Graph Neural Network for Track Finding at LHCb

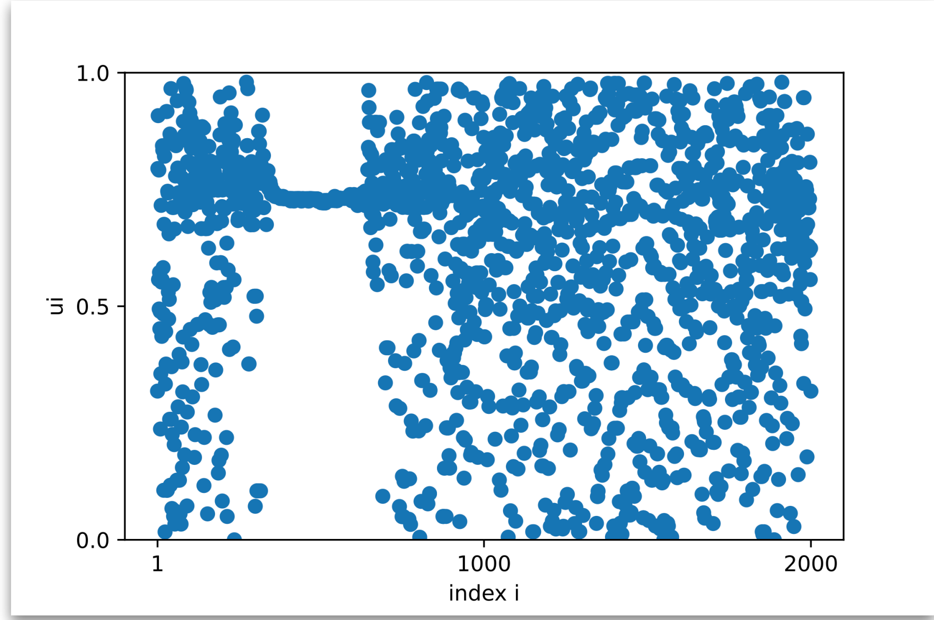
SMARTHEP Annual Meeting
Lund, Sweden, November 27, 2023

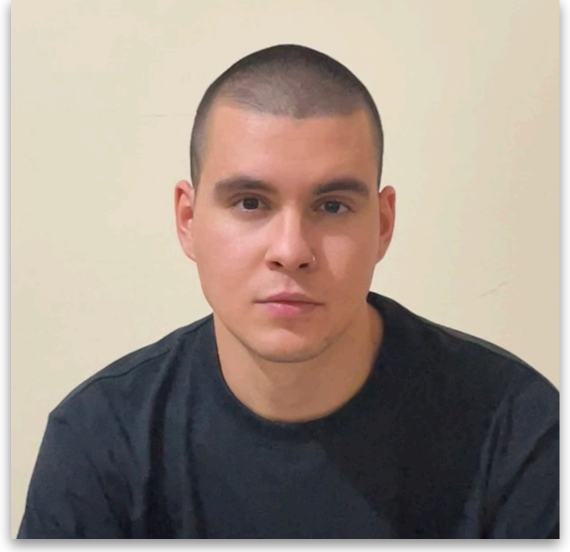
Fotis Giasemis, Anthony Correia, Nabil Garroum, Vladimir Vava Gligorov

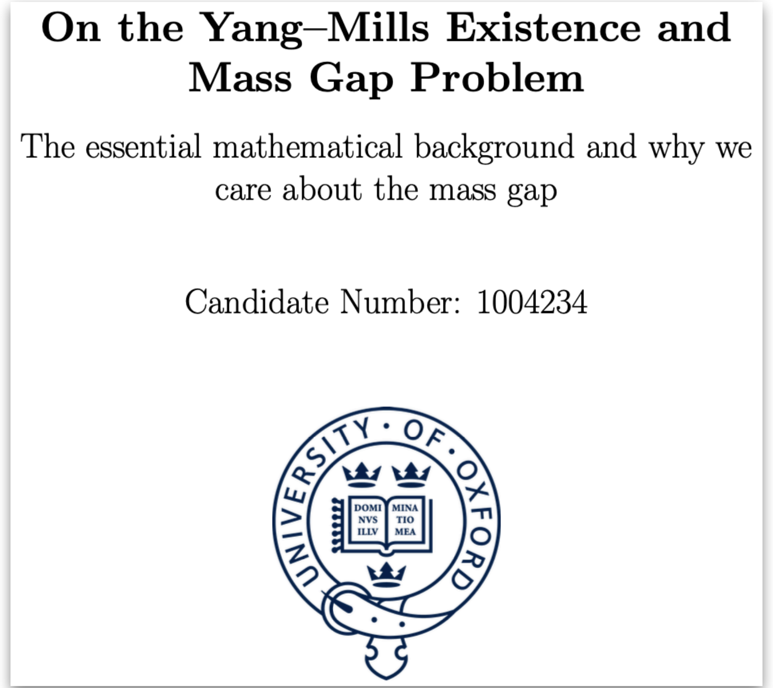
Myself

Fotis

- Hometown: Agia Anna, Euboea, Greece
- MMathPhys Mathematical and Theoretical Physics
    - University of Oxford
    - 4 years
- MSc Applied Mechanics
    - National Technical University of Athens
    - 2 years
    - Thesis: Quantum Chaos
- ESR5: Paris (LIP6 + LPNHE)
    - RTA on heterogeneous architectures for LHC and self-driving cars
    - Vava Gligorov (LPNHE) and Bertrand Granado (LIP6)

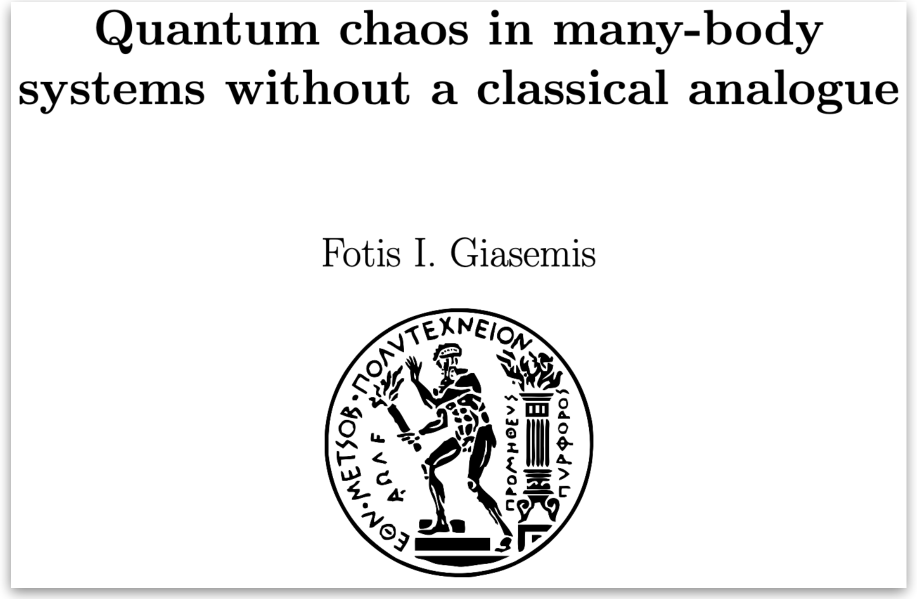




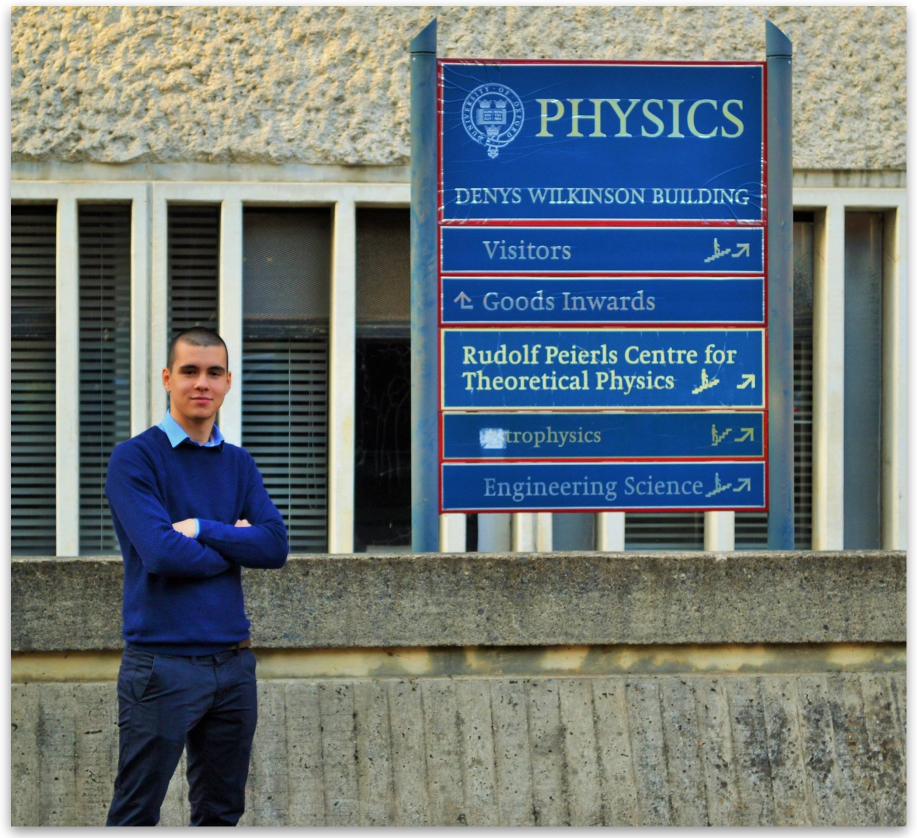

On the Yang–Mills Existence and Mass Gap Problem

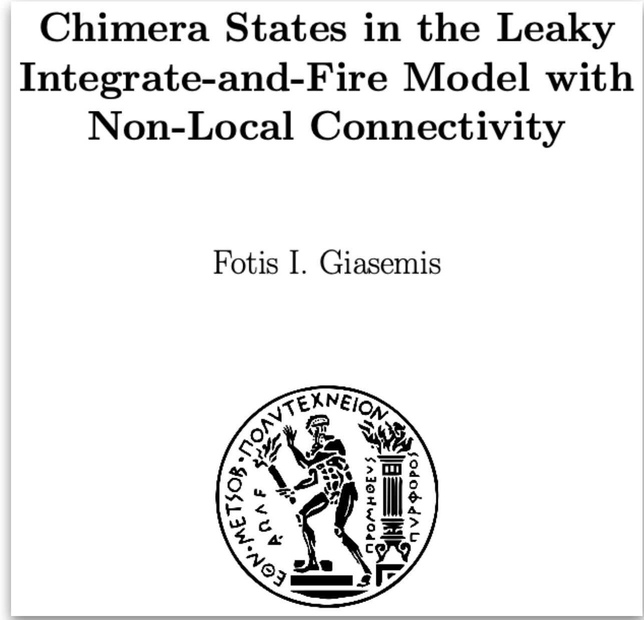
The essential mathematical background and why we care about the mass gap

Candidate Number: 1004234

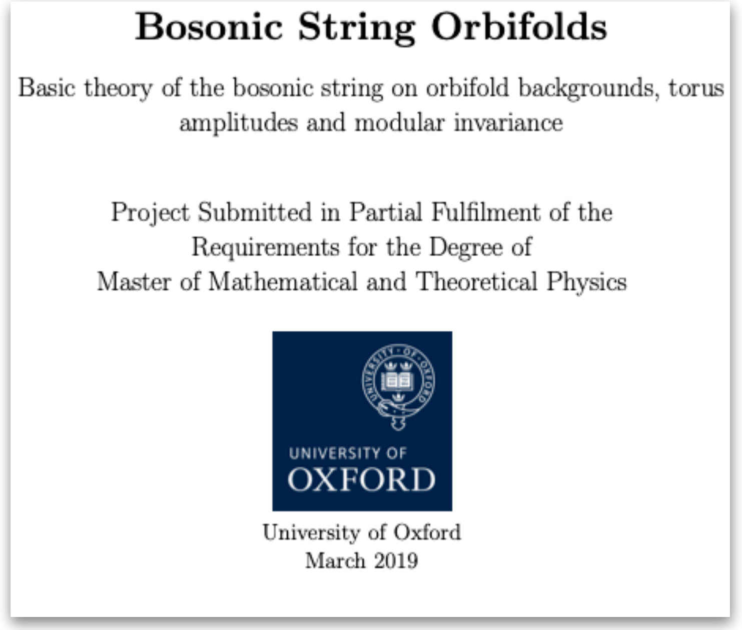

Quantum chaos in many-body systems without a classical analogue

Fotis I. Giasemis




Chimera States in the Leaky Integrate-and-Fire Model with Non-Local Connectivity

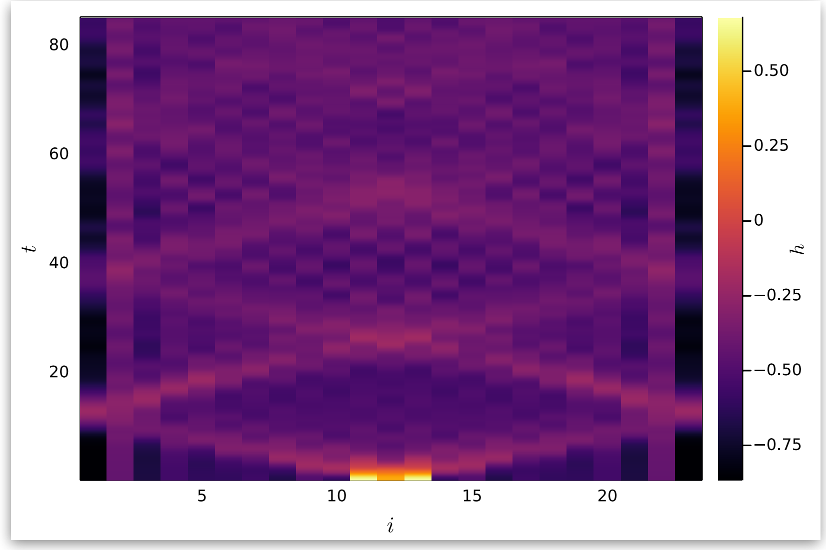
Fotis I. Giasemis

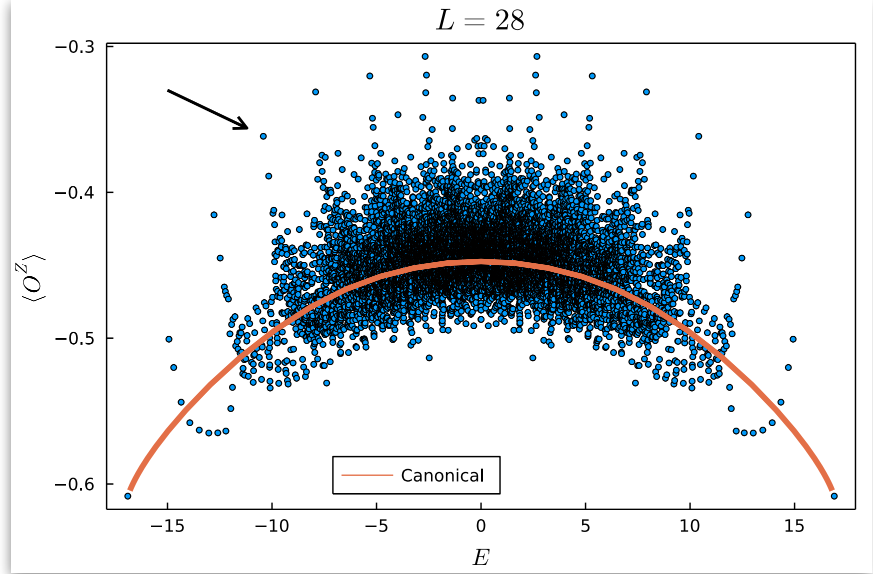

Bosonic String Orbifolds

Basic theory of the bosonic string on orbifold backgrounds, torus amplitudes and modular invariance

Project Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Mathematical and Theoretical Physics

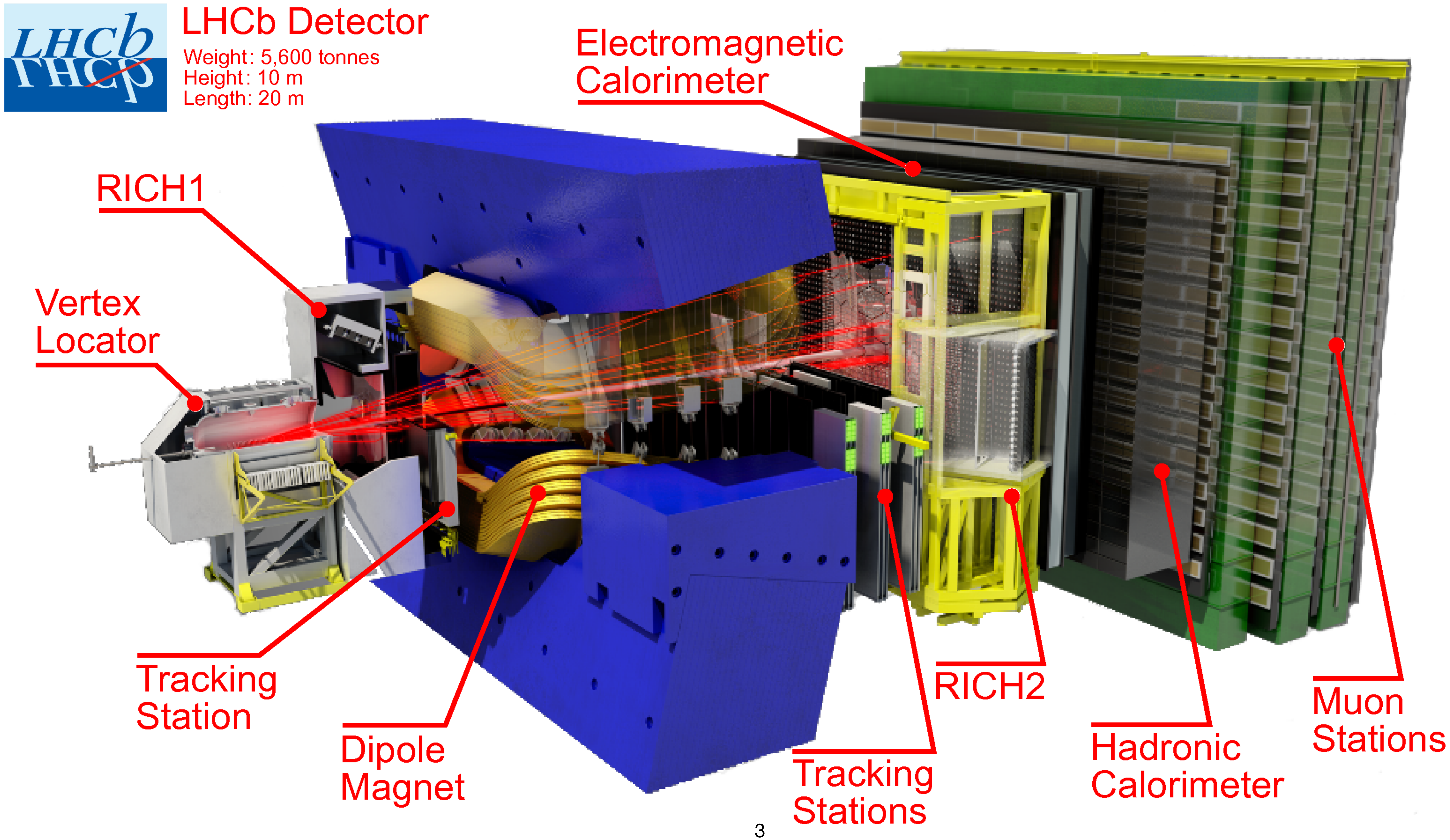
University of Oxford
March 2019





LHCb

The experiment and the detector

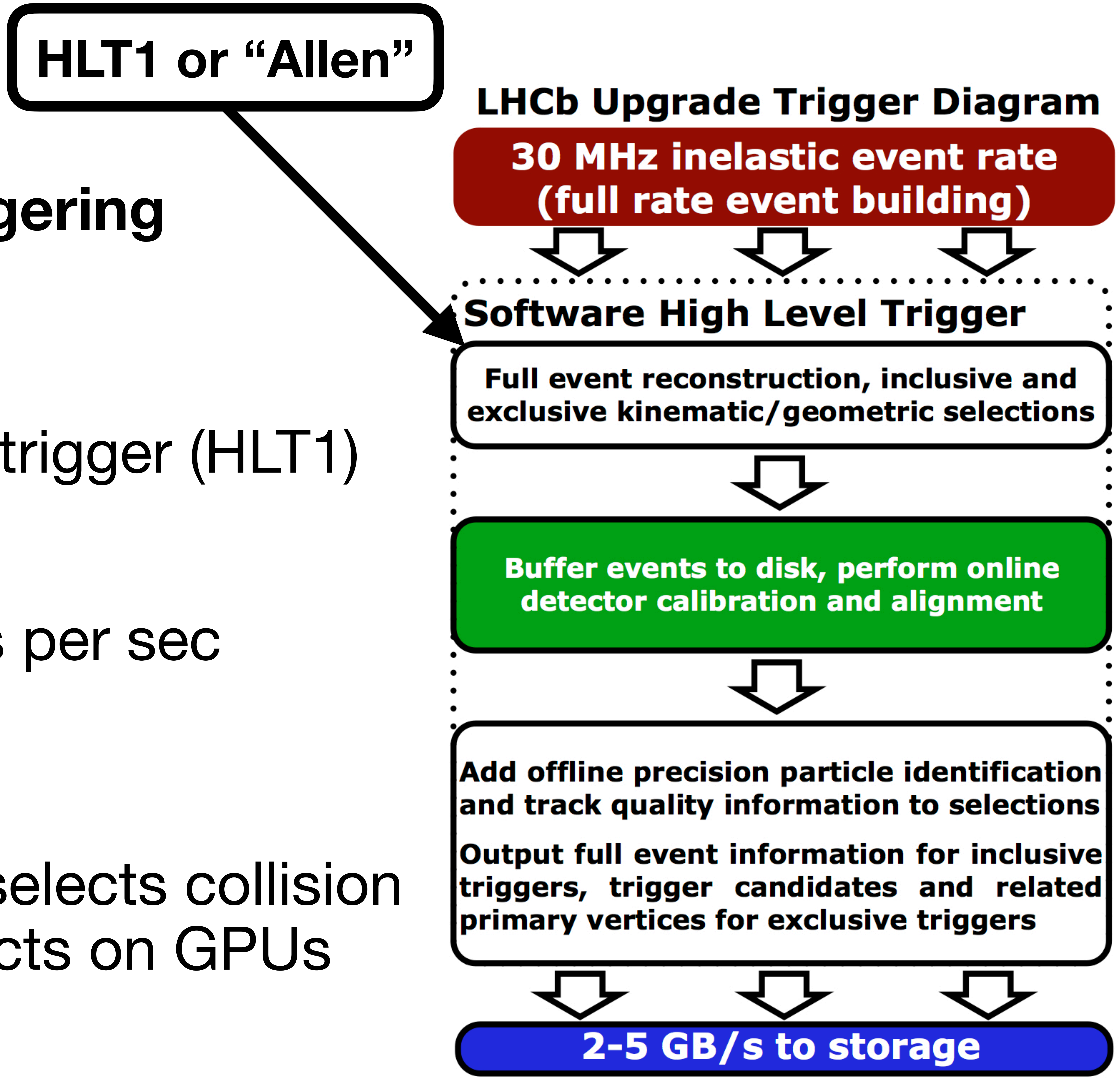


source

The LHCb trigger and Allen

The Software Trigger of LHCb

- Keep only the "interesting" events → **triggering**
- Software high level trigger: 2 levels
- Allen is the level 1 of the LHCb high-level trigger (HLT1) running on **GPUs**
- Filters an input rate of 30 million collisions per sec
- **High throughput constraint**
- Performs fast **track reconstruction** and selects collision events based on one- and two-track objects on GPUs

