First Attempts at using GNNs for Clustering Hits in ECAL

Maria Giulia Ratti, Anne-Mazarine Lyon, Mauro Donegà

ECAL Clustering meeting

September 09th, 2020

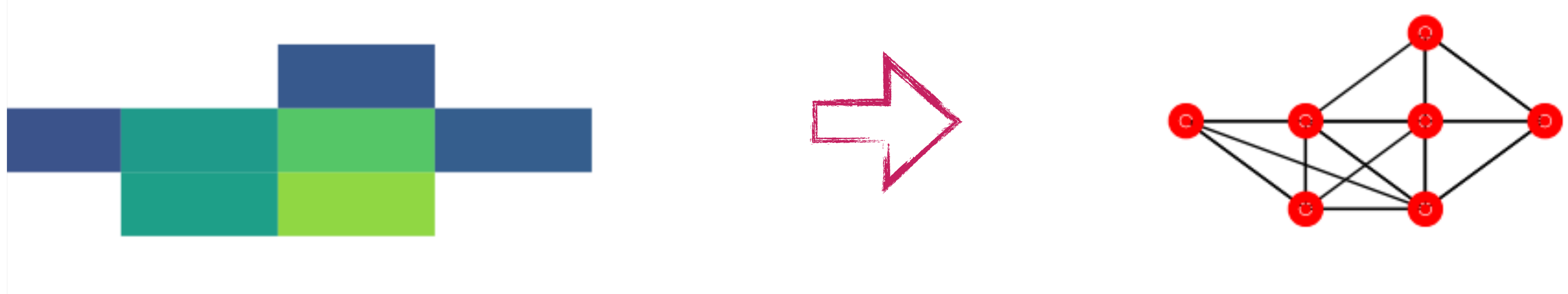
IPA
ETH zürich
CMS Compact Muon Solenoid

Introduction

Idea is to study an alternative algorithm to the PFClustering

One possibility:

- think energy clusters as graphs, were nodes are the PFRechits and edges are connection between hits
- use graph neural networks (GNNs) to do clustering as supervised edge classification problem
  - do the hits connected by the current edge belong to the same particle or not ?



Inspired by "*Graph neural networks for particle reconstruction in high energy physics detectors*" [arXiv:2003.11603]

Implementation via graph_nets library by DeepMind (based on Tensorflow + Sonnet)

Accompanying paper: "*Relational inductive bias, deep learning and graph networks*" [arXiv:1806.01261]

Construction of the Graph

Starting point for the graph:

- nodes: all PFRechits in ECAL EB (features: *energy, ieta, iphi*)
- edges: all possible combinations between any hit and any other hit (feature: ΔR)

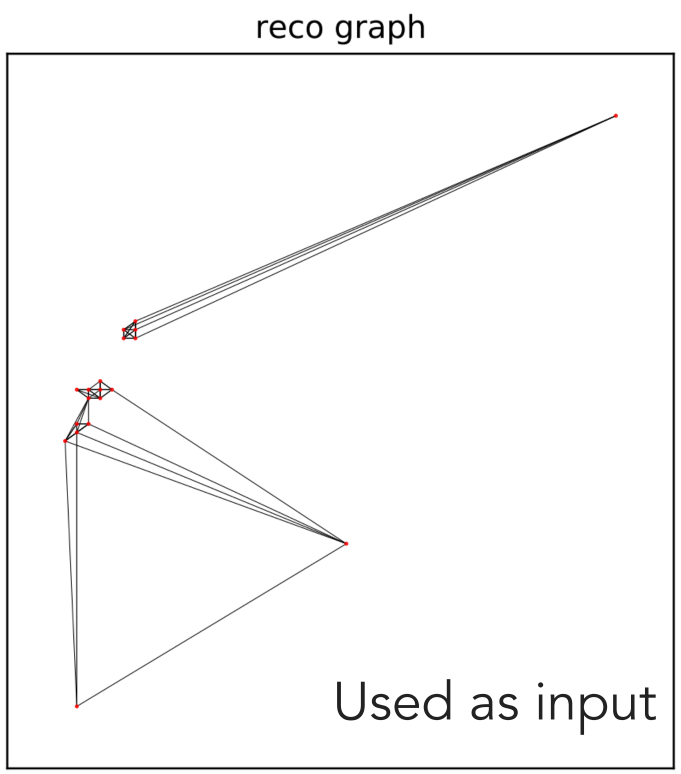
Not usable in a realistic scenario with hundreds of hits (possible edges ~ n^2)

