

Training machines, training people

- Associate Research
 - **Tracking with Graph Neural Networks** (GNNs)
 - Organizing **software training events** with HSF and IRIS-HEP
 - With CMS
- till July '23: PhD with **Belle 2**
 - Calibrating the FEI (aka Belle 2's Skynet candidate) for a V_{cb} measurement
 - Maintaining Belle 2's integration/performance test ("validation") framework
 - Rebuilt Belle 2's onboarding training



Kilian Lieret

 @klieret



Training people

**"TEACHERS CHANGE THE
WORLD ONE CHILD AT A TIME."**



**FALSE: MY CLASS ROSTER HAS
30 KIDS IN EACH CLASS**

Training people in cross-experiment software skills

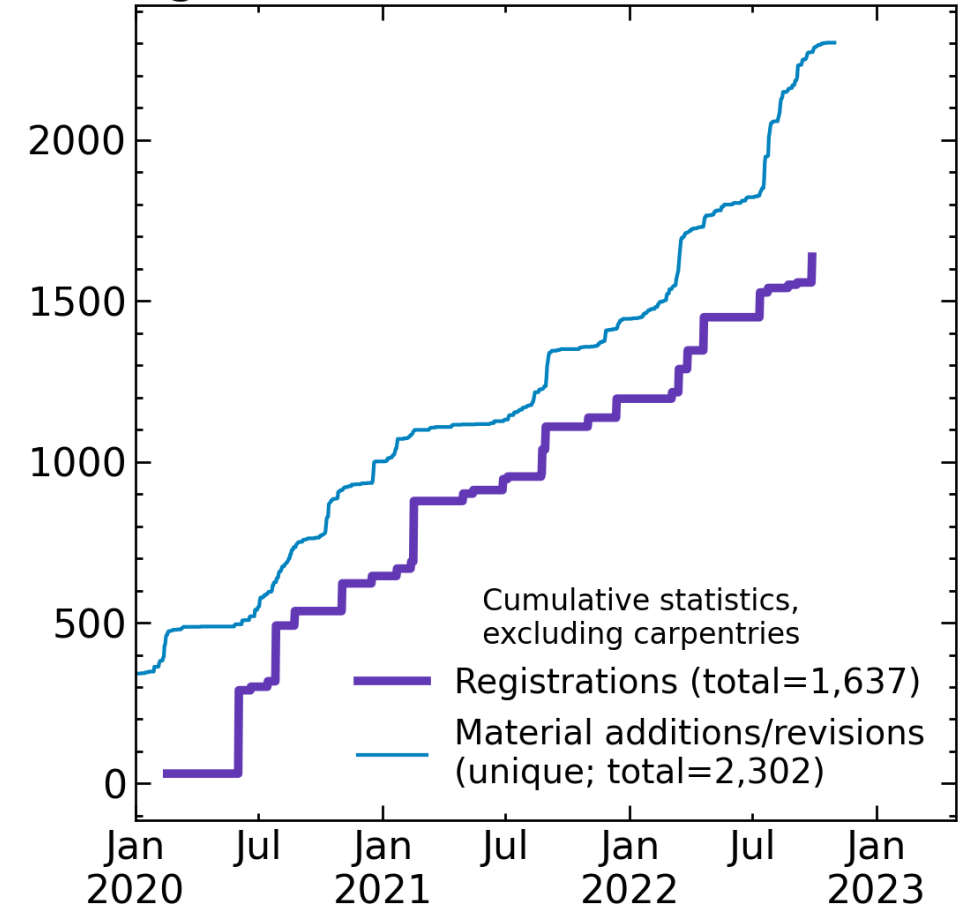
Discuss in [#7](#)

How can we collaborate & be efficient?

Unified training center hepsoftwarefoundation.org/training/curriculum.html

The screenshot displays the 'Training Center for High Energy Physics' website. The main content area is divided into sections: 'Basics' (including The UNIX Shell, Version controlling with git, Programming with python, SSH, Machine learning, and Matplotlib for HEP), 'Software Development and Deployment' (including Version controlling with git, Advanced git, CI/CD (github), and Docker), and 'ROOT'. Each module includes a 'Start learning now!' button and a 'Contribute!' button. A sidebar on the right contains a 'Filters' section with categories: Types (Tutorial, Documentation), Level (Beginner, Advanced), Curriculum (ALTA, HEP tools, Machine Learning Tools), Packages (numpy, pandas, scipy, etc.), and Status (Ready, Beta, In development). The page title is 'Training Center v2.0'.

