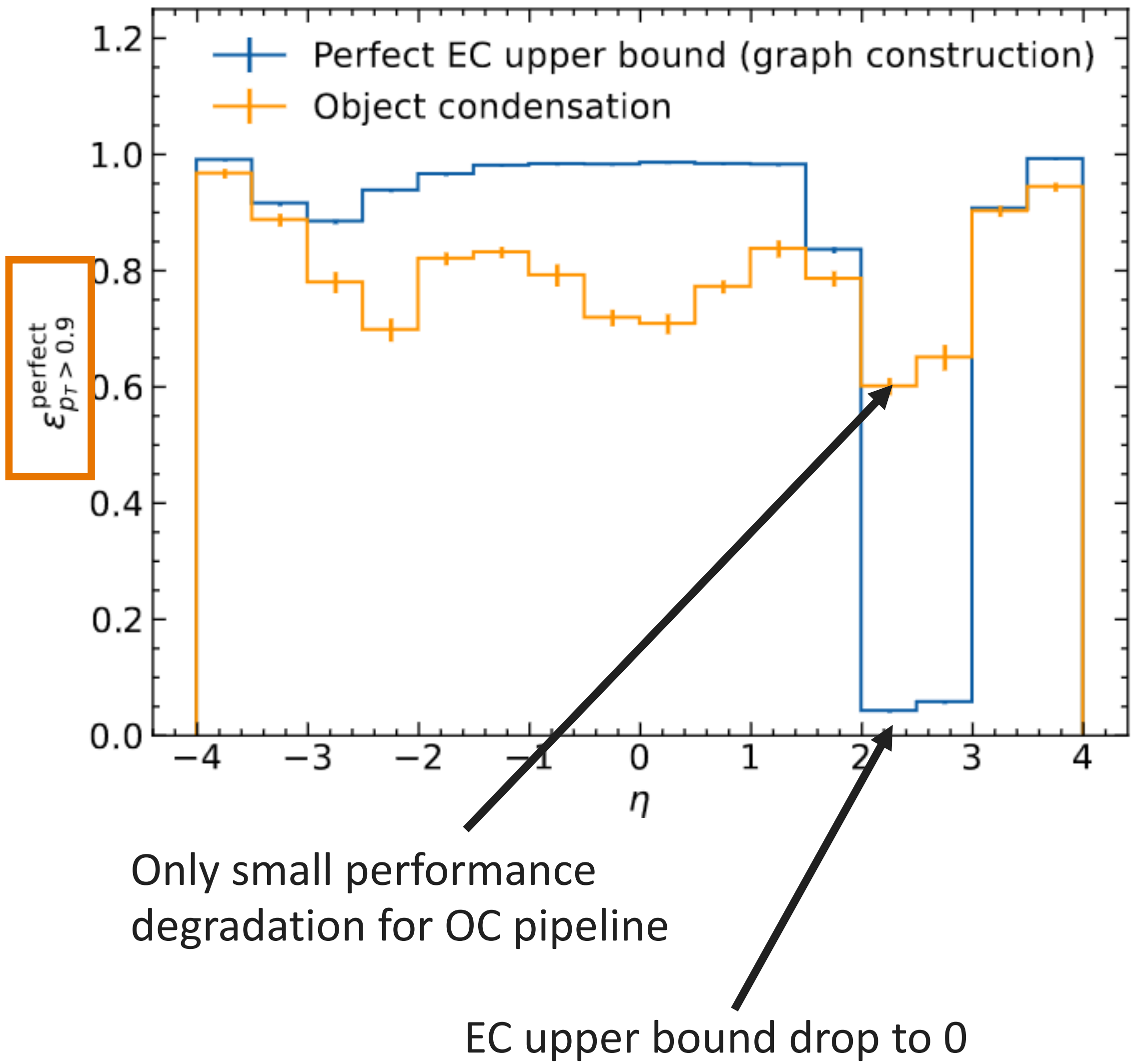
arXiv: 2309.16754

Pipeline 1.1 (@CHEP proceedings): Geometric GC + EC MLP + OC GNN

OC can fix/is more robust to missing edges, i.e., can perfectly reconstruct tracks that are impossible to perfectly reconstruct based on EC scores alone because of missing edges