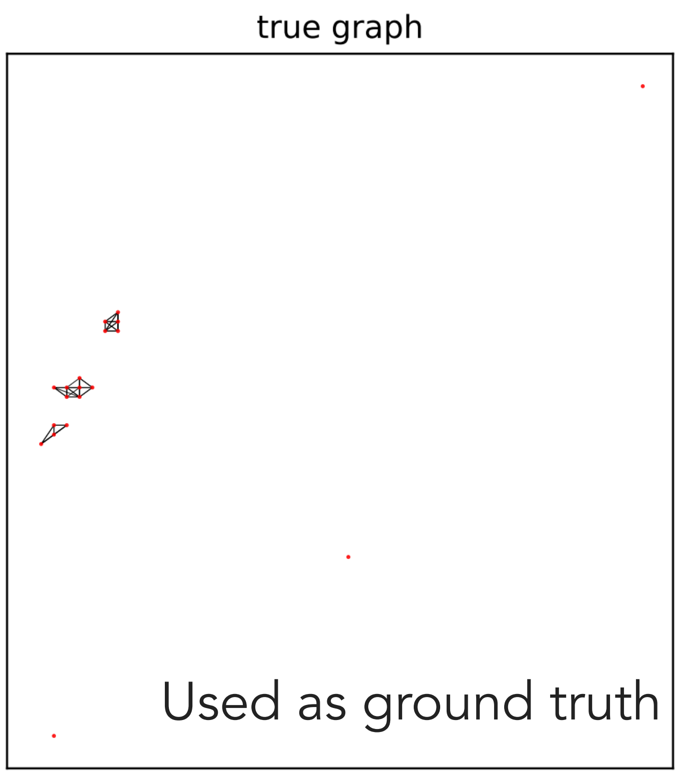
K-NN as an unsupervised pre-clustering step:

- find k=4+(1=self) nearest neighbours based on Euclidean distance based on ieta and iphi coordinates
- no pre-defined windows

Define ground truth:

- edge class:
  - 1 if there are **SimHits** (with the same positions as the current PFRecHits) **belonging to the same caloparticle**
  - 0 otherwise





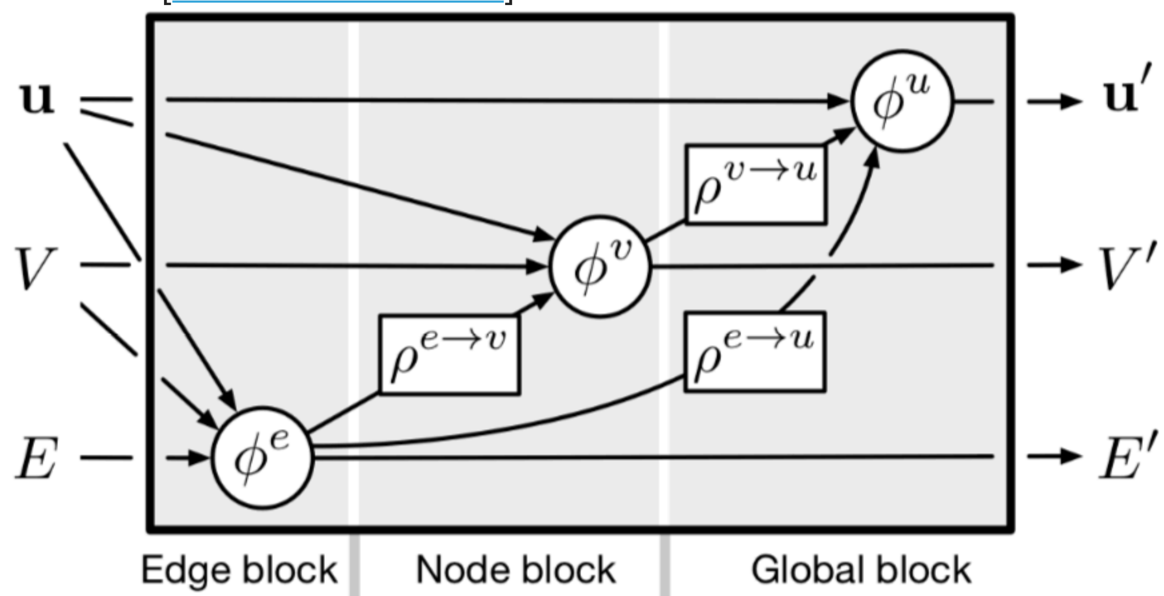
Graph Network Block

Graph Networks (GN) framework: a class of functions to do relational reasoning over graph-structured data