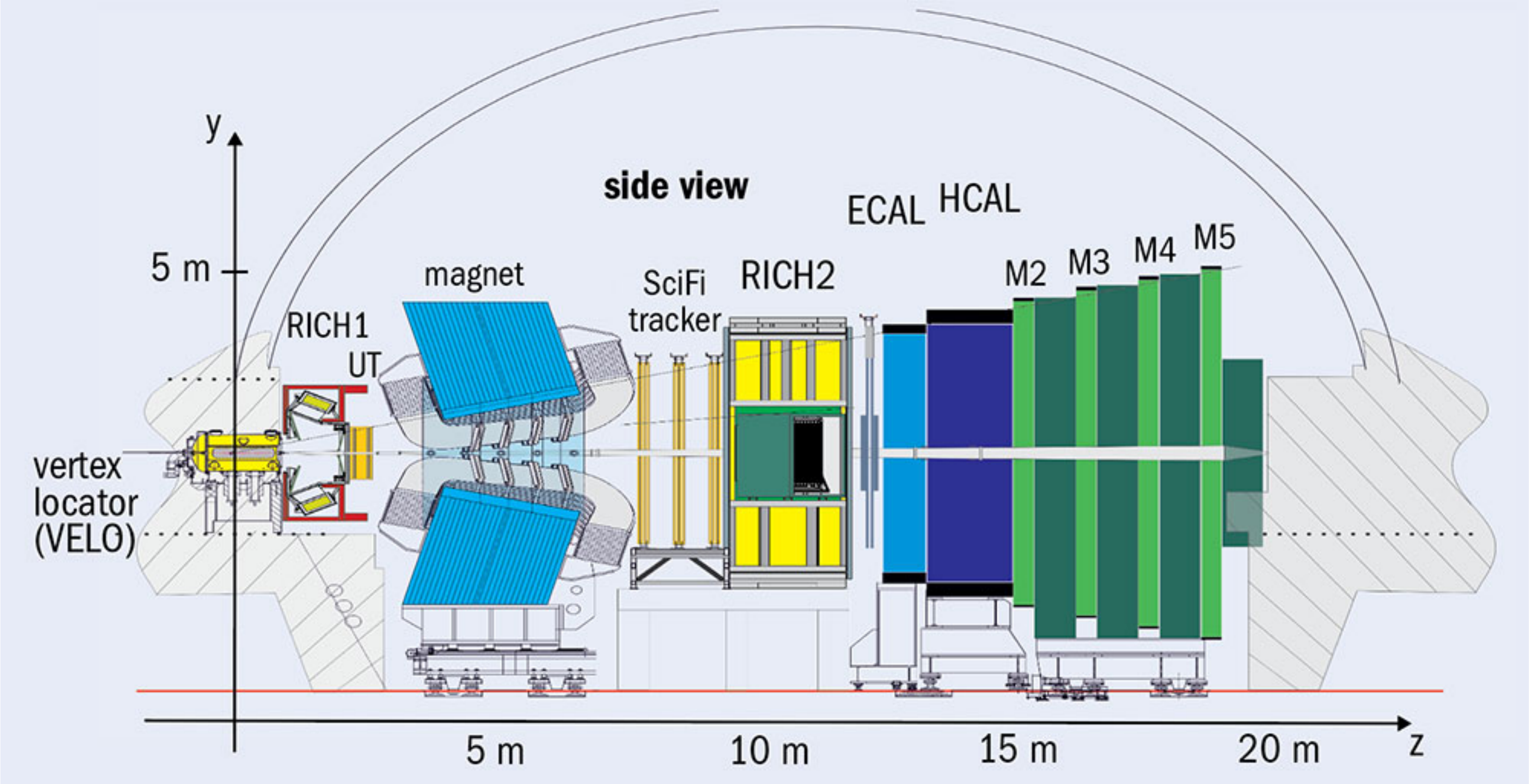


source

Track Finding

Also called
“track reconstruction”, or
“tracking”

Finding tracks from the hits in the detector

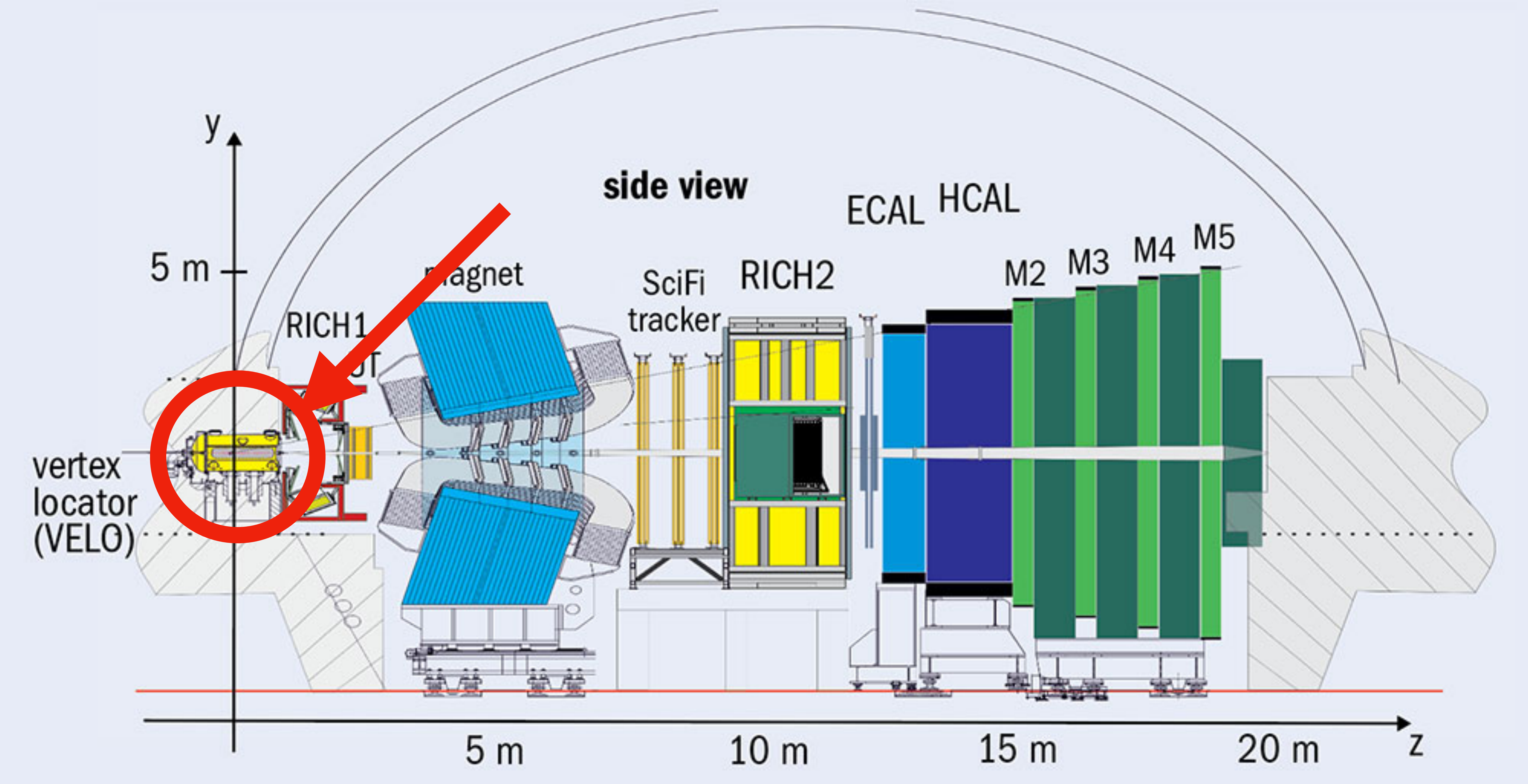


source

Track Finding

Also called
“track reconstruction”, or
“tracking”

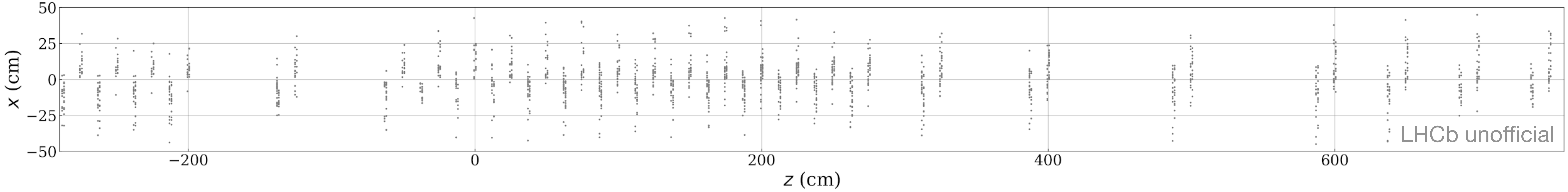
Finding tracks from the hits in the detector

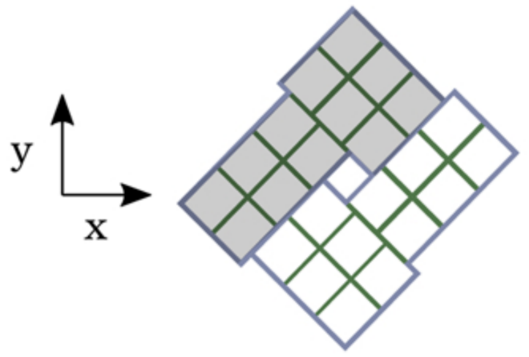


source

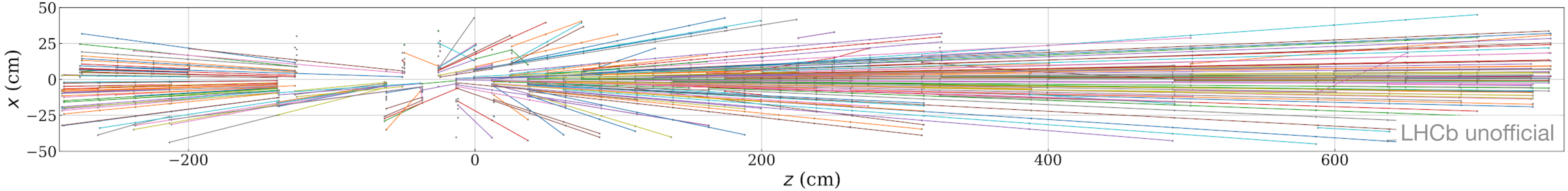
Track Finding

Finding tracks from the hits in the detector





Track finding



Graph Neural Network for Track Finding at LHCb

Main objectives

- Find a NN for tracking at LHCb that achieves state-of-the-art performance

- Optimise network enough in order to meet high throughput constraint

Main objectives

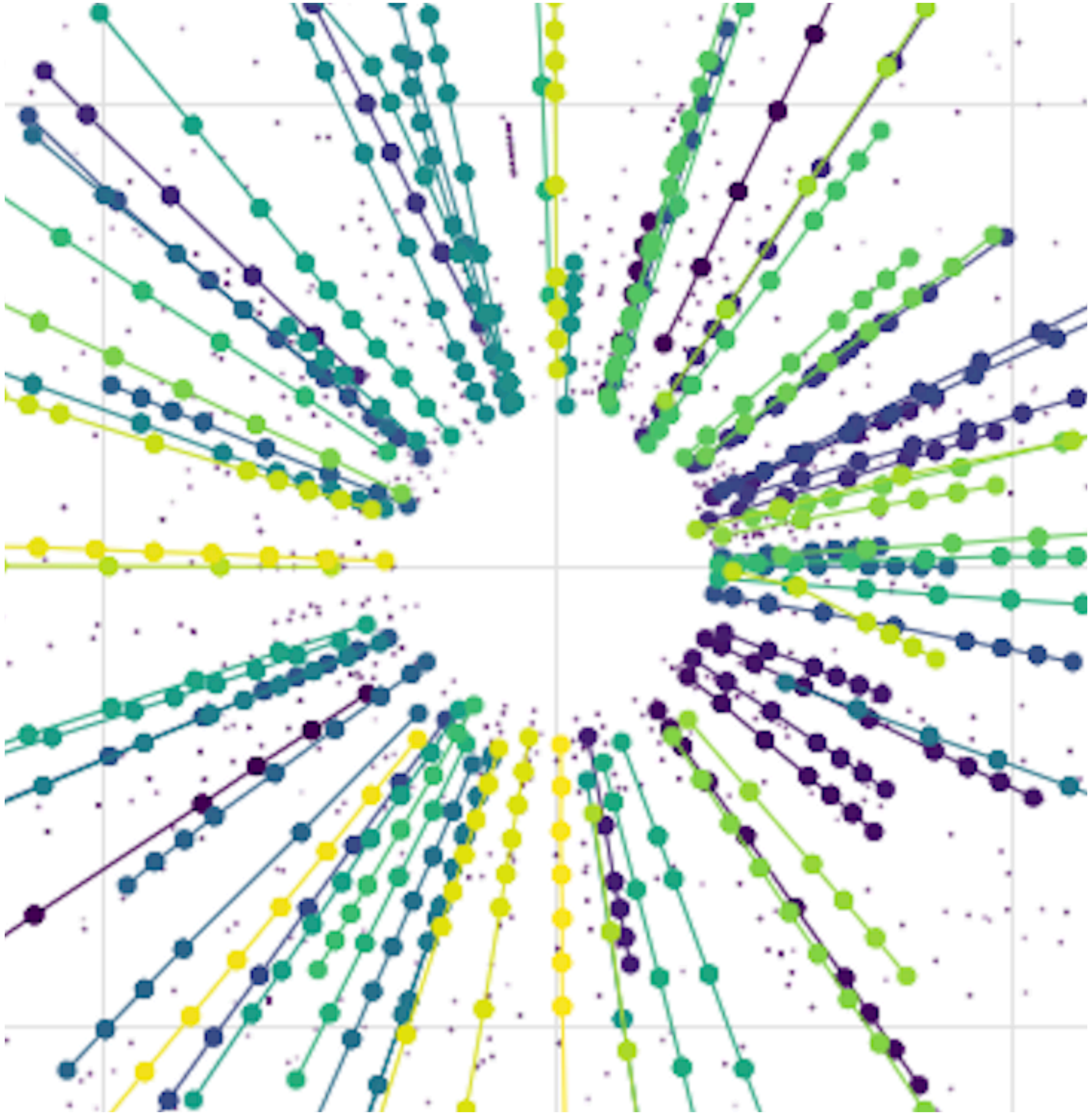
- **Find a NN for tracking at LHCb that achieves state-of-the-art performance**

- Optimise network enough in order to meet high throughput constraint

ETX4VELO

GNN-based pipeline for track finding in the Velo at LHCb, talk@CTD2023,

GitLab repository:
etx4velo@main, etx4velo@dev, etx4velo@ctd2023



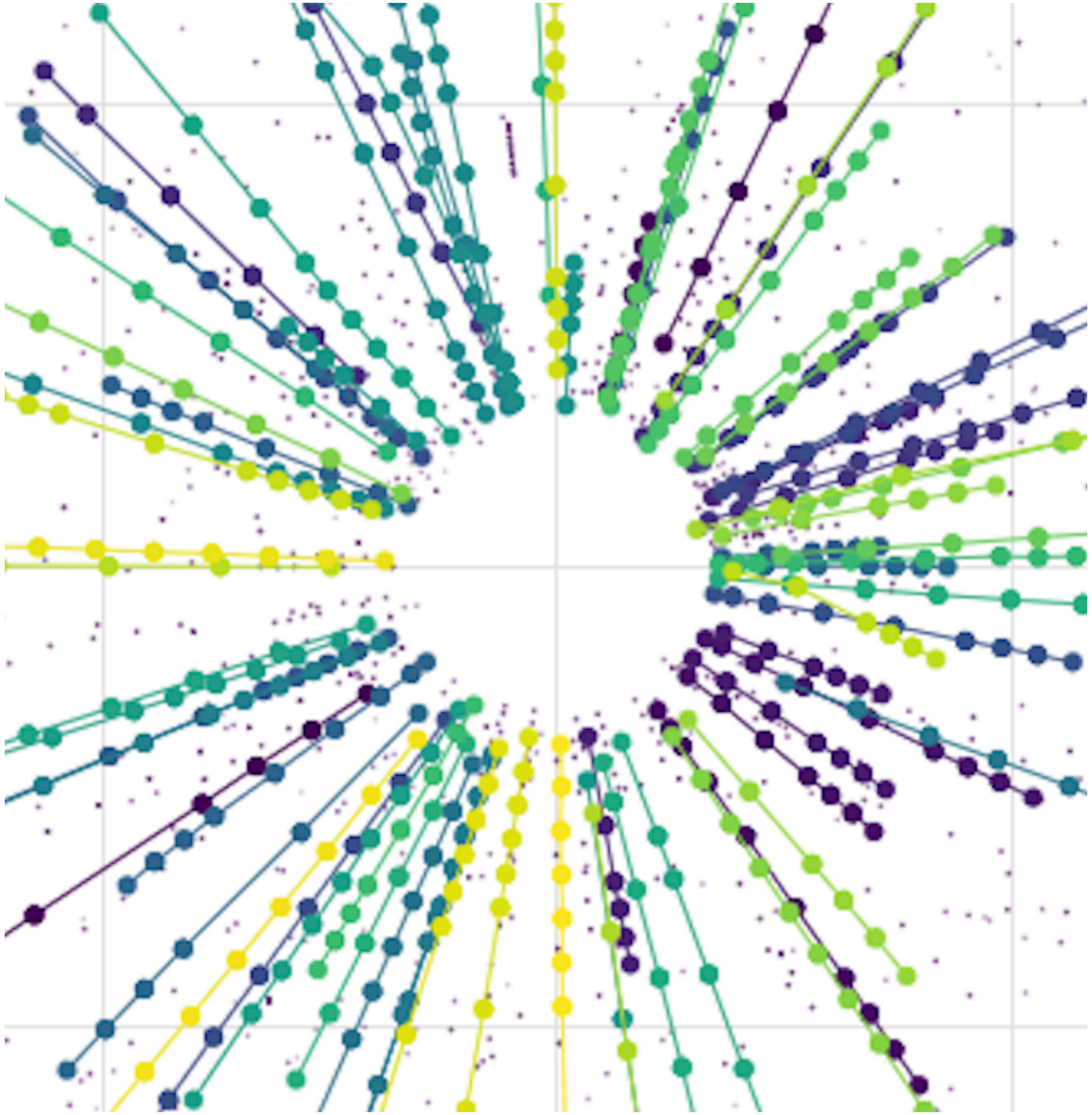
Exa.TrkX

LHCb subdetector

ETX4VELO

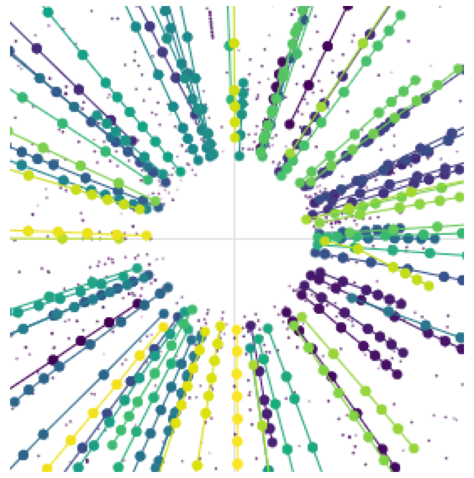
GNN-based pipeline for track finding in the Velo at LHCb, talk@CTD2023,

GitLab repository:
etx4velo@main, etx4velo@dev, etx4velo@ctd2023



ETX4VELO

Graph neural network for track finding in the Velo



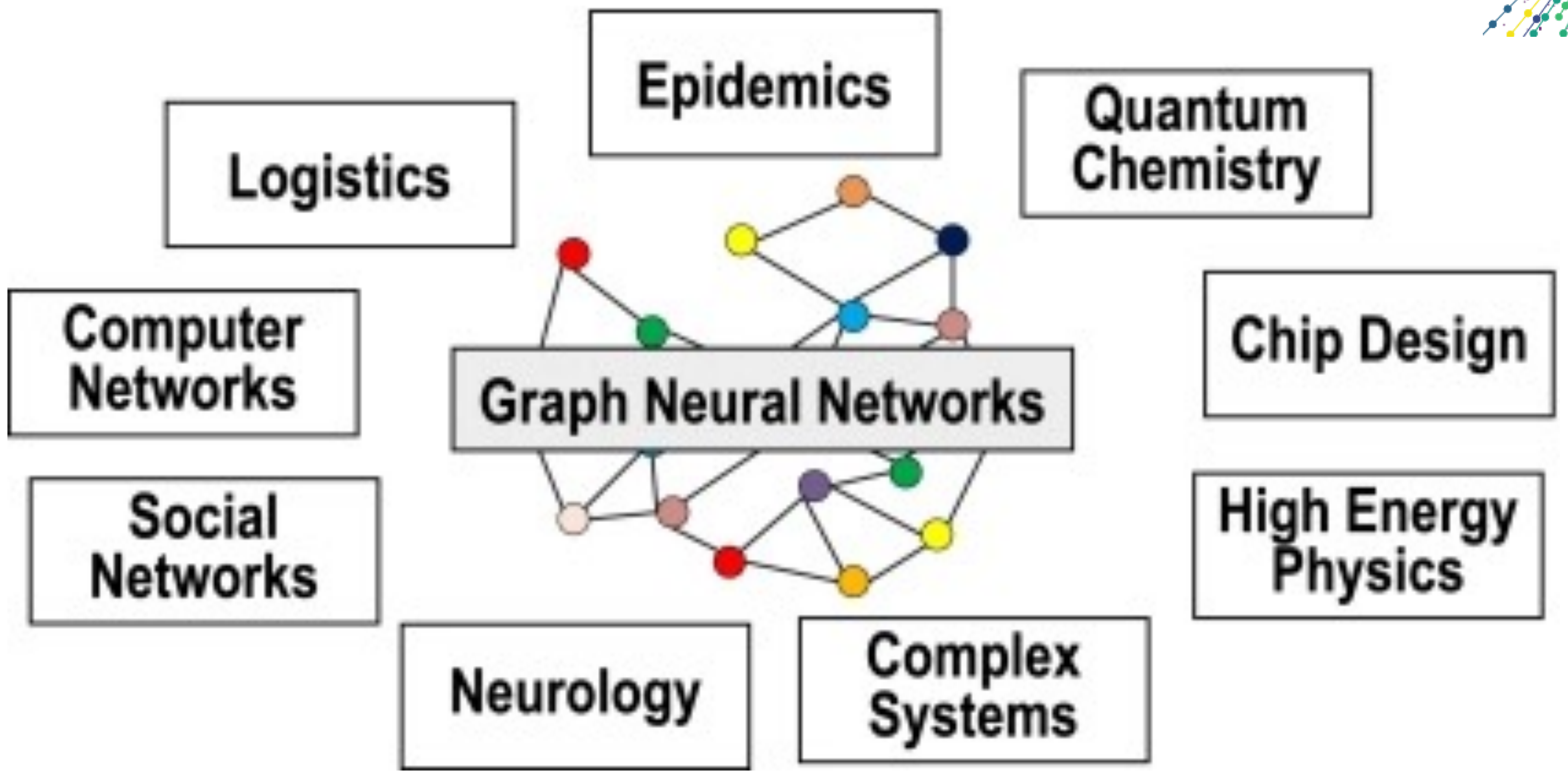
- Why?: Will ML allow a more **efficient** use of computing resources?
- Expected increase in luminosity, next generation of detectors
- Inference time close to **linear** on # hits vs classical **worse-than-quadratic**
- Comparative studies with classical approaches

- Where do we start?: Exa.TrkX collaboration
- exatrkx.github.io, talk@CHEP2021
- PyTorch, PyTorch Geometric, PyTorch Lightning

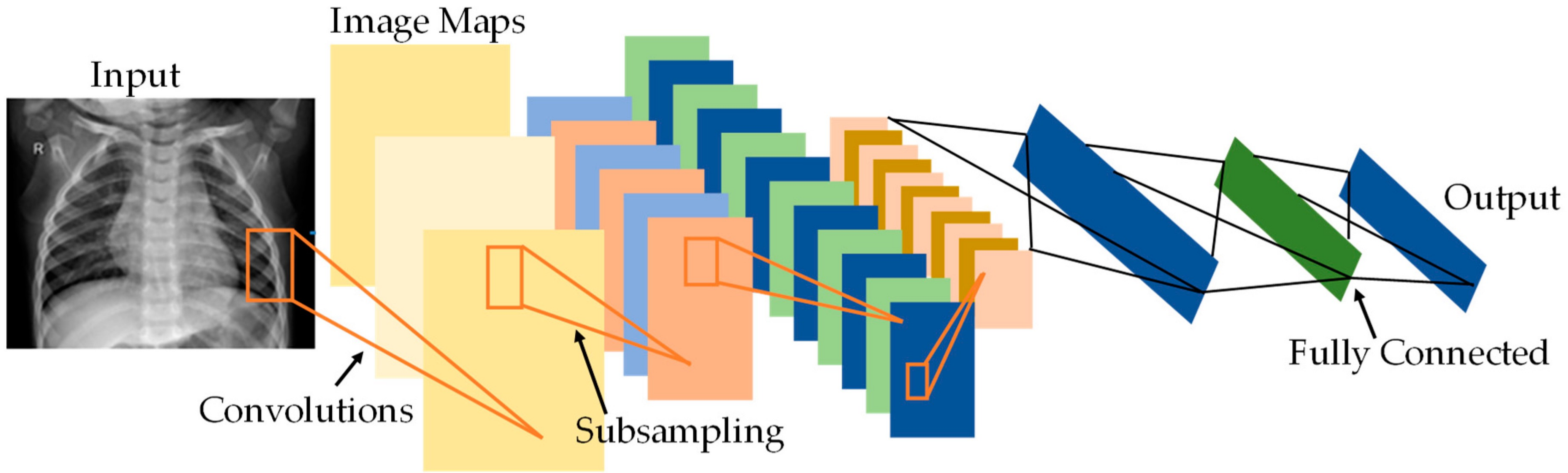
Graph Neural Networks

Why GNNs?