Registrations and material revisions



Training material (technical side)

Discuss in **#7**

- **How to write material:**

- HSF uses a lot of “**carpentry-style**” websites (built in Jekyll)
- SW Carpentry recently switched to new framework (in R): **Good time to reevaluate our choices!**
- In an ideal world, there would be ≥ 3 versions of each course:
 - Self-study writeup/notebook (verbose & complete)
 - Workshop presentation slides/notebook
 - Workshop student notebooks (for exercises)

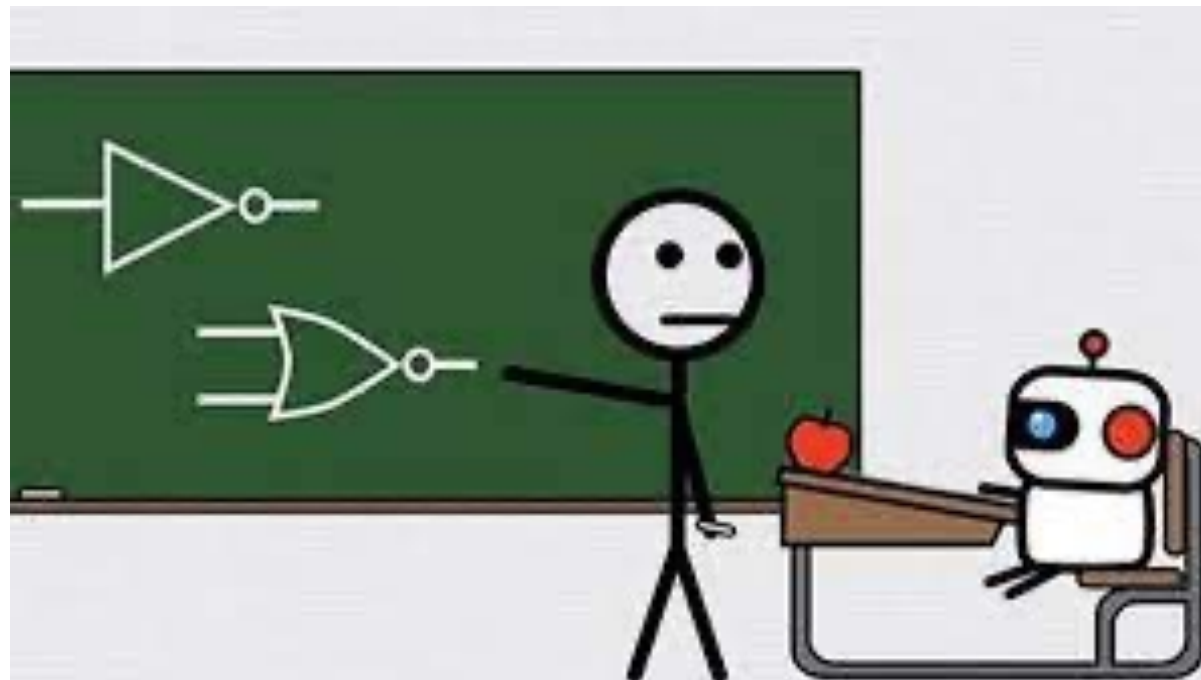
==> Not feasible? How to approach

- **Running code in the browser:**

- Binder is now low on resources
- Google collab will work for anything without big dependencies
- GitHub codespaces looks very promising
- Cool idea (via Jim): Include **feedback buttons** directly in exercise notebook