Main unit of computation: GN block, a graph-to-graph module

[arXiv:1806.01261]

input = graph



output = updated graph

i.e. properties of the graph are updated

In the GN block, for each update, a series of functions is applied

- updater functions ϕ for edge/node/global attributes, **learnable**

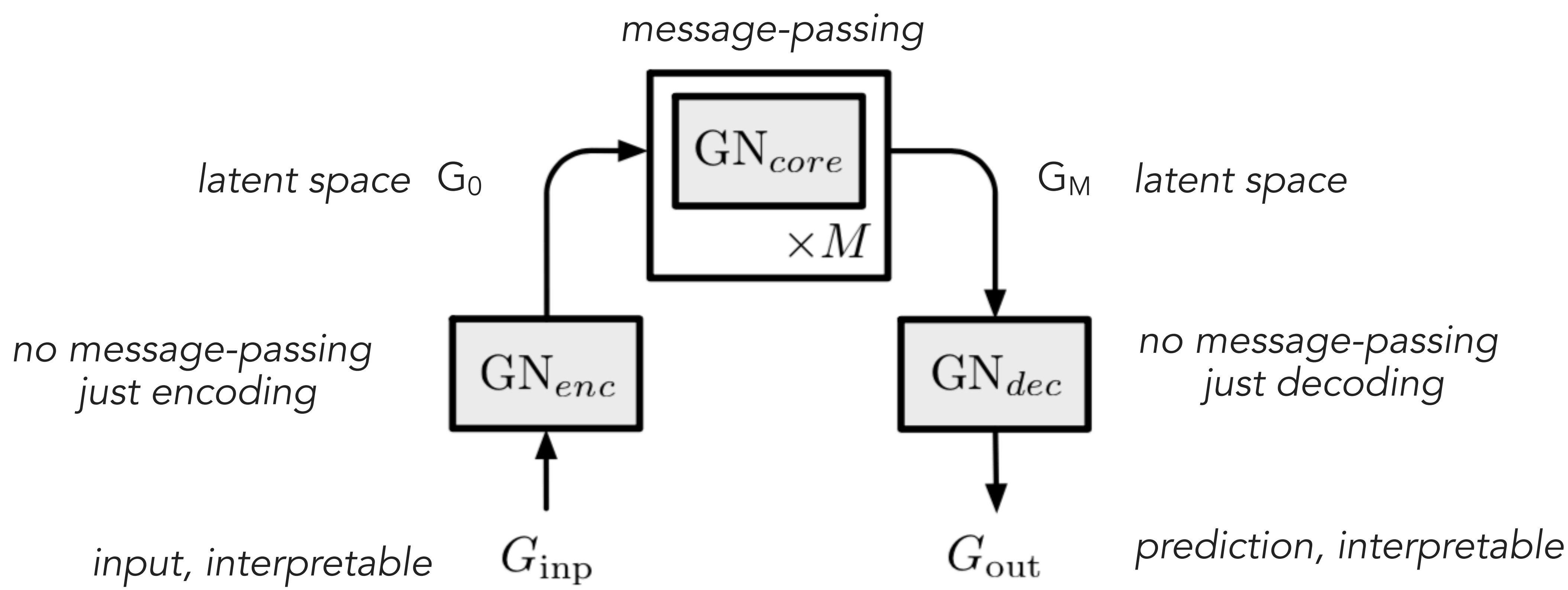
- aggregation functions ρ, aggregate information, **fixed** (in my case, *sum* is used)

$$
\begin{aligned}
\mathbf{e}'_k &= \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) \\
\mathbf{v}'_i &= \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) \\
\mathbf{u}' &= \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right)
\end{aligned}
$$

$$
\begin{aligned}
\bar{\mathbf{e}}'_i &= \rho^{e\to v}\left(E'_i\right) \\
\bar{\mathbf{e}}' &= \rho^{e\to u}\left(E'\right) \\
\bar{\mathbf{v}}' &= \rho^{v\to u}\left(V'\right)
\end{aligned}
$$

Network Architecture

One common architecture: Encode-Process-Decode [arXiv:1806.01261]



- GN_{enc} , GN_{dec} independently act on edge, nodes, global attribute, no message-passing
  GN_{core} does message-passing *(M=1 step in my case)*
- updater functions ϕs: 3 x 3 learnable independent *(in my case, each with 2 layers x 16 nodes, relu activation function)*
- edge-focused loss *(binary cross-entropy)*