To show this:

1. Construct graph as before
2. Remove all edges crossing from barrel to right endcap ($2 < \eta < 3$)
3. Calculate **"perfect EC" uppber bound** by taking all true edges and identify tracks with connected components (drops to 0 for $2 < \eta < 3$)
4. Compare with OC results

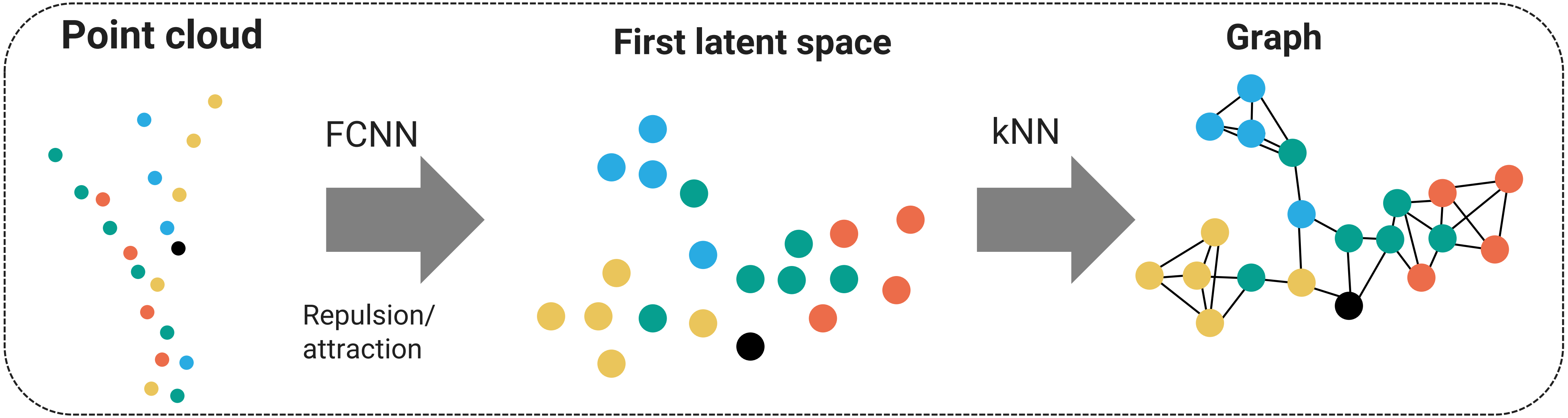


arXiv: 2309.16754

Pipeline 2.0: Metric learning GC + OC GNN

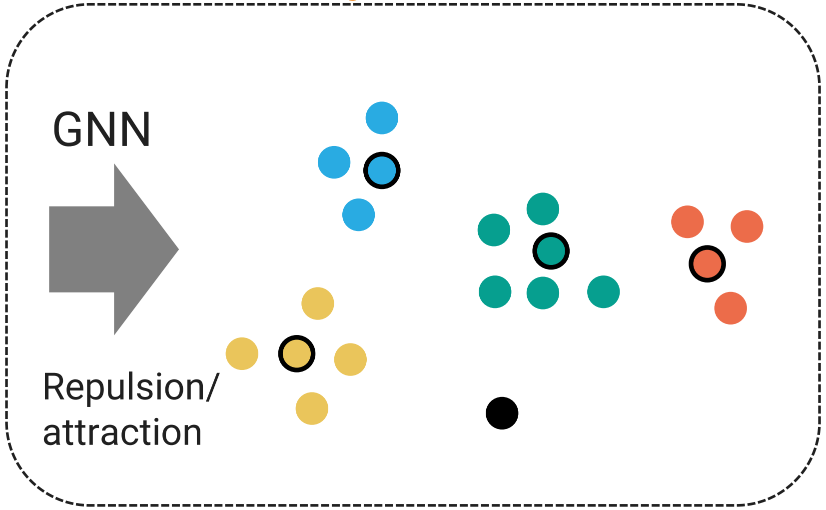