The screenshot shows the HSF website interface. At the top, there's a navigation bar with links like 'Home', 'Code of Conduct', 'Setup', 'Episodes', 'Extras', 'License', and 'Improve this page'. Below that, the main heading is 'Machine Learning on GPU'. There's a video player with a red play button and a 'Watch on YouTube' button. To the right of the video player, there's a red arrow pointing left with the text '< Complete video walkthroughs!'. Below the video player, there's a 'Prerequisites' section with a list of items: 'A Kaggle account. Click here to create an account', 'Basic Python knowledge, e.g. through the Software Carpentry Programming with Python lesson', and 'Basic ML knowledge, e.g. through the Introduction to Machine Learning lesson'. To the right of this section, there's a red arrow pointing up with the text '^ Lessons build on each other'. Below the prerequisites, there's an 'Introduction' section with text about machine learning applications in physics and a list of aims for the lesson: 'demonstrate how to move an existing machine learning model onto a GPU' and 'discuss some of the common issues that come up when using machine learning applications on GPUs'. At the bottom, there's a section titled 'The skills we'll focus on:' with a list of four items: 'Understanding a bit about GPUs', 'Using Python & PyTorch to discover what kind of GPU is available to you', 'Moving a machine learning model onto the GPU', and 'Comparing the performance of the machine learning model between the CPU and the GPU'.

Training machines

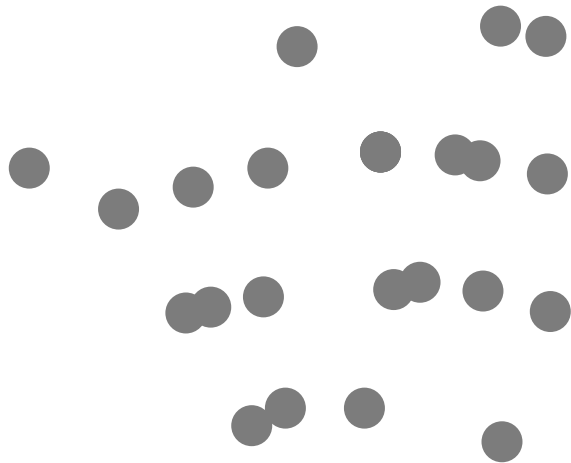


(from CGP Grey)

Tracking with object condensation

Point cloud

(coordinates of hits in detector)



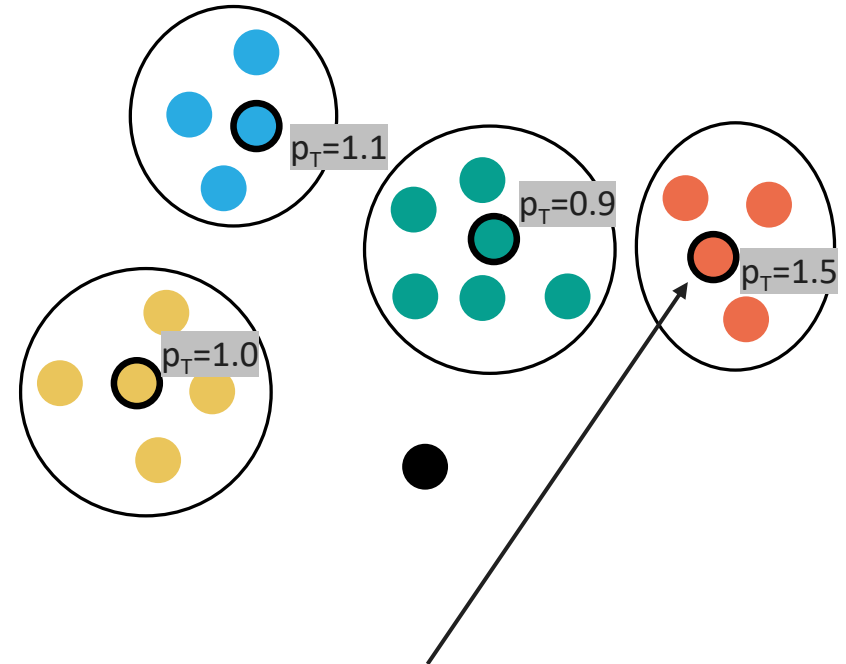
Repulsion & attraction
of points in latent space



No time resolution of points
==> Everything everywhere all at once

Learnt latent space

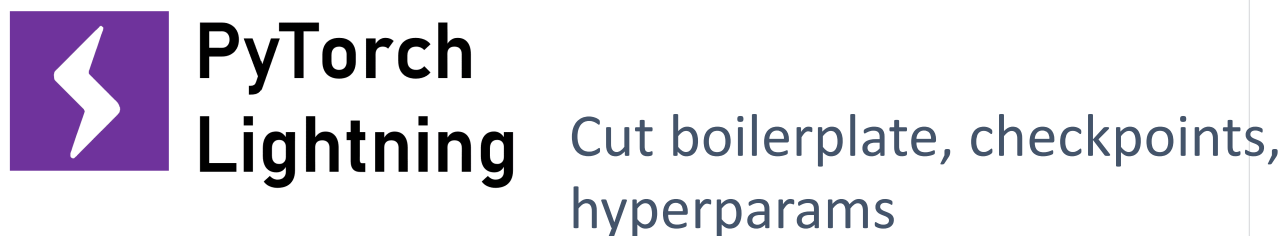
Hits already clustered by particle;
Clusters can be collected trivially



