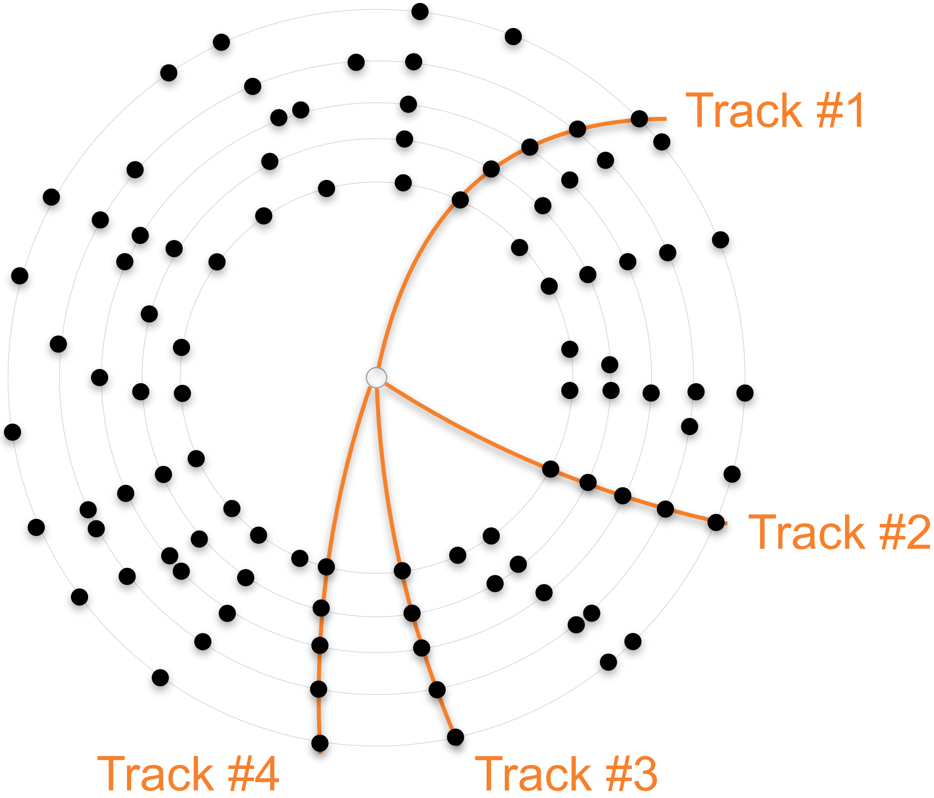
EggNet Track Reconstruction Pipeline

Paolo Calafiura, Jay Chan, Loic Delabrouille, Brandon Wang

Lawrence Berkeley National Laboratory

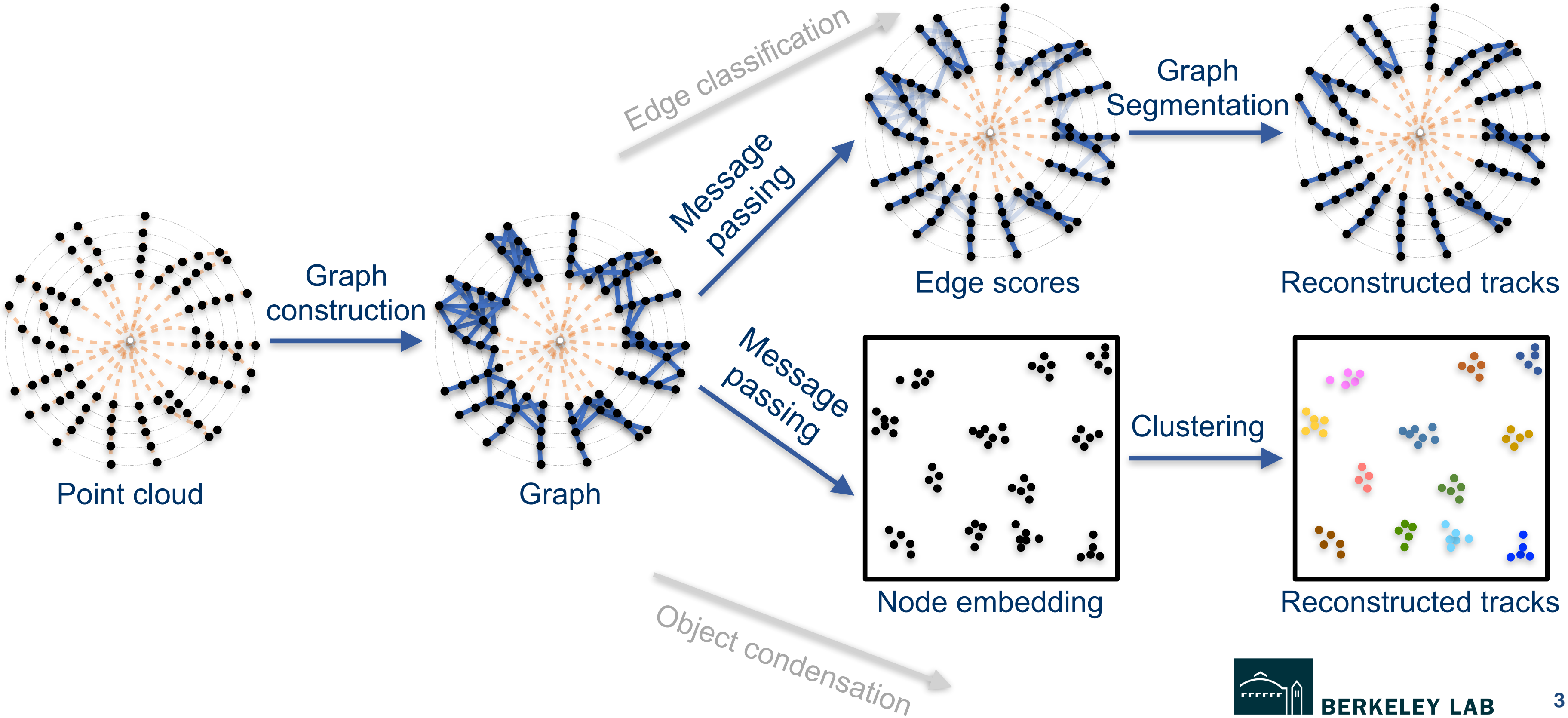
Krakow, Poland, October 24, 2024

Track reconstruction is a challenging task

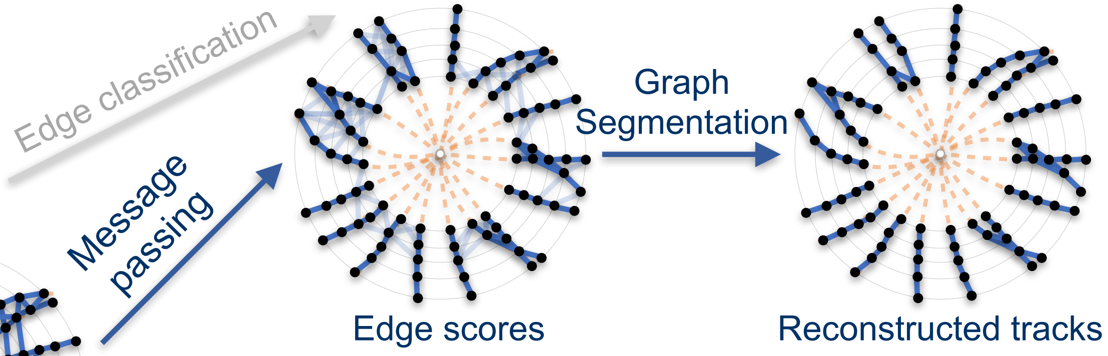


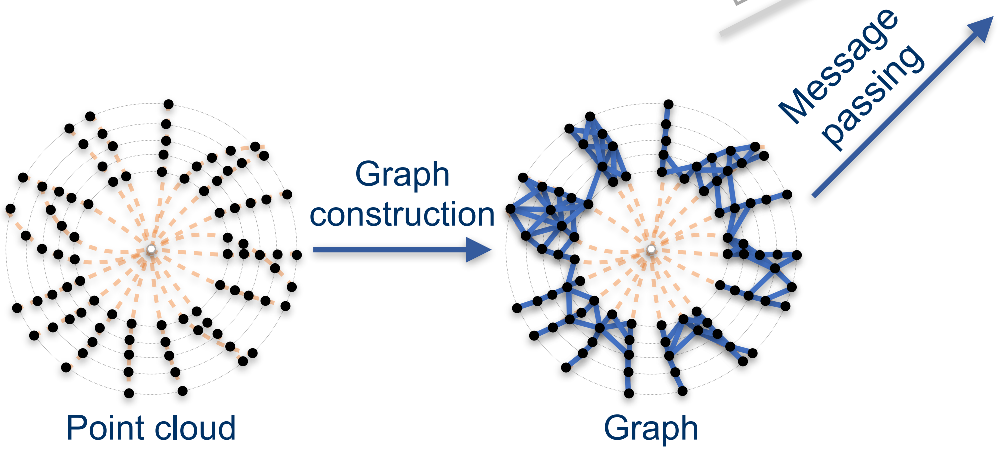
~O(10^4) particles per collision event at HL-LHC
→ ~O(10^5) hits in ATLAS ITK