- Why graphs?
  - To take **connectivity** between data into account
- Why GNNs?
  - Modern DL only for structured data (sequences, grids etc.)
  - Develop NNs that are much more broadly applicable
  - Graphs can have **arbitrary shape and size**



source

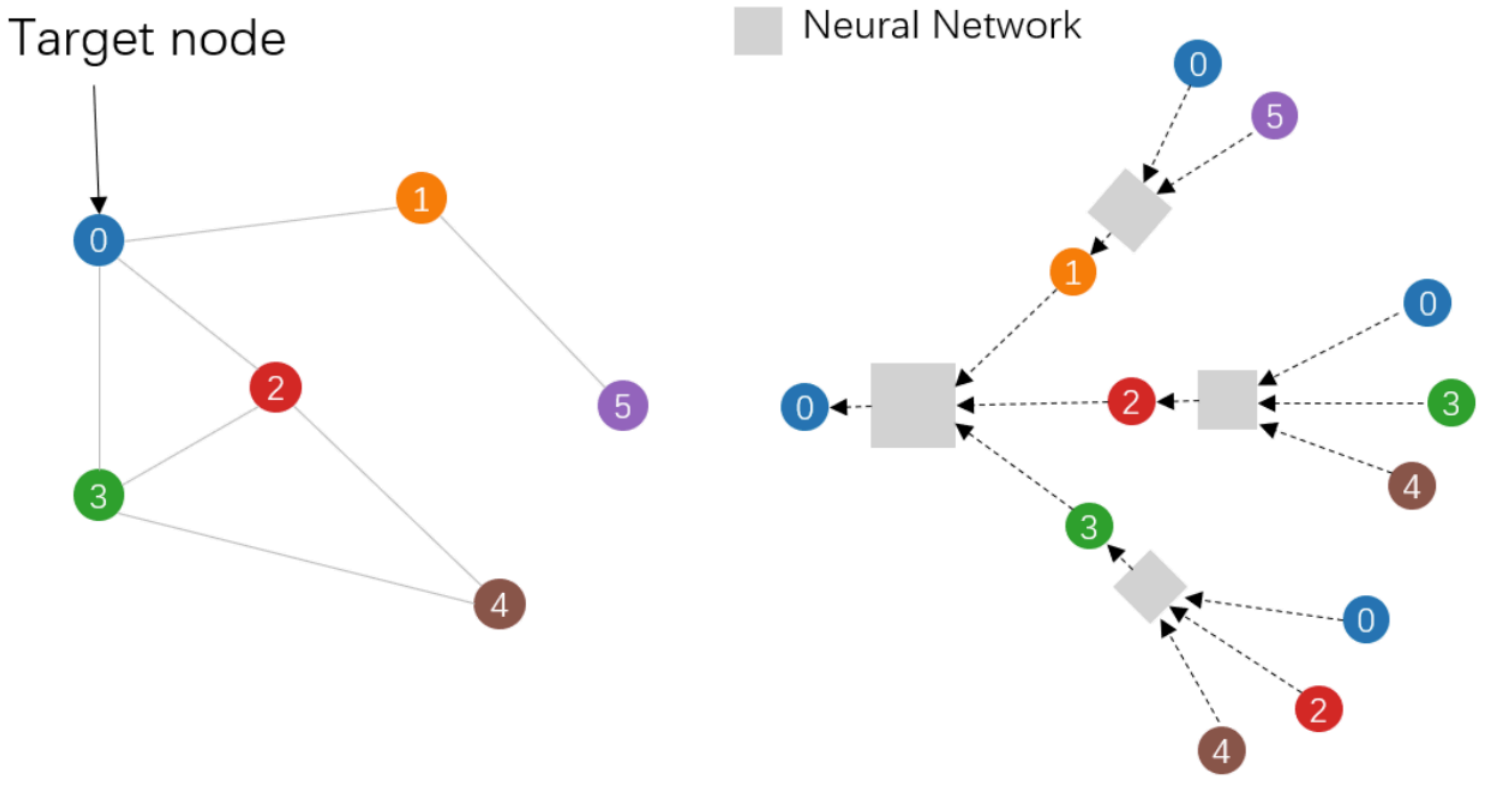


source

Graph Neural Networks

How?

- How do you learn the structure of the data?
  - ~~Normal convolution, as in CNNs~~
  - **“Graph Convolution”**
- Graph Convolution via a computation graph:
  - Node features
  - Aggregation
  - Message passing

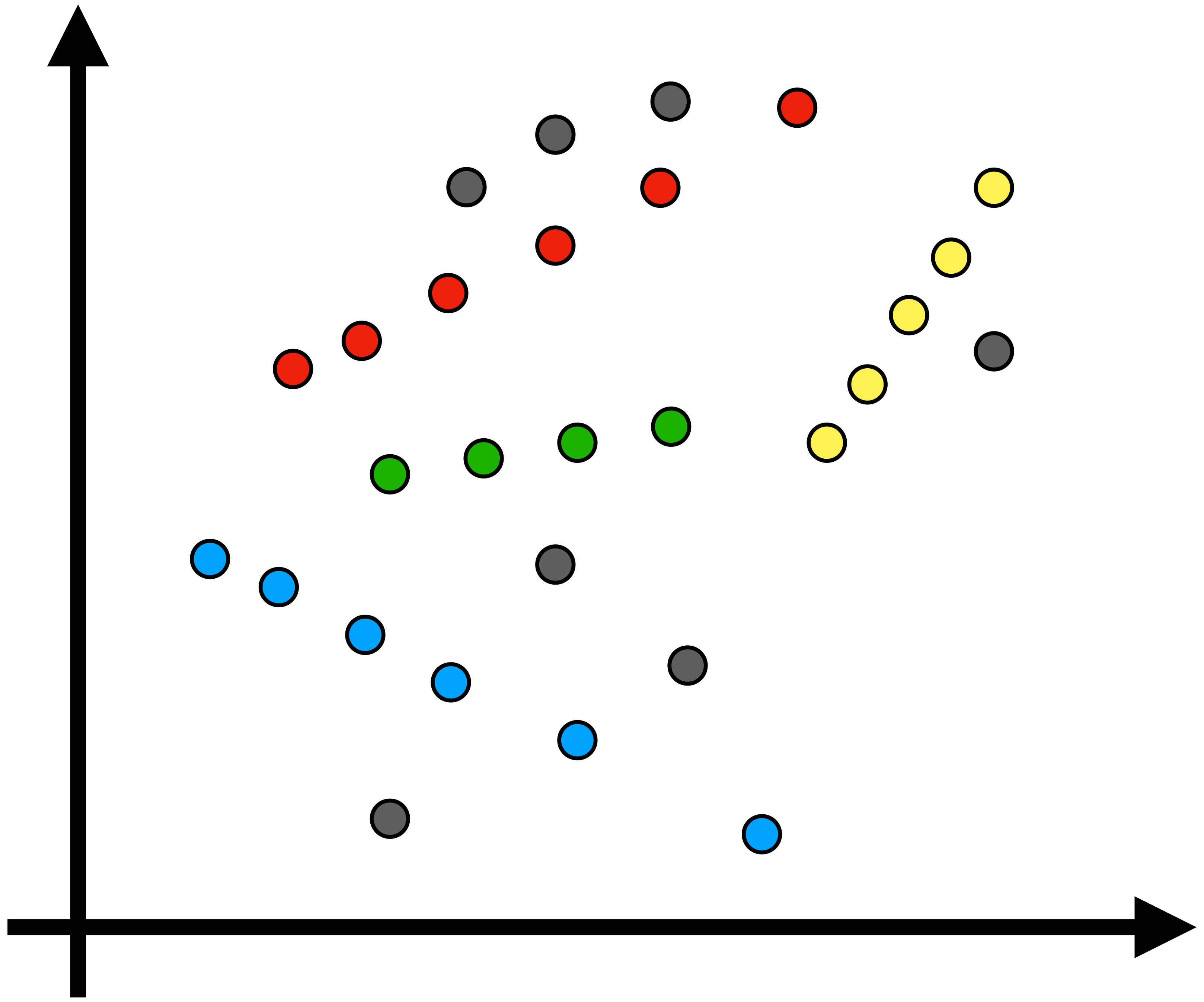


(a) Input graph (b) Neighborhood aggregation

source

ETX4VELO

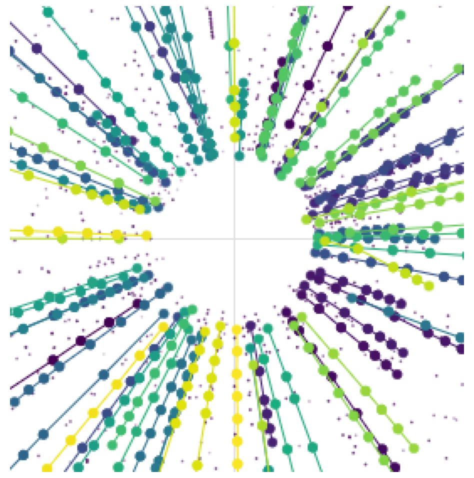
How do we get a graph from the hits?

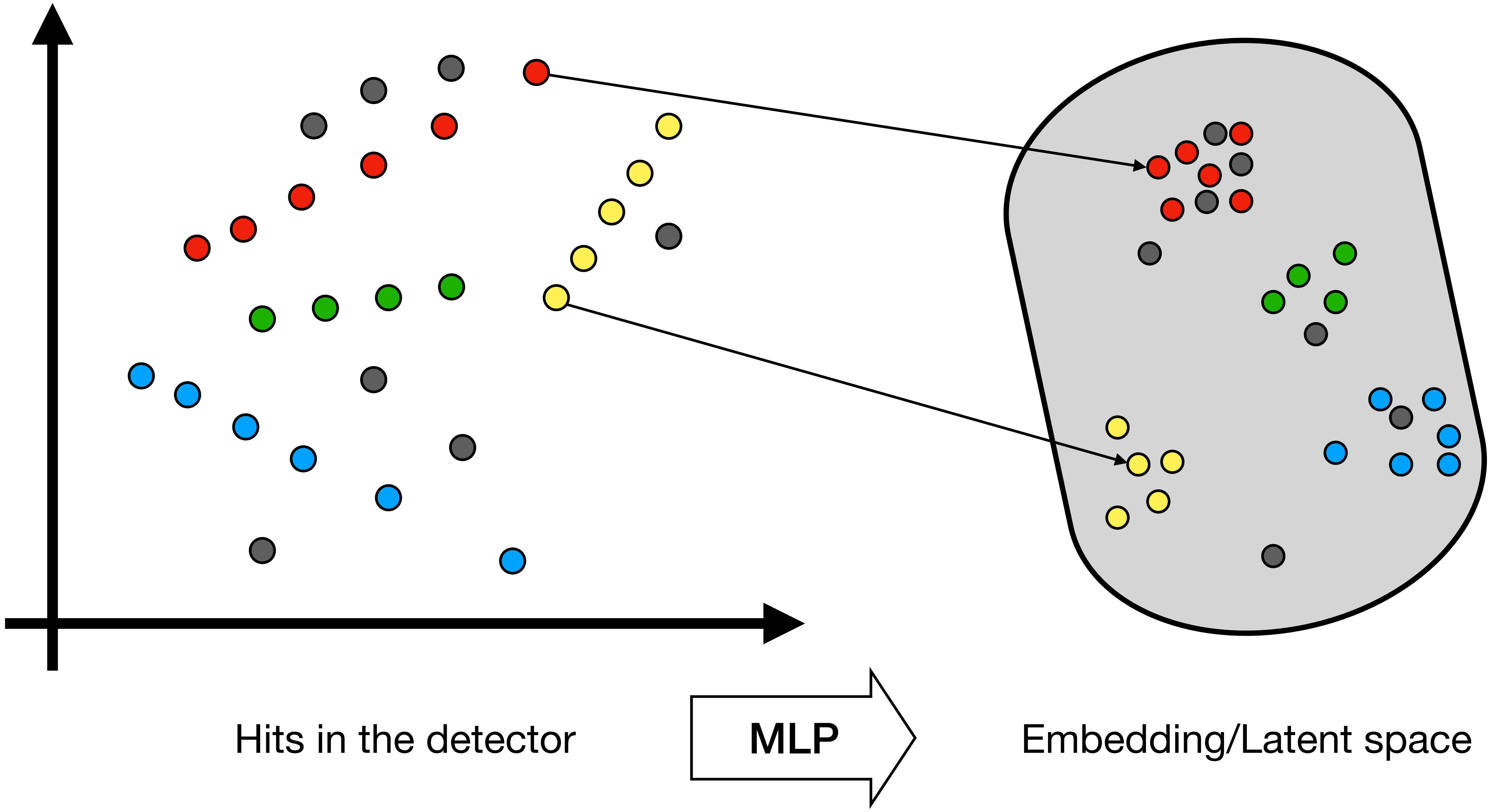


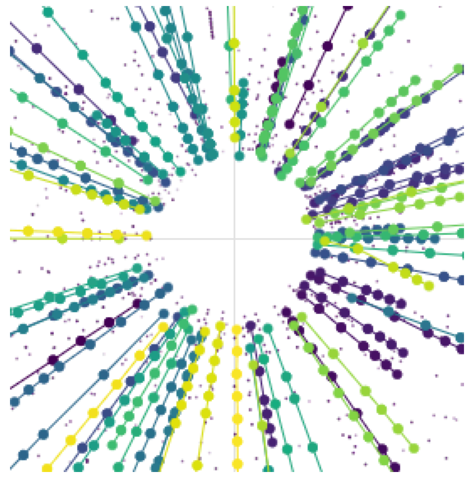
Hits in the detector

ETX4VELO

How do we get a graph from the hits?

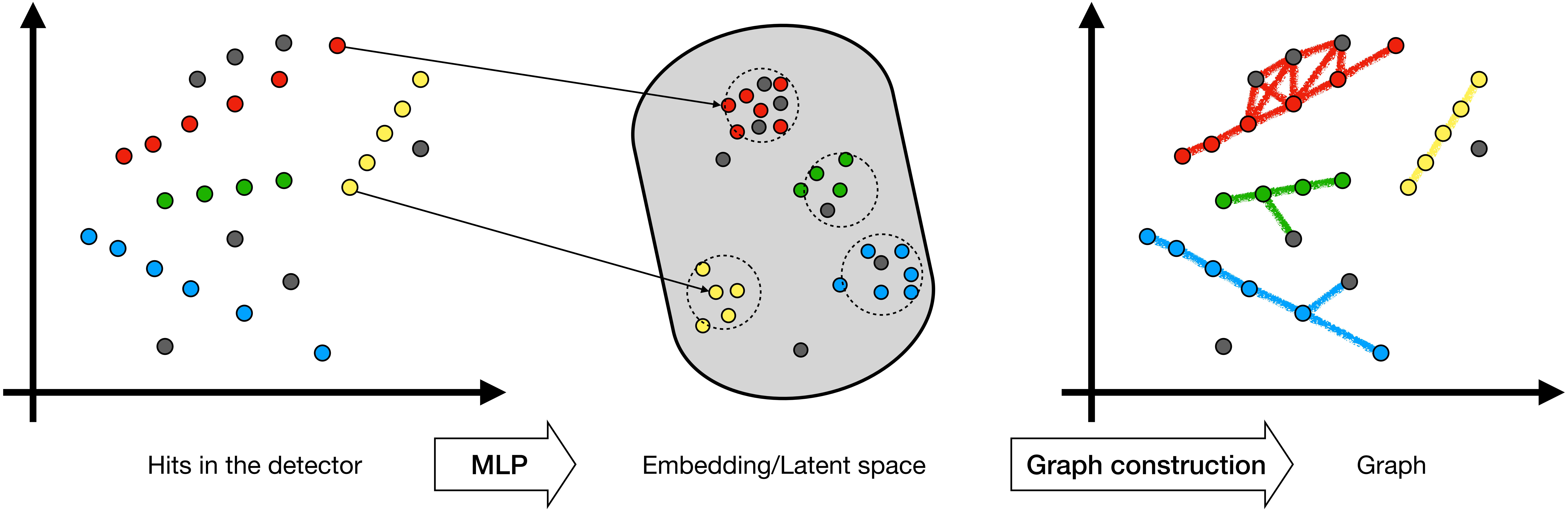


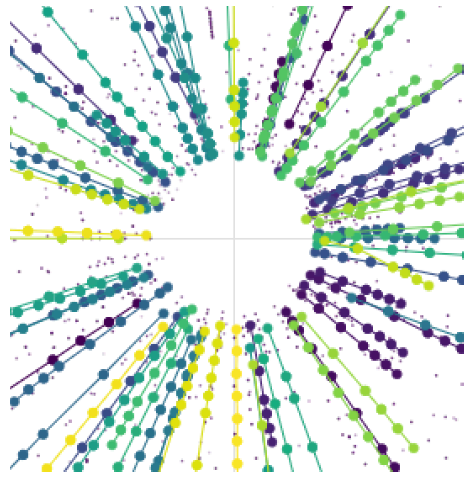




ETX4VELO

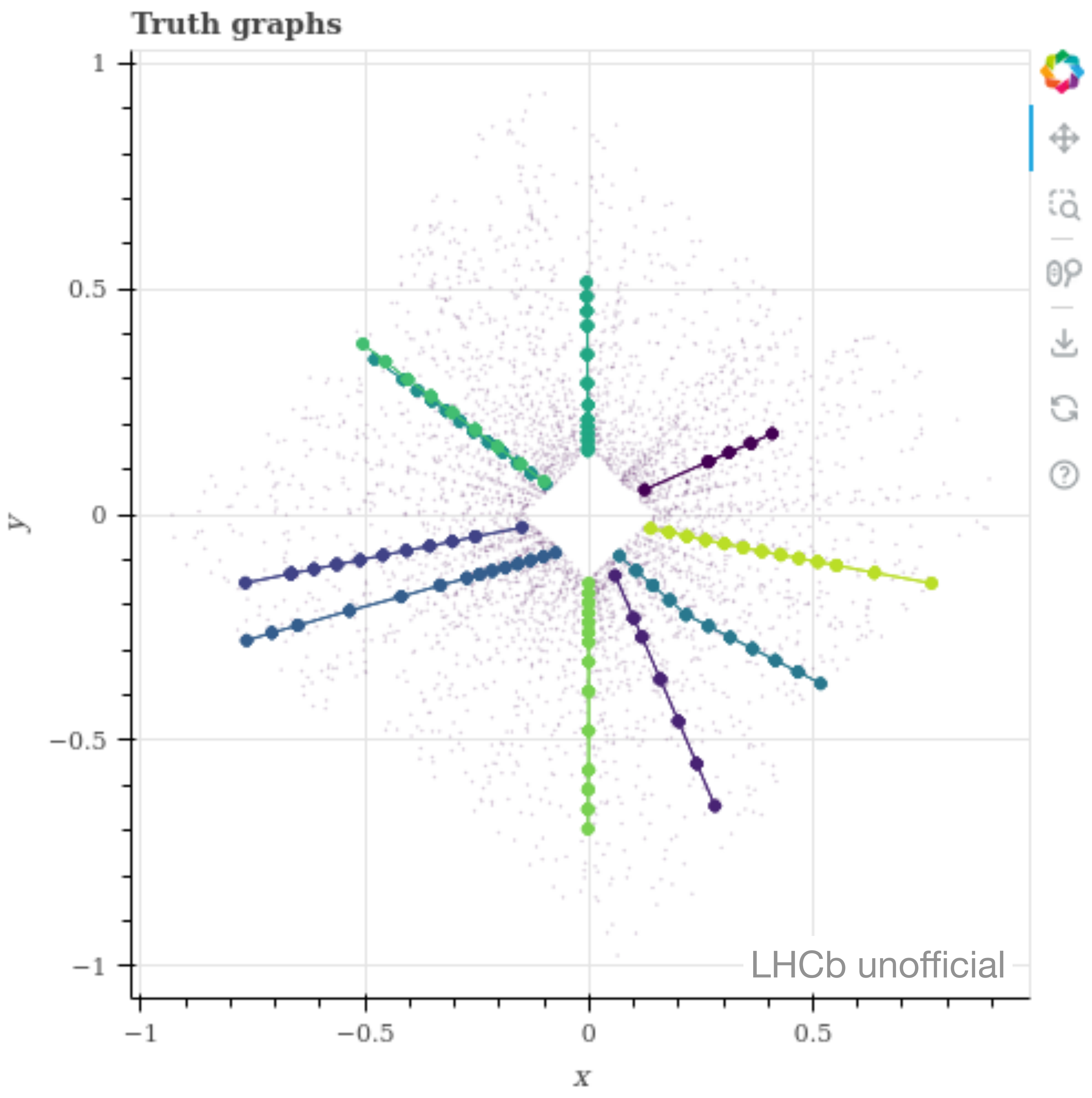
How do we get a graph from the hits?

