FPGA for real-time tracking at LUXE experiment

LUXE Simulation

Ashraf Mohamed

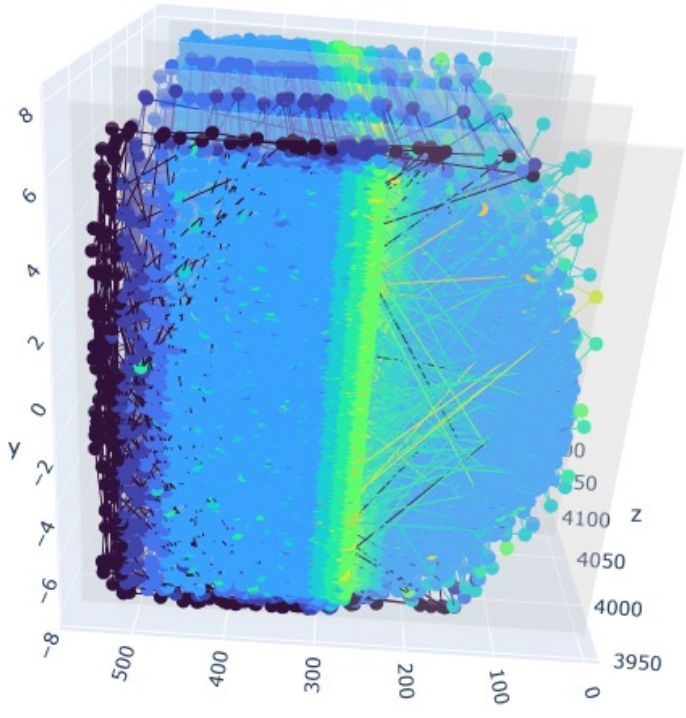
SFT meeting 14/12/2023

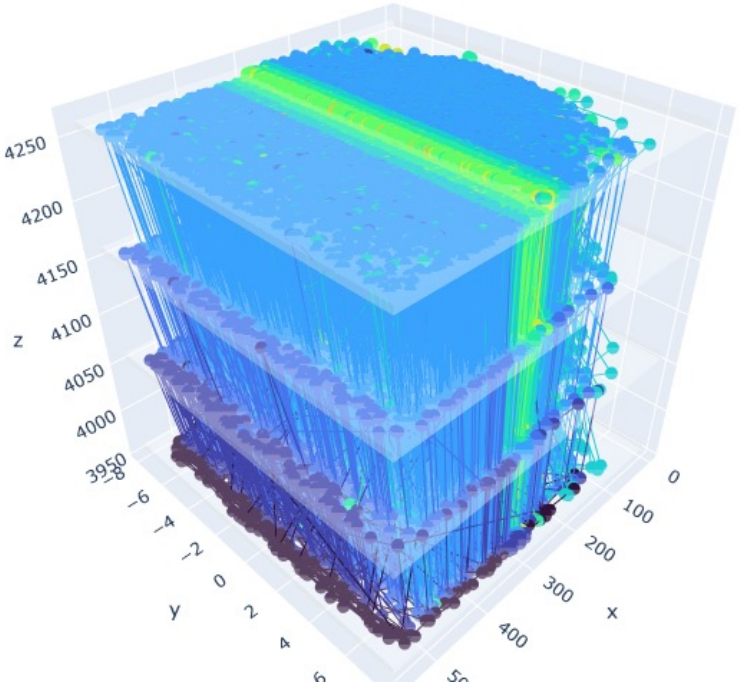
HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

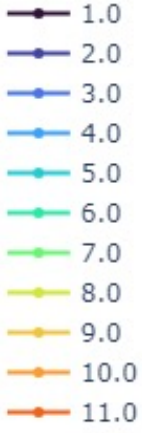
DESY.