GNN-based track reconstruction pipelines



Track reconstruction as edge classification

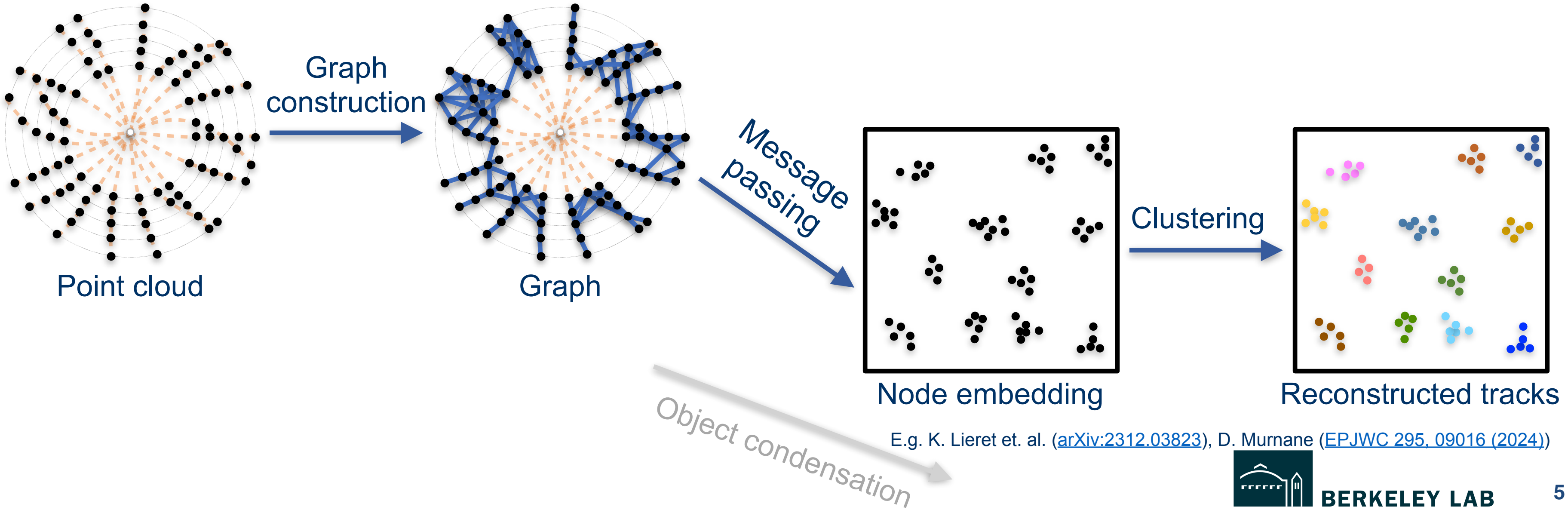




E.g. ATLAS GNN4ITK pipeline (ATL-SOFT-PROC-2023-047)

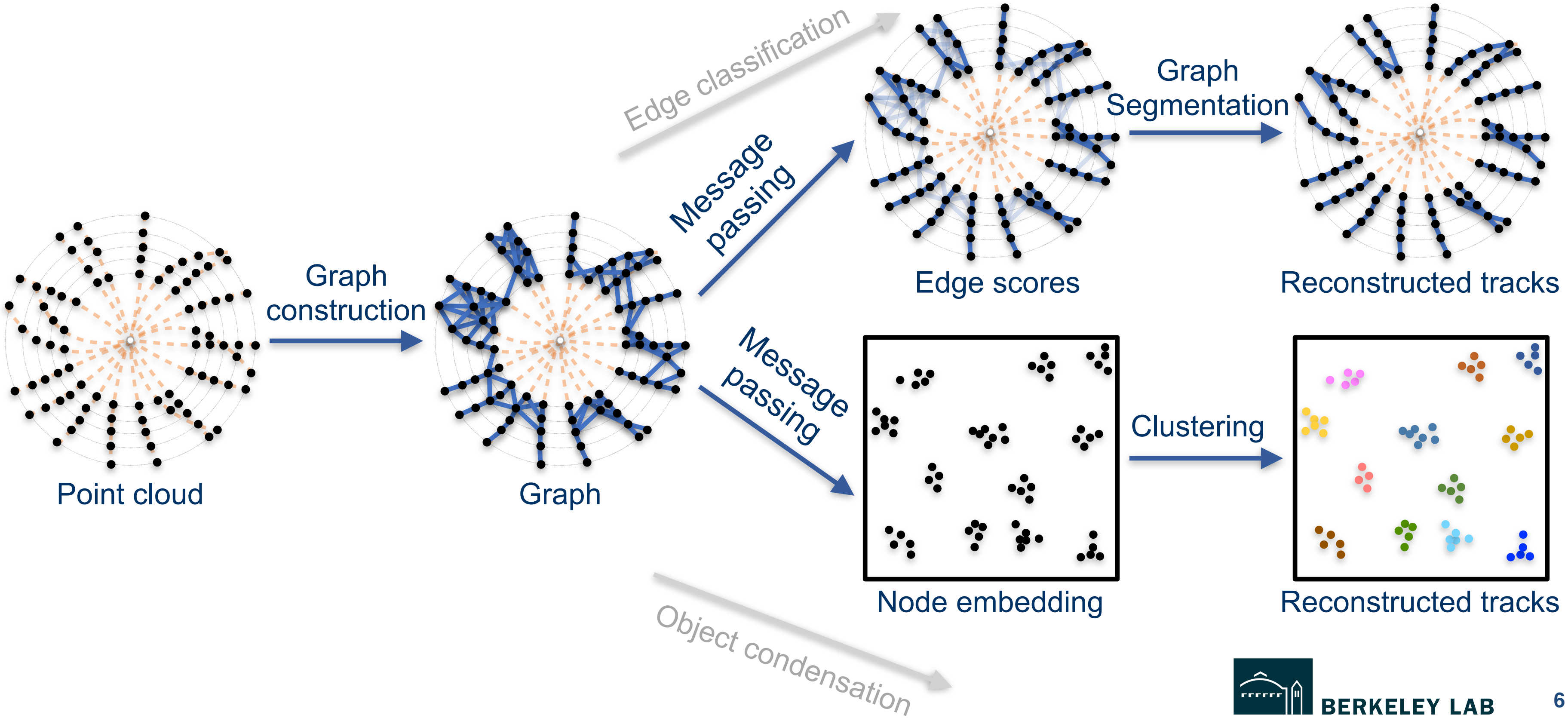
CHEP talks:
[Alina on GNN4ITK overview and computing performance]()
[Daniel on Graph Segmentation]()