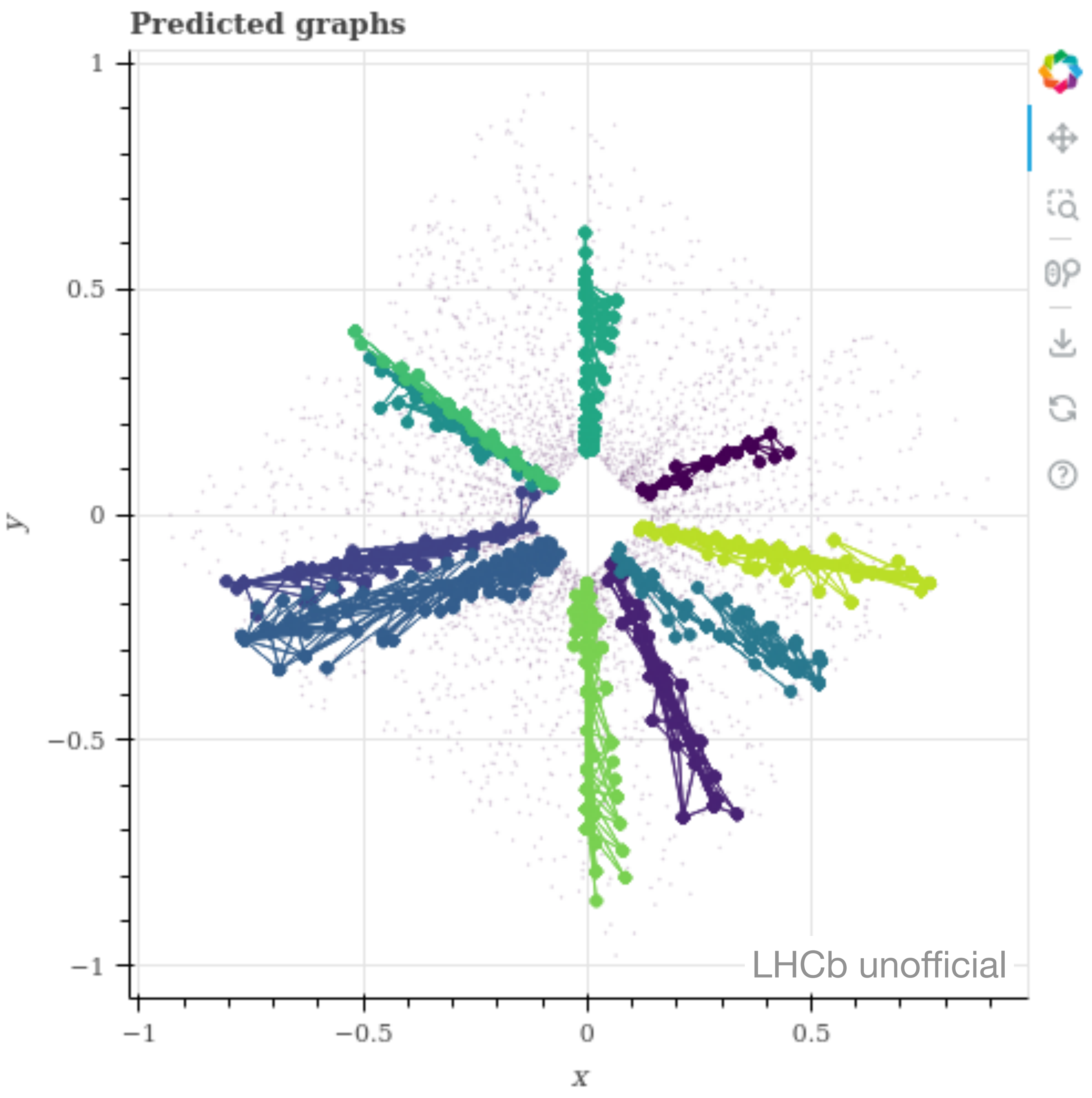




ETX4VELO

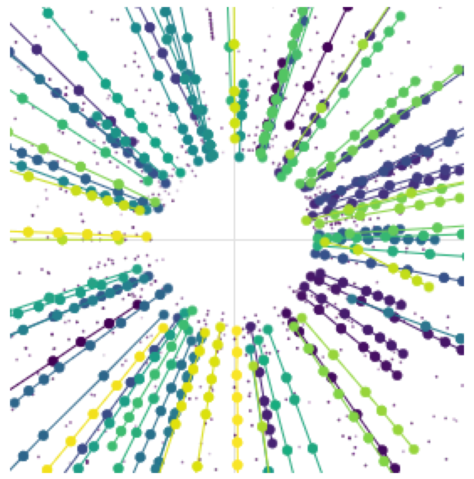
How do we get a graph from the hits?

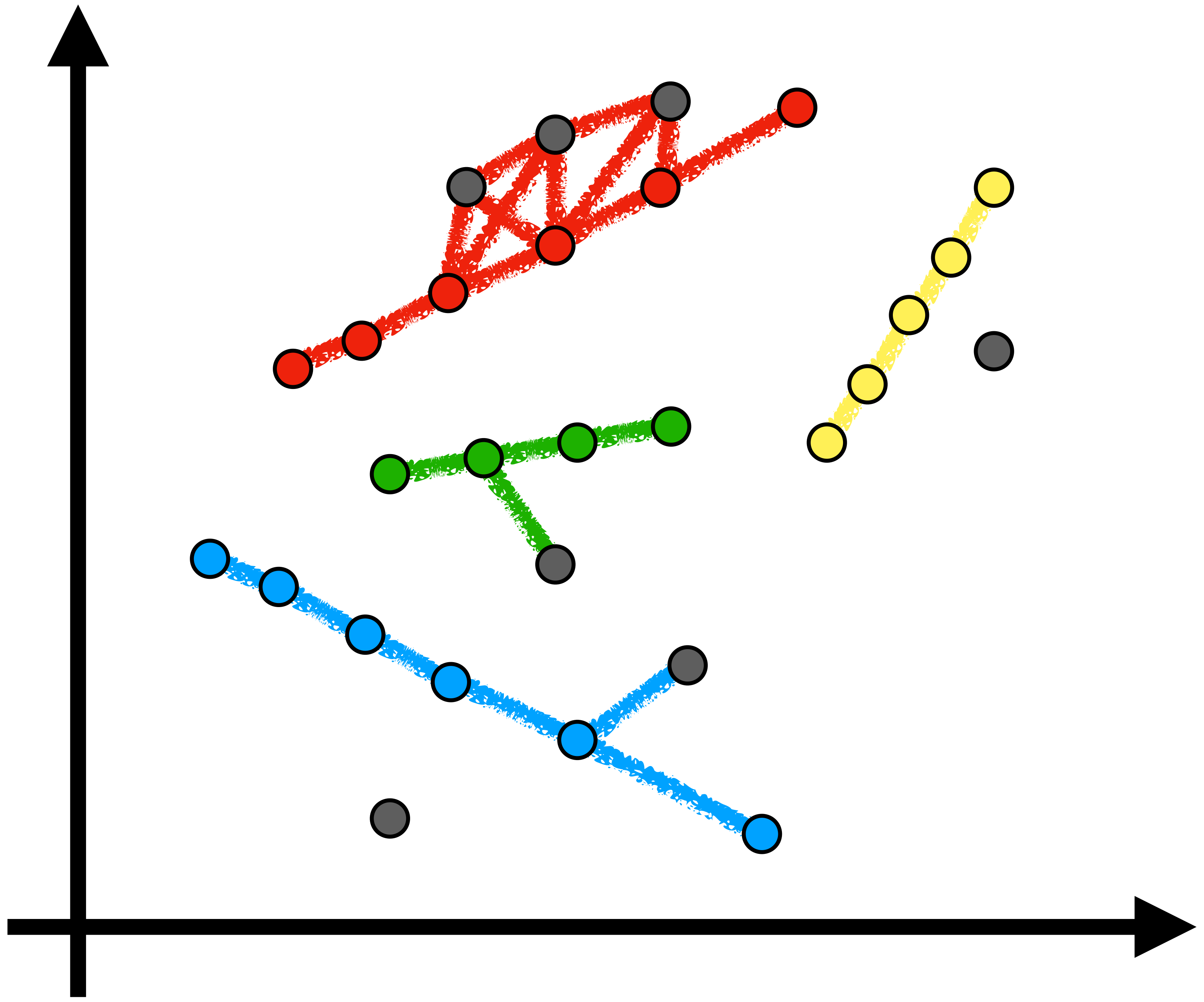




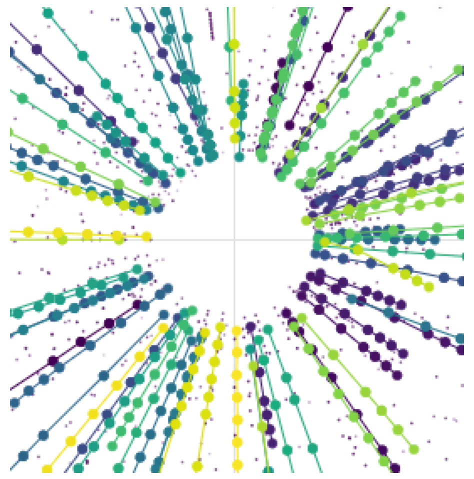
ETX4VELO

How do we get tracks?



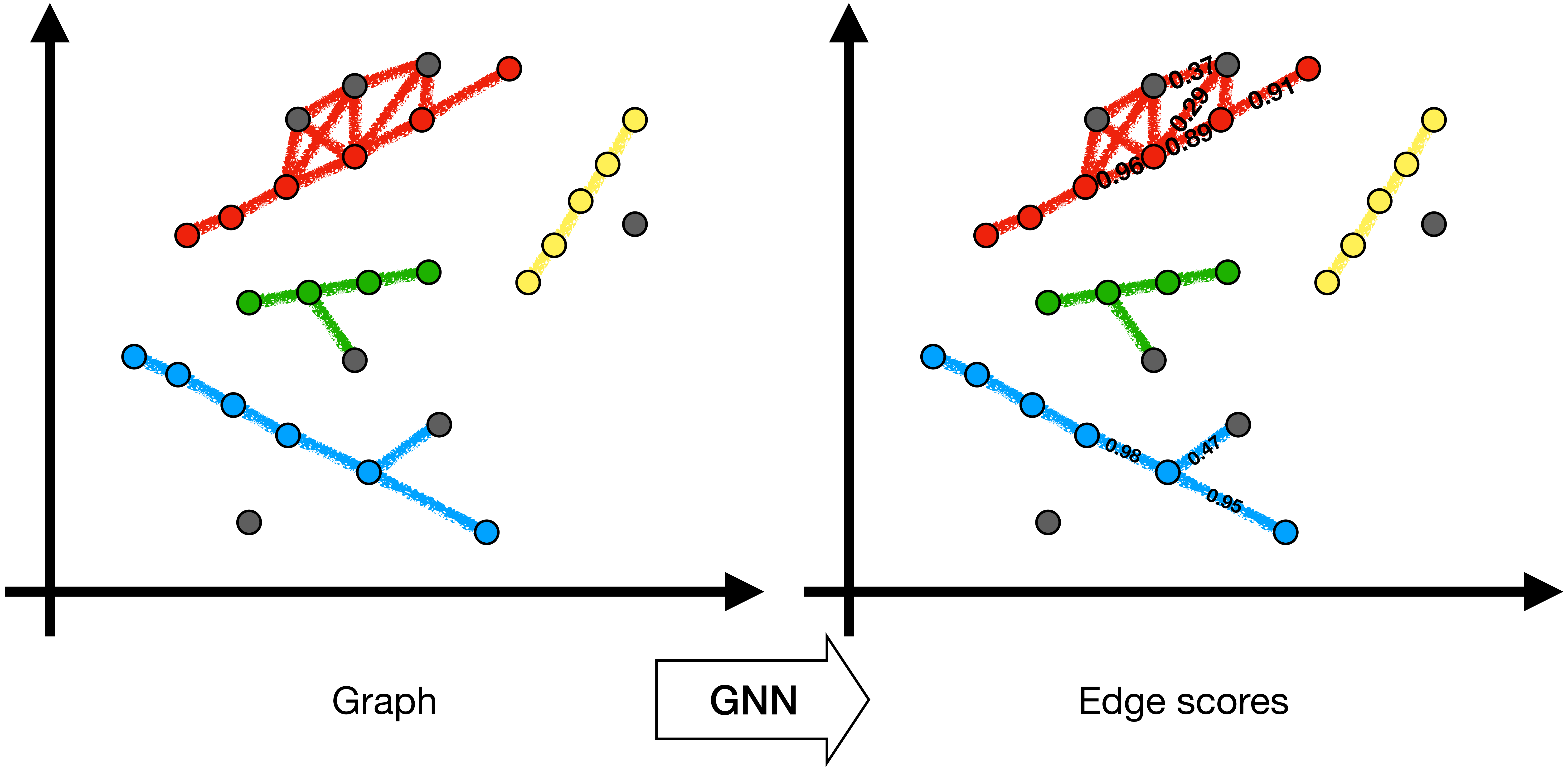


Graph



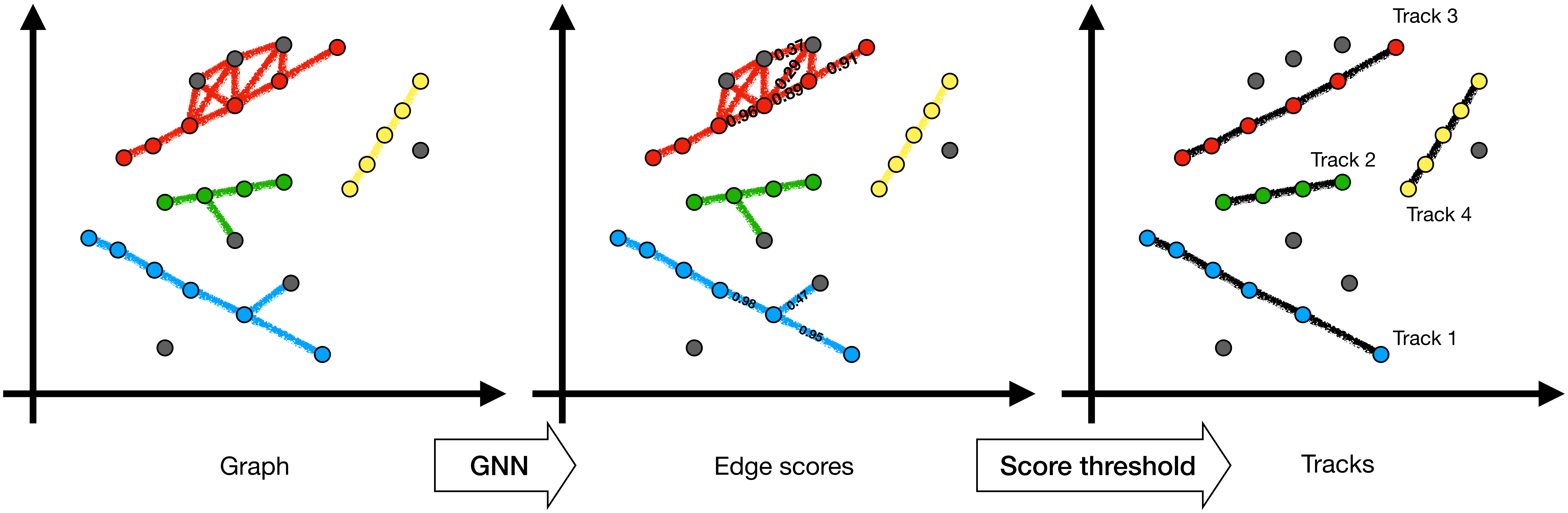
ETX4VELO

How do we get tracks?



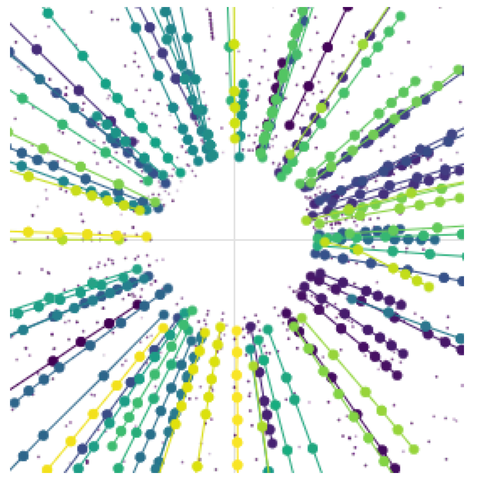
ETX4VELO

How do we get tracks?



ETX4VELO

Refinements



- Performance close to the state of the art
- Problem with electrons:
  - ~ 55% electrons share hits with another electron
  - The 2 electrons share ≥ 1 hit before splitting up
  - Electrons with “long tracks” = **“long electrons”**
  - Important for the LHCb physics program

```
TrackChecker output                   :     38049/  1117828   3.40% ghosts
01_velo                               :    491643/   520515  94.45% ( 95.11%),
02_long                               :    286719/   296345  96.75% ( 97.22%),
03_long_P>5GeV                        :    185866/   189727  97.96% ( 98.30%),
04_long_strange                       :     13654/    15243  89.58% ( 90.68%),
05_long_strange_P>5GeV                :      6606/     7229  91.38% ( 92.00%),
06_long_fromB                         :       497/      513  96.88% ( 96.86%),
07_long_fromB_P>5GeV                  :       335/      343  97.67% ( 97.82%),
08_long_electrons                     :     16634/    21330  77.98% ( 78.93%),
09_long_fromB_electrons               :        41/       58  70.69% ( 76.42%),
10_long_fromB_electrons_P>5GeV        :        30/       38  78.95% ( 81.18%),

*** Benchmark score:  94.01
```

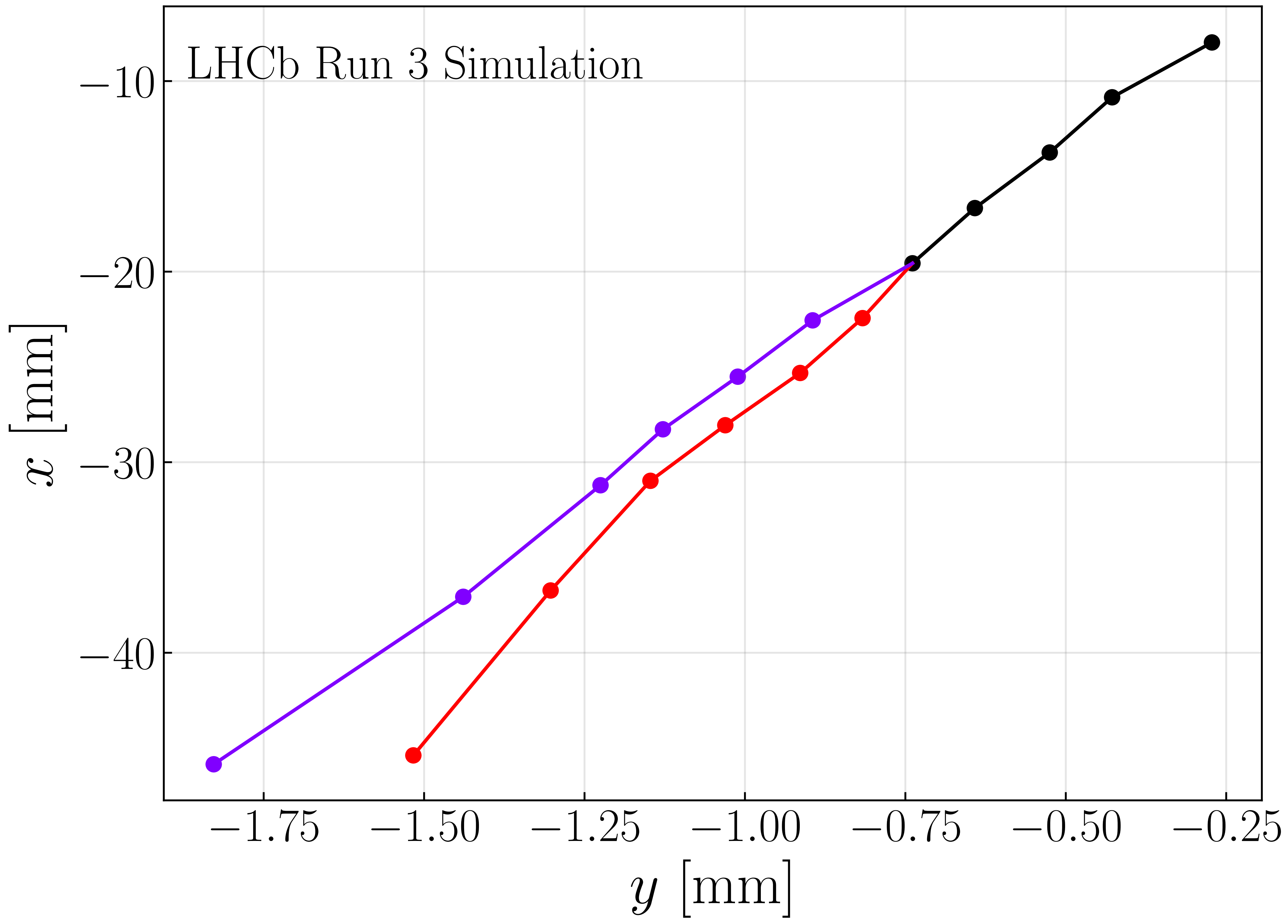
Categories	Efficiency	Average efficiency	% clones	Average hit pur
Velo	90.37%	91.08%	1.41%	99.03%
Long	95.49%	95.97%	0.97%	99.33%
Velo, no electrons	94.45%	95.11%	0.89%	99.30%
Velo, only electrons	69.30%	69.84%	4.91%	97.15%
Long, only electrons	77.98%	78.93%	3.54%	97.36%

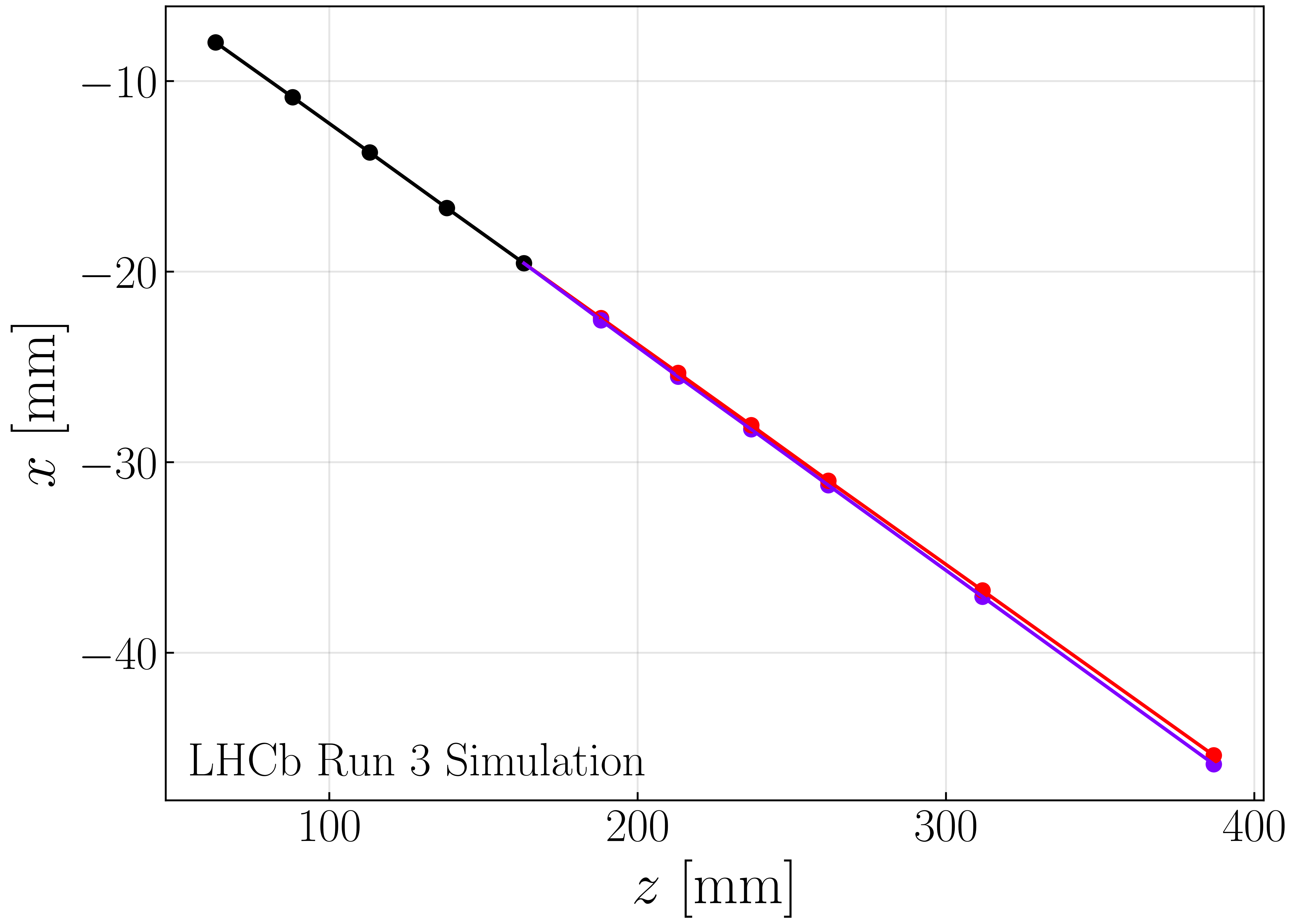
Categories	# ghosts	# tracks	% ghosts
Everything	38,049	1,117,828	3.40%

