Charged Particle Tracking with Quantum Graph Neural Networks

Matteo Argenton [matteo.argenton@unife.it]

Concezio Bozzi, Enrico Calore, Fabio Schifano, Laura Cappelli, Valentina Amitrano

ICHEP24, Prague

Università degli Studi di Ferrara

Finanziato dall'Unione europea NextGenerationEU

Ministero dell'Università e della Ricerca

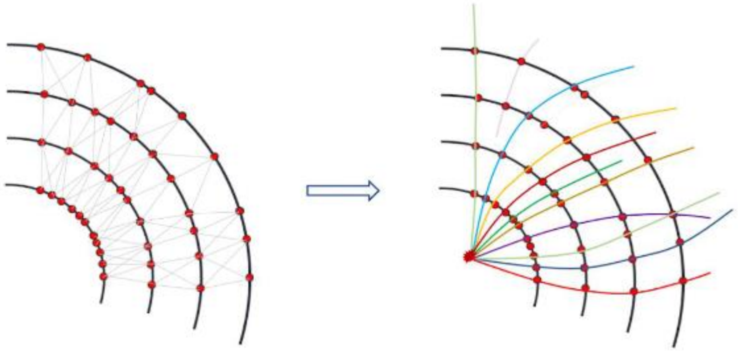
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA

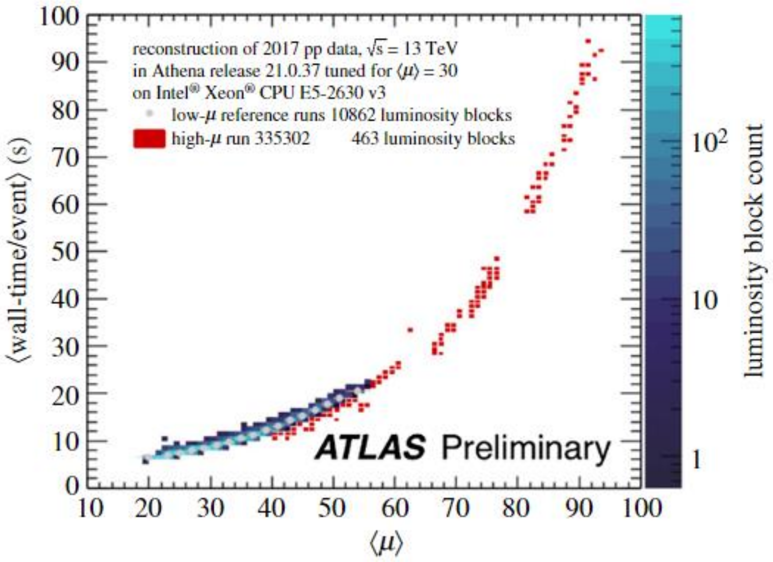
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

INFN

Scientific motivation

Tracking, the reconstruction of particle trajectories starting from particle hits in different detector layers, is already an **extremely computationally demanding task** in the major experiments at CERN



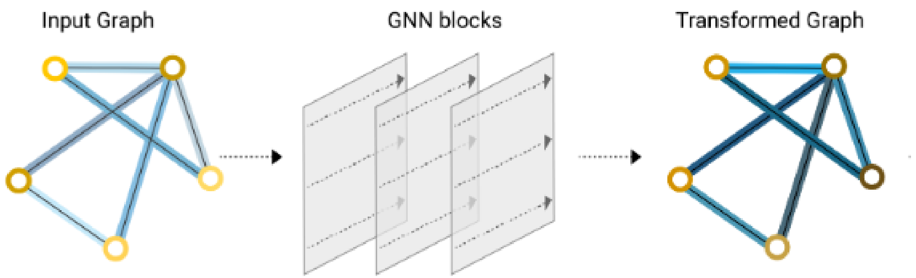


With the **High Luminosity** LHC upgrade the number of proton-proton collisions per event will increase by a factor of 3-5 (140-200 collisions per beam crossing)

A speedup in track reconstruction is mandatory and combining machine learning with quantum computing algorithms is an interesting direction for reaserch