Track reconstruction as object condensation

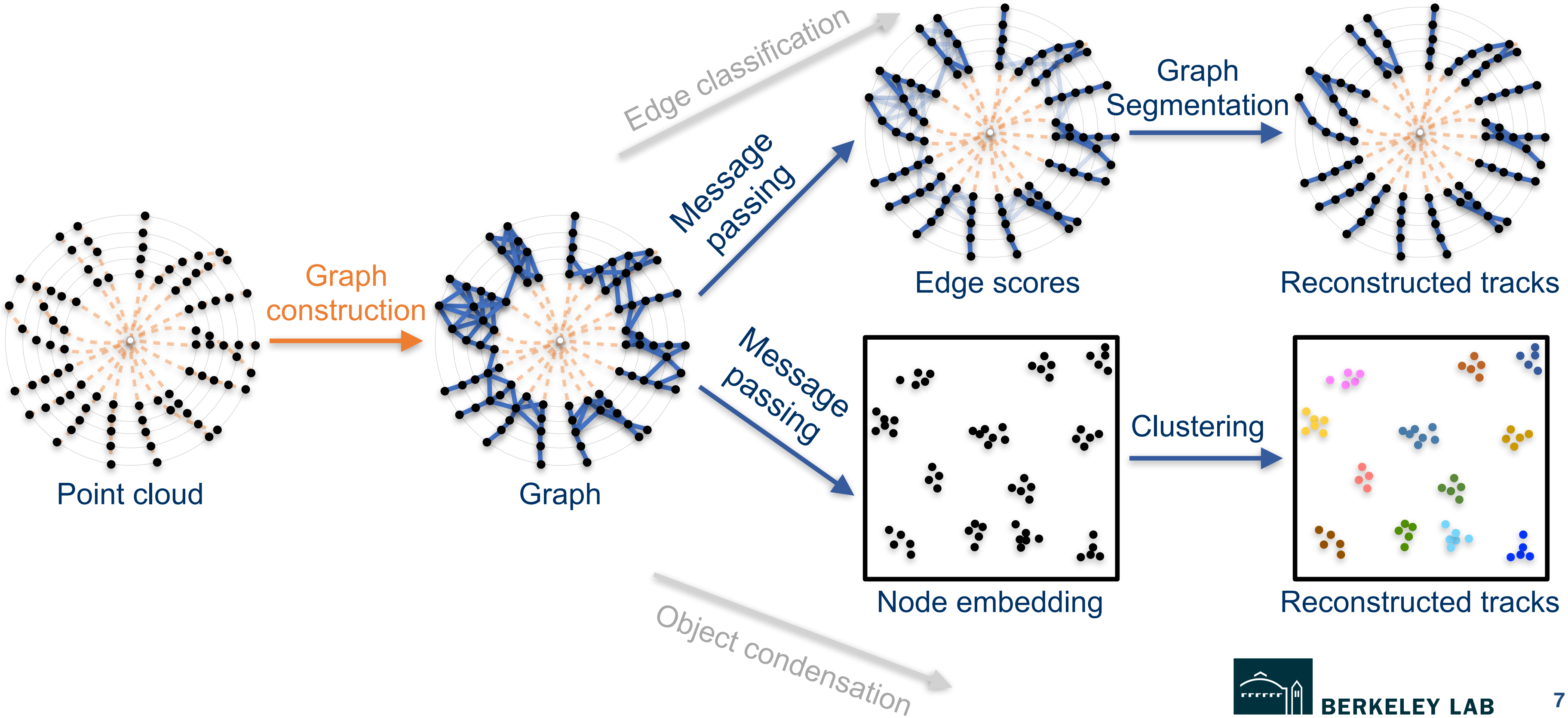


E.g. K. Lieret et. al. (arXiv:2312.03823), D. Murnane (EPJWC 295, 09016 (2024))

GNN-based track reconstruction pipelines

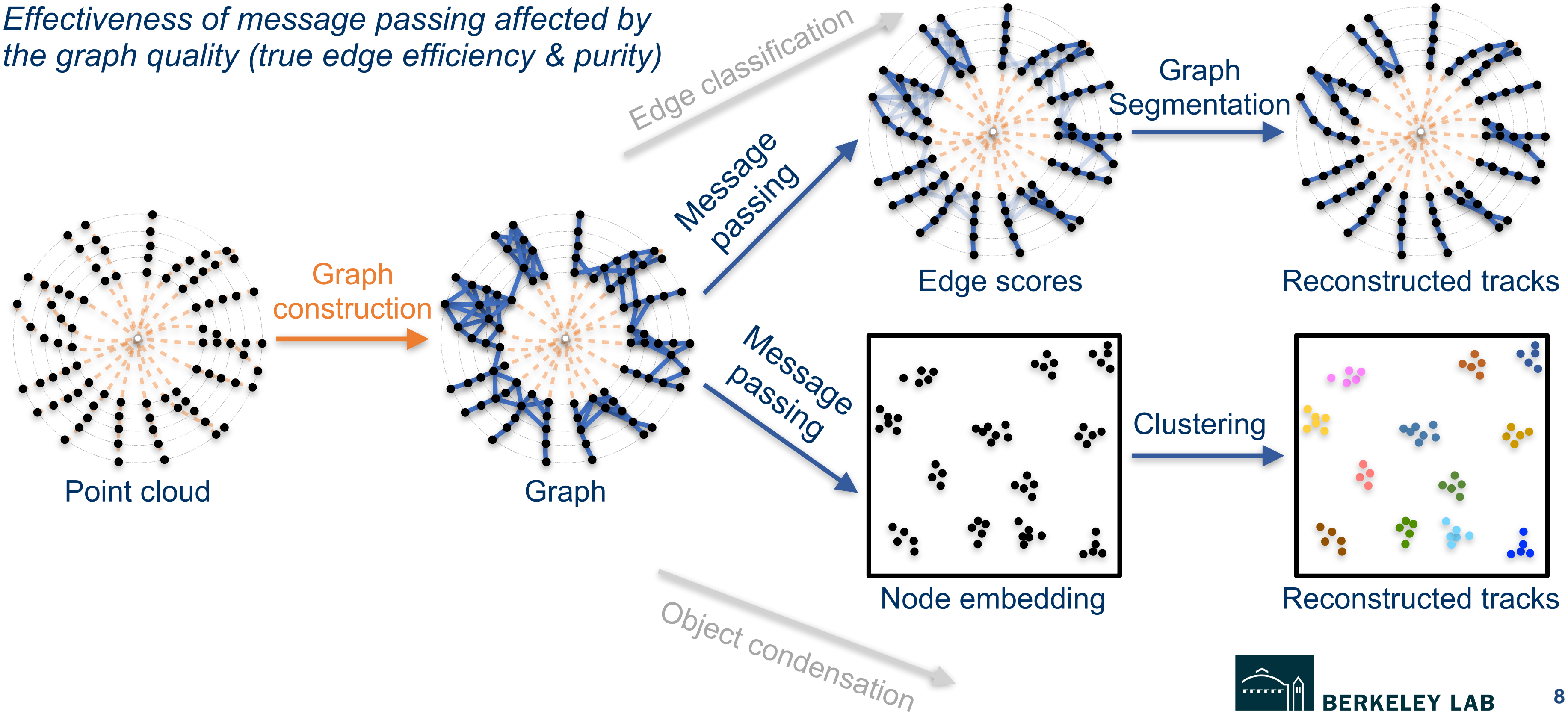


Graph reconstruction required as a first step in pipeline



Graph reconstruction required as a first step in pipeline

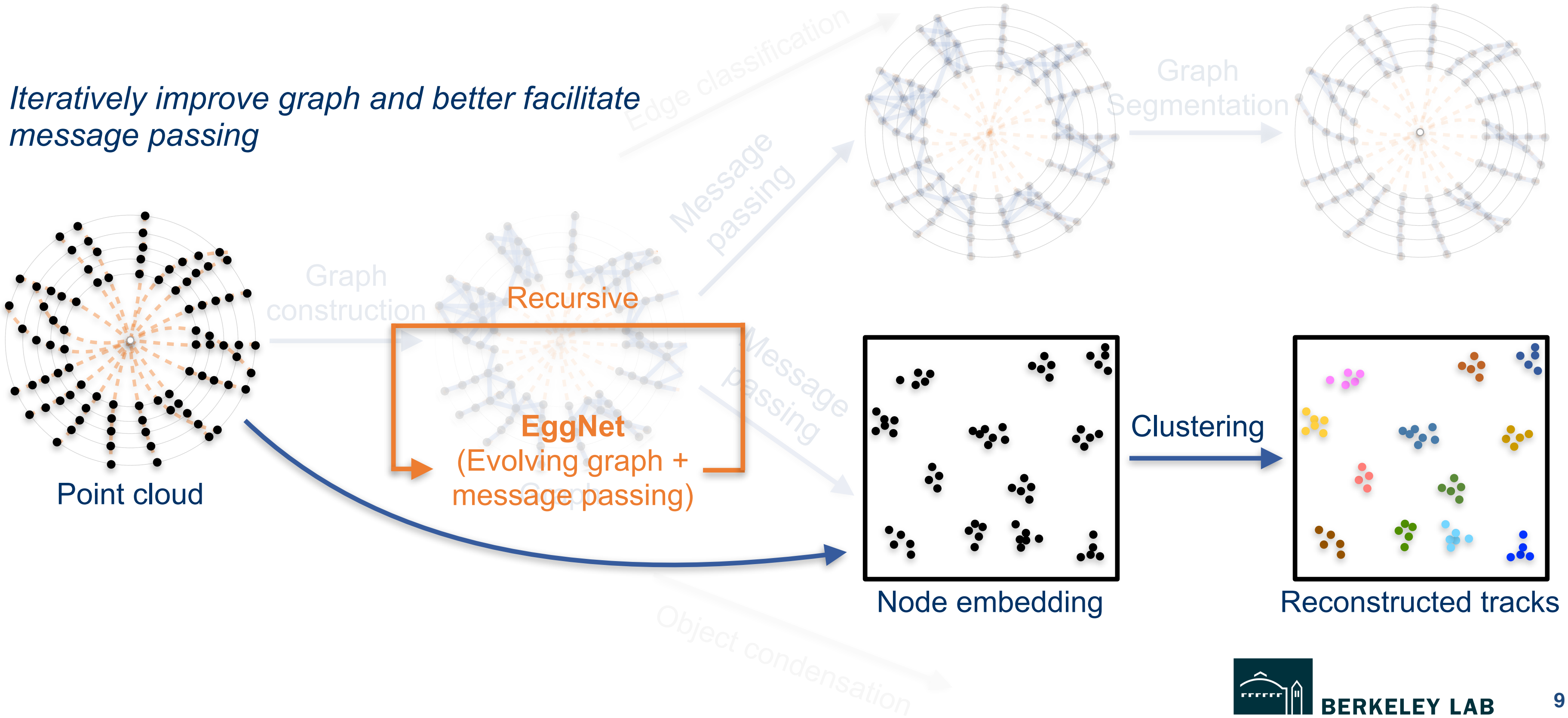
Effectiveness of message passing affected by the graph quality (true edge efficiency & purity)