Heavily inspired by ExaTrkx

STAGE 1: GC with metric learning

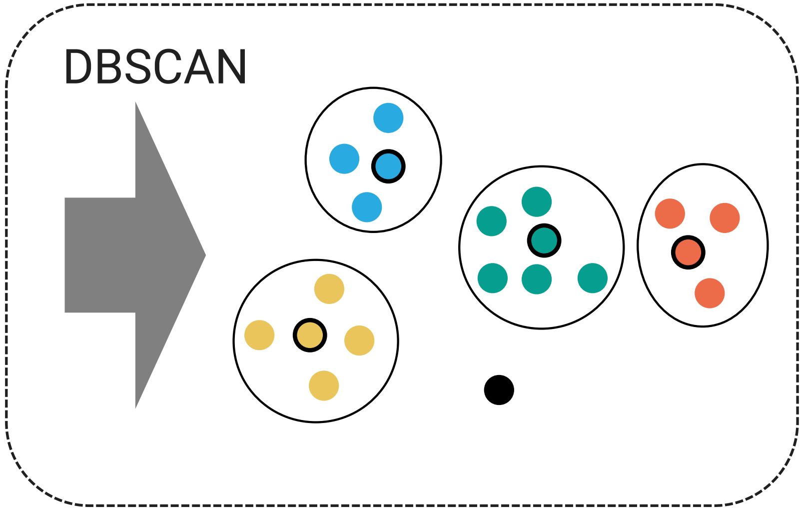


Difference: Currently also training to build edges that skip detector layers

STAGE 2: Object condensation

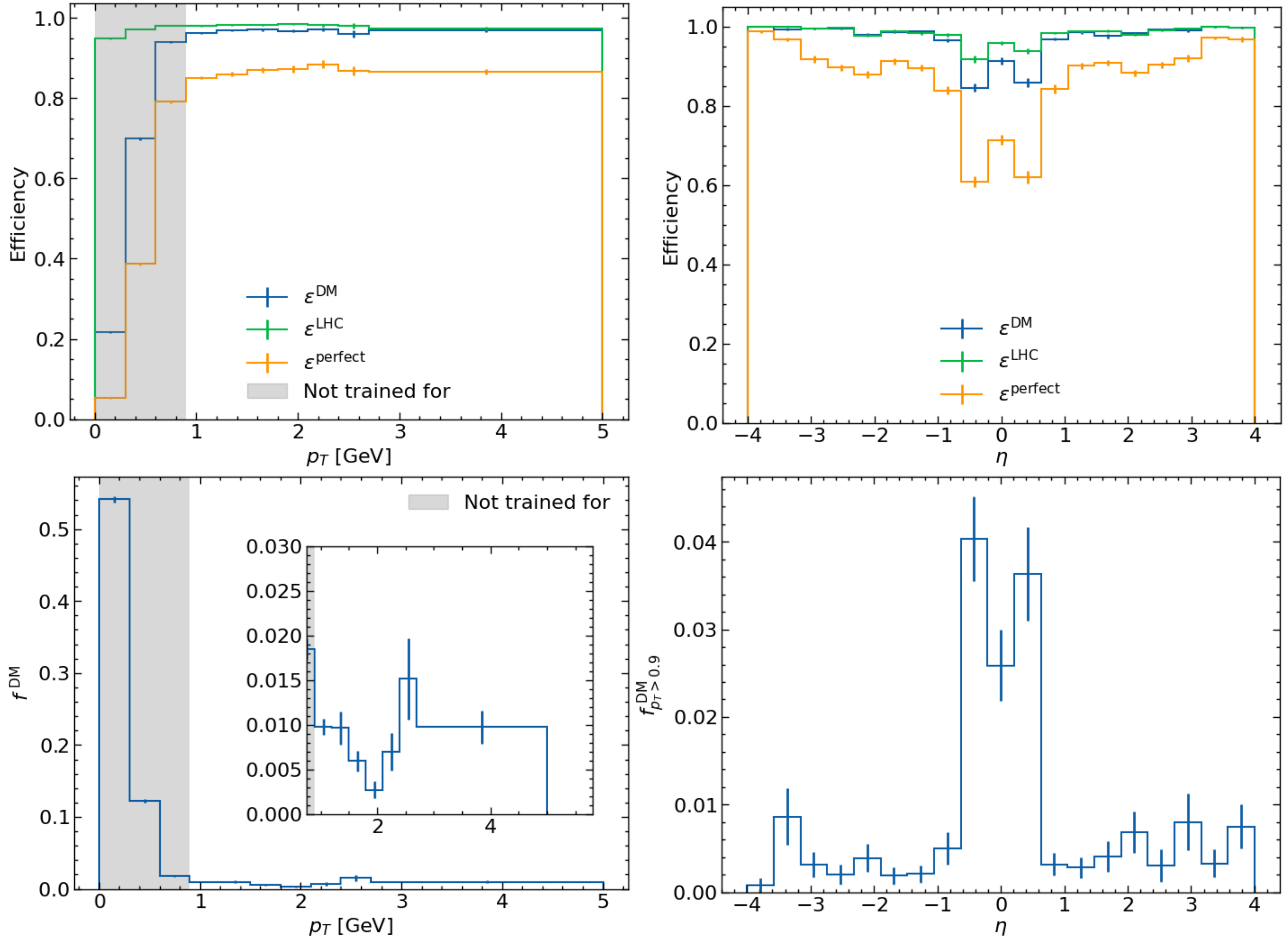


STAGE 3: Collect clusters



- FCNN: 6 layers, hidden dim 256
- Residual connection from GC latent space to OC output
- OC network almost the same as described in arXiv:2309.16754 (5 interaction networks with 192 node/edge dim and residual connections)