LHCb unofficial

ETX4VELO

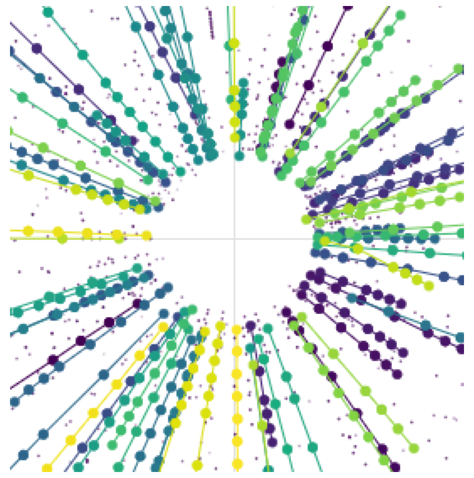
Problem with electrons: shared hits



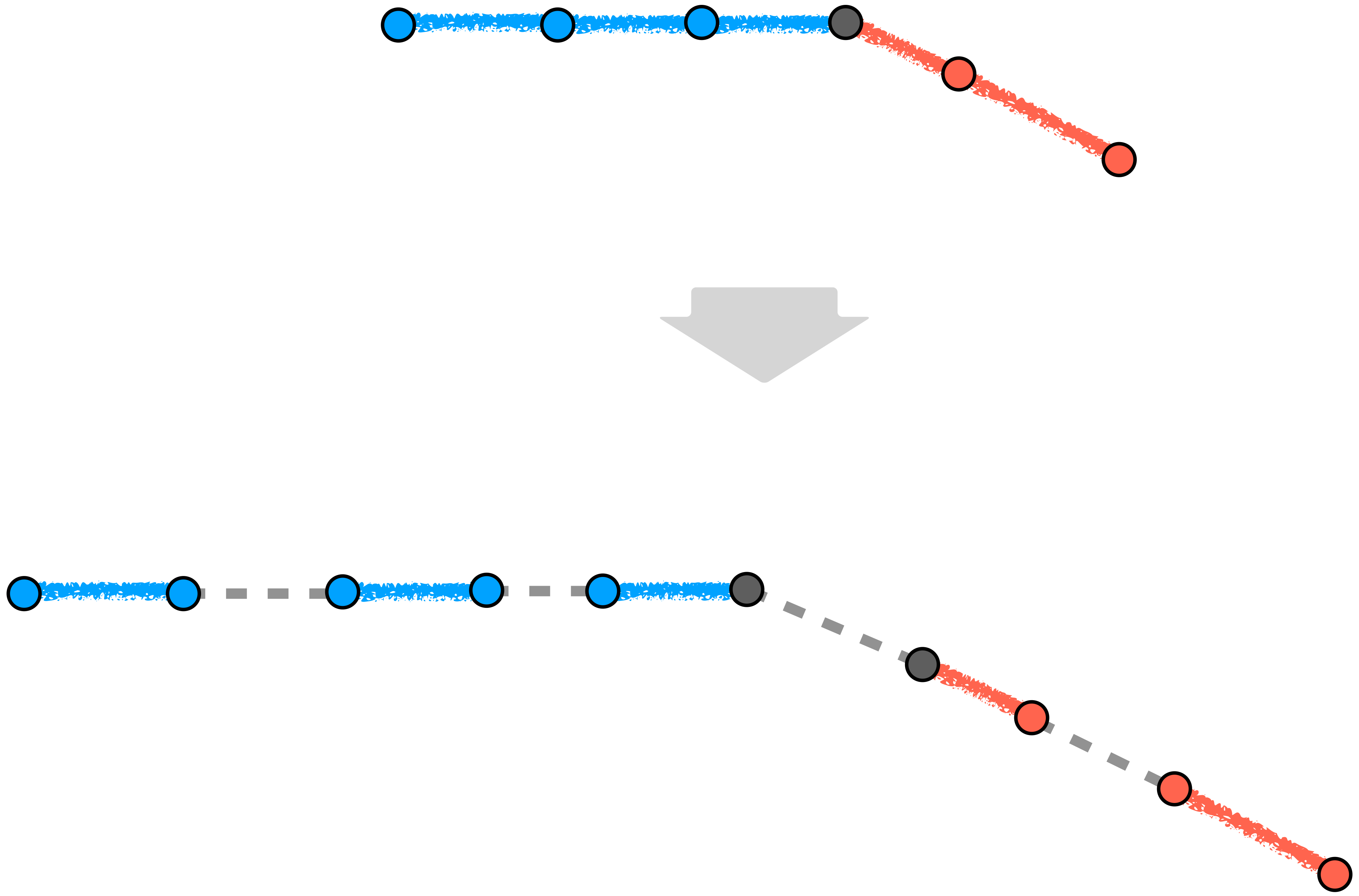


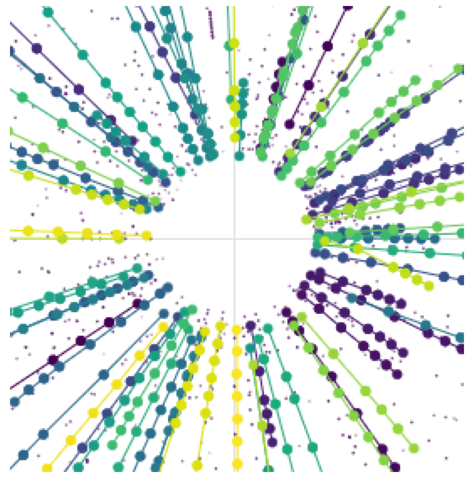
ETX4VELO

Problem with electrons: the solution



- Problem with electrons:
  - Pipeline cannot separate particle with shared edges
  - Hit-hit connections are not enough
  - Solution:
  - Use edge-edge connections (triplets)
  - Use GNN again on triplets





ETX4VELO

Already outperforming the state of the art

```
TrackChecker output                          :     1736/   254023   0.68% ghosts
01_velo                                      :   102725/   104345  98.45% ( 98.48%),      1059 (  1.02%) clones, pur  99.81%, hit eff  98.66%
02_long                                      :    58771/    59167  99.33% ( 99.30%),       566 (  0.95%) clones, pur  99.89%, hit eff  98.93%
03_long_P>5GeV                               :    38035/    38150  99.70% ( 99.65%),       296 (  0.77%) clones, pur  99.91%, hit eff  99.21%
04_long_strange                              :     3066/     3142  97.58% ( 97.64%),        41 (  1.32%) clones, pur  99.48%, hit eff  98.55%
05_long_strange_P>5GeV                       :     1485/     1521  97.63% ( 97.45%),        10 (  0.67%) clones, pur  99.38%, hit eff  99.46%
06_long_fromB                                :      120/      120 100.00% (100.00%),         0 (  0.00%) clones, pur 100.00%, hit eff 100.00%
07_long_fromB_P>5GeV                         :       87/       87 100.00% (100.00%),         0 (  0.00%) clones, pur 100.00%, hit eff 100.00%
08_long_electrons                            :     4169/     4198  99.31% ( 99.44%),       379 (  8.33%) clones, pur  98.39%, hit eff  96.38%
09_long_fromB_electrons                      :       10/       10 100.00% (100.00%),         0 (  0.00%) clones, pur 100.00%, hit eff 100.00%
10_long_fromB_electrons_P>5GeV               :        7/        7 100.00% (100.00%),         0 (  0.00%) clones, pur 100.00%, hit eff 100.00%
```

LHCb unofficial

GDL4HEP > etx4velo

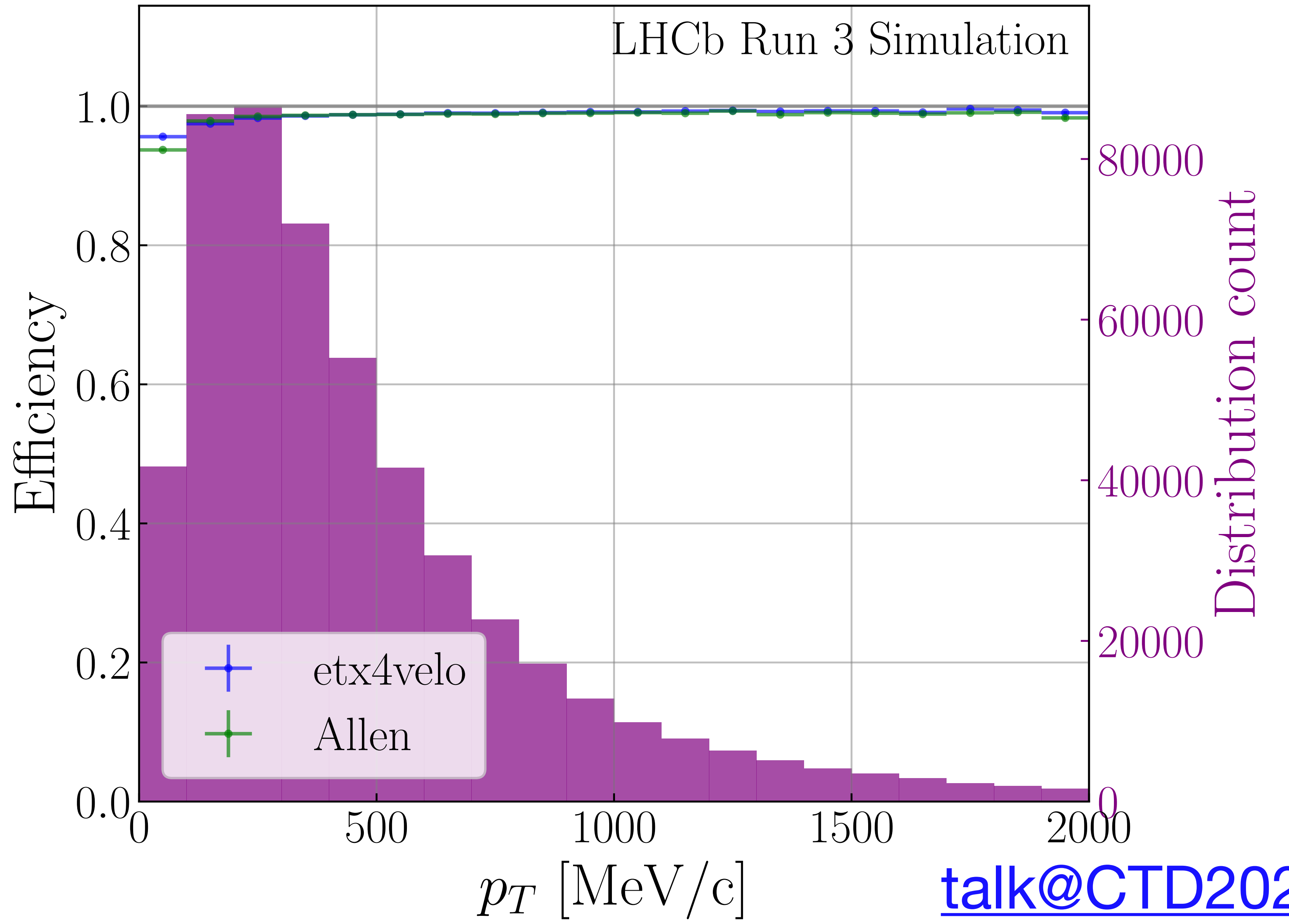
etx4velo

Project ID: 154495

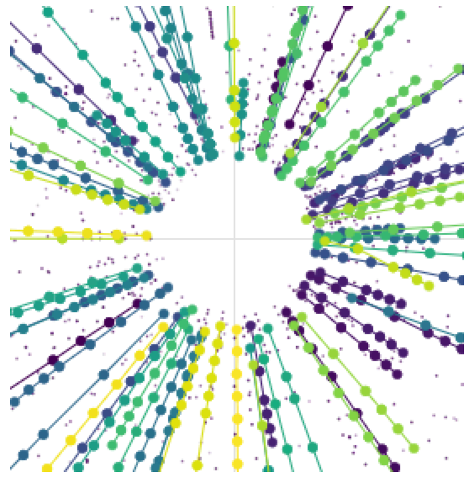
738 Commits · **13** Branches · **4** Tags · **1.1 GiB** Project Storage · **1** Release · **1** Environment

Topics: LHCb · python · Tracking · + 2 more

Track reconstruction in Velo, using the tools of Exa.TrkX.



[talk@CTD2023]



ETX4VELO

Already outperforming the state of the art

```
TrackChecker output              :    1736/   254023   0.68% ghosts
01_velo                          :  102725/   104345  98.45% ( 98.48%),   1059 (  1.02%) clones, pur  99.81%, hit eff  98.66%
02_long                          :   58771/    59167  99.33% ( 99.30%),    566 (  0.95%) clones, pur  99.89%, hit eff  98.93%
03_long_P>5GeV                   :   38035/    38150  99.70% ( 99.65%),    296 (  0.77%) clones, pur  99.91%, hit eff  99.21%
04_long_strange                  :    3066/     3142  97.58% ( 97.64%),     41 (  1.32%) clones, pur  99.48%, hit eff  98.55%
05_long_strange_P>5GeV           :    1485/     1521  97.63% ( 97.45%),     10 (  0.67%) clones, pur  99.38%, hit eff  99.46%
06_long_fromB                    :     120/      120 100.00% (100.00%),      0 (  0.00%) clones, pur 100.00%, hit eff 100.00%
07_long_fromB_P>5GeV             :      87/       87 100.00% (100.00%),      0 (  0.00%) clones, pur 100.00%, hit eff 100.00%
08_long_electrons                :    4169/     4198  99.31% ( 99.44%),    379 (  8.33%) clones, pur  98.39%, hit eff  96.38%
09_long_fromB_electrons          :      10/       10 100.00% (100.00%),      0 (  0.00%) clones, pur 100.00%, hit eff 100.00%
10_long_fromB_electrons_P>5GeV   :       7/        7 100.00% (100.00%),      0 (  0.00%) clones, pur 100.00%, hit eff 100.00%
```

LHCb unofficial

GDL4HEP › etx4velo

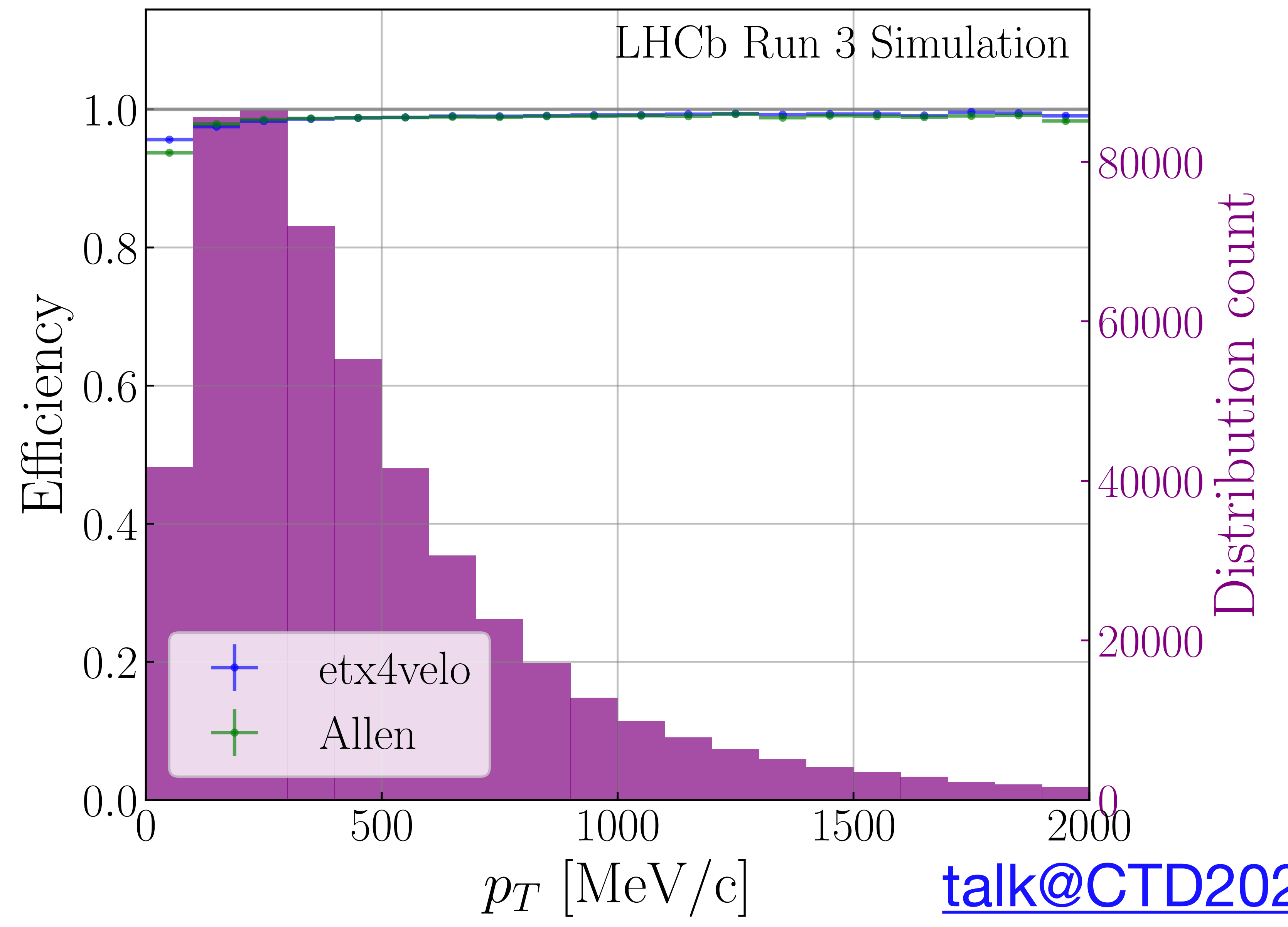
etx4velo

Project ID: 154495

738 Commits · 13 Branches · 4 Tags · 1.1 GiB Project Storage · 1 Release · 1 Environment

Topics: LHCb · python · Tracking · + 2 more

Track reconstruction in Velo, using the tools of Exa.TrkX.



talk@CTD2023

Main objectives

- **Find a NN for tracking at LHCb that achieves state-of-the-art performance** ✓

- Optimise network enough in order to meet high throughput constraint

Main objectives

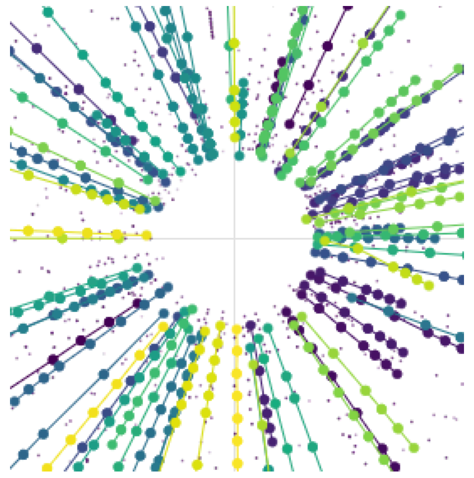
- Find a NN for tracking at LHCb that achieves state-of-the-art performance
- **Optimise network enough in order to meet high throughput constraint**

Main objectives

- Find a NN for tracking at LHCb that achieves state-of-the-art performance

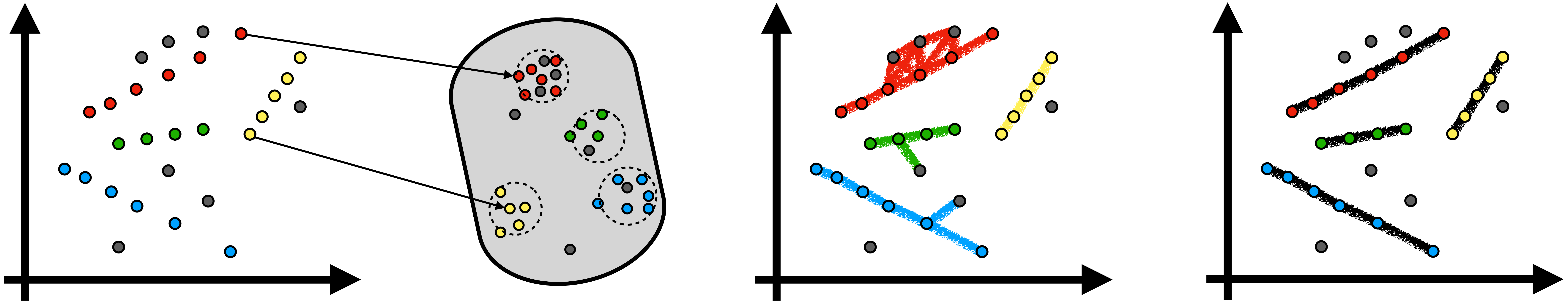
- **Optimise network enough in order to meet high throughput constraint**

Switch from Python to C++/CUDA



From ETX4VELO to ETX4VELO_CPP

Inference: what is slowing us down?

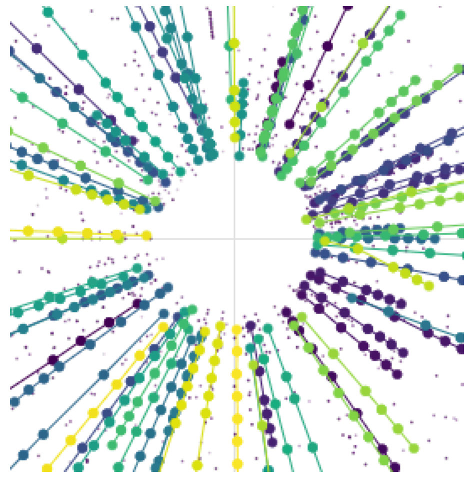


Hits in the detector → **MLP** → Embedding/Latent space → **Graph construction** → Graph

Graph → **GNN** → Edge scores → **Score threshold** → Tracks

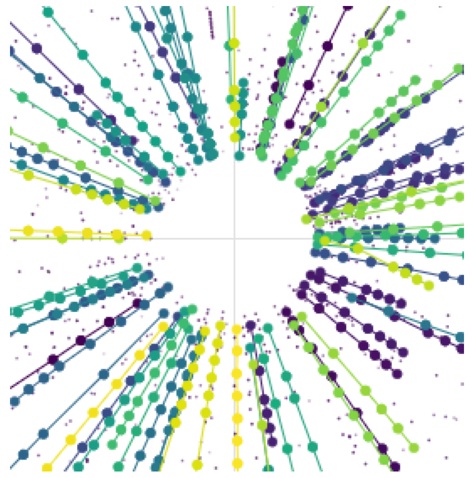
From ETX4VELO to ETX4VELO_CPP

Inference: what is slowing us down?



Hits in the detector → **MLP** → Embedding/Latent space → **Graph construction** → Graph

Graph → **GNN** → Edge scores → **Score threshold** → Tracks

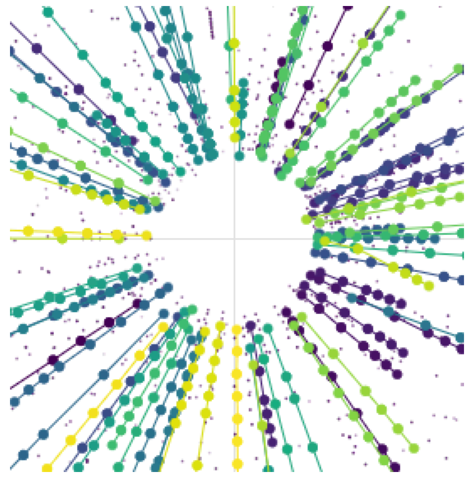


From ETX4VELO to ETX4VELO_CPP

Inference: what is slowing us down?

- Throughput depends on the sizes of the networks
- Can use tools for inference on GPU: TensorRT, ONNX runtime, libTorch

- **EXPORT: ONNX or PyTorch**
- **IMPLEMENT: C++/CUDA**



From ETX4VELO to ETX4VELO_CPP

Inference: what is slowing us down?