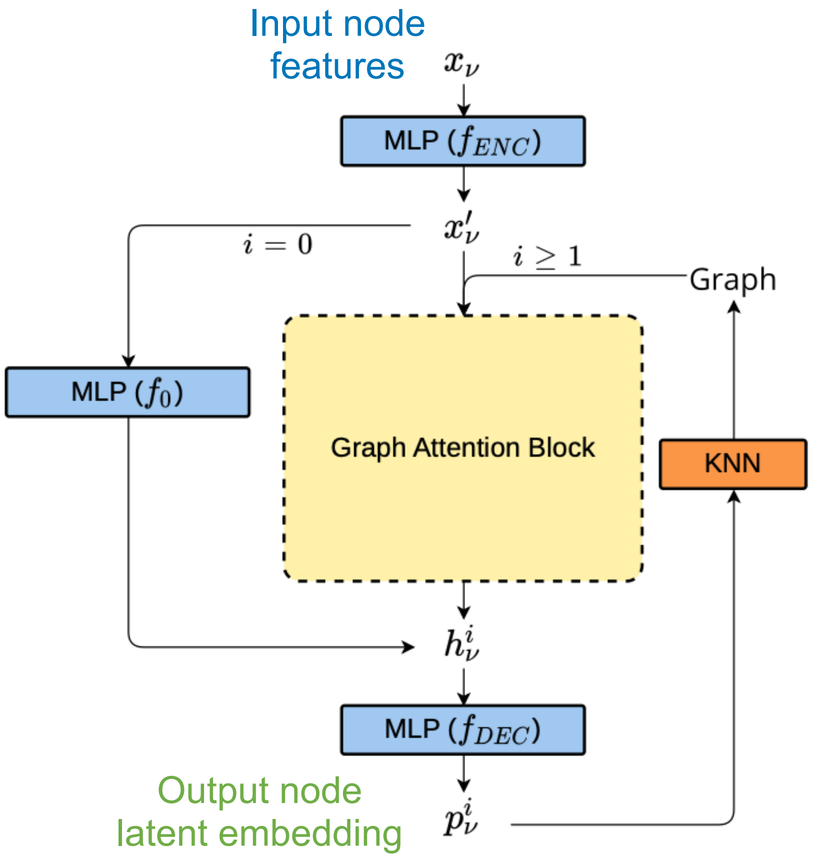
BERKELEY LAB

EggNet tracking pipeline

Iteratively improve graph and better facilitate message passing

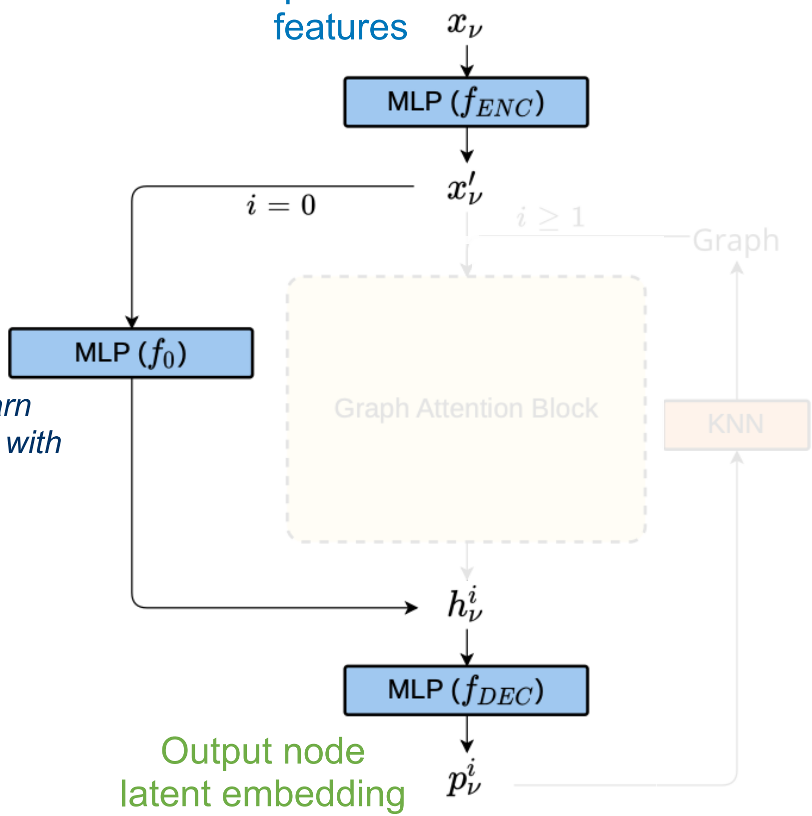


EggNet


Input node features
x_ν
MLP (f_{ENC})
x'_ν
$i = 0$
$i \geq 1$
Graph
MLP (f_0)
Graph Attention Block
KNN
h^i_ν
MLP (f_{DEC})
Output node latent embedding
p^i_ν