Summary of simulations

Less is more



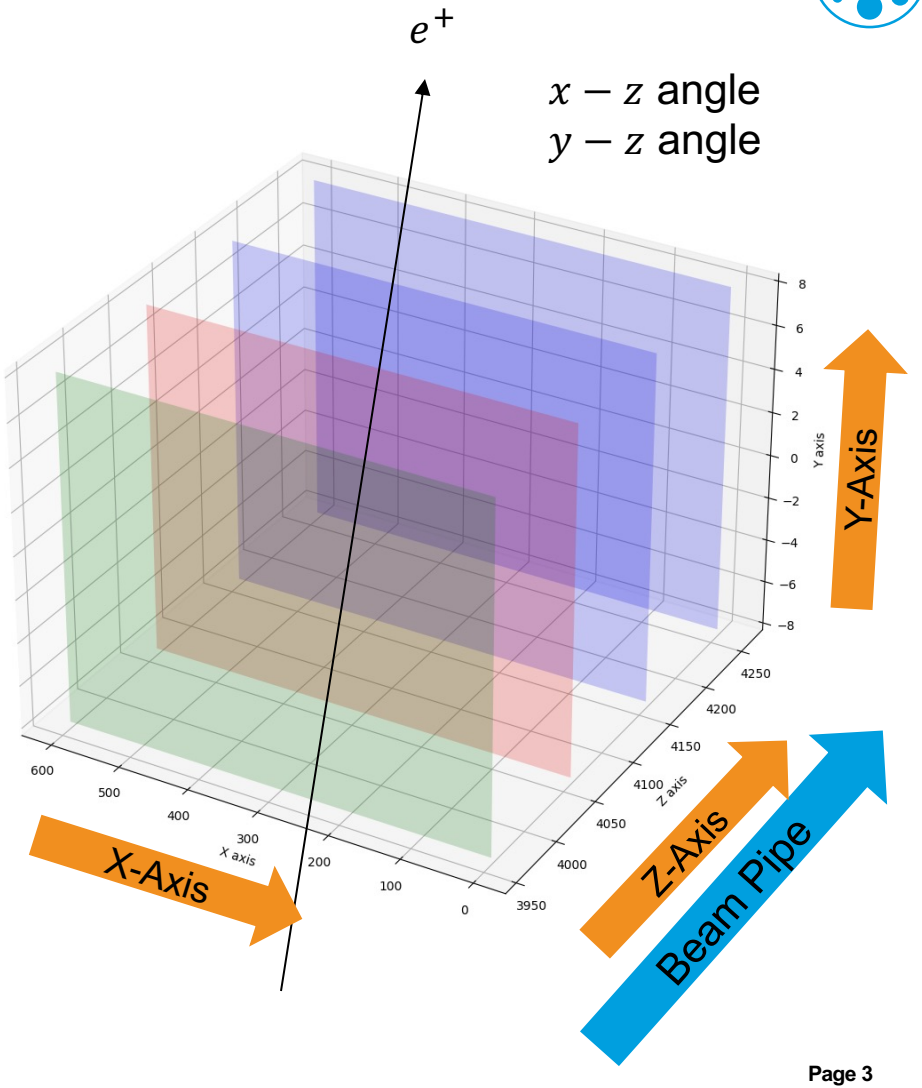




Coordinate system

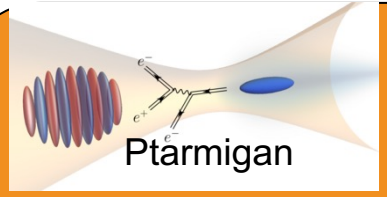
For graph building

- For graph building, we would choose every two hits that are potentially from the same track
- In order to make the graph efficient, we should make the selection window as narrow as possible
- A hit-pair should be selected if the two hits are within:
  - Two different consecutive layers (keep in mind the staves)
  - A window of cone around the expected direction of possible tracks
- Building a graph considering all the hits is not possible, even if we apply the preselection