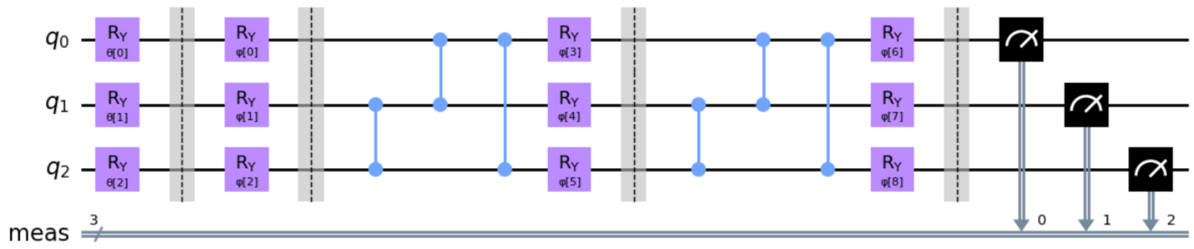
What *we have been working on

A hybrid **quantum machine learning** application for charged particle tracking



Graph neural networks

+

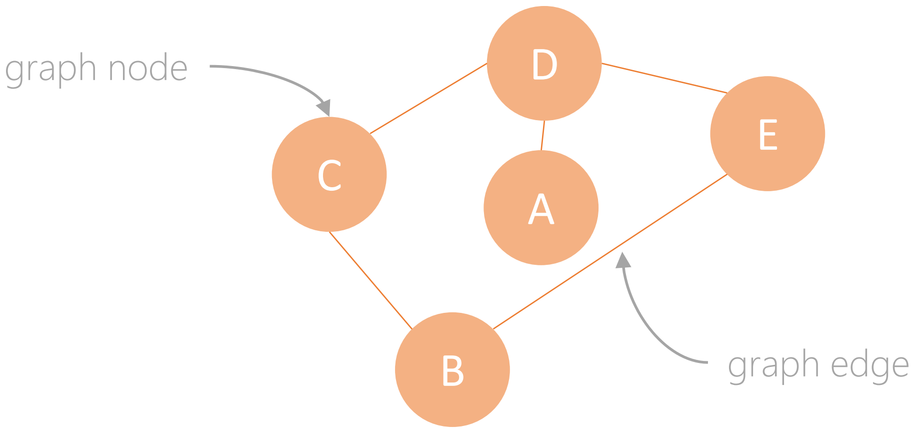


Parametric quantum circuits

*we →

Our input: graphs

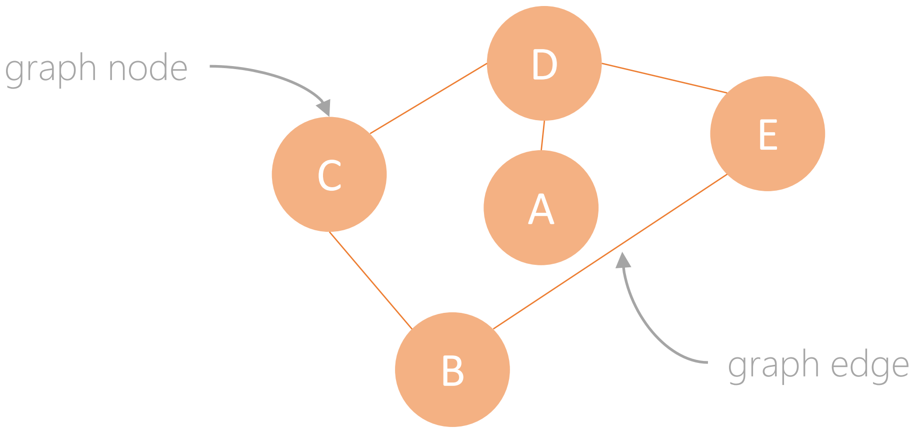
We represent charged particle tracks in a detector as a graph

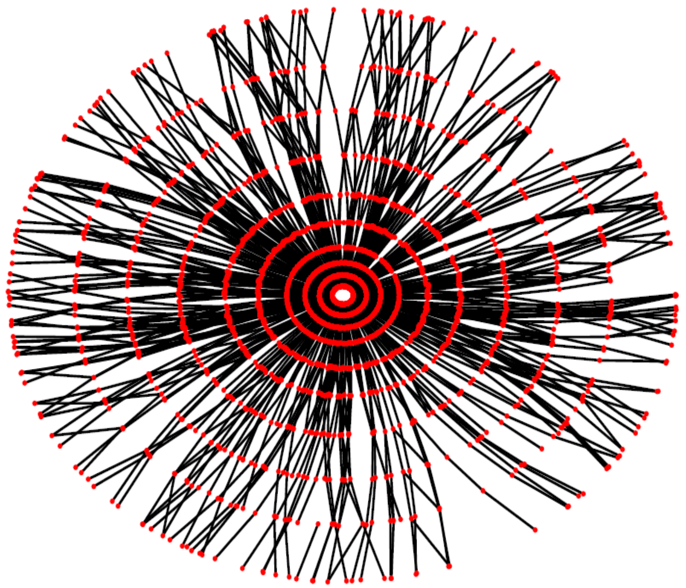


this is a simple graph → **nodes**, **edges**

Our input: graphs

We represent charged particle tracks in a detector as a graph



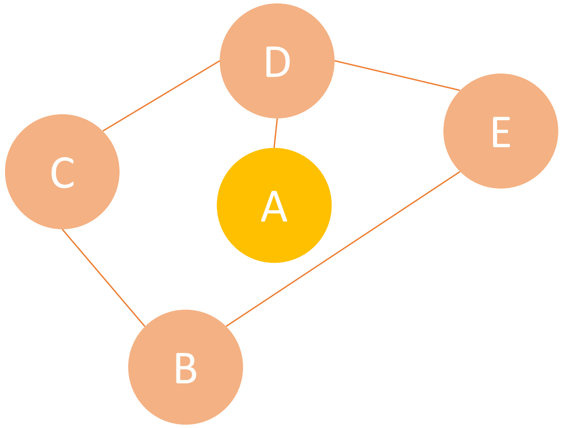


this is a simple graph → **nodes** → hits in a detector layer ← *our* graphs

this is a simple graph → **edges** → track segments ← *our* graphs

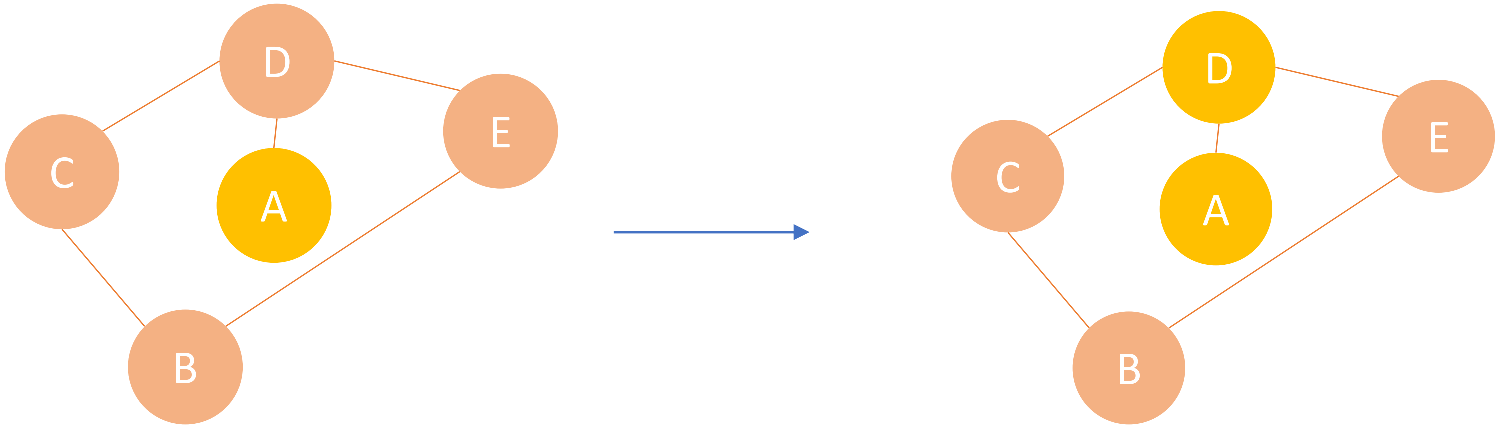
Graphs and graph neural networks

Information is propagated through the graph (with an attention mechanism: some nodes can be made more important than others)