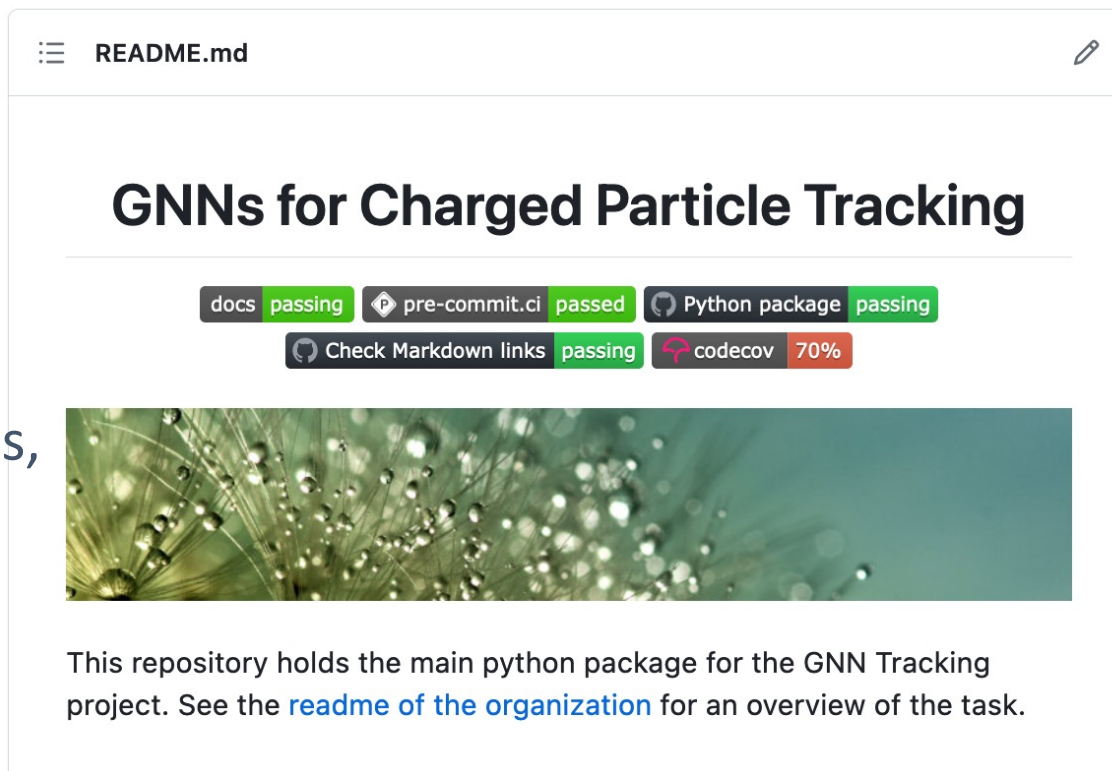
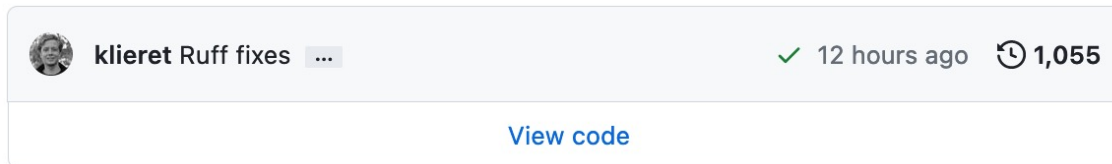
Condensation point

Represents the track, can learn
track parameters like p_T (WIP)

Tech stack & implementation



Fully open source framework:
github.com/gnn-tracking



Ongoing fights: SLURM & ML

Perhaps to discuss in **#19**

- Our infrastructure (in my case, **SLURM**) is often not a first class citizen for ML frameworks: How can we get them to play nice together?