Pipeline 2.0: Metric learning GC + OC GNN



Model:

- GC: 300K parameters
- OC: 1.9M parameters
- kNN k=10

Performance for pT > 0.9 GeV

- **DM: 96%**
- **LHC: 98%**
- **Perfect: 86%**
- **Fake DM: 0.9%**

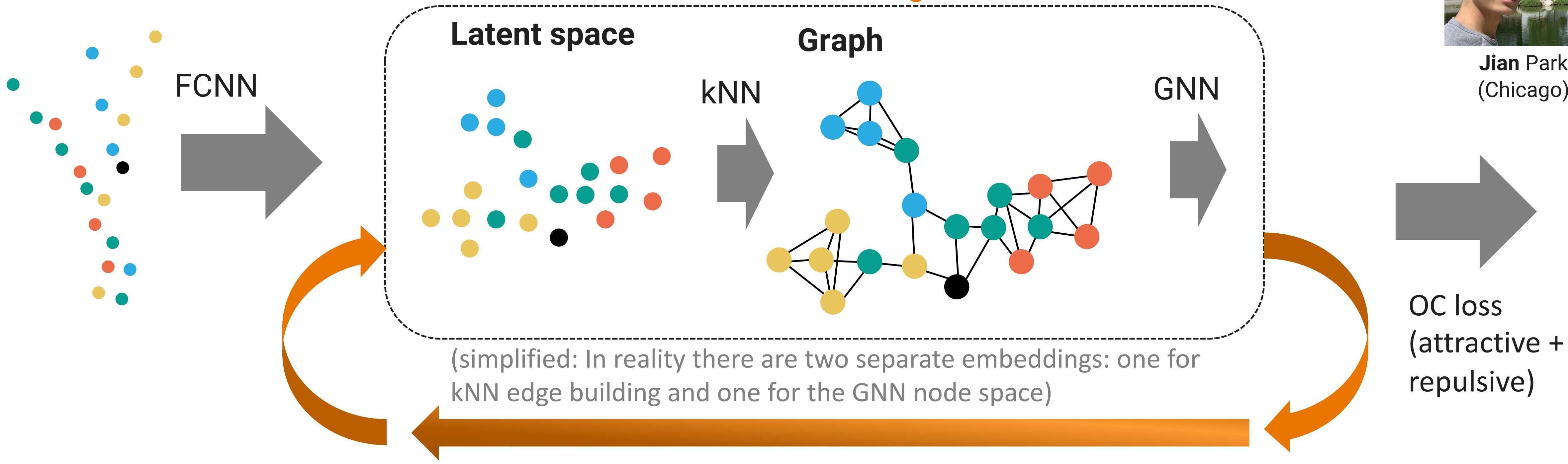
Training time ~30h (GC) + 60h (OC) on A100; probably still some performance left to recover with careful fine-tuning & training