Graphs and graph neural networks

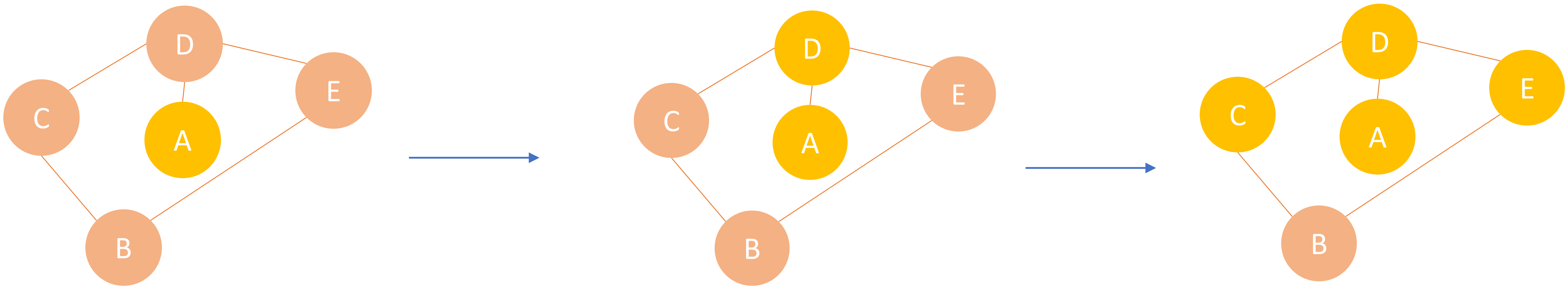
Information is propagated through the graph (with an attention mechanism: some nodes can be made more important than others)



The information associated with node A (e.g. the coordinates of a specific hit) is propagated to its **neighbors**

Graphs and graph neural networks

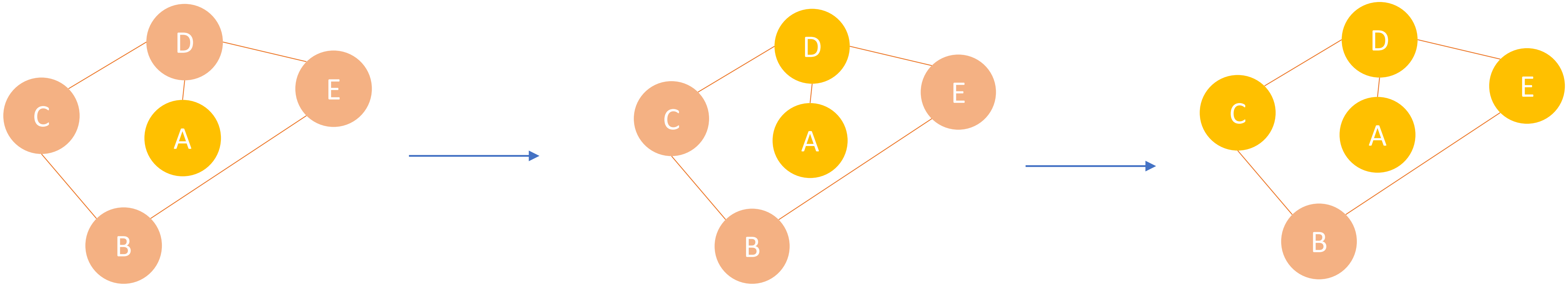
Information is propagated through the graph (with an attention mechanism: some nodes can be made more important than others)



The information associated with node A (e.g. the coordinates of a specific hit) is propagated to its **neighbors**

Graphs and graph neural networks

Information is propagated through the graph (with an attention mechanism: some nodes can be made more important than others)



The information associated with node A (e.g. the coordinates of a specific hit) is propagated to its **neighbors**

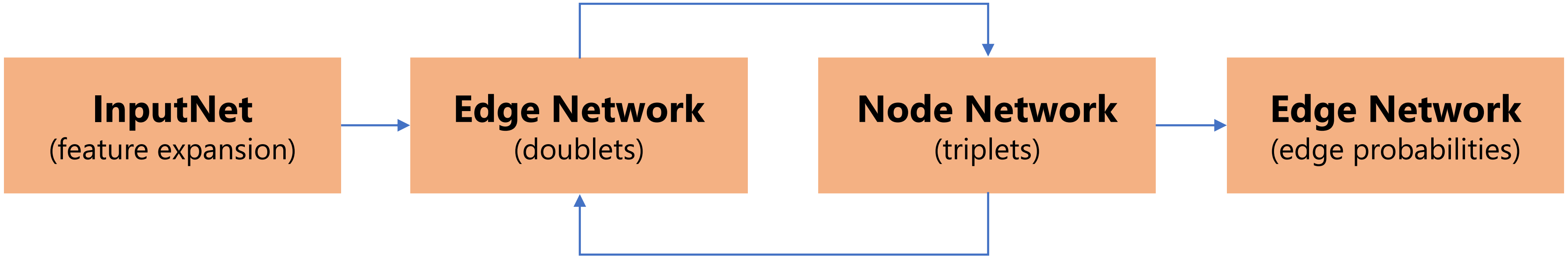
This can be done for information at edge level as well

Graphs and graph neural networks

Graphs are represented by data structures such as adjacency and feature matrices which are fed to a graph neural network

we split the propagation mechanism in two parts

- **Edge Network** → each edge in the graph is associated with a probability of being a physical link between two hits (doublet)
- **Node Network** → each node aggregates the information from its neighbors (triplet)



the number of iterations is related to the flow of information between the nodes of each graph

Quantum Computing and QGNNs

Why:

GNNs provide an interesting **global** approach to tracking

QC offers a an entirely new computing paradigm with built-in

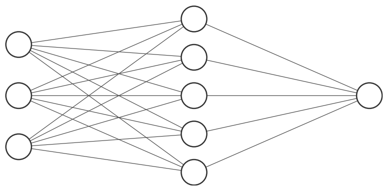
- **parallelism** (quantum state superposition and linear operators)
- **entanglement**
- **exponentially-scaling** Hilbert space in the **linear** number of qubits

Hybrid QGNNs[1] are a good candidate for the NISQ (Noisy Intermediate Scale Quantum) era

[1] first implemented by Tüysüz et. Al. https://doi.org/10.48550/arXiv.2109.12636

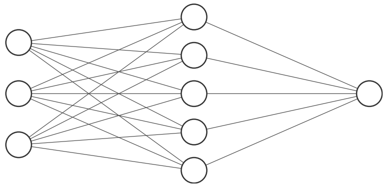
Graph neural networks and quantum circuits

Classically the Edge and Node Networks are dense fully connected layers

Edge Network ≡ 

Graph neural networks and quantum circuits

Classically the Edge and Node Networks are dense fully connected layers

Edge Network $\equiv$ 

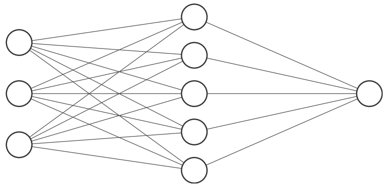
we are exploring the possibility of using a hybrid architecture employing quantum circuits

Edge Network

Graph neural networks and quantum circuits

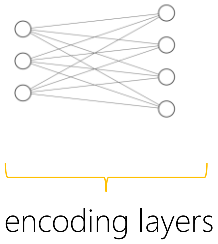
Classically the Edge and Node Networks are dense fully connected layers

Edge Network ≡



we are exploring the possibility of using a hybrid architecture employing quantum circuits

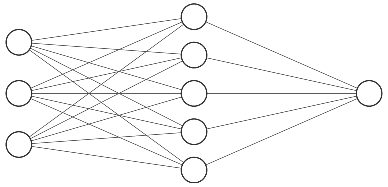
Edge Network ≡



Graph neural networks and quantum circuits

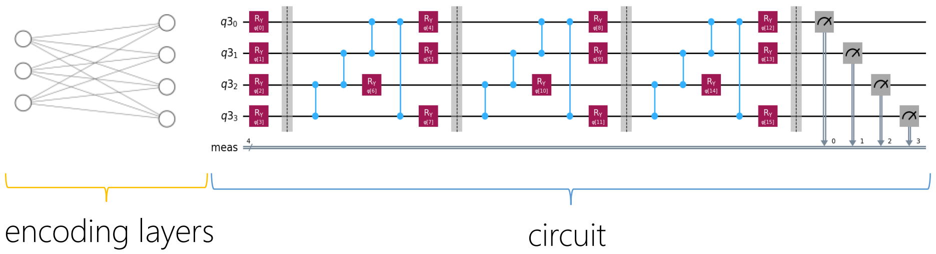
Classically the Edge and Node Networks are dense fully connected layers

Edge Network ≡



we are exploring the possibility of using a hybrid architecture employing quantum circuits

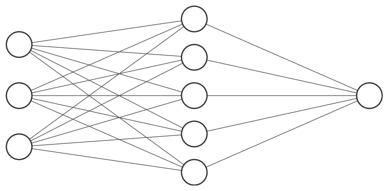
Edge Network ≡



Graph neural networks and quantum circuits

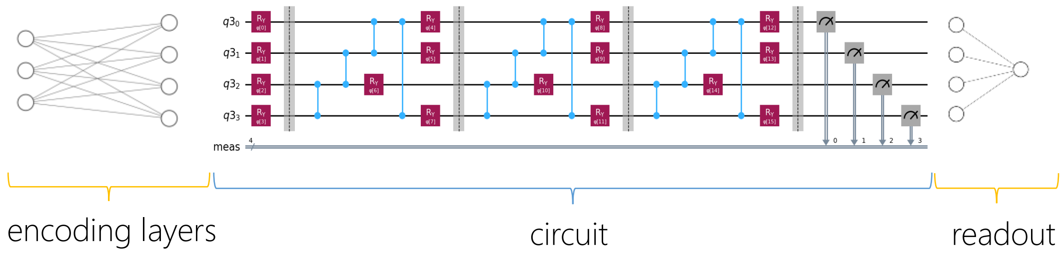
Classically the Edge and Node Networks are dense fully connected layers

Edge Network ≡



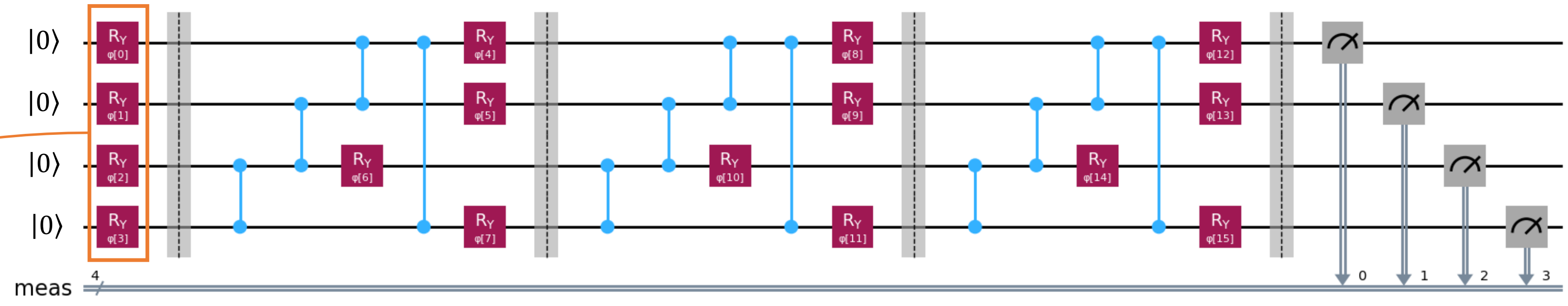
we are exploring the possibility of using a hybrid architecture employing quantum circuits