Example 1: SLURM + **Weights & Biases**

Weights & Biases

- Batch nodes usually don't have internet, but Weights & Biases syncs to a cloud.
- Solution (self-advertisement): Trigger synchronizations from the login node
github.com/klieret/wandb-offline-sync-hook
- Alternative: Use ray

Example 2: SLURM + **Ray Tune**



- SLURM example from Ray docs is probably not what you want
- But can start ray head on login node, then allocate ray workers ([example](#))
- But how to handle timeout of nodes with long training jobs? Ideally would like to not accept job if we cannot finish it and instead resubmit request for new node

Positive example: **SLURM + Lightning**

SLURMEnvironments can checkpoint + resubmit itself

Collaborating on ML R&D

Discuss in **#26**

How can we structure **ML frameworks for R & D** such that we can get multiple developers & scientists **develop models together**

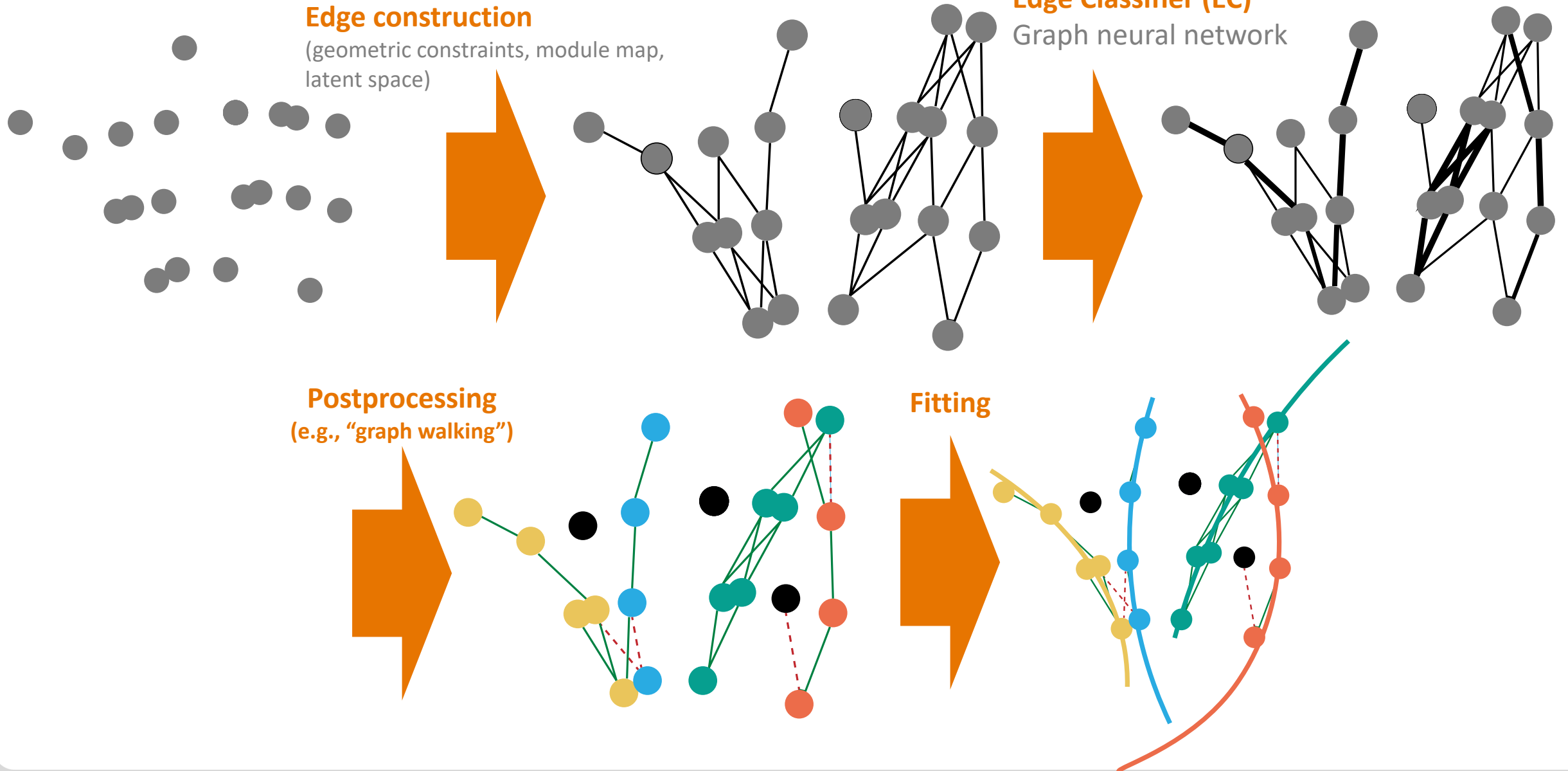
1. Want: **Cutting boiler plate** & making it fun
2. Want: **Mix & match models**
3. Don't want: **“Fork & forget”**, people starting forks for some experiment that never contribute back to the original project
4. Don't want: **Constrain creativity**