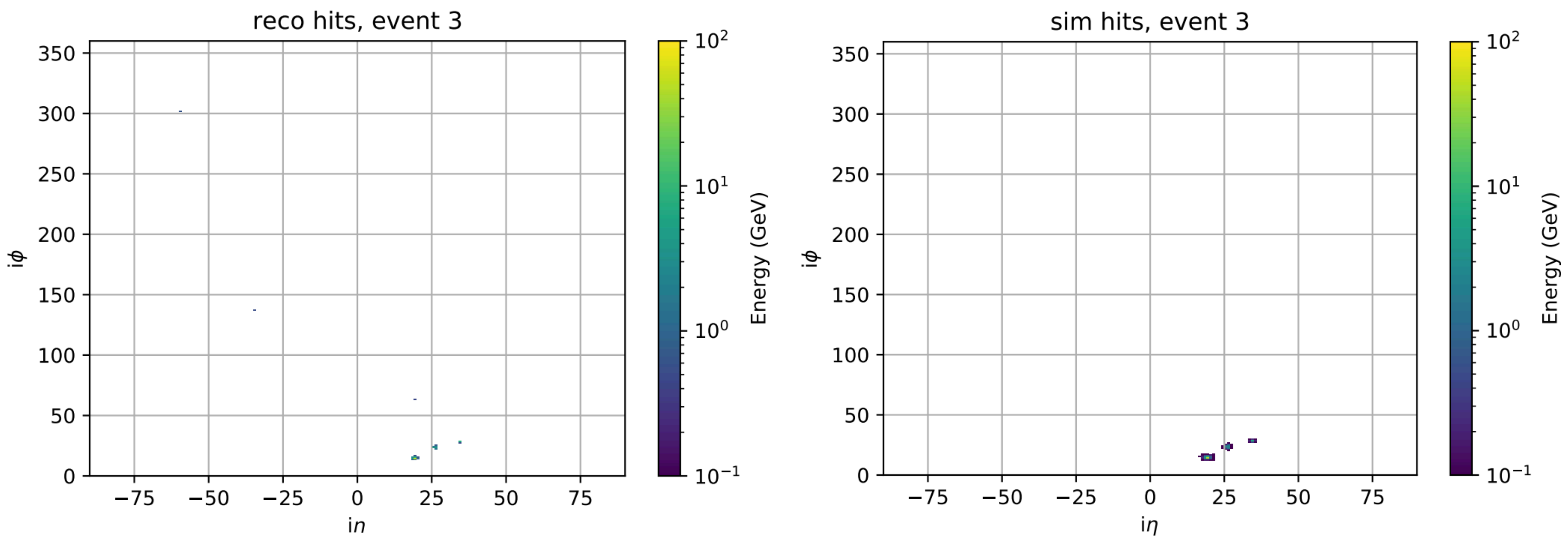
Samples and Examples

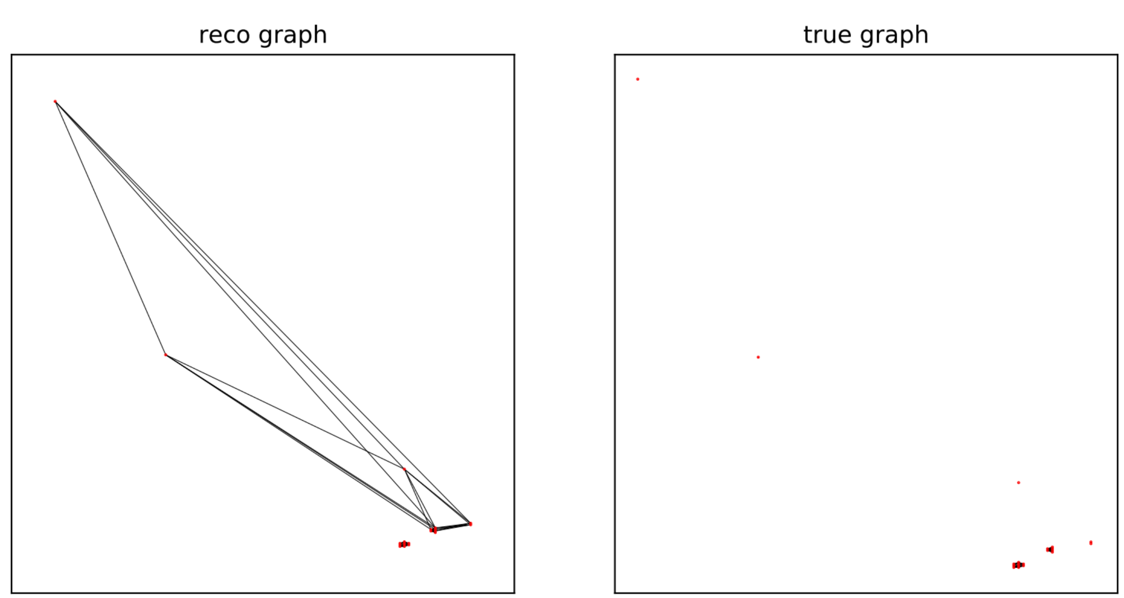
N.B.: only simhits with E>5 MeV were kept