- Throughput depends on the sizes of the networks
- Can use tools for inference on GPU: TensorRT, ONNX runtime, libTorch

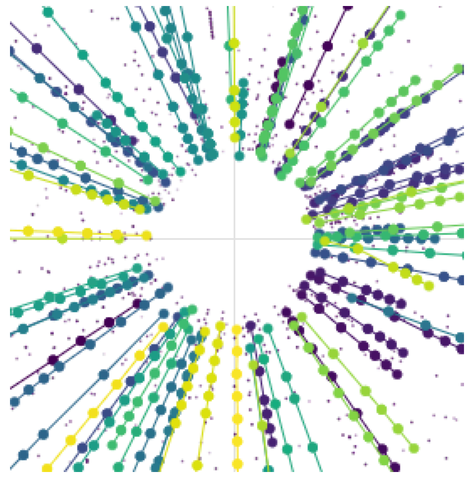
IMPLEMENT

"k nearest neighbours (kNN)" algorithm: computationally expensive

Hits in the detector → **MLP** (EXPORT) → Embedding/Latent space → **Graph construction** → Graph

Graph → **GNN** (EXPORT) → Edge scores → **Score threshold** → Tracks

- **EXPORT: ONNX or PyTorch**
- **IMPLEMENT: C++/CUDA**



From ETX4VELO to ETX4VELO_CPP

Inference: what is slowing us down?

- Throughput depends on the sizes of the networks
- Can use tools for inference on GPU: TensorRT, ONNX runtime, libTorch

IMPLEMENT

“k nearest neighbours (kNN)” algorithm: computationally expensive

Hits in the detector → **MLP** (EXPORT) → Embedding/Latent space → **Graph construction** → Graph

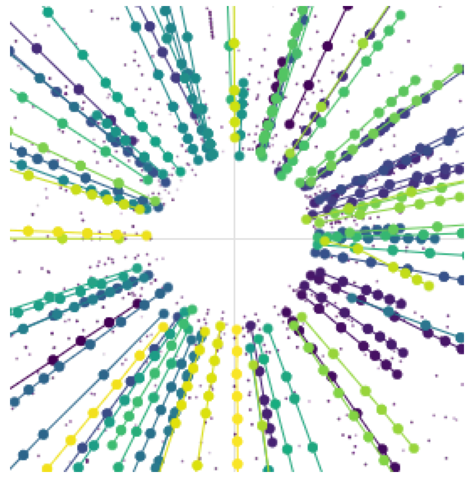
Graph → **GNN** (EXPORT) → Edge scores → **Score threshold** → Tracks

IMPLEMENT

“Weakly connected components (WCC)” algorithm

- **EXPORT: ONNX or PyTorch**
- **IMPLEMENT: C++/CUDA**

Comparison of Python and C++ pipelines



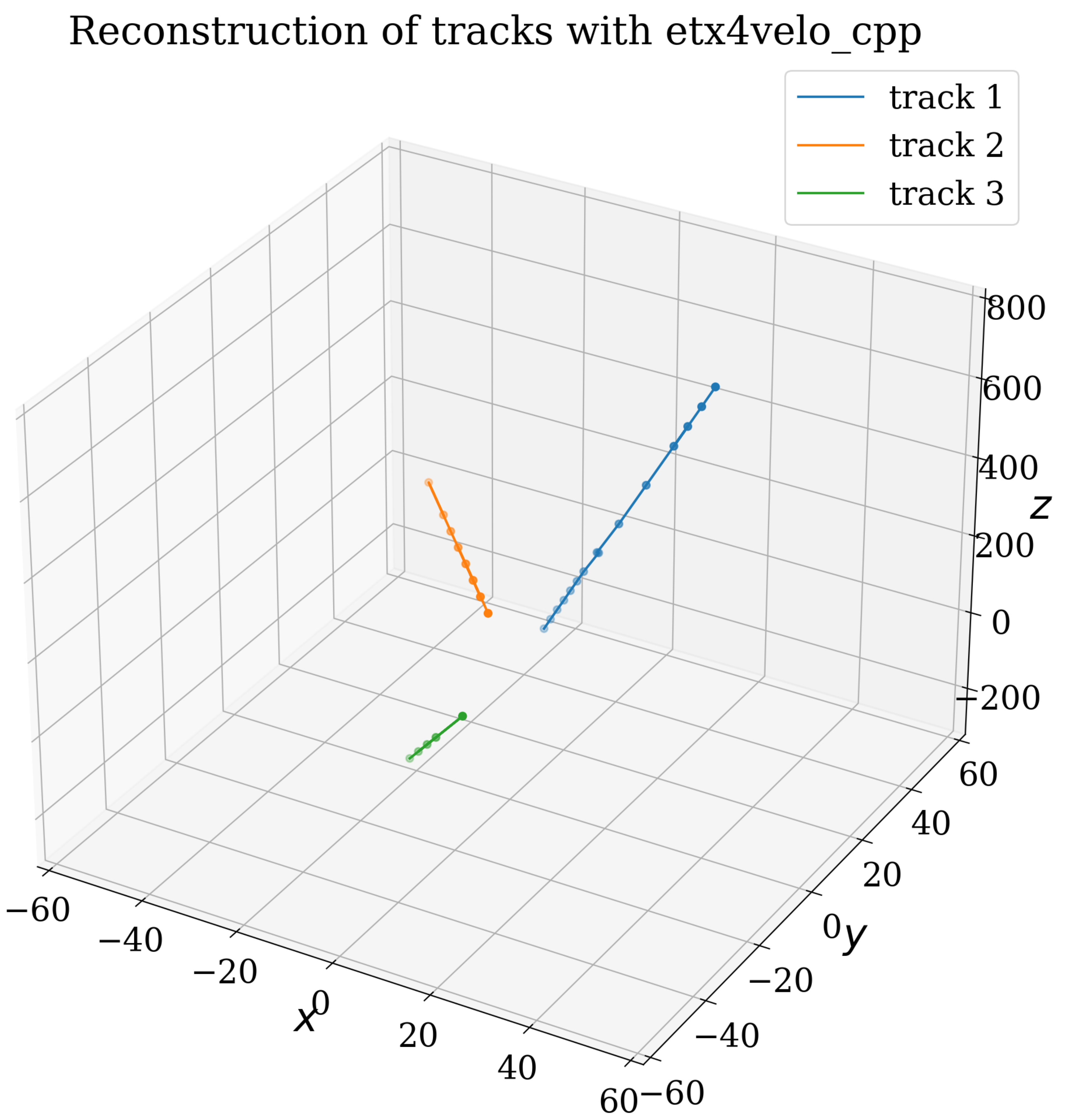
Physics performance comparison

ETX4VELO

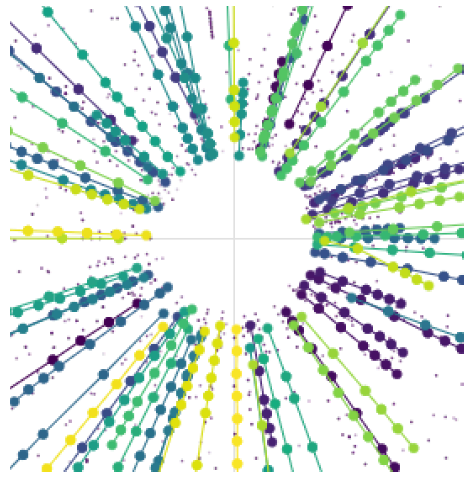
TrackChecker output	16/ 250 6.40% ghosts			
01_velo	117/ 123 95.12% (95.12%)	3 (2.50%) clones	pur 98.26%	hit eff 92.95%
02_long	71/ 73 97.26% (97.26%)	1 (1.39%) clones	pur 98.57%	hit eff 95.62%
03_long_P>5GeV	50/ 52 96.15% (96.15%)	0 (0.00%) clones	pur 98.54%	hit eff 97.36%
04_long_strange	3/ 3 100.00% (100.00%)	0 (0.00%) clones	pur 100.00%	hit eff 100.00%
05_long_strange_P>5GeV	2/ 2 100.00% (100.00%)	0 (0.00%) clones	pur 100.00%	hit eff 100.00%
08_long_electrons	6/ 11 54.55% (54.55%)	0 (0.00%) clones	pur 87.13%	hit eff 89.54%

ETX4VELO_CPP

TrackChecker output	16/ 251 6.37% ghosts			
01_velo	117/ 123 95.12% (95.12%)	3 (2.50%) clones	pur 98.57%	hit eff 92.72%
02_long	71/ 73 97.26% (97.26%)	1 (1.39%) clones	pur 98.86%	hit eff 94.98%
03_long_P>5GeV	50/ 52 96.15% (96.15%)	0 (0.00%) clones	pur 98.35%	hit eff 96.64%
04_long_strange	3/ 3 100.00% (100.00%)	0 (0.00%) clones	pur 100.00%	hit eff 100.00%
05_long_strange_P>5GeV	2/ 2 100.00% (100.00%)	0 (0.00%) clones	pur 100.00%	hit eff 100.00%
08_long_electrons	9/ 11 81.82% (81.82%)	0 (0.00%) clones	pur 89.51%	hit eff 90.80%



- Both pipelines produce exactly the same results
- Can now focus on throughput optimisations



ETX4VELO_CPP inference

Throughput considerations

- First implementation using the Exa.TrkX repository, talk@ACAT2021, arXiv:2202.06929
- **Optimizations** to do:
  - **Parallelise** kNN and WCC across the events
  - Infer MLP and GNN in large **batches**
  - Optimize **data transfers** between host and device
  - Reduce **neural network size** or change architecture, pruning
  - Write custom implementations
  - Accelerate parts of the pipeline on **FPGAs**

Conclusion

Track finding with ETX4VELO

- Comparable or superior performance to current state of the art
- Excellent electron reconstruction

C++ version of pipeline: ETX4VELO_CPP

- Identical physics performance with ETX4VELO (without triplets)
- Progress towards the implementation in LHCb framework (Allen)

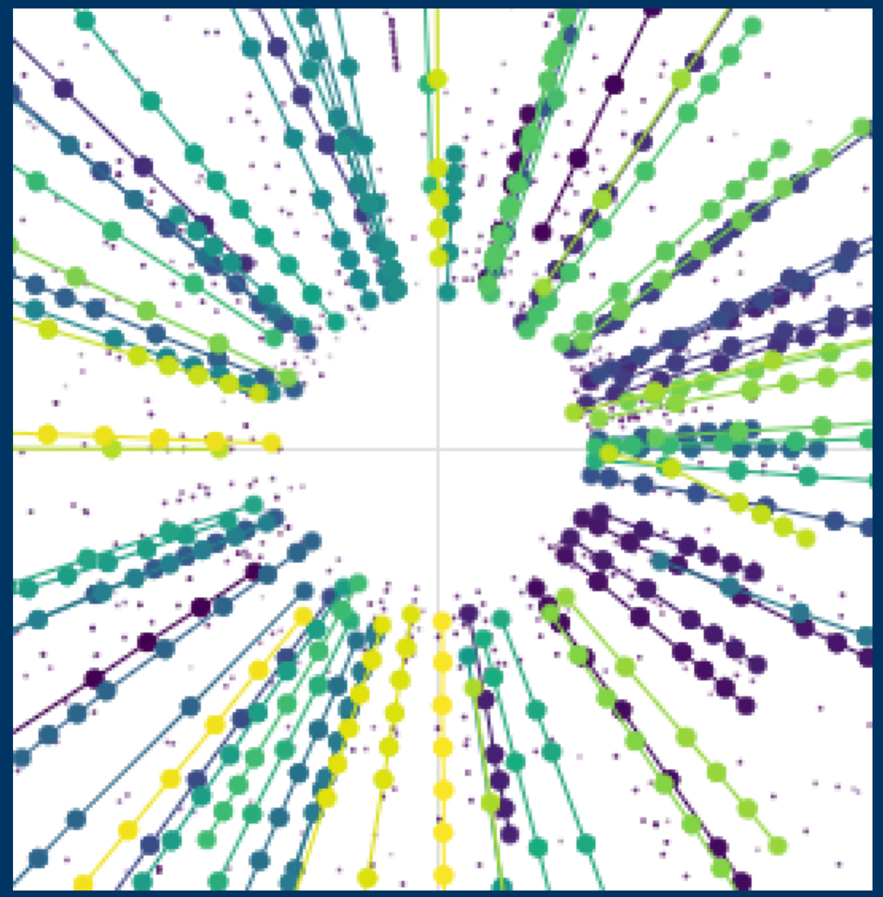
Ongoing work

- Optimise throughput of the pipeline
- Compare optimal throughput with current classical algorithms
- Extension to other LHCb tracking detectors, starting from SciFi

Thank you!

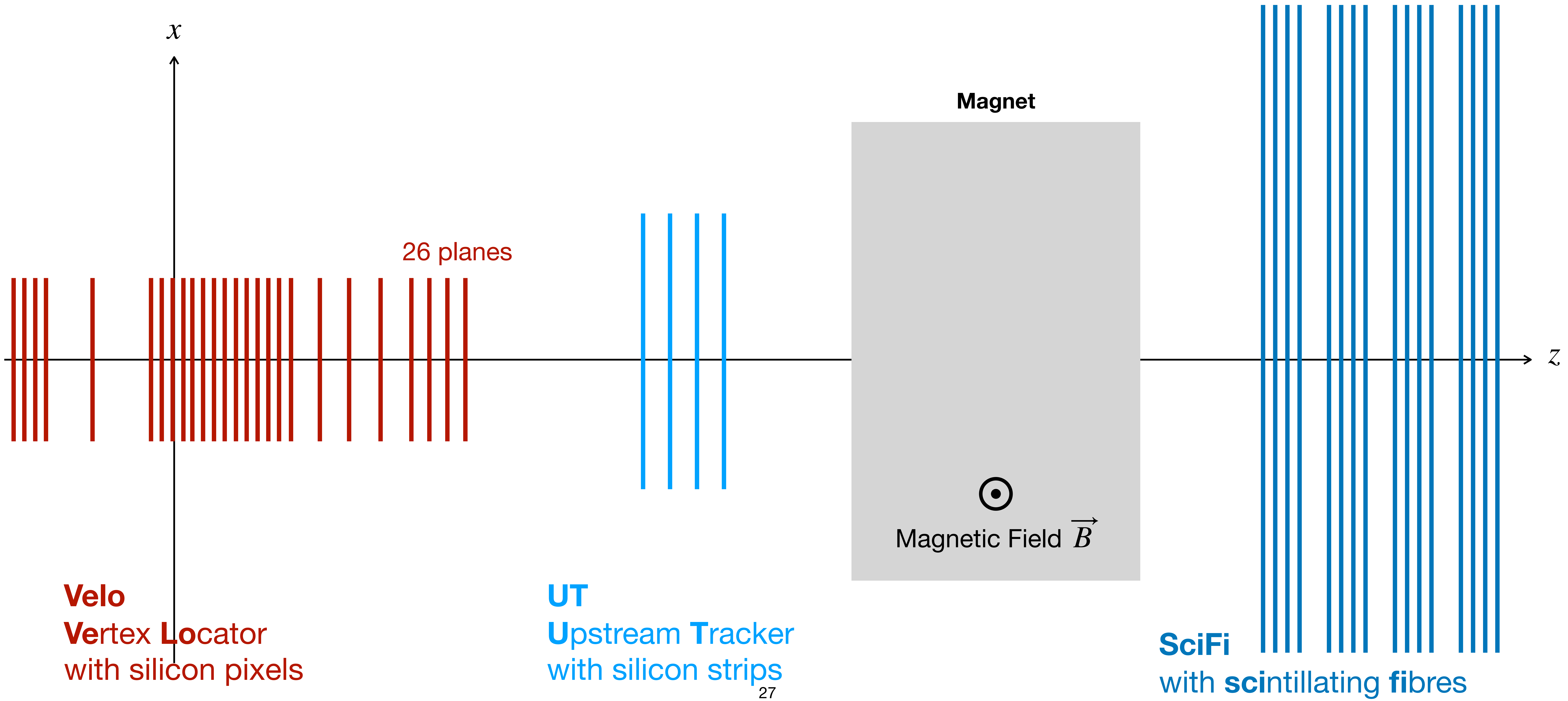
This work is part of the SMARTHEP network and it is funded by the European Union's Horizon 2020 research and innovation programme, call H2020-MSCA-ITN-2020, under Grant Agreement n. 956086.