Experimental pipeline: GravNet

This block is similar to pipeline 2.0, only repeated and trained all at once

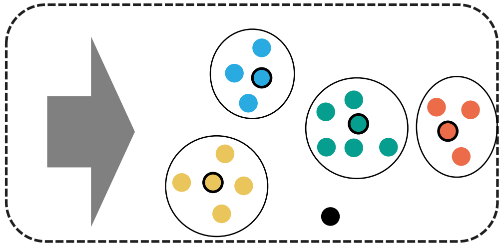
Jian Park
(Chicago)

STAGE 1+2: Embedding



(simplified: In reality there are two separate embeddings: one for kNN edge building and one for the GNN node space)

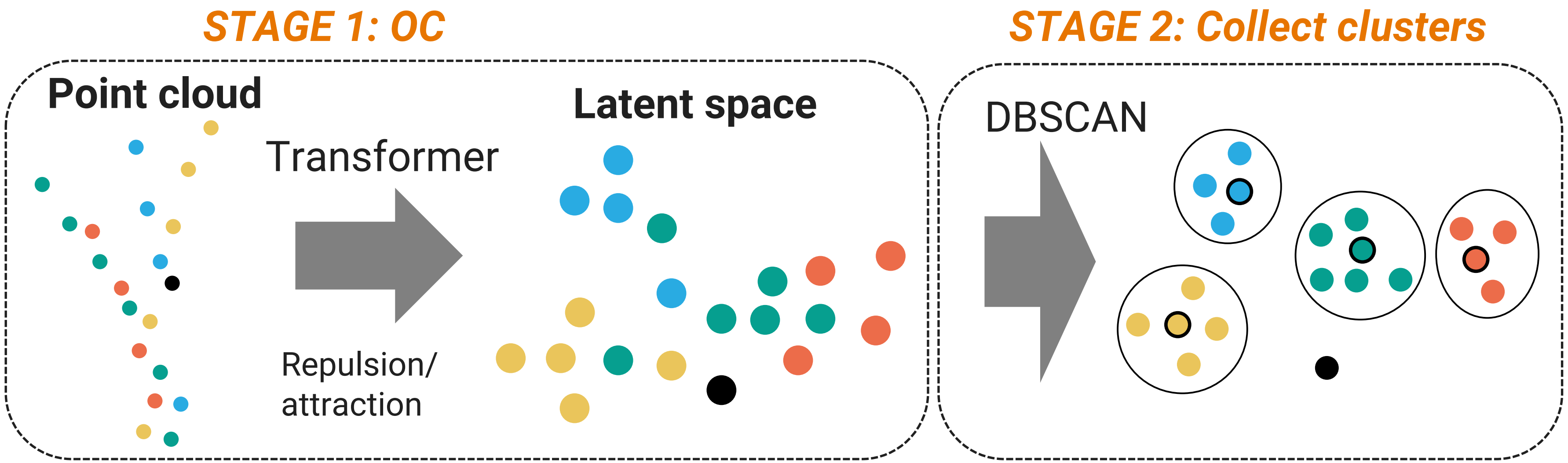
STAGE 3: Collect clusters



- **End-to-end training** (which is good and bad)
- As the embedding gets better, so do the message passing edges ⇨ only need small k
- GravNet slightly modified (e.g., FCNNs instead of simple linear layers)
- Currently only prototype; **confirmed to reach around 90% DM eff.**, but probably more given enough training time
- OC with GravNet seems to work very well for the Belle II outer tracker (Lea Reuter et al.)

https://github.com/gnn-tracking/jian-gnn-tracking-experiments

Experimental pipeline: Transformer



Siqi Miao
(Georgia Tech)