Samples specifications:

- 2.5 K events
- no PU, ECAL conditions expected for 450/fb, EB only
- three photons in each event, no tracker, photons can overlap

Example: non-overlapping photons





all plots : https://mratti.web.cern.ch/mratti/EcalDPG/Clu/plots_wOverlap/

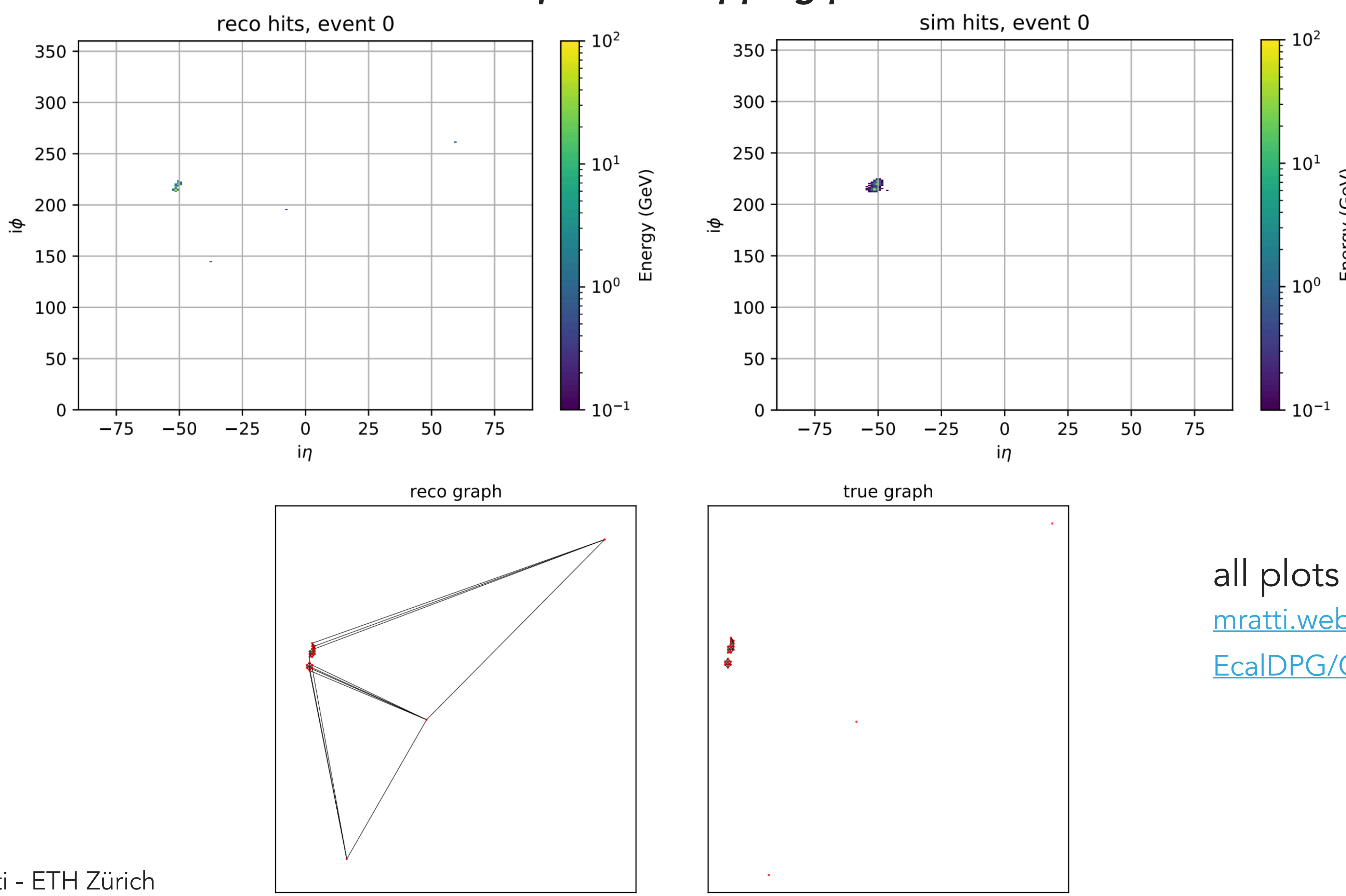
Samples and Examples

Samples specifications:

N.B.: only simhits with E>5 MeV were kept

- 2.5 K events
- no PU, ECAL conditions expected for 450/fb, EB only
- three photons in each event, no tracker, photons can overlap