Backup



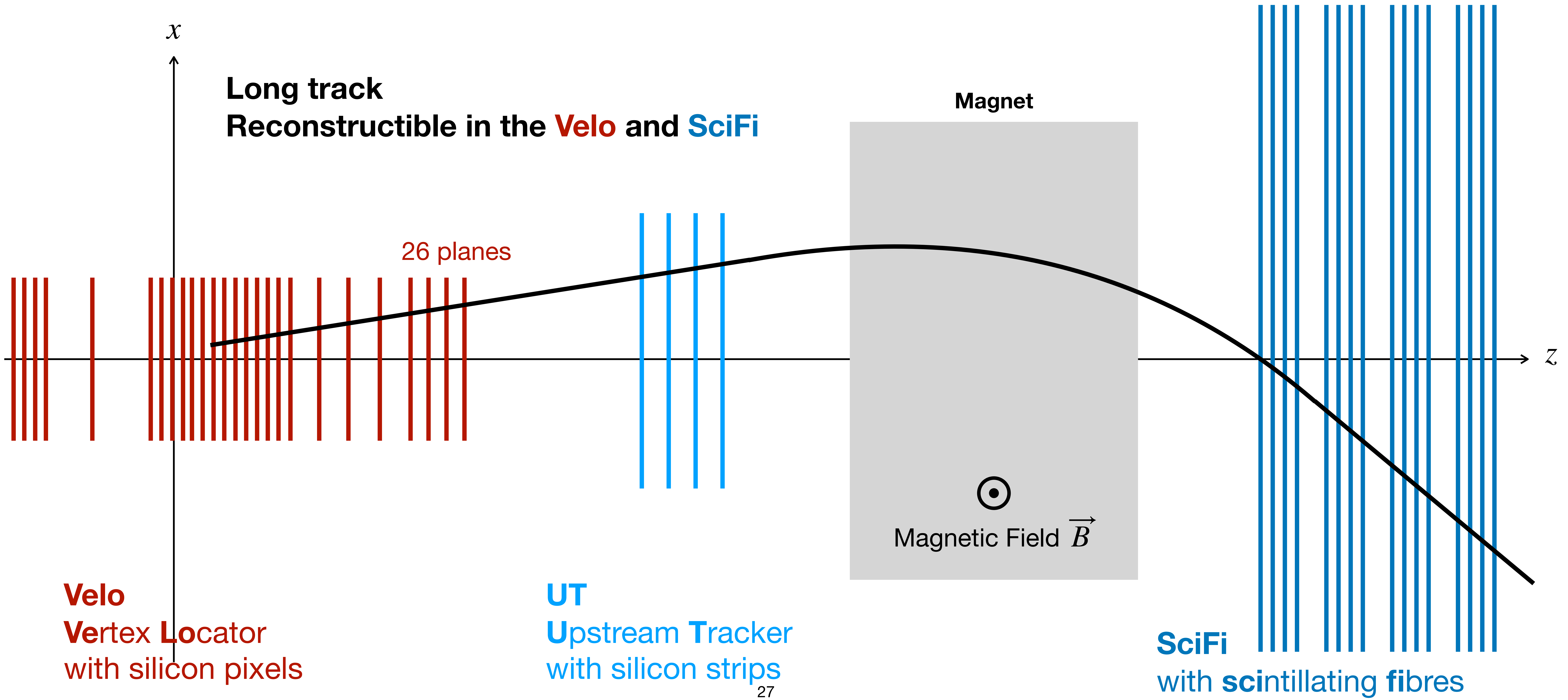
Tracks in LHCb

Long tracks



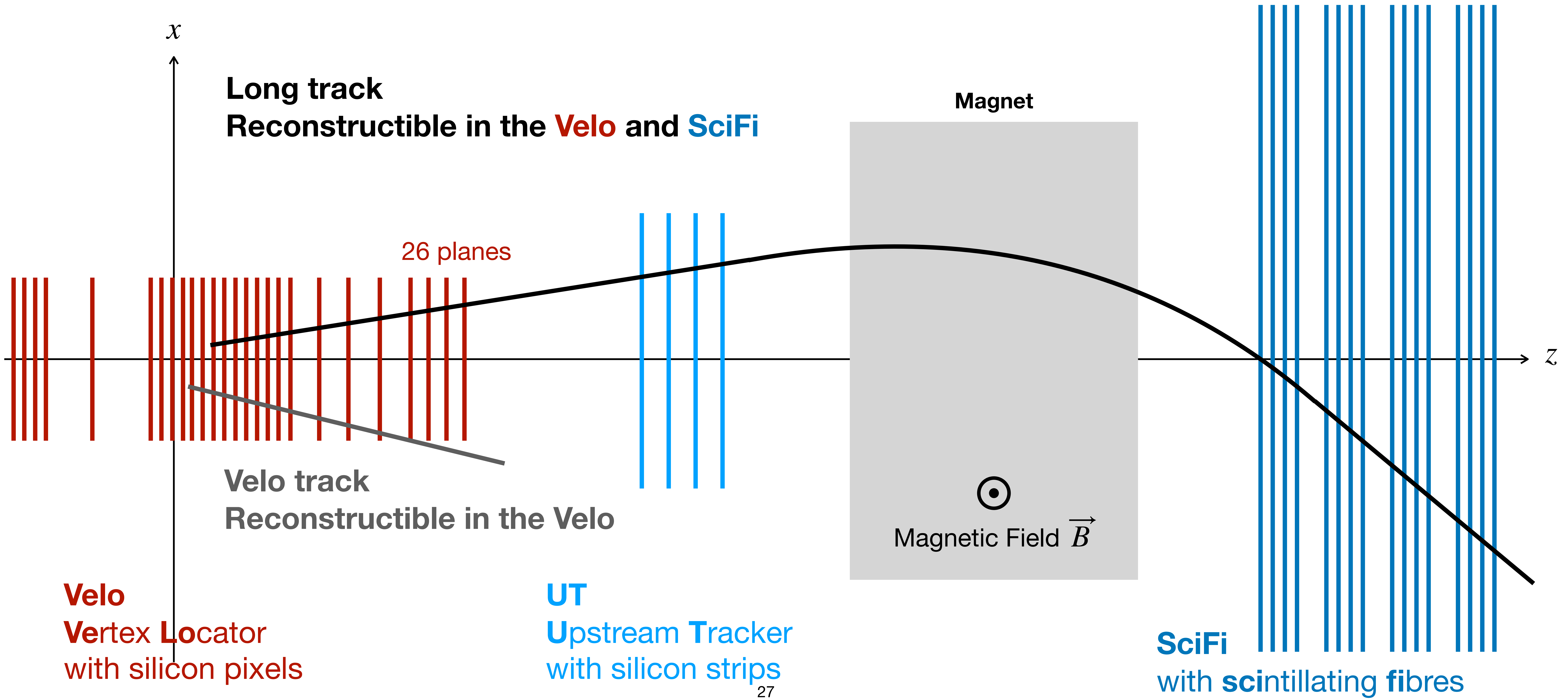
Tracks in LHCb

Long tracks



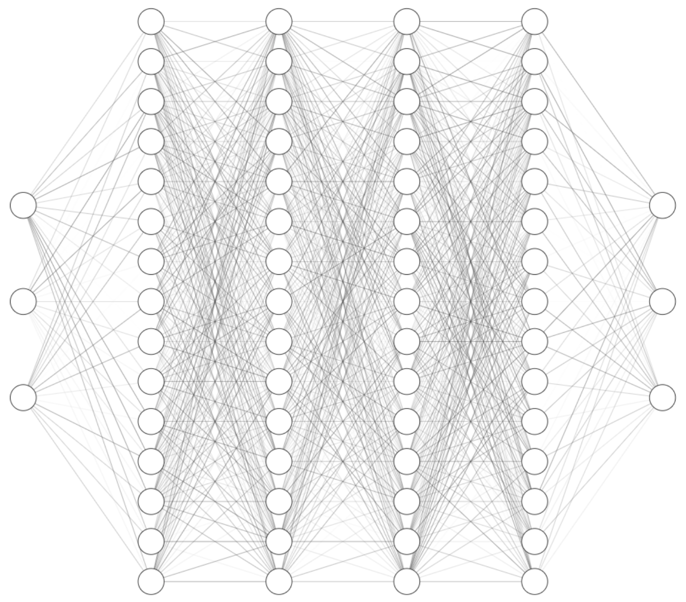
Tracks in LHCb

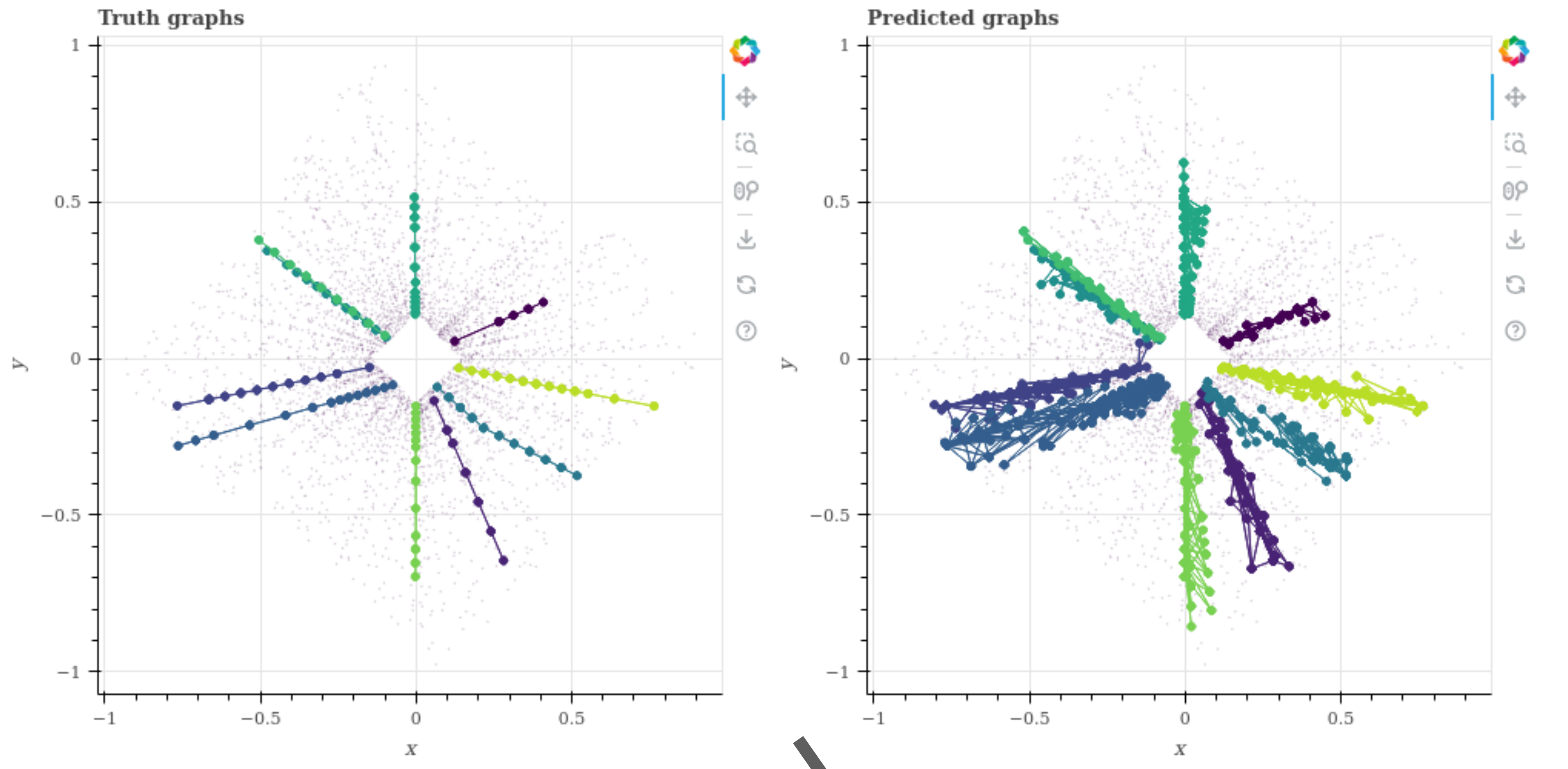
Long tracks

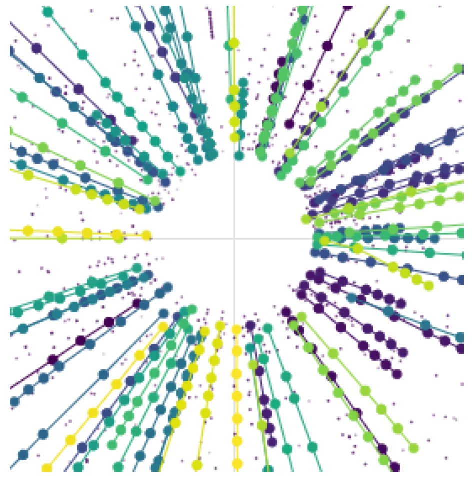


ETX4VELO

Training pipeline

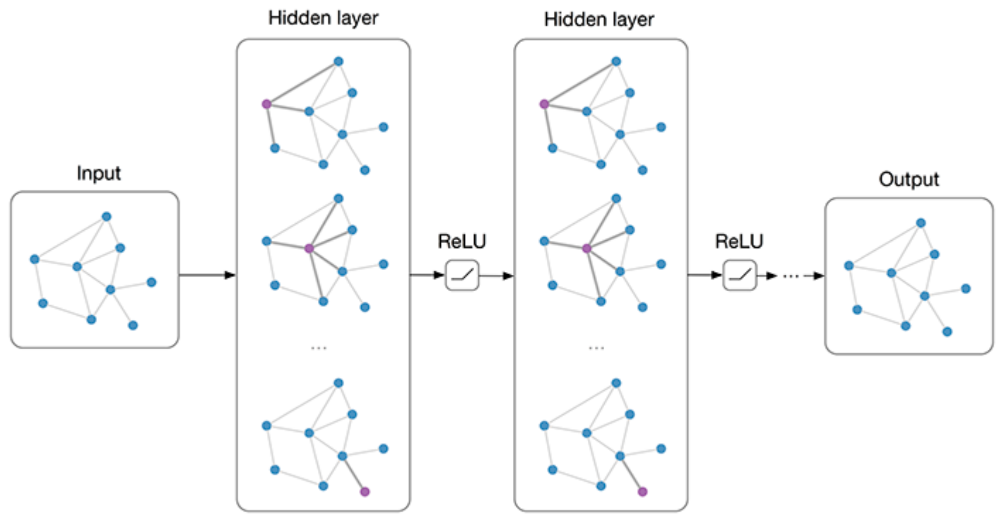


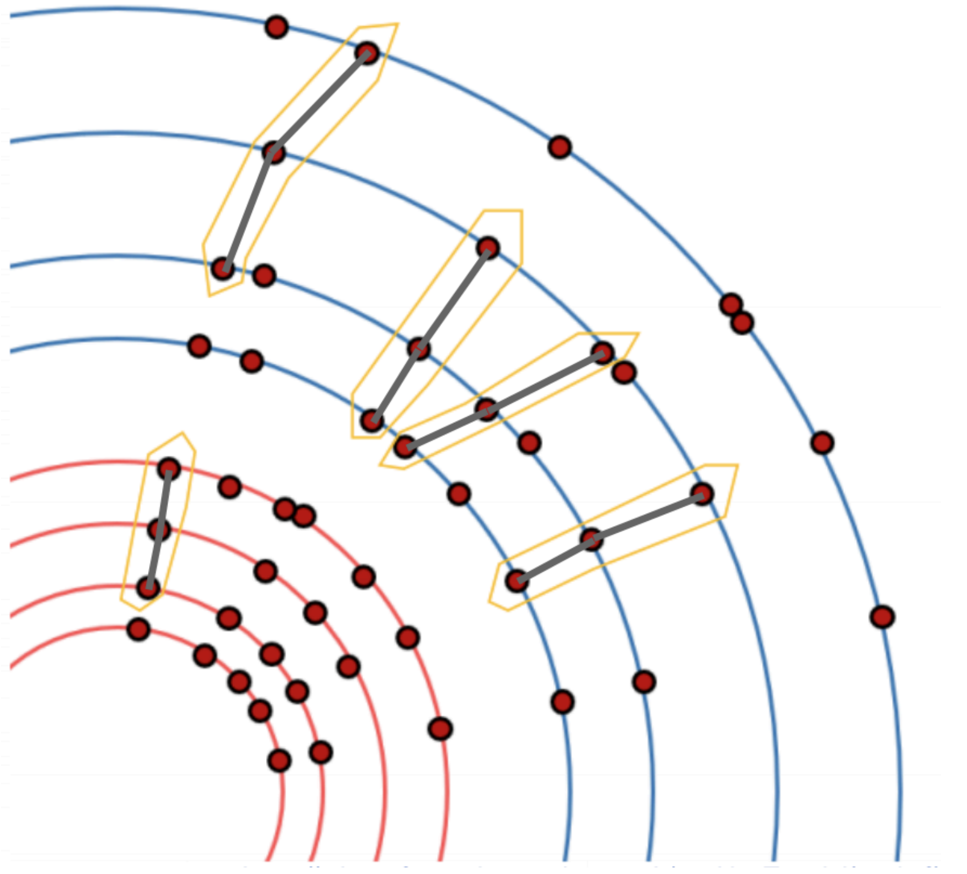




Processing → Metric Learning Training → Metric Learning Inference → Graph Construction → GNN Training

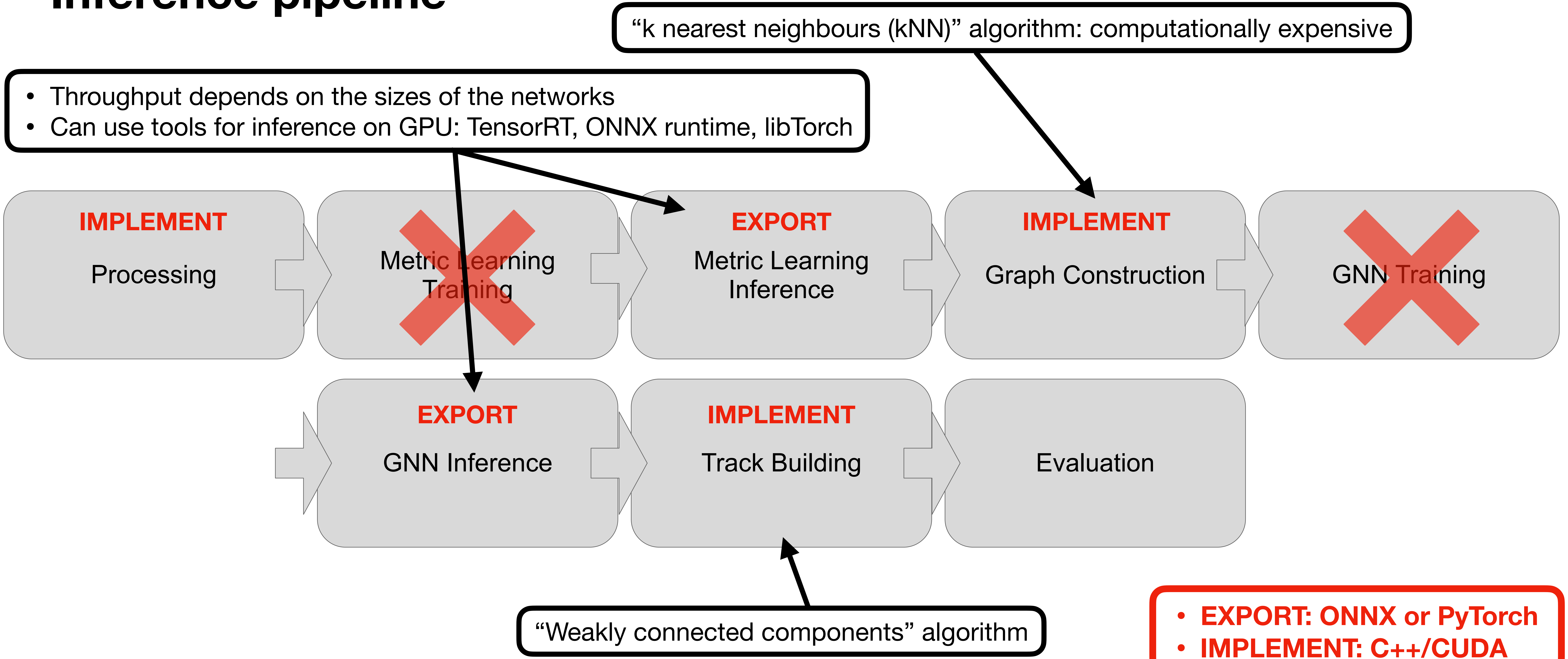
→ GNN Inference → Track Building → Evaluation





ETX4VELO

Inference pipeline

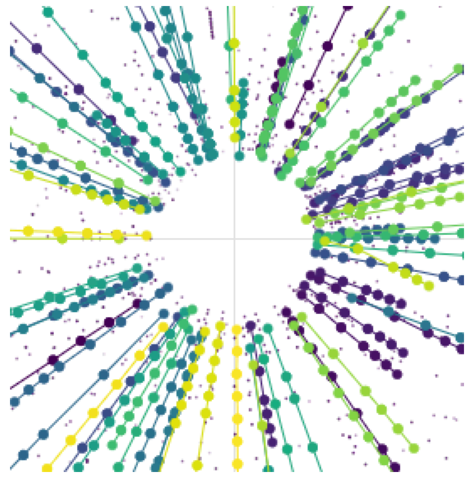


- **EXPORT: ONNX or PyTorch**
- **IMPLEMENT: C++/CUDA**

ETX4VELO

Processing

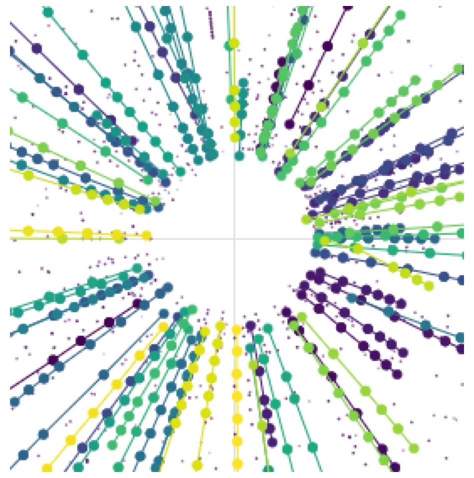
CSV → Torch



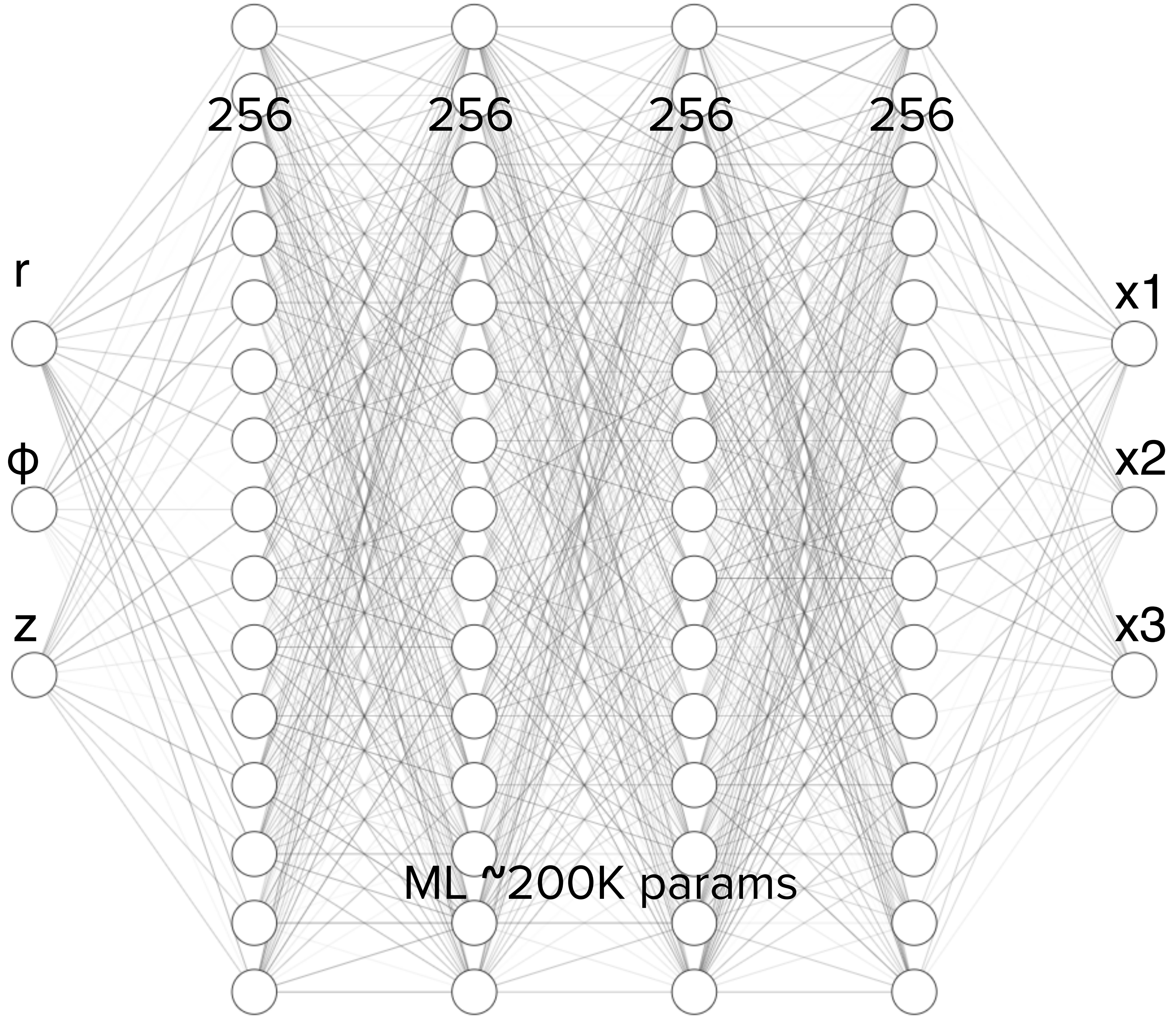
- Split data by event
- Selection on data / cuts
- Transform the data from Cartesian to cylindrical coordinates
- Calculate true edges of the graph
  - Find all the hits with the same mcid
  - Order them wrt the distance from the origin vertex
  - True edges are between these ordered successive hits
- Store data into torch tensors

ETX4VELO
Metric Learning

Hits → Embedding → Graph



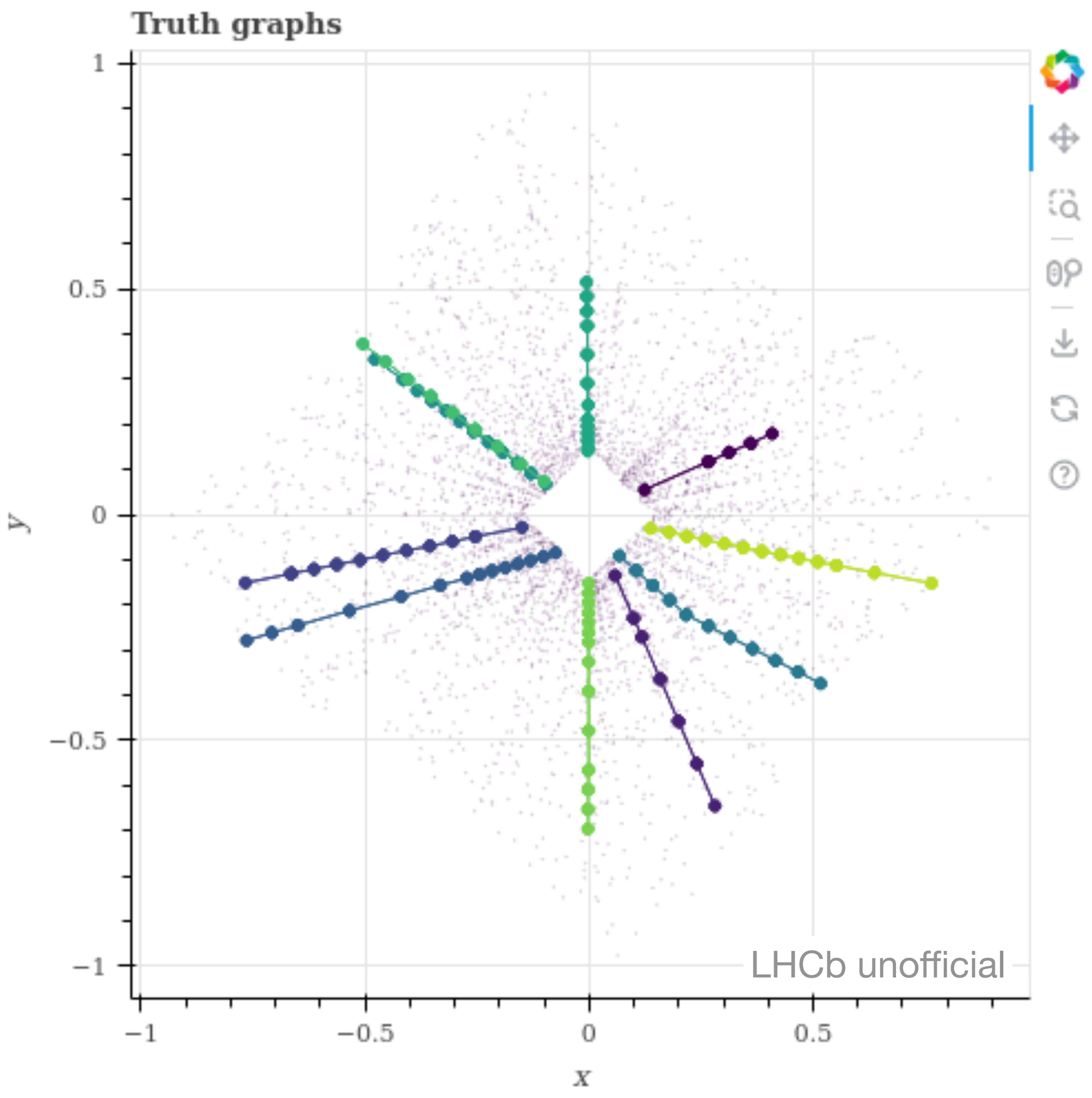
- **Metric Learning Training**
  - Train an MLP to map the features to an embedding space
  - **Distance is reduced for successive hits** (same edge)
  - Distance is amplified if not successive
  - Create the graph for the event
    - For each hit in the embedding space
    - Create hypersphere around it
    - Connect target hit with all hits inside hypersphere
    - **faiss.knn_gpu** github.com/facebookresearch/faiss
- **Metric Learning Inference**
  - With the now trained network, **generate the graphs** for each of the events