Pan Li
(Georgia Tech)

Motivation:

- kNN used in GC is often O(n^2) in GPU implementations
- GNNs have lots of irregular computations → not optimal on GPU; want model that is hardware-friendly/as fast as possible
- Transformer pipeline can be trained end-to-end

Proposition: Efficient sparse transformers

- Scaled dot product attention with relative positional encoding and E2 locality sensitive hashing (E2LSH)
- Trained with contrastive learning & hard negative mining

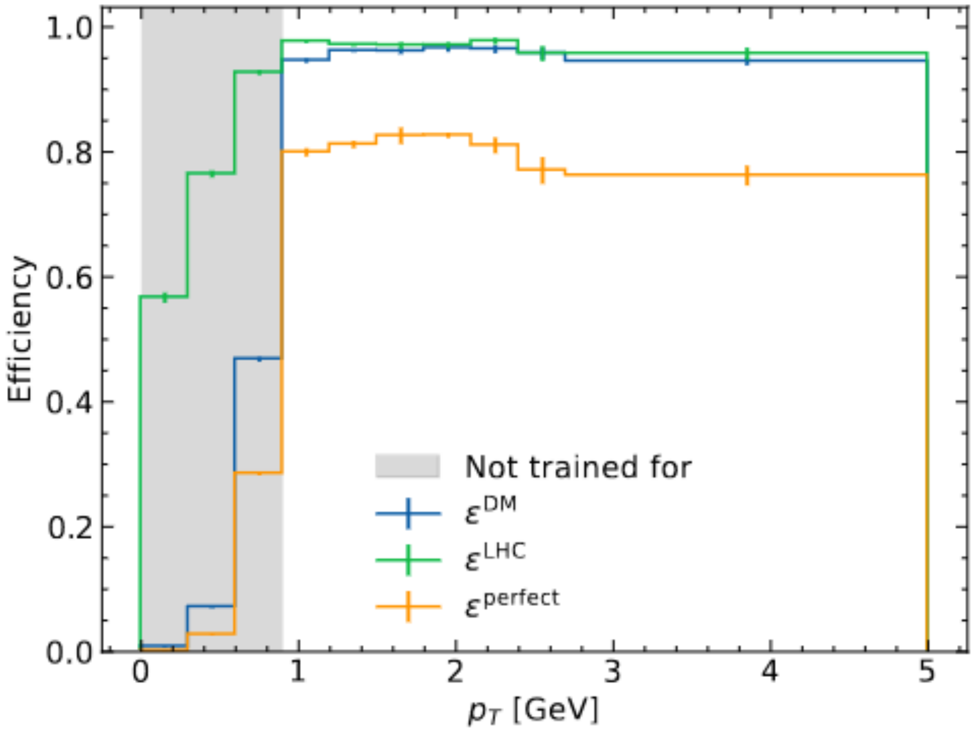
Result:

- Computations parallelizable and regular O(n log n)
- Inference on Quadro RTX 6000 around 500x faster at similar