workflow

https://github.com/ashrafkasem/LUXEsimulationHTC



.h5

pyLCIO

.slcio

luxego

DD4hep

edm4hep

pandas

DESY

HTCondor Software Suite

Hits

Segments

GNN

Track segments

MarlinACTSTracking

https://github.com/ashrafkasem/LUXETrackML

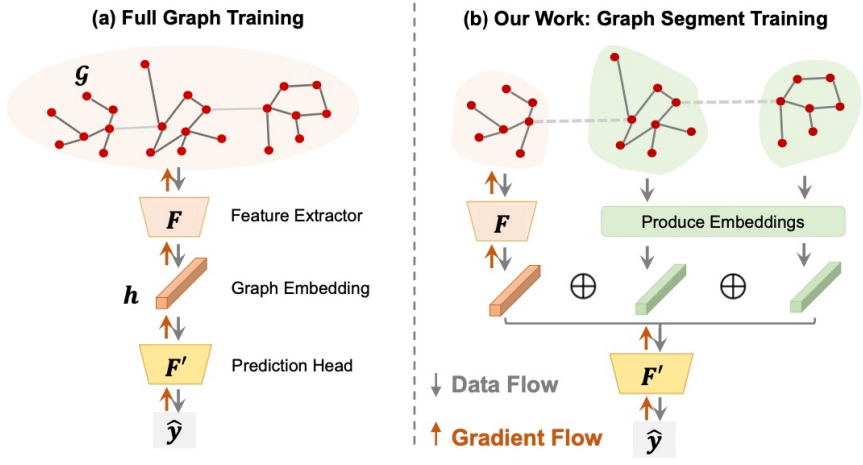
https://github.com/LUXEsoftware/MarlinACTSTracking

Sampling graph

Large graph concept

arXiv:2305.12322v2

- Full Graph Training: Classically, models are trained using the entire graph → all nodes and edges are used to compute gradients
- Graph Segment Training: partition each large graph into smaller segments and select a random subset of segments to update the model
- How large our graph is:
  on average: 70k particles per BX produces ~250k hits
  every layer receives ~ 70k hits
- While we build the graph we don't know the truth information we rely only on geometrical cuts to reject fakes
- Graph Production Statistics:
  Using 656 events, 46218570 particles in total
  True edges: 143009166 Fake edges: 7780995637



Bootstrapping

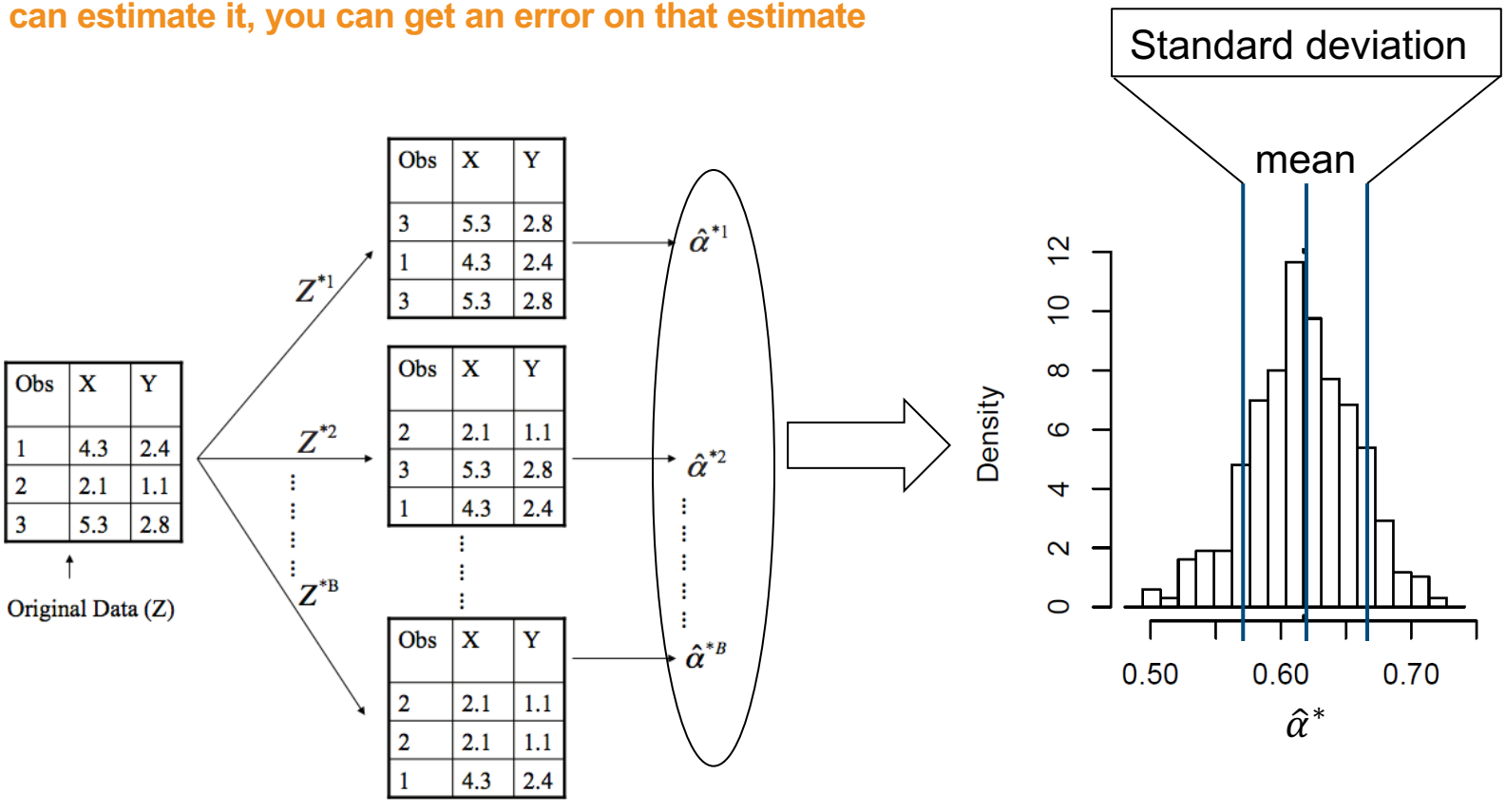
if you can estimate it, you can get an error on that estimate