EggNet

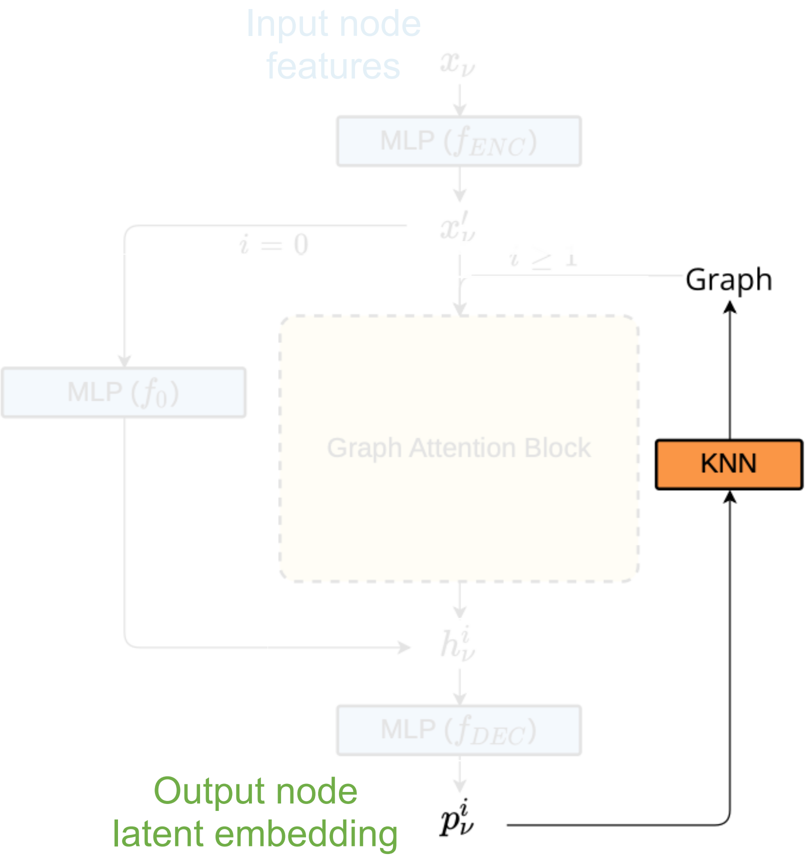
Input node features



First iteration: learn node embedding with deep sets

Output node latent embedding

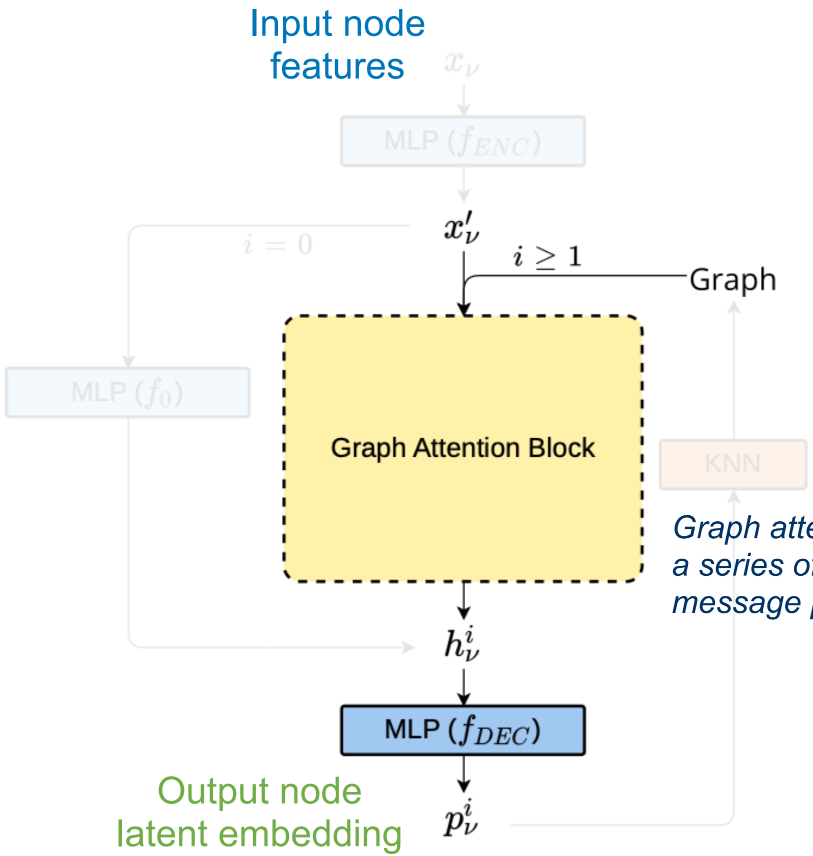
EggNet



Evolutional
KNN graph

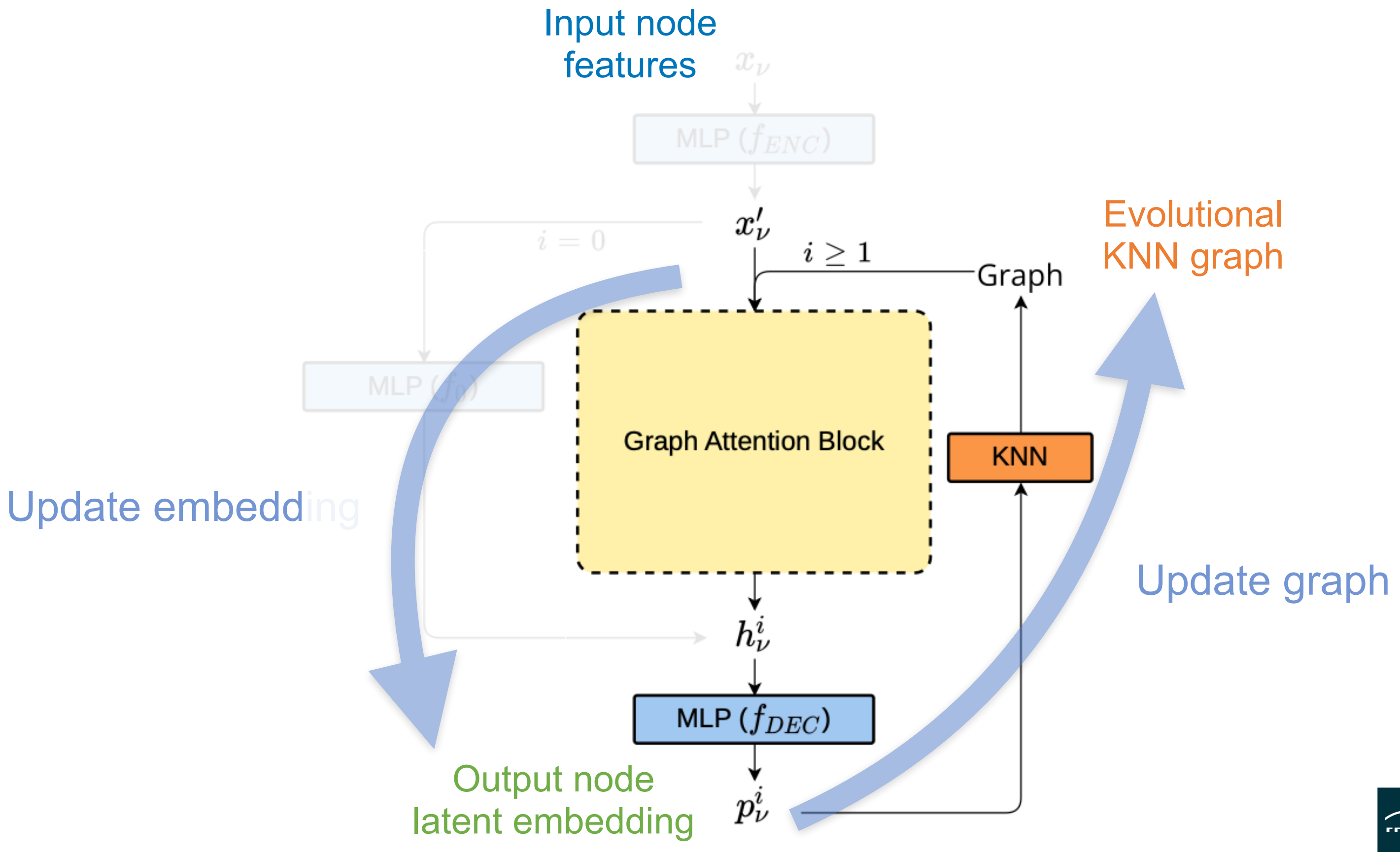
K-nearest-neighbor (KNN):
Connect each node to its k closest nodes in the embedding space

EggNet



Graph attention block consists of a series of graph-attention-style message passings

EggNet



Input node features
x_v
MLP (f_{ENC})
x'_v
$i = 0$
$i \geq 1$
Graph
Evolutional KNN graph
MLP (f_0)
Graph Attention Block
KNN
Update embedding
Update graph
h^i_v
MLP (f_{DEC})
Output node latent embedding
p^i_v

Contrastive loss

For each pair of nodes (edge):

$$L(e) = y_e\, d_e^2 + (1 - y_e)\, max^2(0, m - d_e)$$

Attractive loss for positive pair y = 1
(hits come from the same particle)

Repulsive loss for negative pair y = 0
(hits come from different particles)

d = Euclidean distance between two hits

Three categories of edges:

$$L_{\mathrm{tot}} = < L(e_{\mathrm{signal}}) > + < L(e_{\mathrm{random}}) > + < L(e_{\mathrm{KNN}}) >$$

Signal edges (hits from same particles)

~10x #nodes

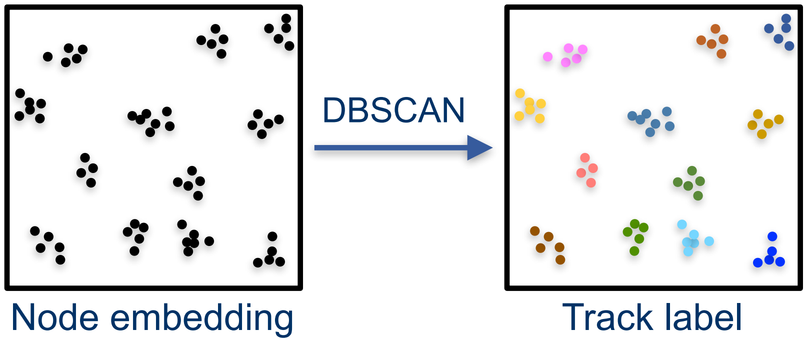
Random edges (randomly select 2 hits)

Select 10^5

KNN edges for “hard negative mining”