Bottom line:

- This very different from the requirements ML in production
- Both a “soft topic” and a framework question

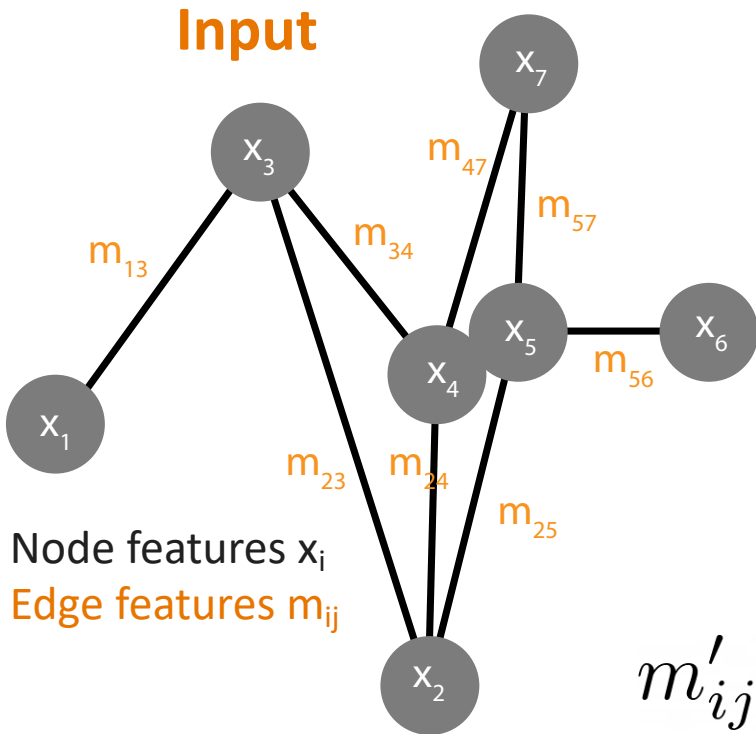
Backup

Tracking as an edge classification task

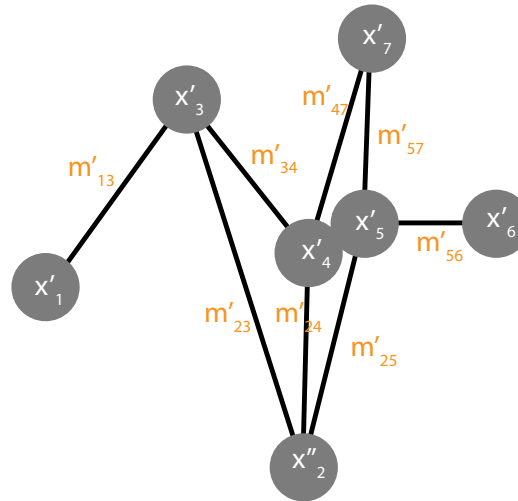


Graph Neural Networks

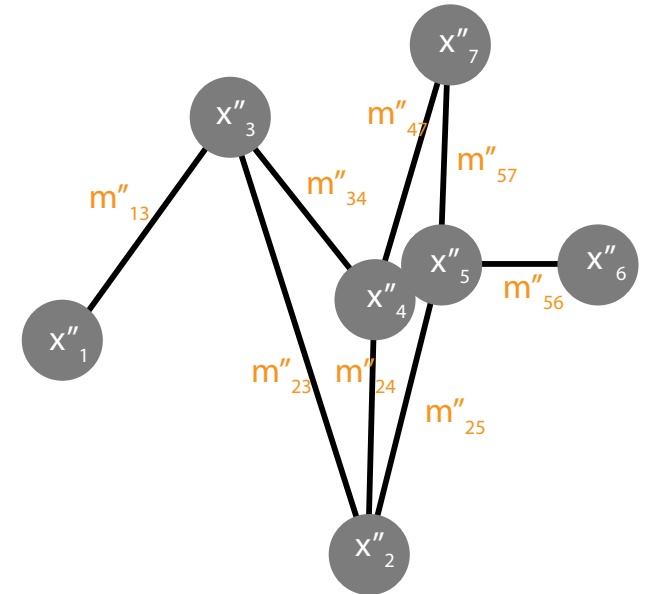