- Given a sample of n independent identically distributed observations $X_1, X_2, \dots, X_n$ from a distribution F
- A parameter θ of the distribution F with a real valued estimator $\theta(X_1, X_2, \dots, X_n)$
- The bootstrap estimates the accuracy of the estimator by replacing F with F_n,
- The basic idea of bootstrap (resampling) is making inference about an estimator θ (such as the sample mean) for a population parameter θ on a data sample
- It is a resampling method by independently sampling with replacement from an existing sample data with same sample size $m < n$, and performing inference among these resampled data
- In any one bootstrap sample some events will be selected multiple times whereas some others will not be selected at all
- The distribution of the bootstrap samples is reasonably Gaussian
- The bootstrap, estimates the mean of $\theta(X_1, X_2, \dots, X_n)$ by computing or approximating the mean of

$$\theta_m{}^* = \theta_m(X_{m1}{}^*, X_{m2}{}^*, \dots, X_{mn}{}^*)$$

Bootstrapping

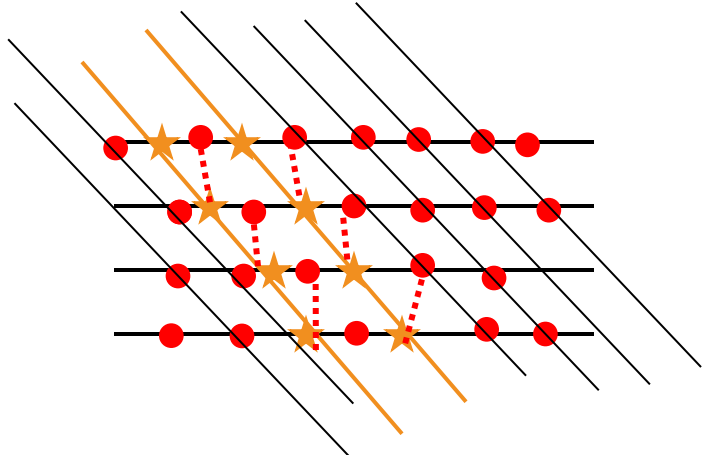
if you can estimate it, you can get an error on that estimate



Bootstrapping large graph

Sample particles not hits from google to LUXE (arXiv:2305.12322v2)