Edge Network ≡



Quantum circuits and encoding

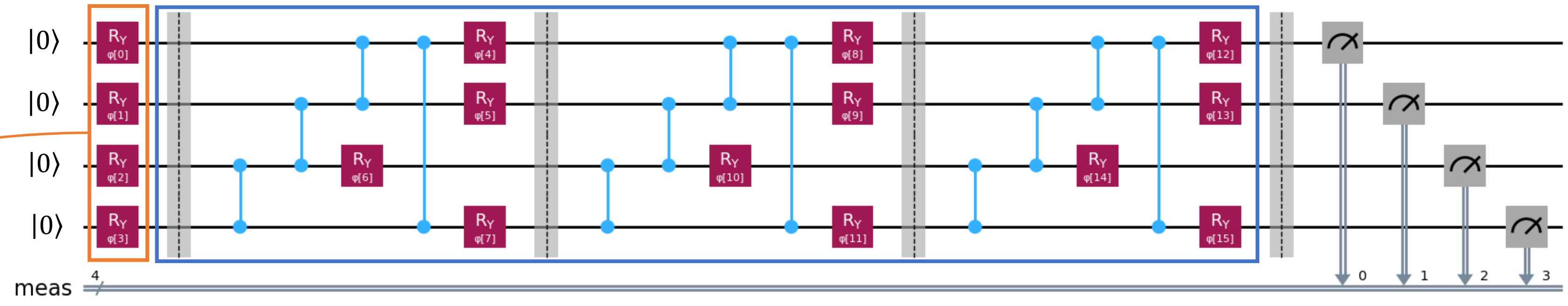
The crucial step is the embedding of classical information into the quantum circuit



the output of the encoding layers is embedded as rotation angles → these parameters are used by the encoding $R_y(\theta)$ gates to rotate the initial $|0000\rangle$ state to a new state in the Hilbert space

Quantum circuits and encoding

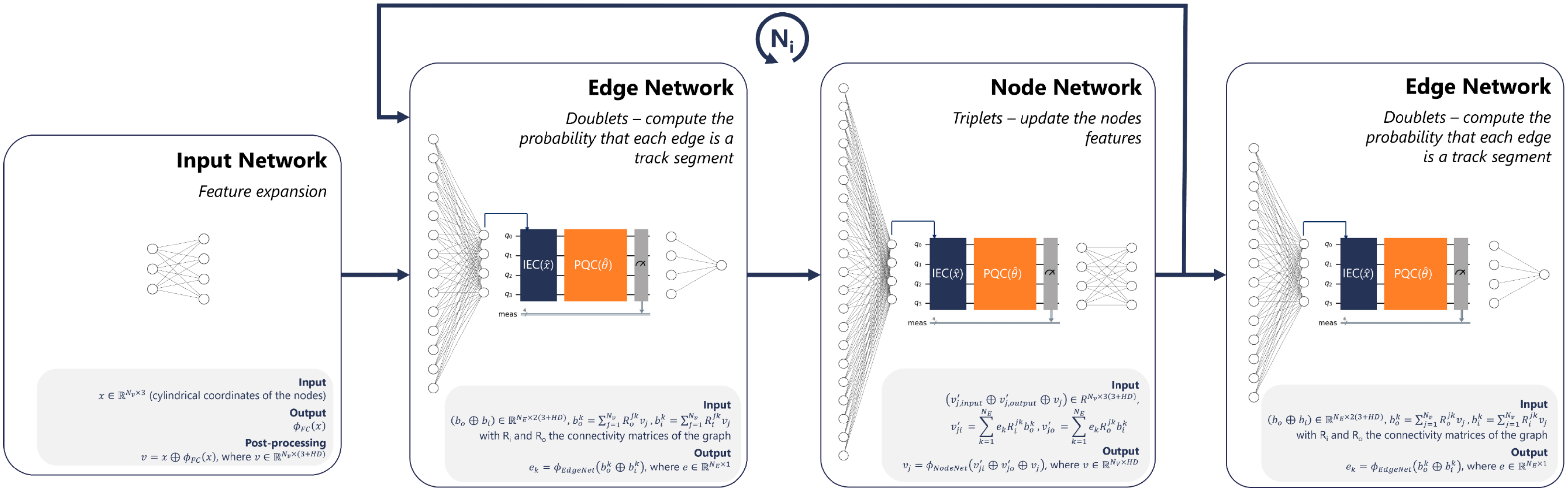
The crucial step is the embedding of classical information into the quantum circuit



the output of the encoding layers is embedded as rotation angles → these parameters are used by the encoding $R_y(\theta)$ gates to rotate the initial $|0000\rangle$ state to a new state in the Hilbert space

The second part of the circuit is called PQC (Parametrized Quantum Circuit) and its free parameters are the ones we train

Hybrid QGNN model



Our implementation(s)

We have been working on an efficient implementation of the QGNN model using different frameworks

- We use the **TrackML** dataset, which provides collision events **simulated with HL-LHC conditions** in a generic tracker

- Jax + Flax + Pennylane is the most promising version of our software (up to an order of magnitude less training time –from a few days to a few hours – compared to Torch + Qiskit and TensorFlow Quantum + Cirq)

- We study the hybrid QGNN model in terms of:
  - **accuracy** and other metrics for **increasing pileup values**
  - **noiseless**, **noisy** and **real IBM quantum hardware backends** [2]

[2] Access to the IBM Quantum Services was obtained through the IBM Quantum Hub at CERN under the CERN-INFN agreement contract KR5386/IT.