10x #nodes

DBSCAN and track performance

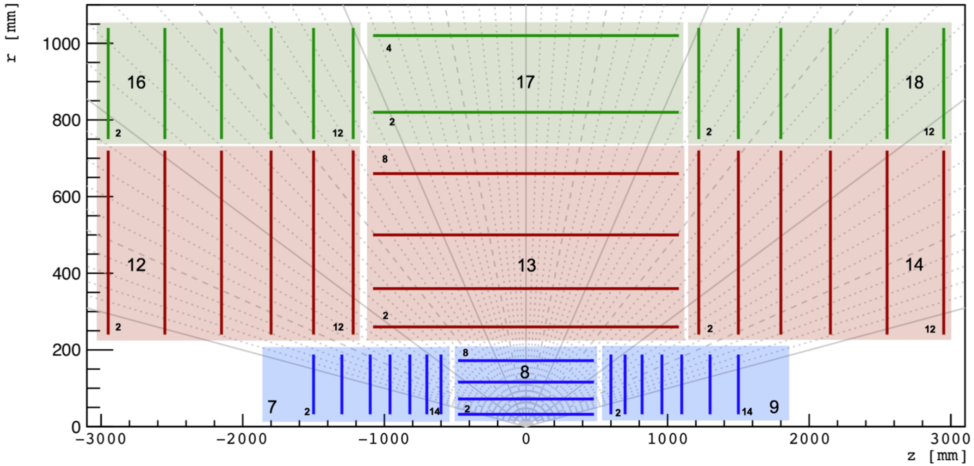


Evaluate track performance with the standard ATLAS definition ([arXiv: 2103.06995](https://arxiv.org/abs/2103.06995))

A matched track = (>50% hits in this track candidate come from same particle)

$$\mathrm{Eff}_{\mathrm{track}} = \frac{N^{\mathrm{reco}}_{\mathrm{particles}}}{N_{\mathrm{particles}}} \qquad r_{\mathrm{fake}} = \frac{N_{\mathrm{tracks}} - N^{\mathrm{matched}}_{\mathrm{tracks}}}{N_{\mathrm{tracks}}} \qquad r_{\mathrm{duplicate}} = \frac{N^{\mathrm{matched}}_{\mathrm{tracks}} - N^{\mathrm{reco}}_{\mathrm{particles}}}{N^{\mathrm{reco}}_{\mathrm{particles}}}$$

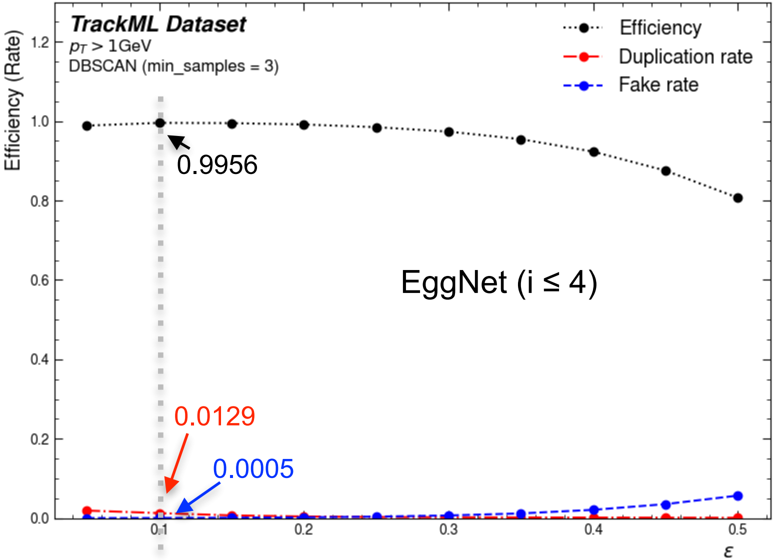
Test case with TrackML dataset



- Formulated in the [Kaggle TrackML challenge]() (HL-LHC like detector)
- Each event ~O(10^4) particles; ~O(10^5) hits

DBSCAN track performance (pT = 1 GeV hard cuts)

Track performance vs ε (DBSCAN)



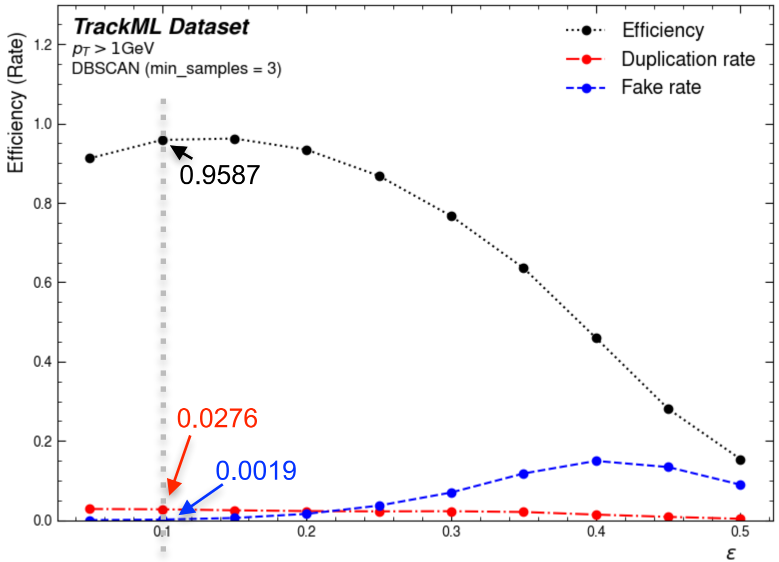
METHOD	EFFICIENCY	DUP. RATE	FAKE RATE
EC	0.9898 ± 0.0009	0.0421 ± 0.0011	0.0012 ± 0.0000
OC	0.9902 ± 0.0007	0.0328 ± 0.0007	0.0015 ± 0.0001
EGGNET ($i \leq 1$)	0.7454 ± 0.0027	0.2202 ± 0.0024	0.0099 ± 0.0004
EGGNET ($i \leq 2$)	0.9905 ± 0.0005	0.0179 ± 0.0006	0.0011 ± 0.0000
EGGNET ($i \leq 3$)	0.9940 ± 0.0002	$\mathbf{0.0117 \pm 0.0003}$	$\mathbf{0.0005 \pm 0.0000}$
EGGNET ($\mathbf{i \leq 4}$)	$\mathbf{0.9956 \pm 0.0003}$	0.0129 ± 0.0002	0.0006 ± 0.0000

arXiv:2407.13925

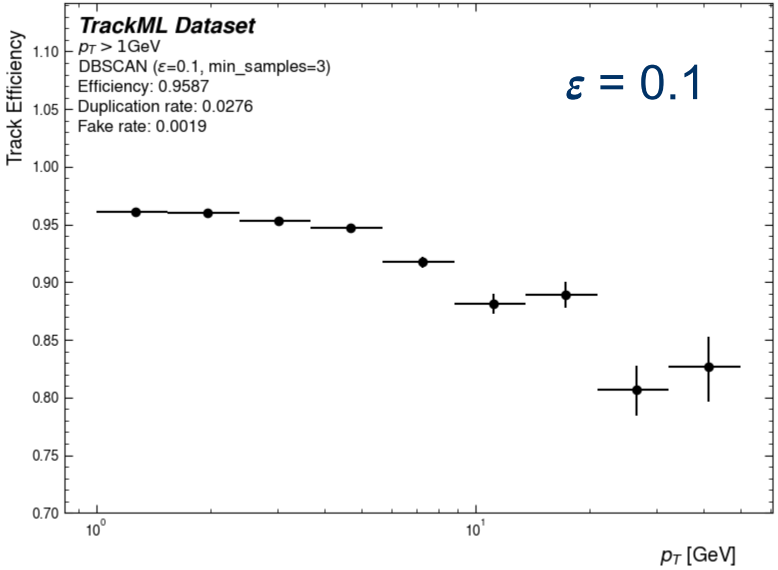
- Remove hits associated with particles of pT<1GeV for simplicity
  - Reduce graph size to ~$O(10^4)$ nodes
- EggNet outperforms prebuilt-graph-based methods with ≥3 iterations

DBSCAN track performance (full TrackML events)

Track performance vs ε (DBSCAN)