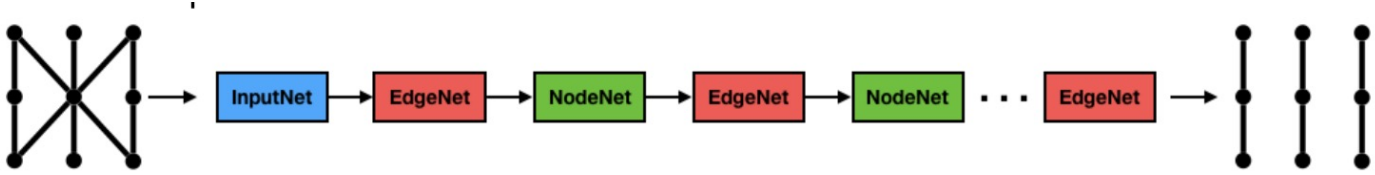
- In the aforementioned paper the graph is for social network
- In our case the graphs is of geometric nature, i.e. the segments need to stack together to form a complete track
- Sampling from the hits will lead to many incomplete tracks i.e fakes will be high
- Sampling from track segments is computationally expensive → need to build the entire graph segments and then sample from them
- Sampling from the particles list is remarkably faster and doesn't lead to discontinuous tracks
- Sampling size is dependent on the number of desired epochs --> each epoch while making the batch, we pick different sample of the same events



Graph Production Statistics.....
Using 8260 events, 29094320 particles in total
Total: True edges: 90026301 Fake edges: 244828904

GNN

Architecture



Input transformation layer

A module which computes weights for edges of the graph. For each edge, it selects the associated nodes' features and applies some fully-connected network layers with a final sigmoid activation.

A module which computes new node features on the graph. For each node, it aggregates the neighbour node features (separately on the input and output side), and combines them with the node's previous features in a fully-connected network to compute the new features

A message-passing graph neural network model which performs binary classification of nodes.