In our case: Input, latent and output is (almost) the same graph (but different features)



Latent
(usually more than one graph)



Output



$$m'_{ij} = \text{NN}(x_i, x_j, m_{ij})$$

$$x'_i = \text{NN}\left(x_i, \bigoplus_{j \in \mathcal{N}_i} m_{ij}\right)$$

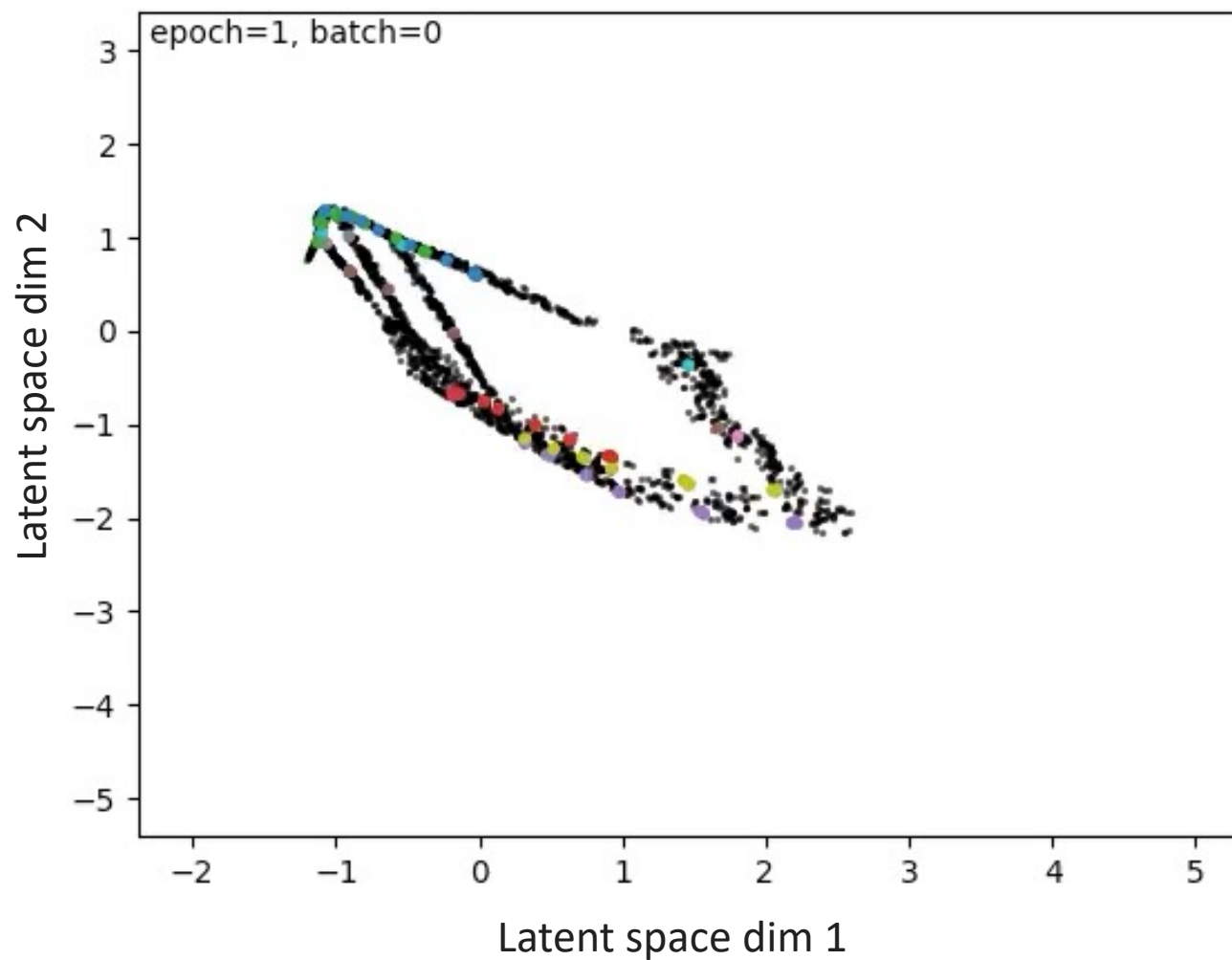
all neighboring nodes $\rightarrow j \in \mathcal{N}_i$ \leftarrow some permutation invariant aggregation