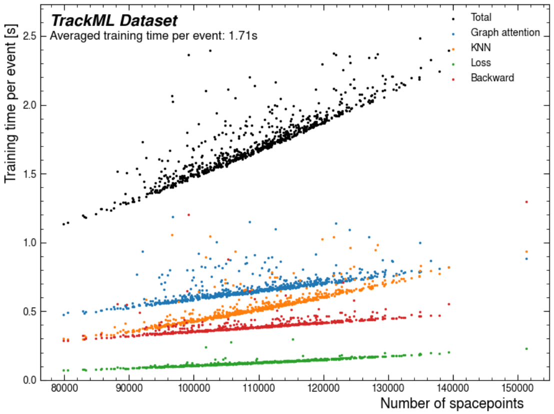
Track efficiency vs p_T



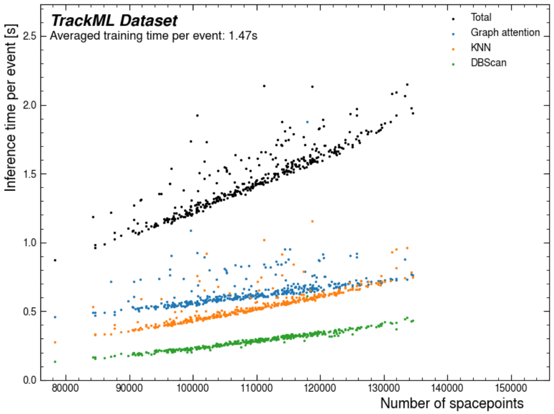
BERKELEY LAB

Computing challenges

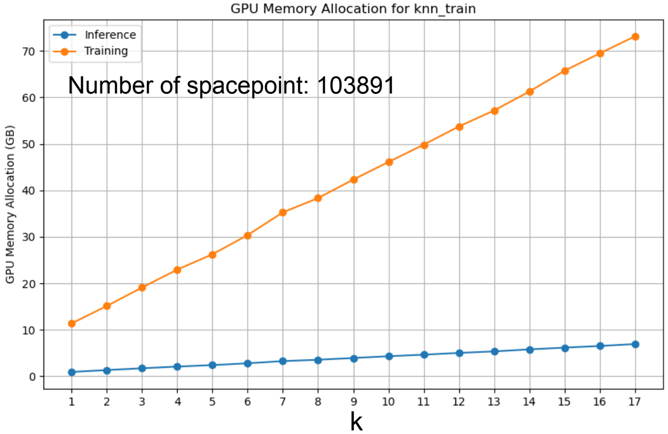
Training time



Inference time

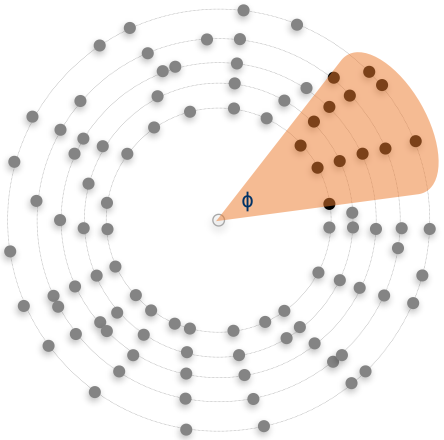


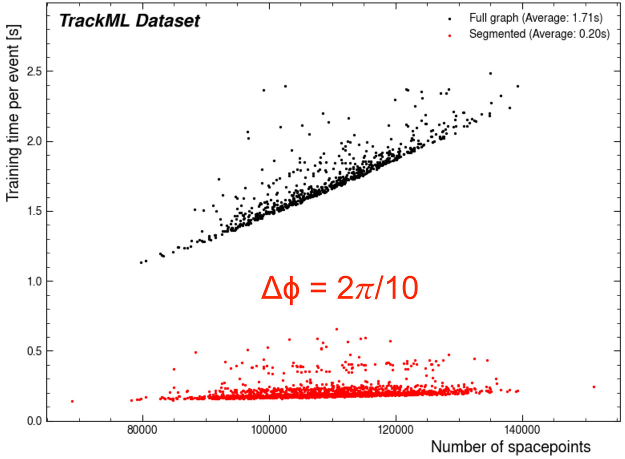
GPU memory

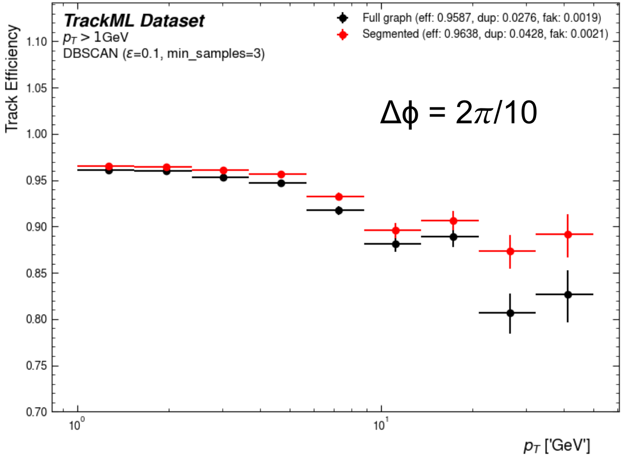


- Computing performance evaluated on an NVIDIA A100 80GB GPU
- KNN and DBSCAN ran on GPU (cuml library) -> ~6x speed up compared to CPU
- Computing time mainly comes from graph attention and KNN
- KNN scales quadratically with number of spacepoints
- High demands on GPU memory. An event with ~150k spacepoints requires ~50GB GPU memory

Training on segmented subgraphs



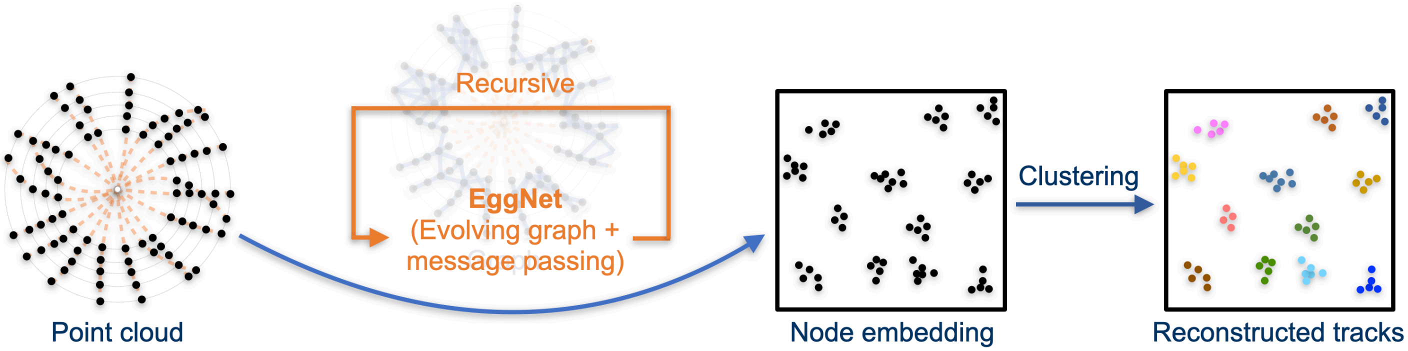




- Only look at a subset of spacepoints at a time
- Train with the subgraphs segmented by ϕ (fixed range at a random central value)
- Significantly reduce GPU memory requirement as well as training time
- Obtain similar track performance to training with full graphs
- Can potentially perform inference on segmented graphs as well (future work)

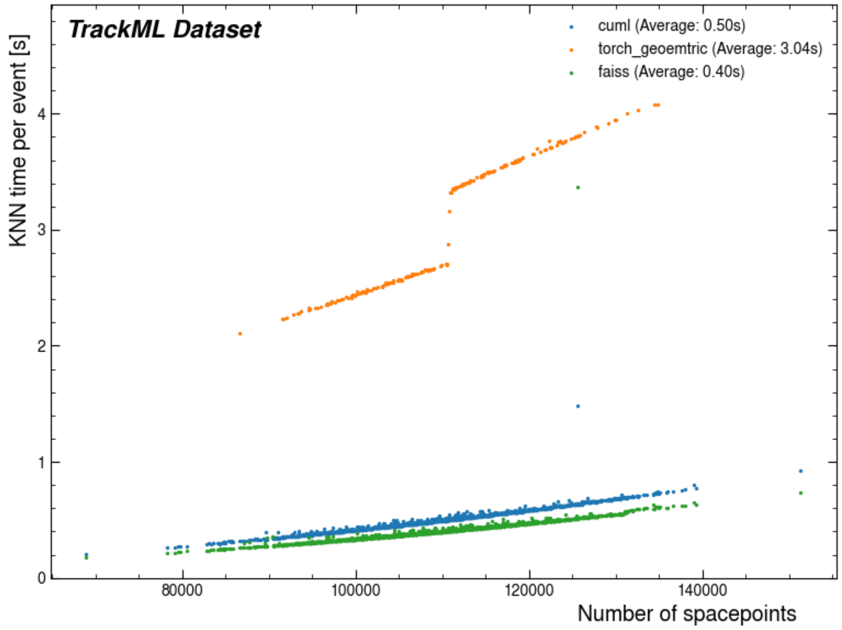
Summary

- Propose a one-shot object-condensation tracking algorithm using an Evolving-graph-based Graph Attention Network
  - Better facilitate message passing with updated graphs
  - Test it with full trackML events; achieve excellent track performance
- Next step: address challenges in computational cost
  - Scalability of KNN: exploring approximate algorithm with GPU implementation
  - High demands on GPU memory: training on segmented graphs give similar physics performance