gpu : A100-SXM4-80GB

Validation size : 10%

Training size : 90%

Lr : 0.001

batch_size : 1

n_iters : 4

n_epoch : 100

hid_dim : 128

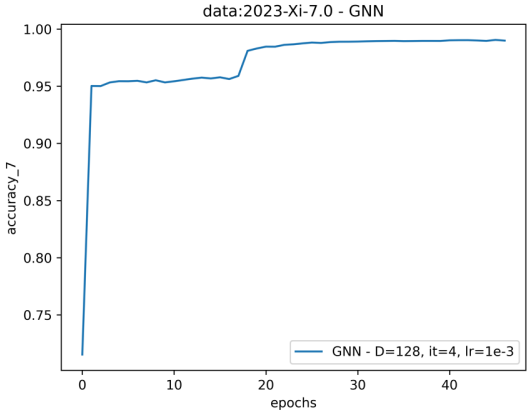
optimizer : Adam

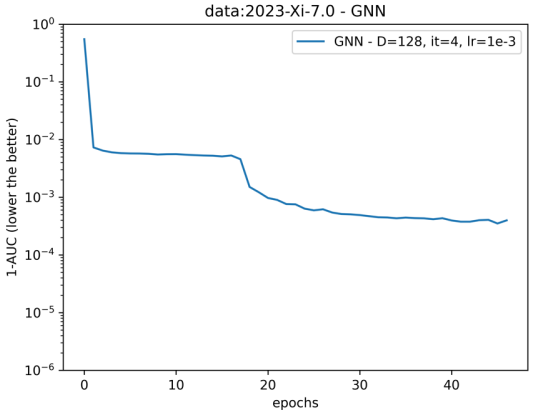
loss_func : BinaryCrossentropy

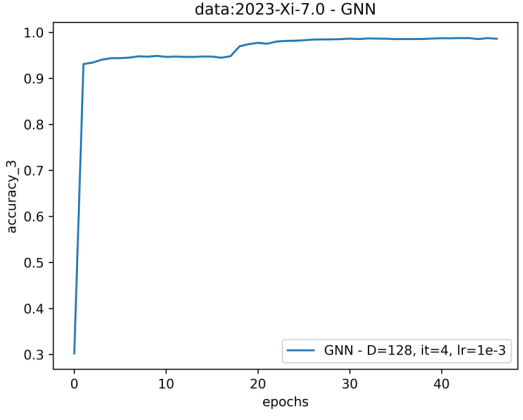
Performance

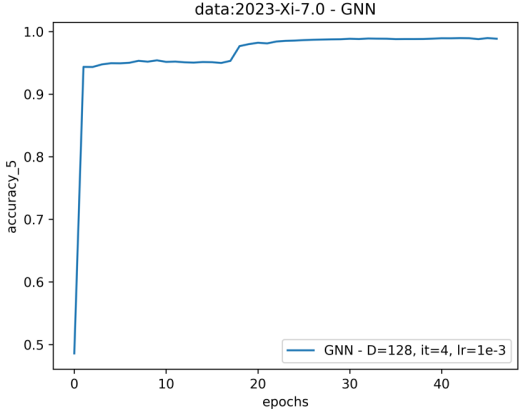
Preliminarily

- $accuracy = \frac{tp+tn}{tn+fp+fn+tp}$
- $precision = \frac{tp}{tp+fp}$ # also named purity
- $recall = \frac{tp}{tp+fn}$ # also named efficiency
- $f1 = \frac{2*precision*recall}{precision+recall}$







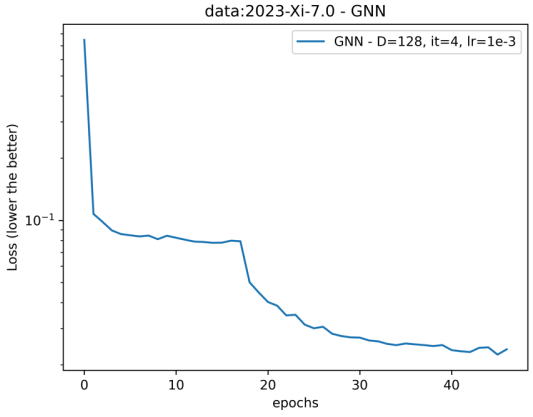


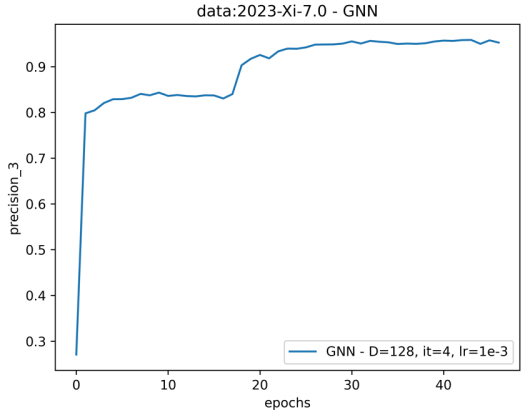
- Need closer look and some iterations on the batch_size, optimizer, learning_rates (Stochastic nature of the gradient)
- Need to validate (us the model to predict) on full original graphs as validation datasets
- Training is not yet finished, but finishes today

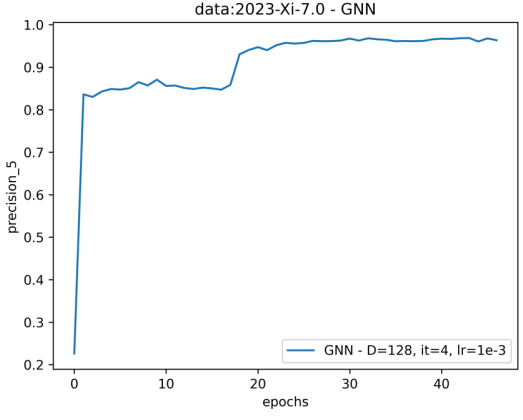
Performance

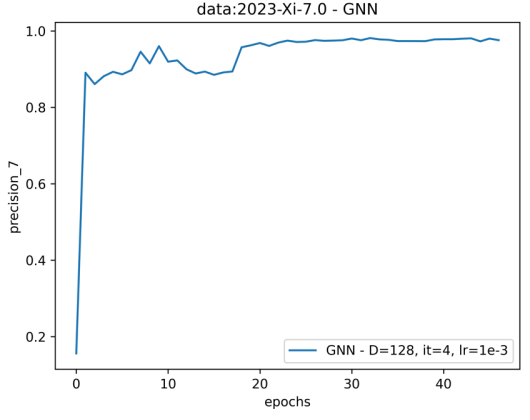
Preliminarily

- $accuracy = \frac{tp+tn}{tn+fp+fn+tp}$
- $precision = \frac{tp}{tp+fp}$ # also named purity
- $recall = \frac{tp}{tp+fn}$ # also named efficiency
- $f1 = \frac{2*precision*recall}{precision+recall}$