Edge classification: Final edge features are probability that edge is correct (connects hits that belong to the same particle)

Object condensation in action

2D latent space; random selection of particles colored

Early simplified study (much fewer hits than in real life)



[Click here if video doesn't play](#)

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 their CP
quadratic potential
Attraction stronger if CP's CL is high

Repulsive loss function

penalizes hits close to other 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](#))

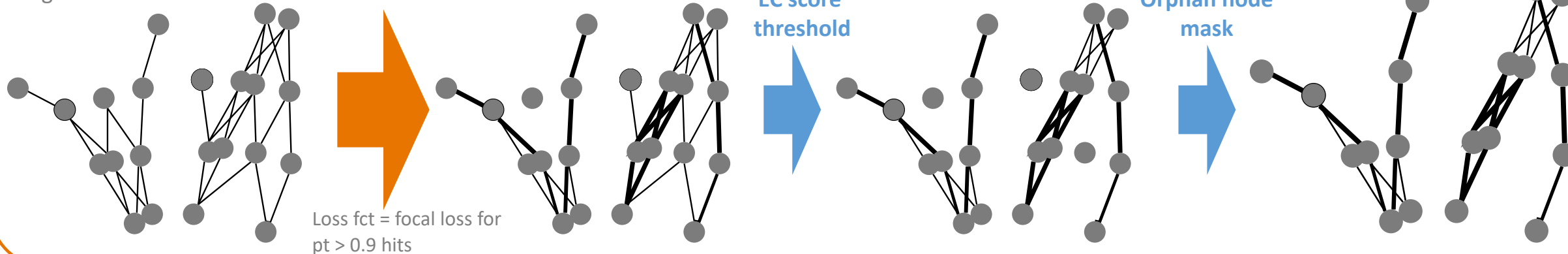
Object condensation: Our current pipeline

STAGE 1: EC

→ Working on replacing this with dynamic edge creation “Point cloud network”

Graph construction based on geometric cuts

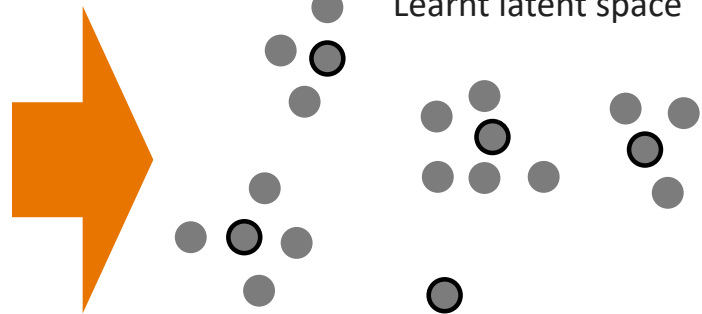
EC GNN



STAGE 2: OC

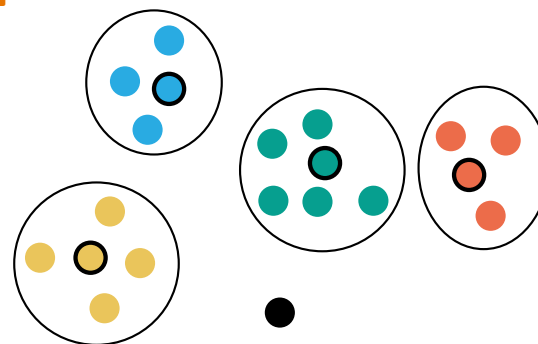
OC GNN

Learnt latent space



STAGE 3: Collect clusters

DBSCAN



- All three stages have their own hyperparameters
- Can be trained/optimized separately (fixing the previous stage)