Backups

Nearest Neighboring Algorithms



- Significant (>6x) speed up of KNN performed on GPU (cuml and Faiss) compared to CPU implementation (torch_geometric)
- Yet to explore approximate nearest neighboring (ANN) algorithms
  - Faiss ANN
  - Annoy (only CPU implementation available)

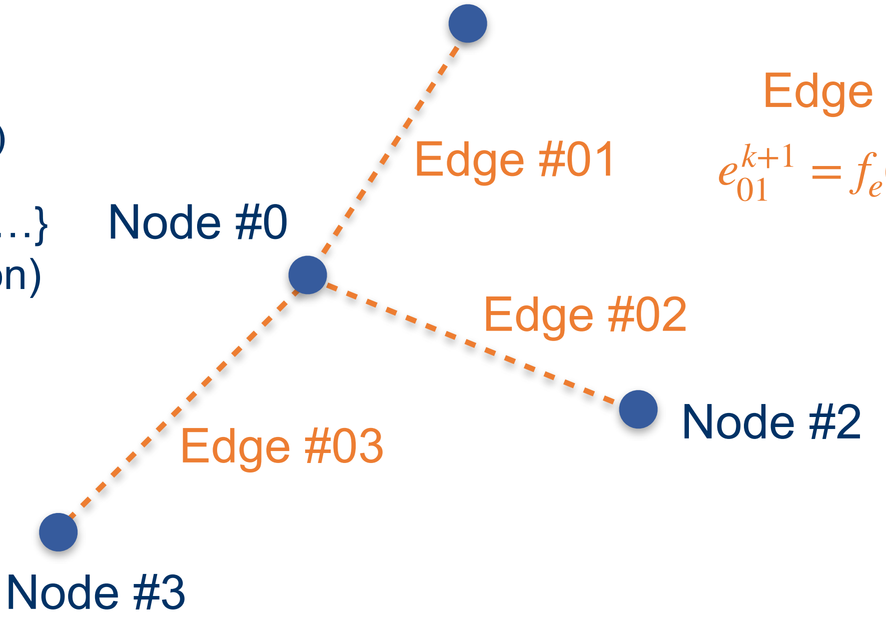
Message passing for track reconstruction

<u>Hits = Nodes</u>

Node update:

$$n_0^{k+1} = f_n(n_o^k, \mathrm{agg}(e_{0j}^k))$$

agg ∋ {sum, mean, max…}
^ weighted sum (attention)

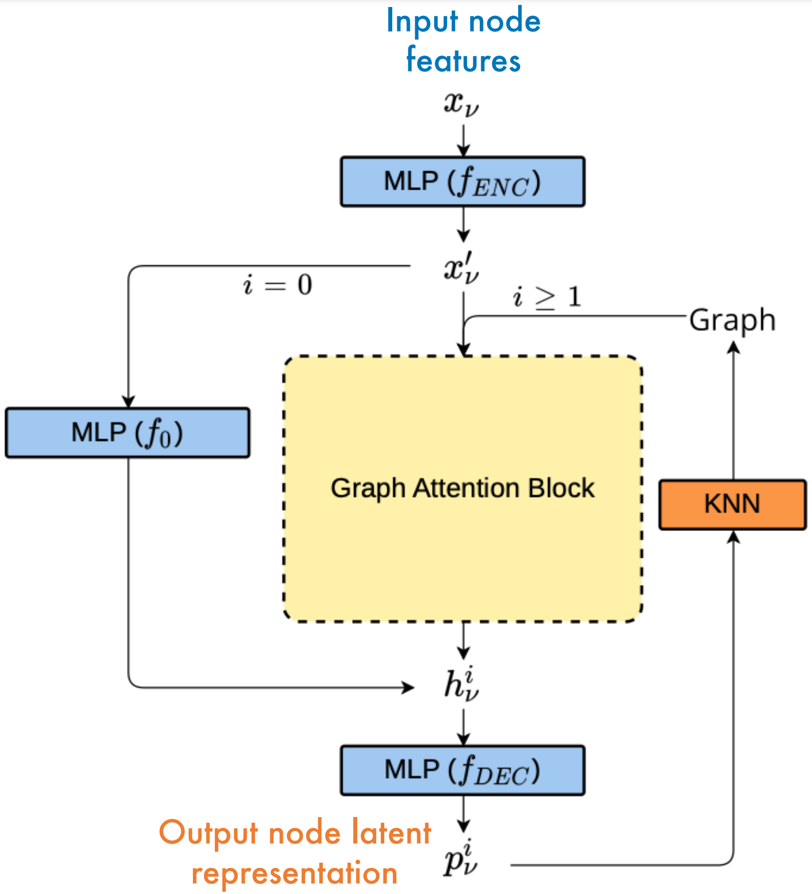


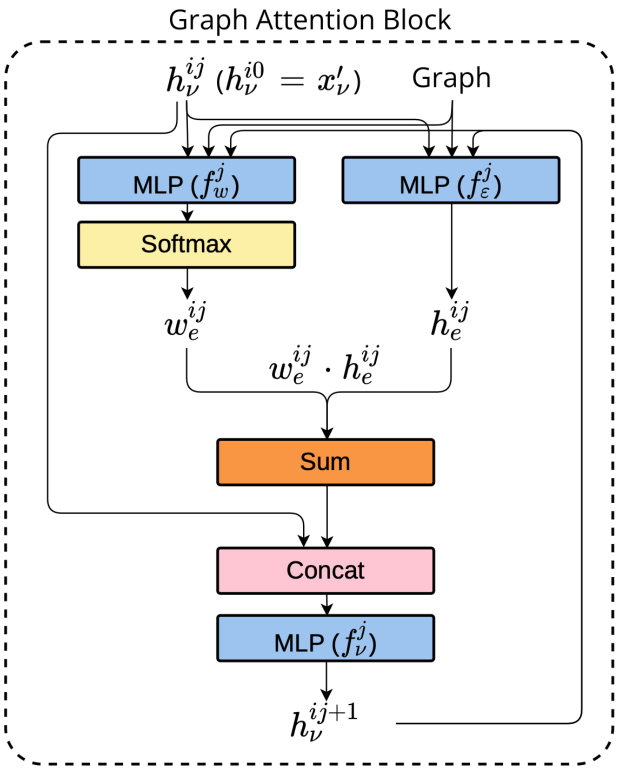
Edge update:

$$e_{01}^{k+1} = f_e(e_{o1}^k, n_0^k, n_1^k)$$

Learn key node / edge features from the whole graph structure in an event

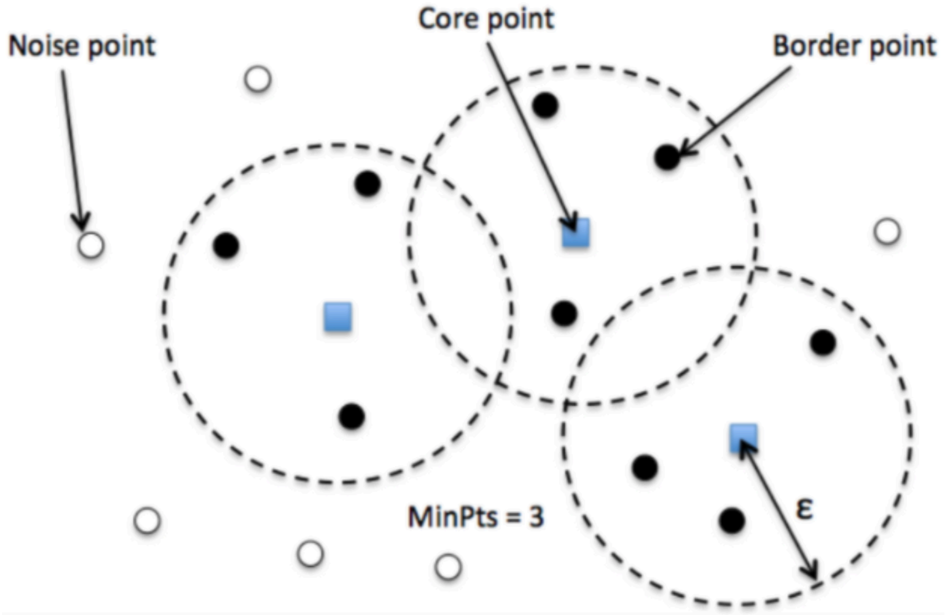
EggNet





BERKELEY LAB

Density-Based Spatial Clustering of Applications with Noise



Idea: a cluster in data space is a contiguous region of high point density, separated from other such clusters by contiguous regions of low point density