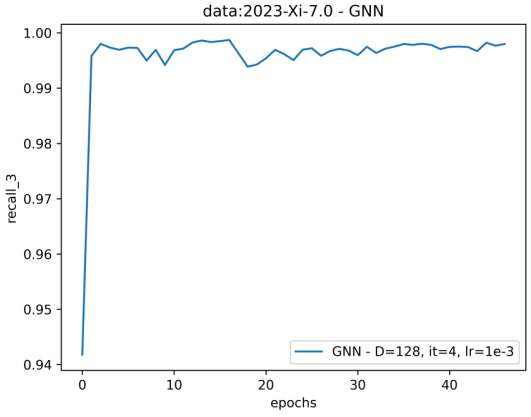


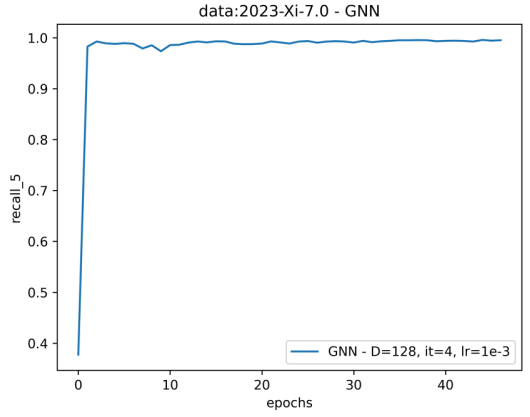
- Need closer look and some iterations on the batch_size, optimizer, learning_rates (Stochastic nature of the gradient)
- Need to validate (us the model to predict) on full original graphs as validation datasets
- Training is not yet finished, but finishes today

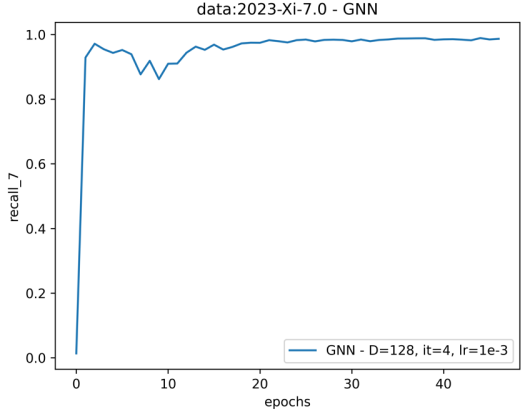
Performance

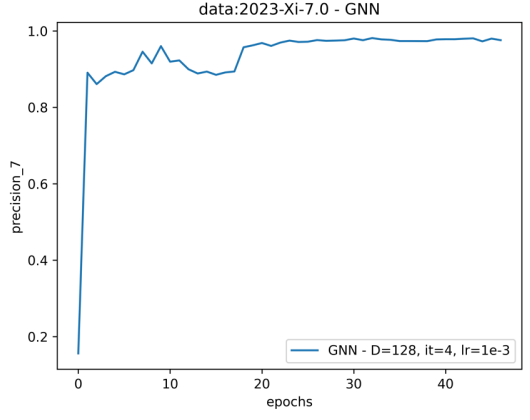
Preliminarily

- $accuracy = \frac{tp+tn}{tn+fp+fn+tp}$
- $precision = \frac{tp}{tp+fp}$ # also named purity
- $recall = \frac{tp}{tp+fn}$ # also named efficiency
- $f1 = \frac{2*precision*recall}{precision+recall}$









- Need closer look and some iterations on the batch_size, optimizer, learning_rates (Stochastic nature of the gradient)
- Need to validate (us the model to predict) on full original graphs as validation datasets
- Training is not yet finished, but finishes today

Thanks