Example: overlapping photons



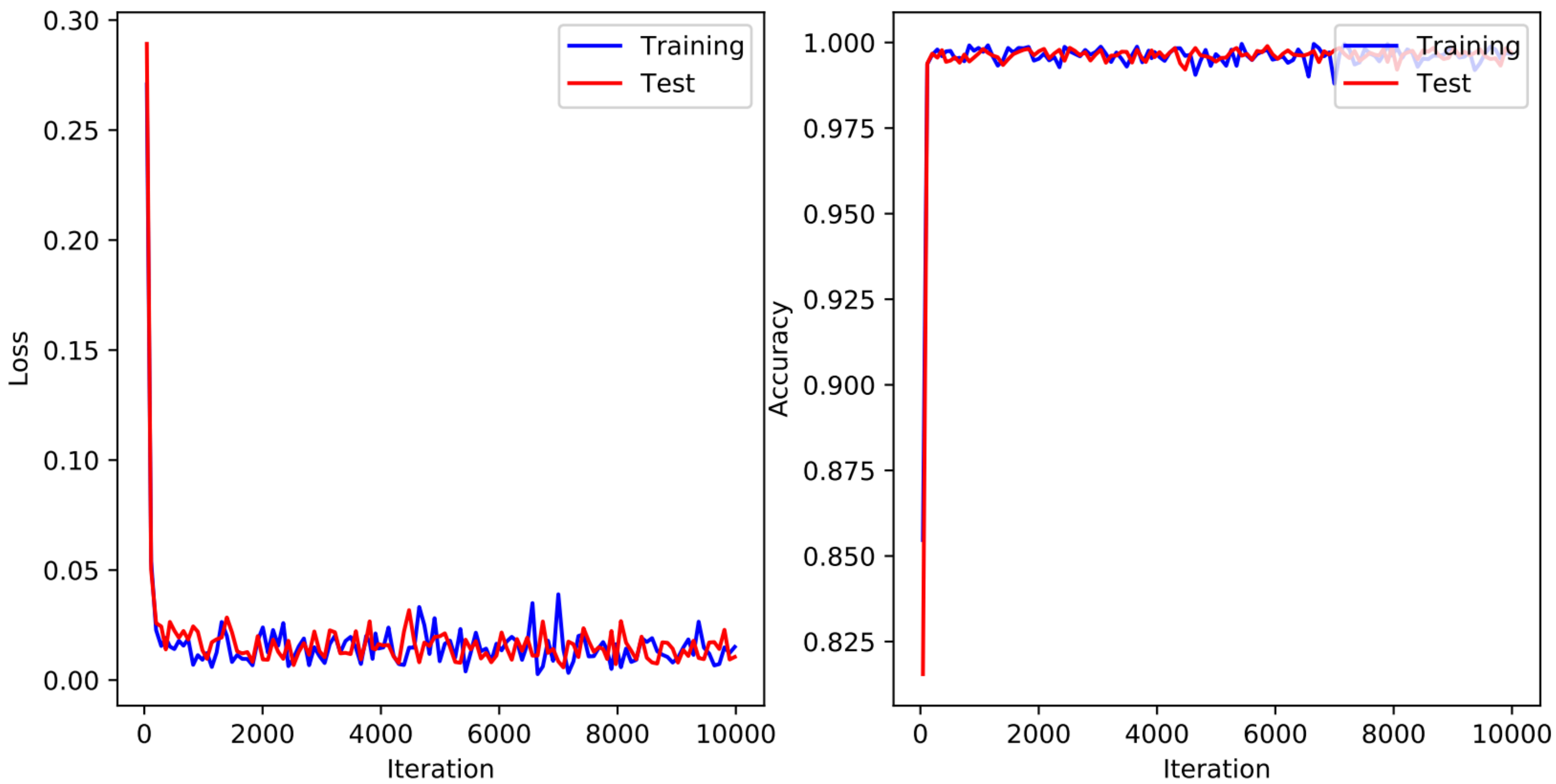
all plots : https://mratti.web.cern.ch/mratti/EcalDPG/Clu/plots_wOverlap/