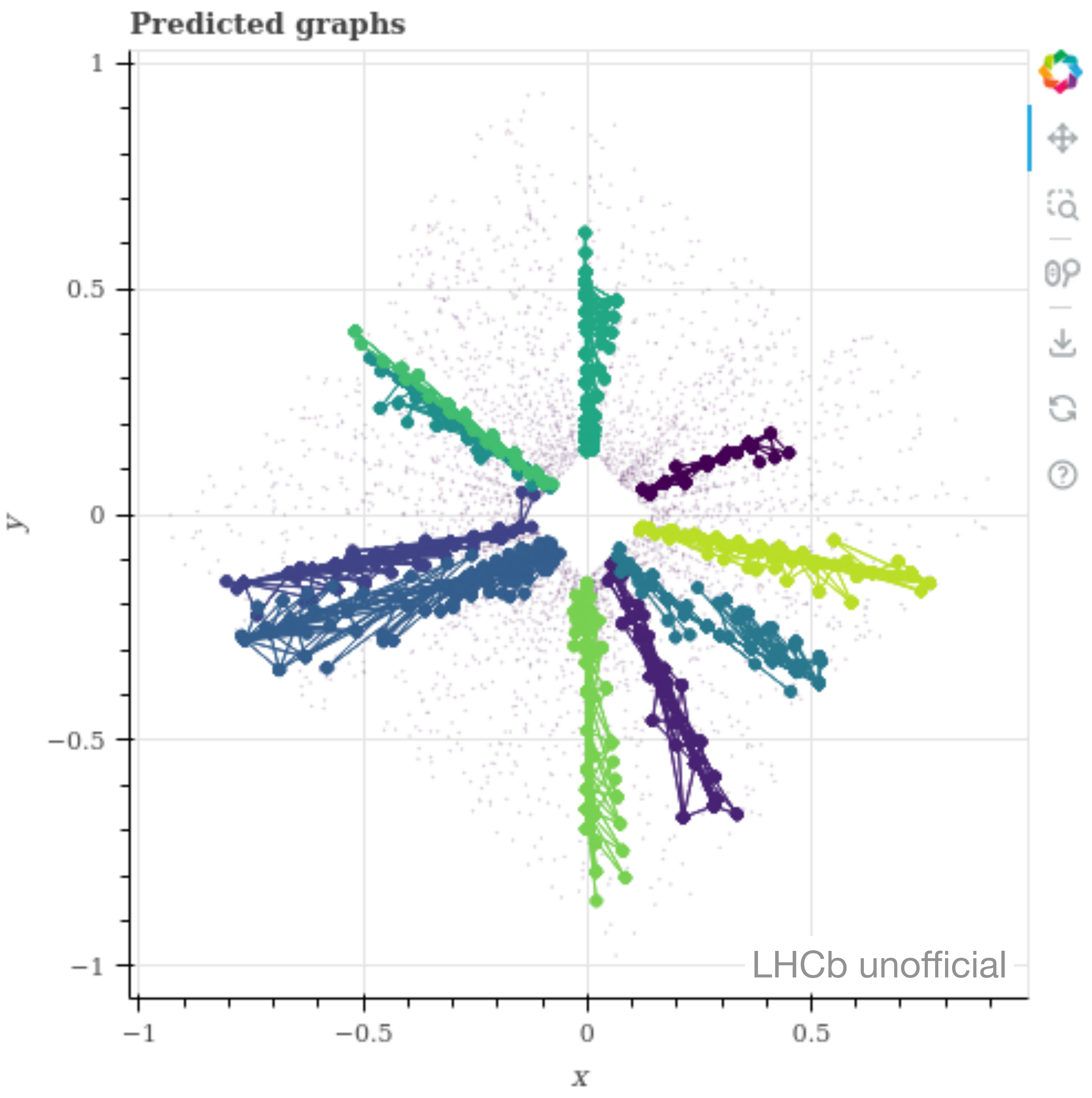


source

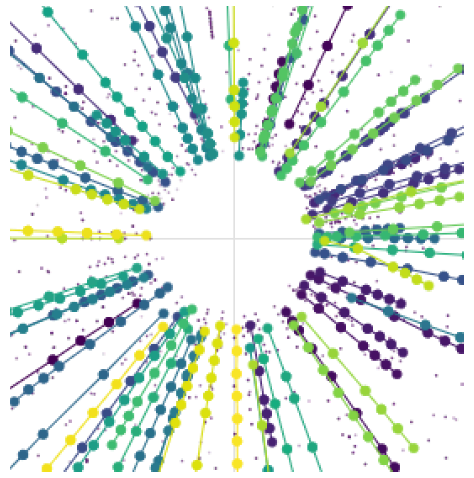
ETX4VELO
Metric Learning

Hits → Embedding → Graph





ETX4VELO

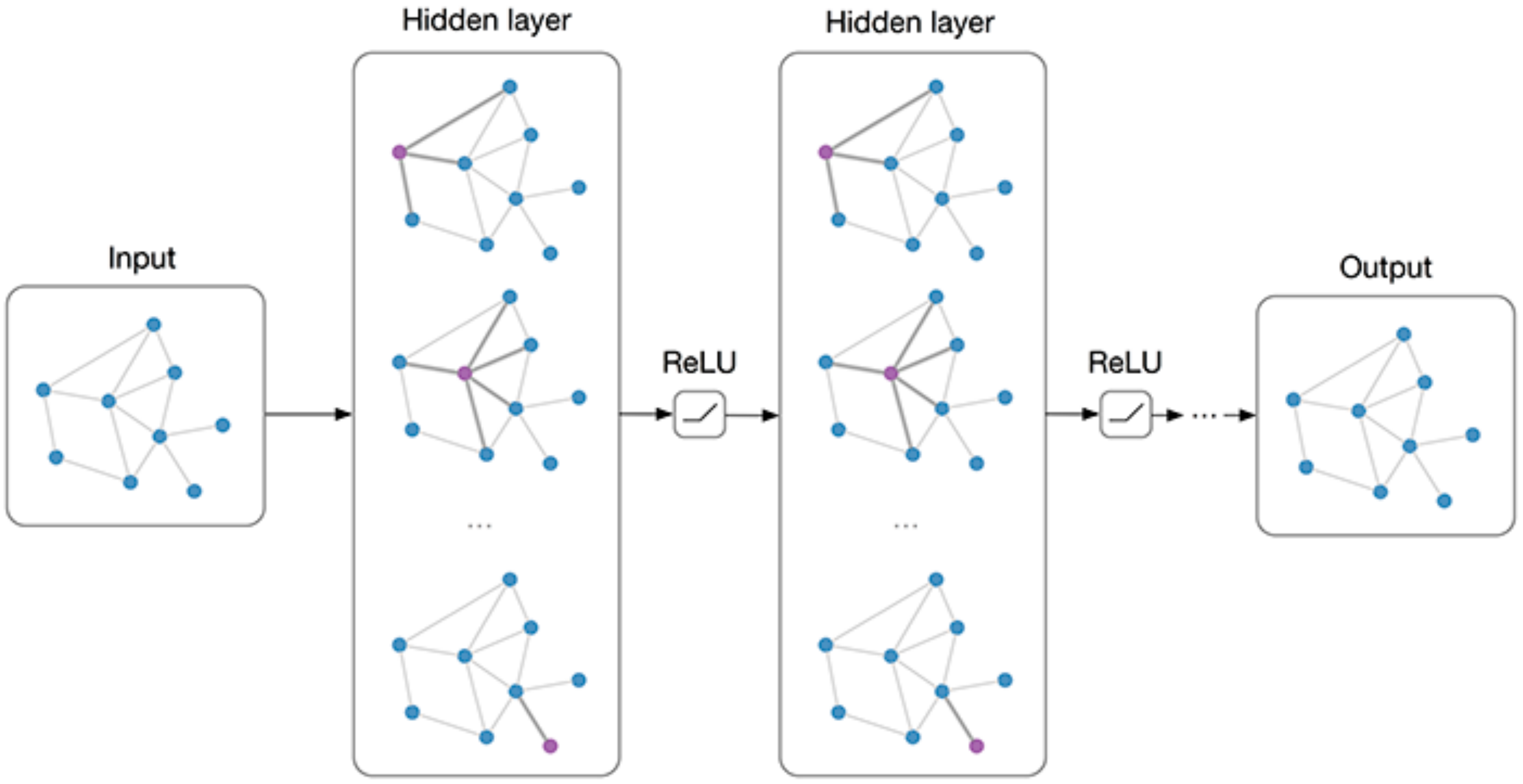


GNN

Graph → Edge scores

- **GNN Training**
  - With the generated graphs, train the GNN to give scores to each edge
  - True edge score = 1
  - GNN: Interaction network, Battaglia et al. “Interaction Networks for Learning about Objects, Relations and Physics”, arXiv:1612.00222
- **GNN Inference**
  - For each generated graph for the events, give scores to all the edges

GNN ~2M params (no pruning yet)

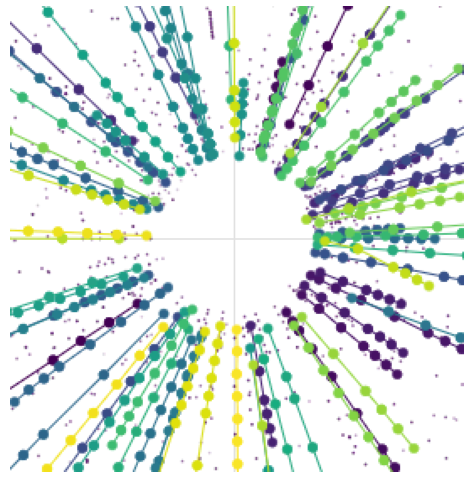


source

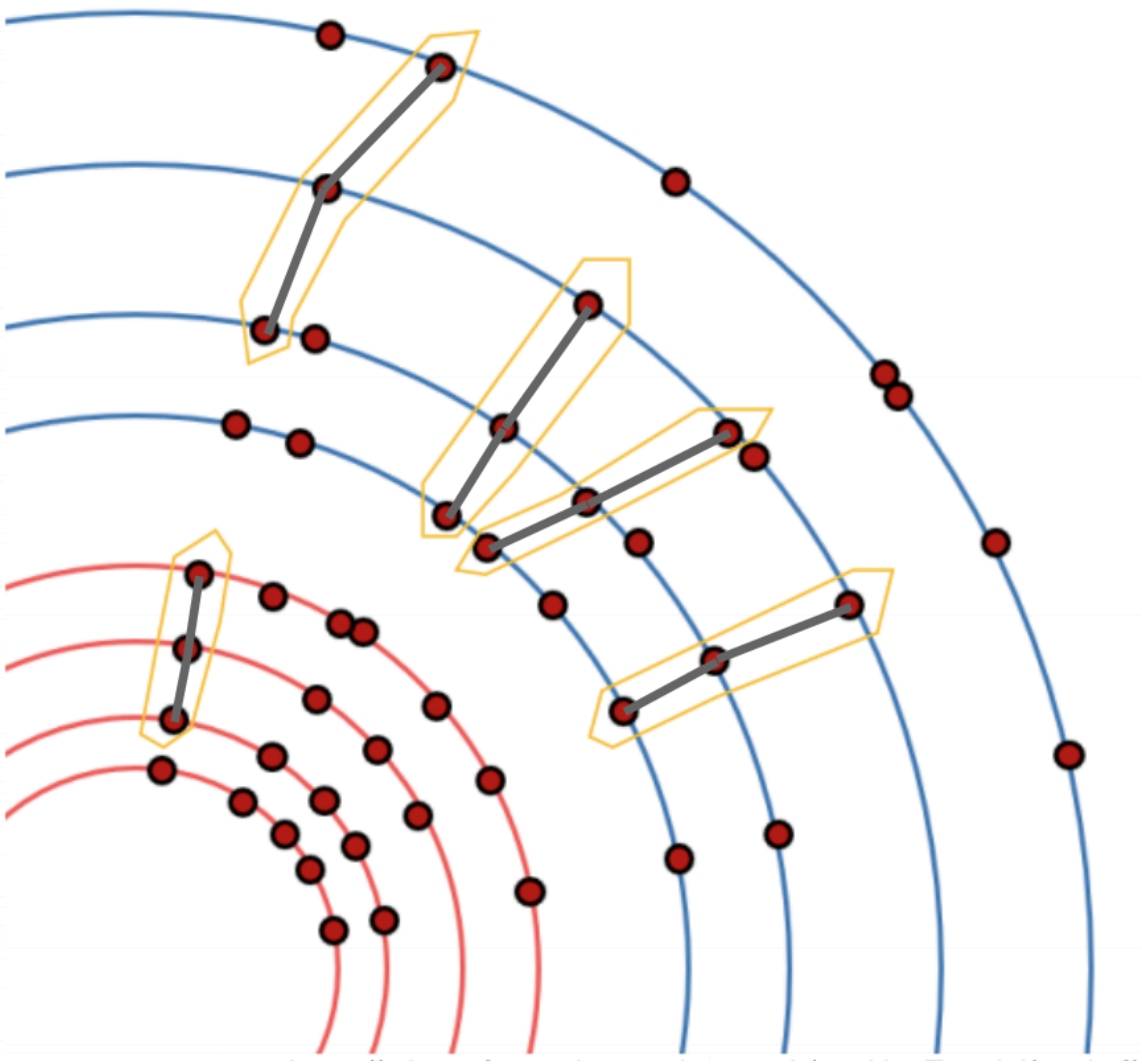
ETX4VELO

Track building

Edge scores → Tracks



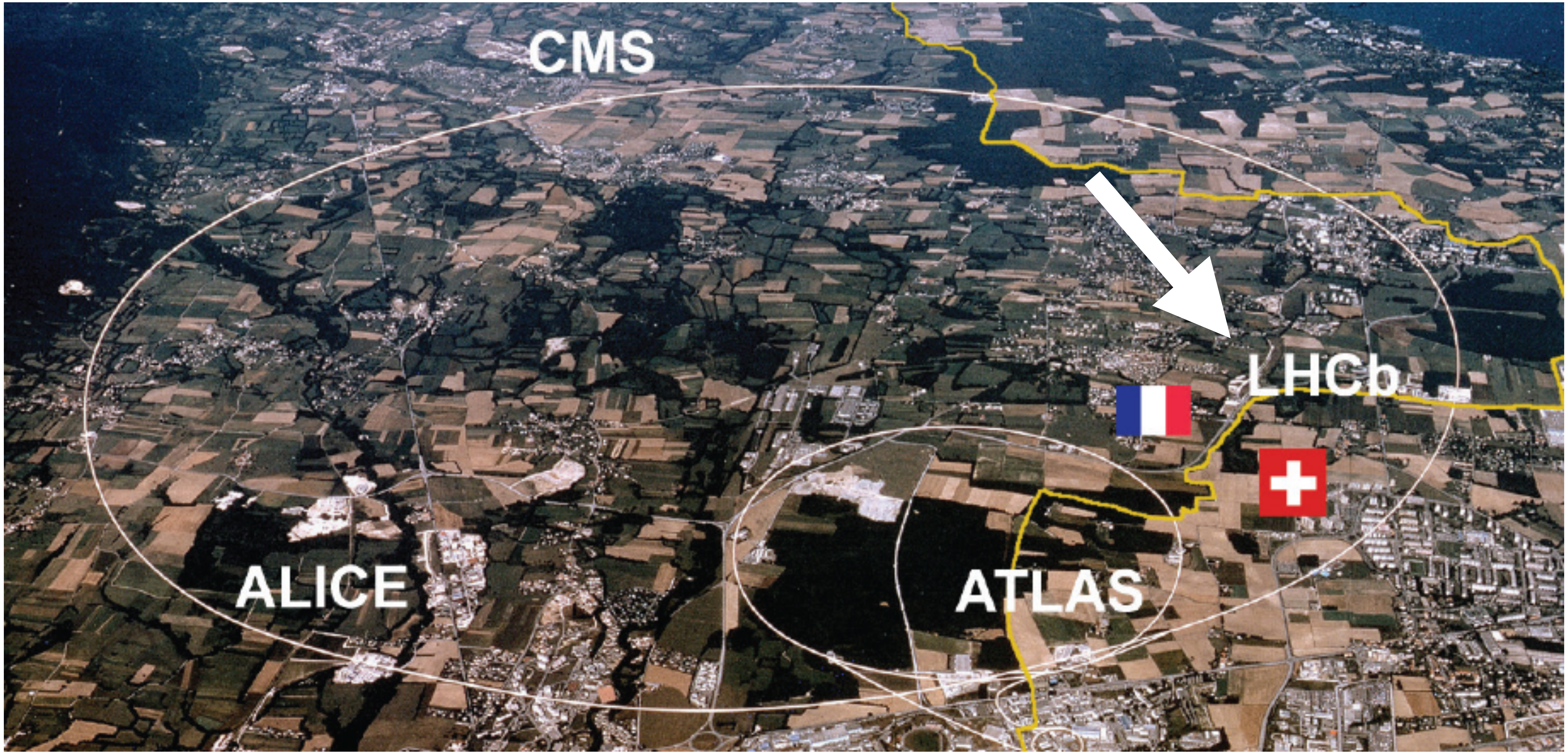
- Graph: sparse
- Choose score cut, e.g. 0.9
- If edge score < 0.9: remove edge
- Graph with disconnected components
- Break graph down to its connected components, [scipy.sparse.csgraph.connected_components]
- → Track candidates



source

CERN

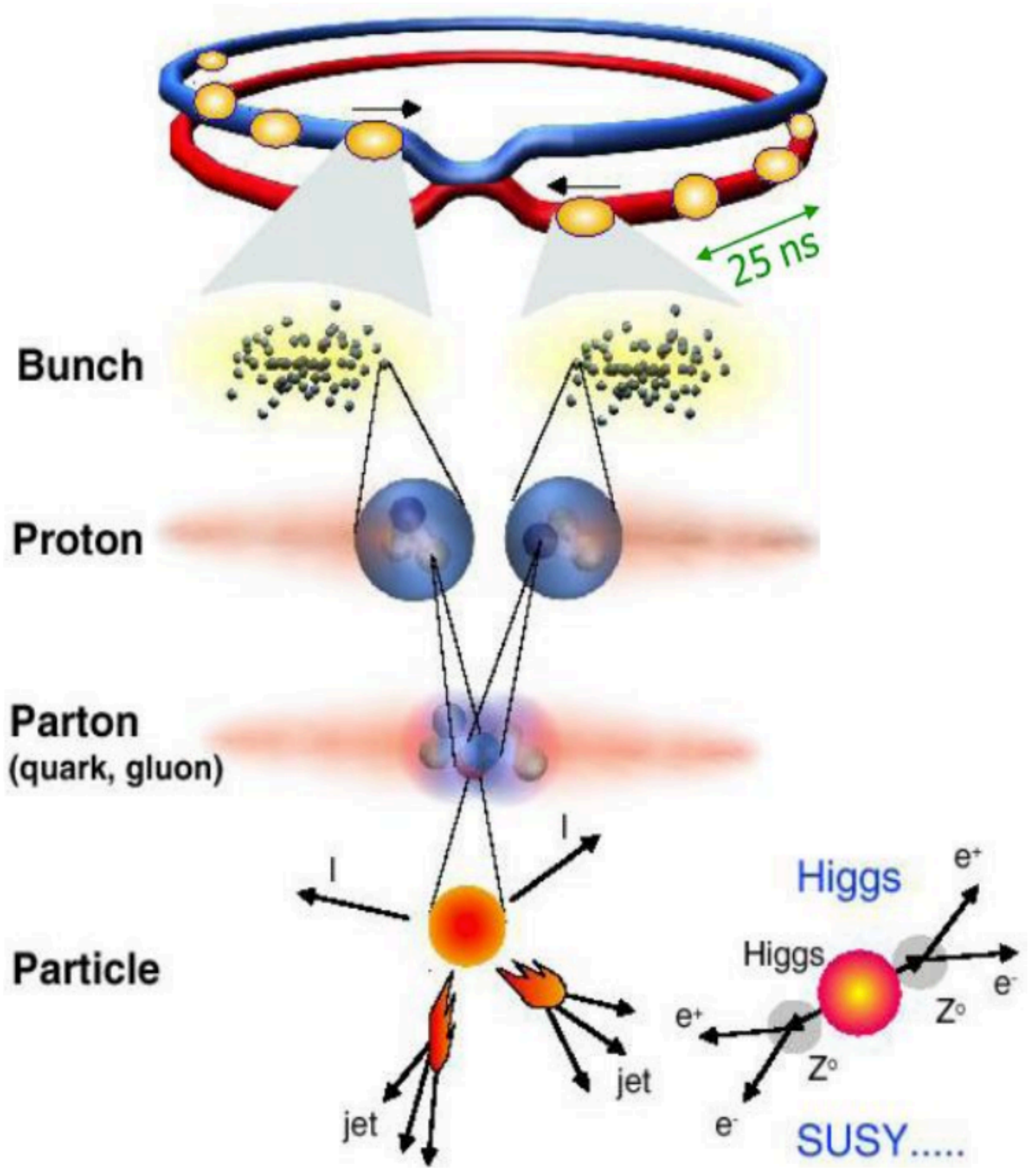
The Large Hadron Collider and LHCb



source

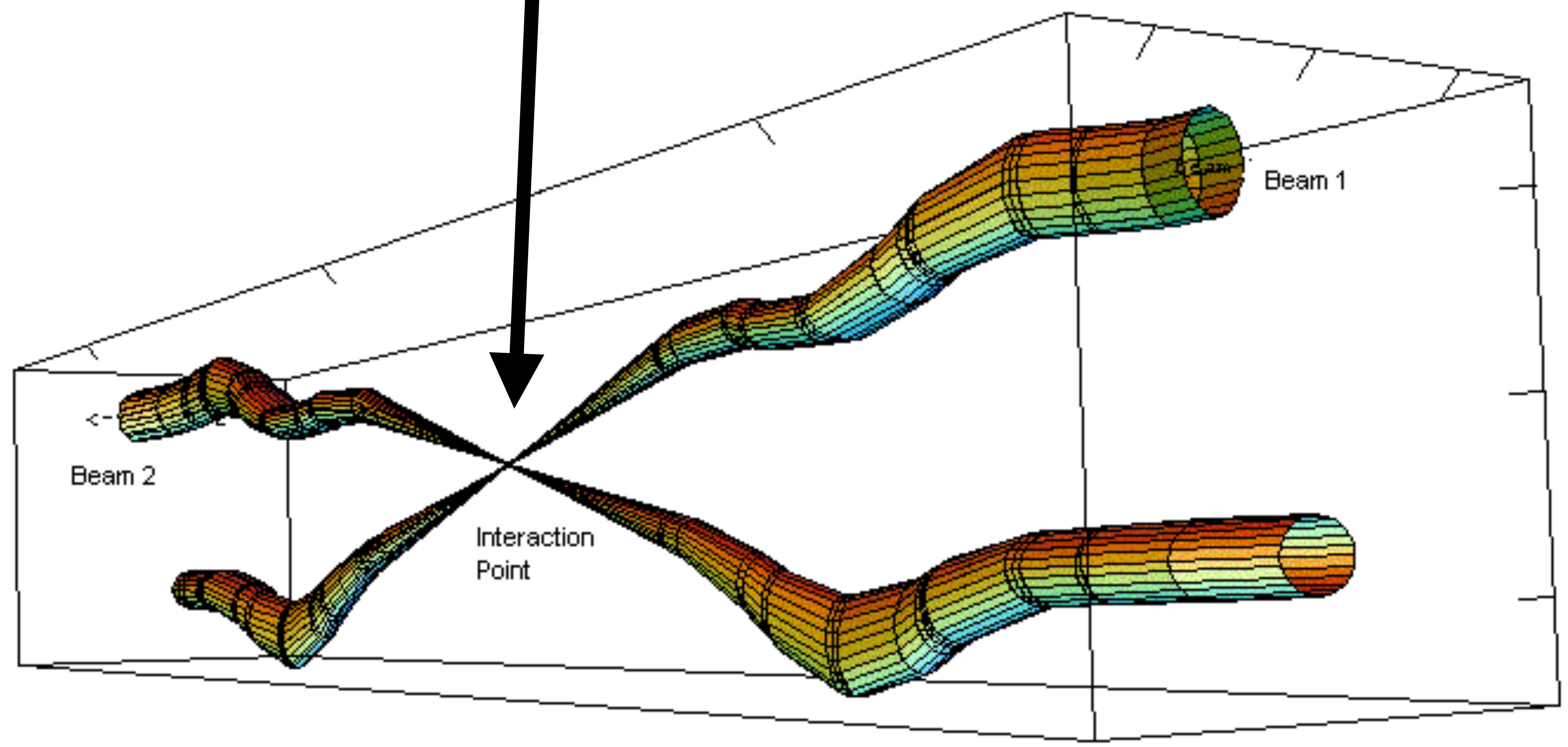
The Large Hadron Collider

Collisions at the LHC



source

Protons colliding at 0.99999999 the speed of light



Relative beam sizes around IP1 (Atlas) in collision

source