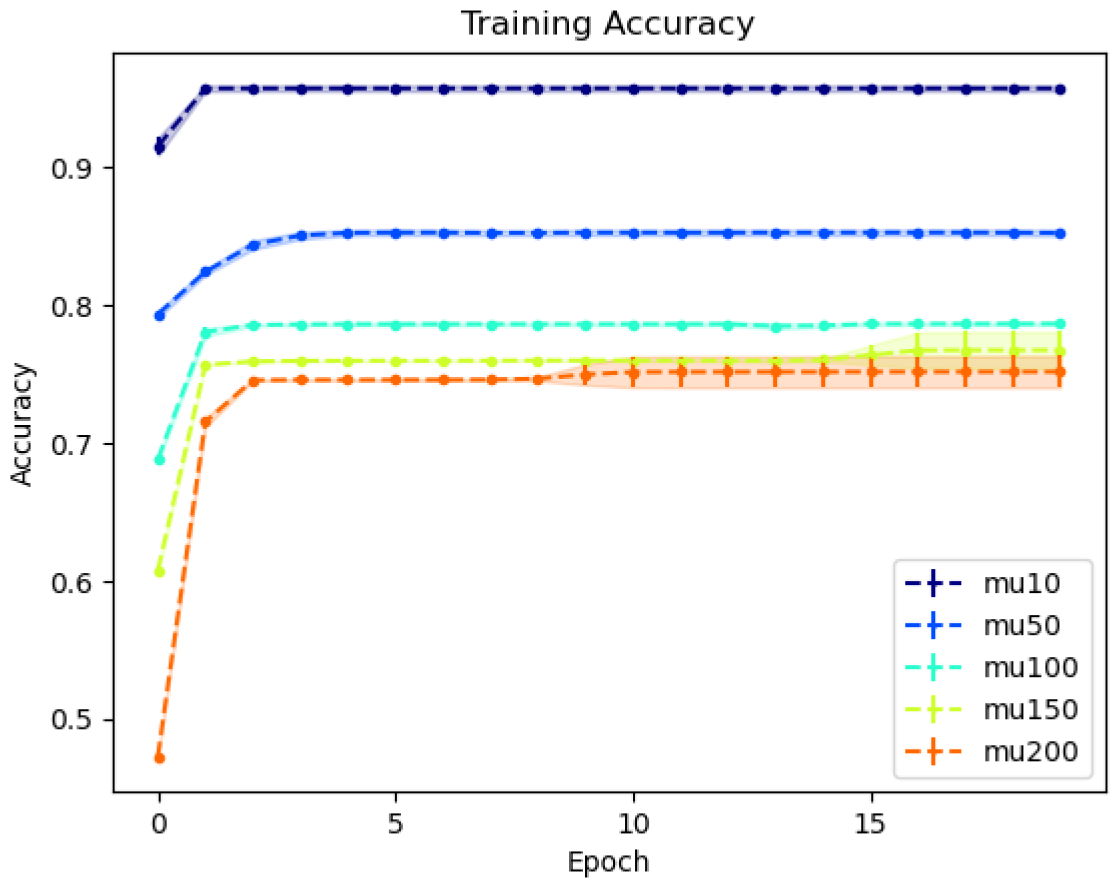
PENNYLANE

Training the QGNN

Accuracy is, as expected, higher with lower pileup

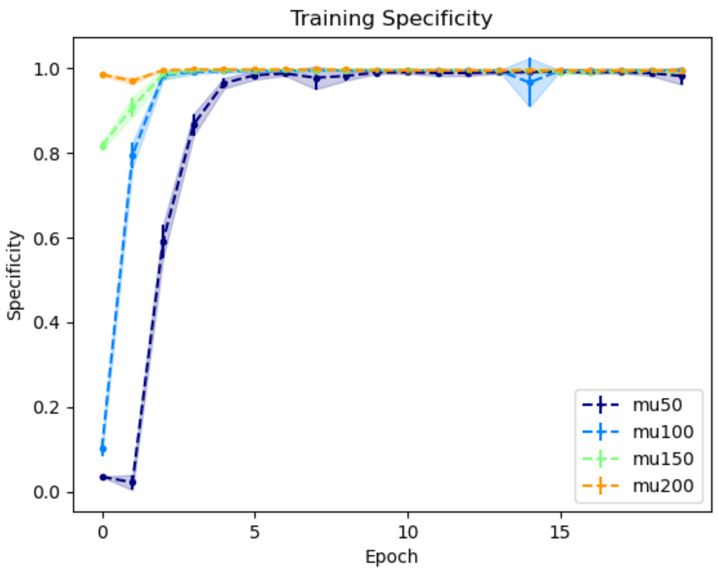
$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

- The dataset is increasingly unbalanced for decreasing pileup
- Error bars are obtained by k-folding