Training

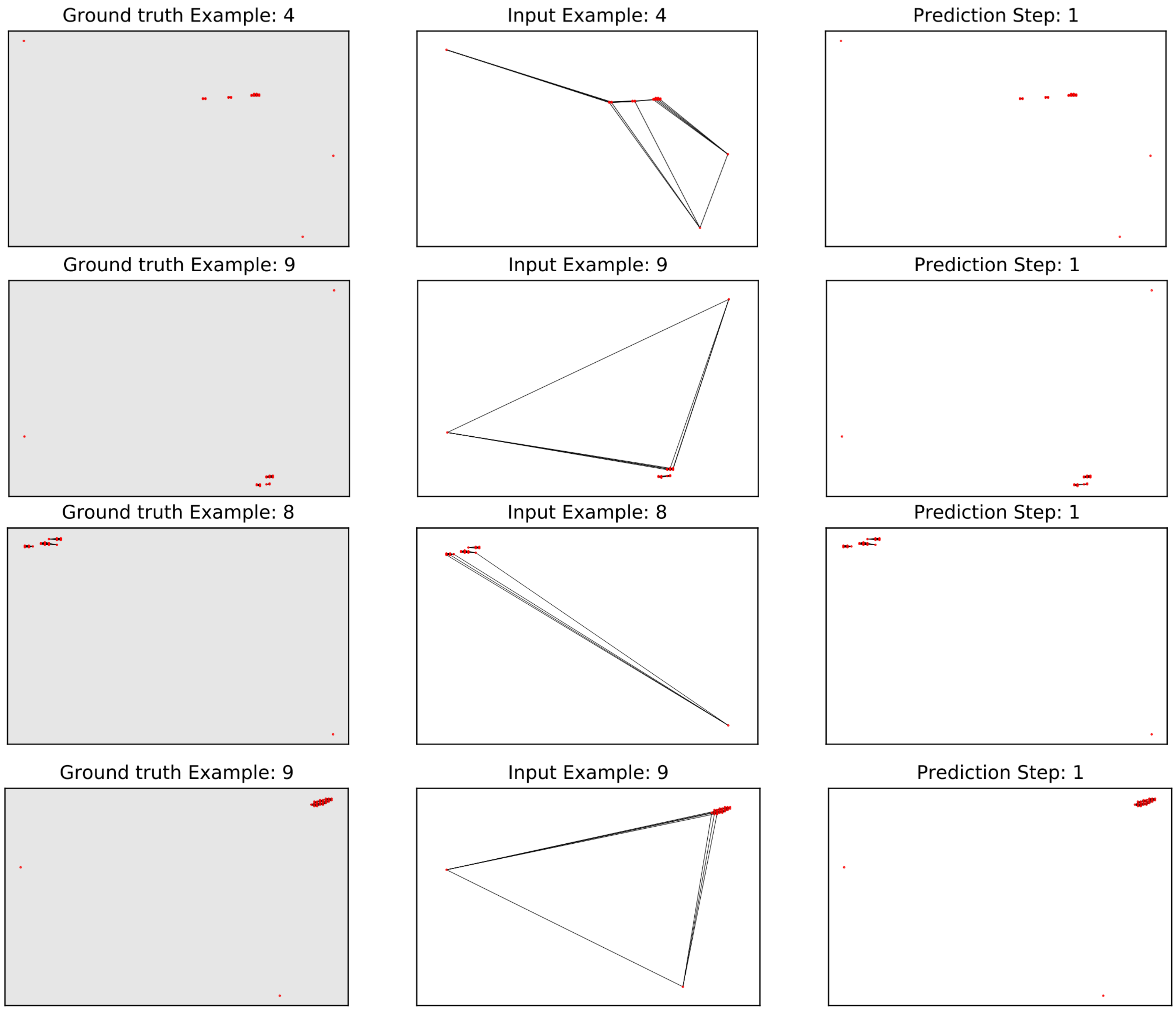
Training specifications:

- batch size: 32 for training, 50 for test
- 2.5K events (but tens of edges per event), 60% training, 40% test
- ~300 learnable parameters
- 10K training iterations
- Adam optimizer w/ learning rate=0.001

Very good performance reached very quickly: accuracy ~ 99.5%



EXAMPLE RESULTS



Summary

Good preliminary results on this simplified example

- but for the moment limited applicability:
  - in some cases the overlap cannot be resolved in the ground truth
  - for non-overlapping particles, probably not better than the existing algorithm

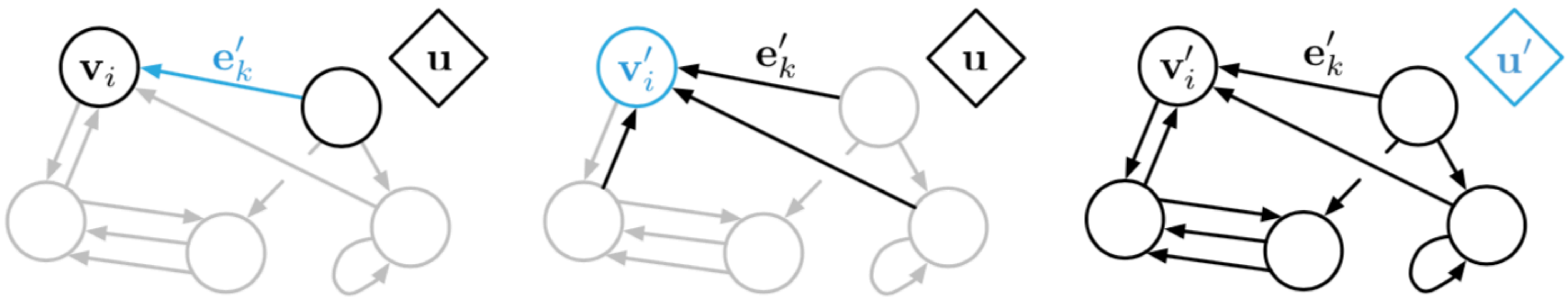
Possible improvements:

- predict to which caloparticle each edge belongs
- predict energy fractions of each node according to the simulation
- however, this requires that one specifies the maximum number of fractions per hit and that one always predicts them all

Back-up Slides

Graph Network Block

element updated in blue
other elements involved



(a) Edge update (b) Node update (c) Global update

1. for each edge, apply update:

$$\mathbf{e}'_k \leftarrow \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right)$$

e.g. compute updated potential

2. for each node, apply aggregation on the updated edges attached to the node

$$E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i,\ k=1:N^e}$$

$$\bar{\mathbf{e}}'_i \leftarrow \rho^{e\to v}\left(E'_i\right)$$

e.g. sum the potentials for each particle

3. for each node, apply update:

$$\mathbf{v}'_i \leftarrow \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right)$$

e.g. compute updated position, velocity

4. globally on all updated nodes, apply aggregation

$$\textbf{let } V' = \{\mathbf{v}'\}_{i=1:N^v}$$
$$\bar{\mathbf{v}}' \leftarrow \rho^{v\to u}\left(V'\right)$$

e.g. compute total kinetic energy

5. globally on all updated edges, apply aggregation

$$\textbf{let } E' = \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:N^e}$$
$$\bar{\mathbf{e}}' \leftarrow \rho^{e\to u}\left(E'\right)$$

e.g. compute total potential energies

6. on the graph globally, apply update of the global attribute:

$$\mathbf{u}' \leftarrow \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right)$$

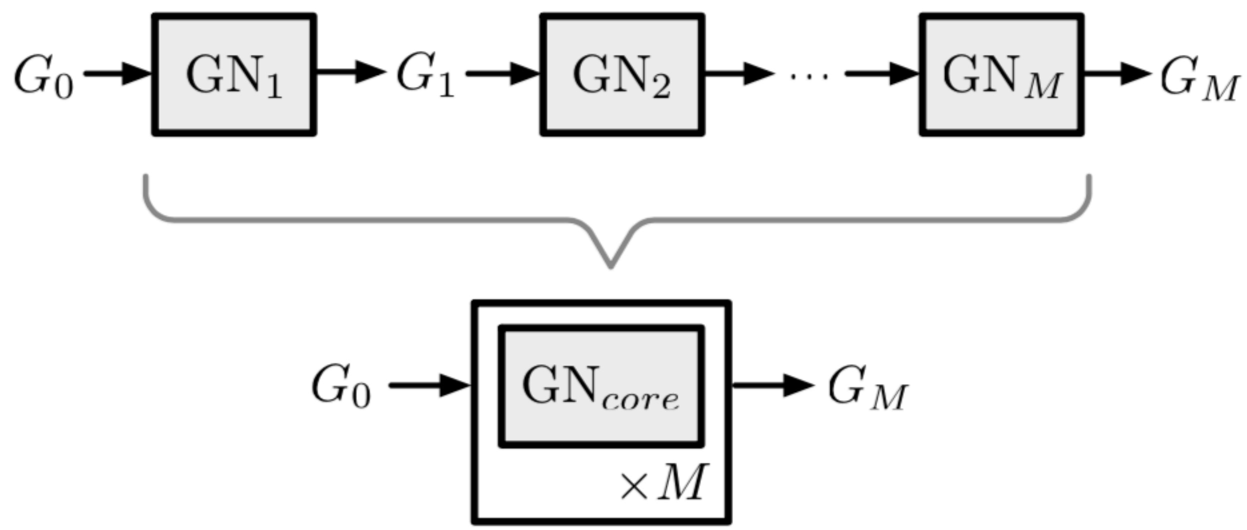
e.g. compute updated global energy

7. return the updated graph

$$\textbf{return } (E', V', \mathbf{u}')$$

Multi-Block Architectures

You can compose an arbitrary number of GN blocks with **shared** or **unshared** ϕ functions



(a) Composition of GN blocks

With shared functions, multiple GN blocks equals multiple steps of information propagation
⇒ information propagates farther in the graph (**message-passing**)