Summary & Outlook

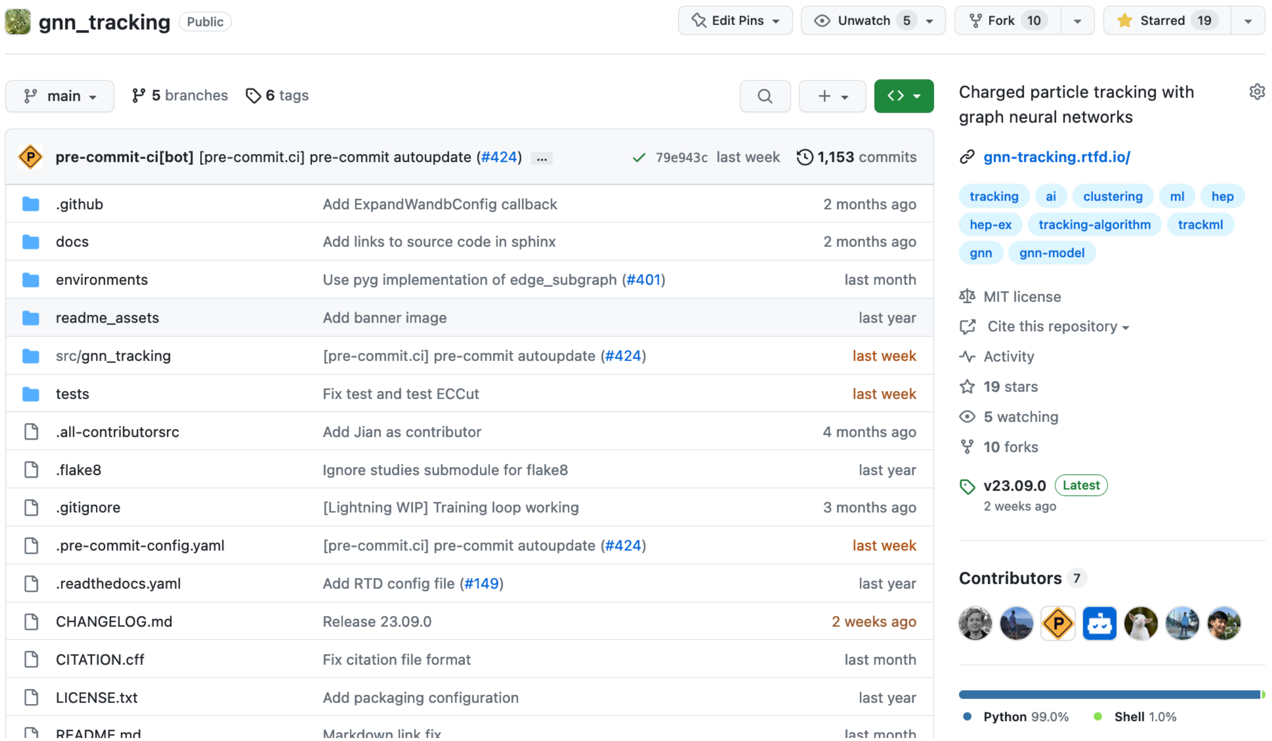
- **Learned clustering** (OC) is an alternative to EC-based track reconstruction
- Ran experiments on **pixel layers of trackML** dataset
- Two different architectures achieved high efficiencies:
  - **Geometric GC + FCNN EC + OC:** 95% DM, 80% perfect (pT > 0.9) (details in arXiv:2309.16754)
  - **Metric learning GC + OC:** 96% DM, 85% perfect (pT > 0.9)
- Several other architectures under consideration:
  - **GravNet** layers (repeated embedding + kNN edge building)
  - **Kernalized Local Transformers**
- OC can **handle missing edges** to a certain degree
- WIP:
  - Application to full detector
  - Studies with CMS data

Results from the 2.0 pipeline



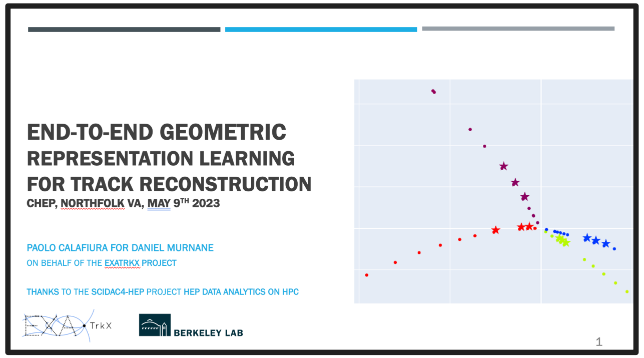
Thanks!

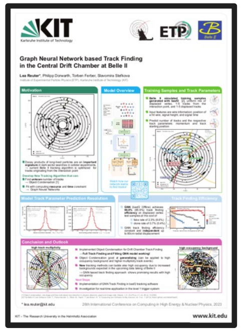
Find us on GitHub! New contributors welcome!
https://github.com/gnn-tracking



Shoutouts: More object condensation

Daniel Murnane
"Influencer" approach (next up!)





Lea Reuter
Object condensation tracking for the Belle II outer tracker @CHEP23