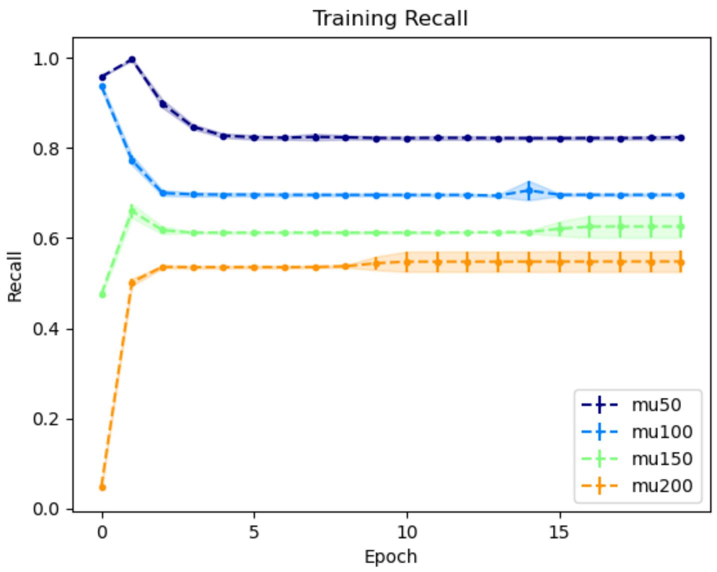


Training the AQGNN

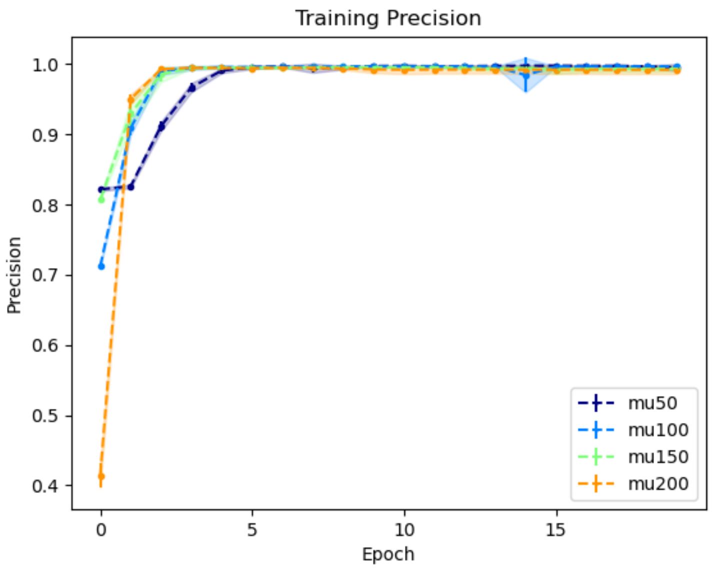
Other metrics show that the QGNN is able to correctly recognize fake edges, but struggles with true edge classification



$$specificity = \frac{TN}{TN + FP}$$



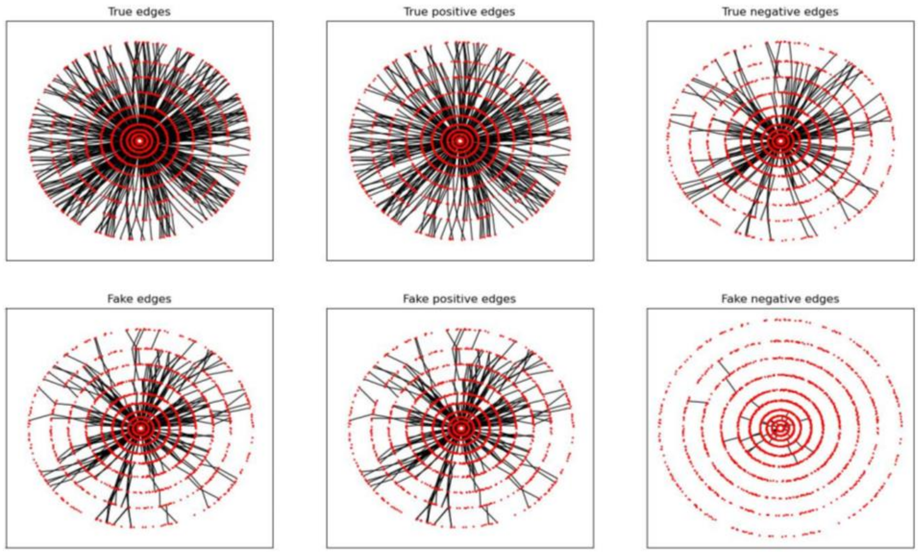
$$recall = \frac{TP}{TP + FN}$$

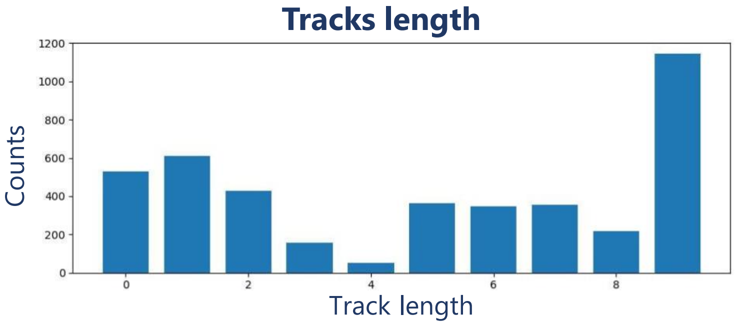


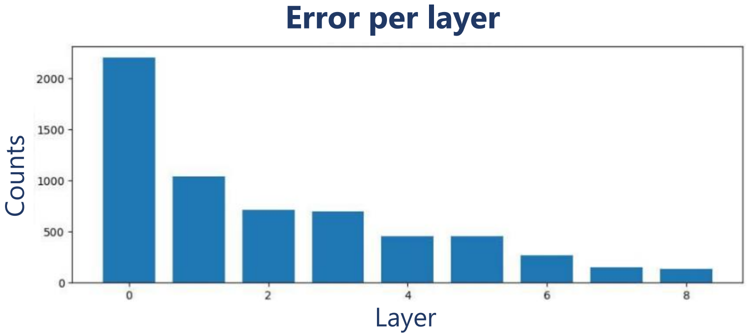
$$precision = \frac{TP}{TP + FP}$$

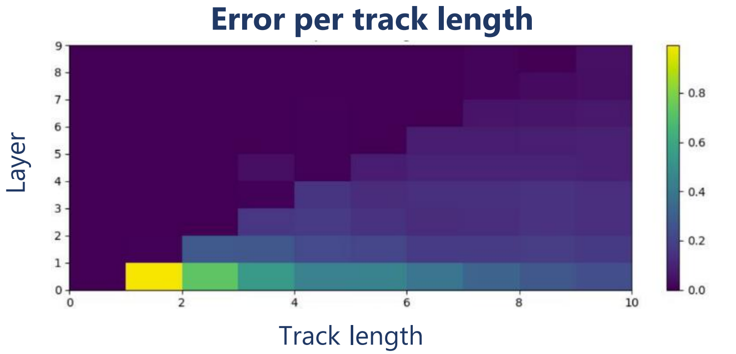
Training the QGNN

- In particular the majority of the errors occur in the innermost layers of the detector
- This is an expected behavior since because in layers 0-1 we find the vast majority of the combinatorial for the track segment candidates





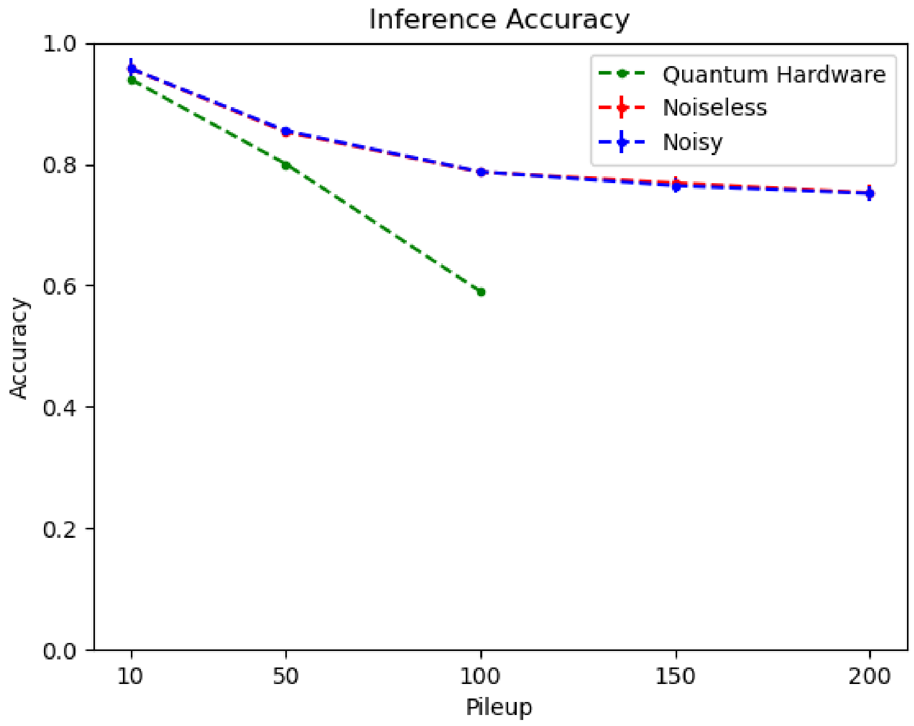




Inference

We tested the QGNN model on different backends

- ideal noiseless simulator
- Qiskit Aer noisy simulator
- IBM Quantum hardware (IBM_Osaka)



- There is no significant difference between the results for noiseless and noisy simulated values, the two curves are essentially overlapped
- Test set is reduced for inference on IBM Quantum Hardware due to limitations in QPU time and resources availability

our repo

Conclusion and prospects

- We have successfully implemented a QGNN model and we **trained** and performed **inferences** on **simulators** and **real quantum hardware**

- Our implementation can be **trained in reasonable times**, which is an important starting point for future studies

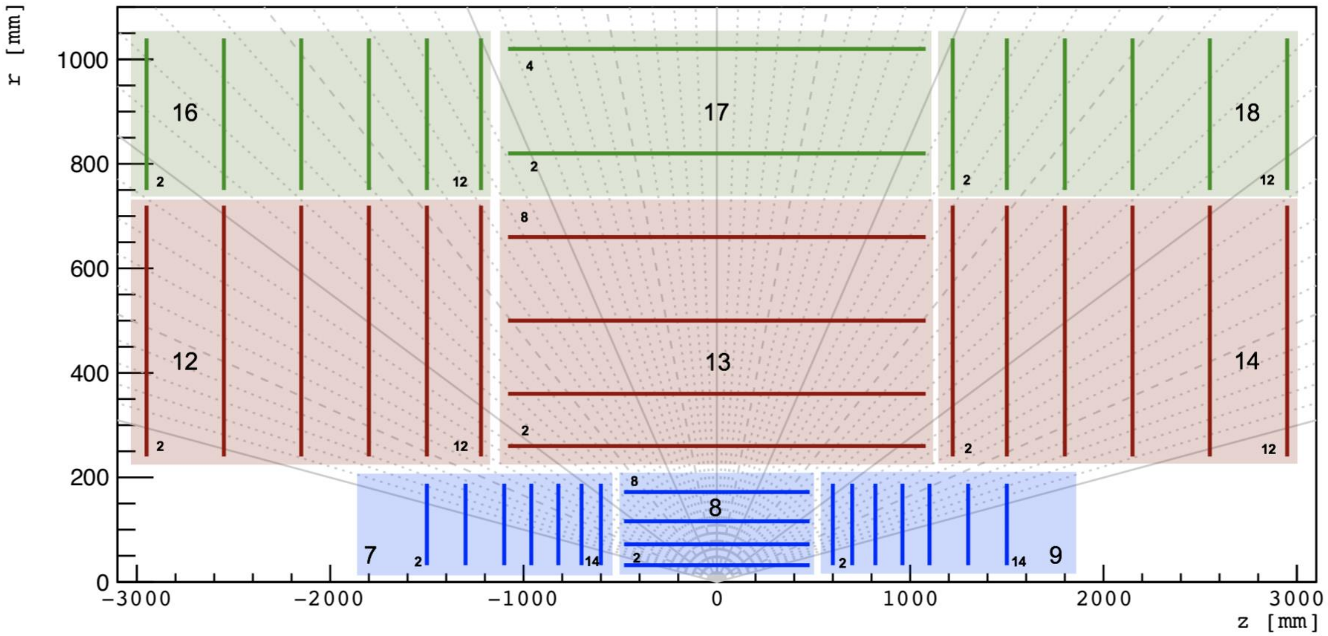
- Tracking is a complex problem, and definitely **not a low hanging fruit application for quantum machine learning**

- It would be very interesting to experiment with other **(quantum) message passing algorithms**

Thank you for your attention

Backup – detector and dataset

The dataset we use comes from the TrackML Kaggle challenge [2]



- only the barrel region (8,13-17) is considered
- selection:

```
pt_min: 1. # GeV
phi_slope_max: 0.0006
z0_max: 100
n_phi_sections: 1
n_eta_sections: 1
eta_range: [-5, 5]
```


[2] https://www.kaggle.com/competitions/trackml-particle-identification

Backup – Input Graphs

Pileup 200

Graph with 5653 hits, 8837 edges, 53% true

Pileup 150

Graph with 4223 hits, 5630 edges, 58% true

Pileup 100

Graph with 2728 hits, 3117 edges, 71% true

Pileup 50

Graph with 1512 hits, 1553 edges, 83% true

Pileup 10

Graph with 291 hits, 240 edges, 98% true