

Quantum Machine Learning for Track Reconstruction

Laura Cappelli

Matteo Argenton, Concezio Bozzi, Enrico Calore, Sebastiano Fabio Schifano

CHEP 2024

Kraków, 21 October 2024



**Università
degli Studi
di Ferrara**



Finanziato
dall'Unione europea
NextGenerationEU



**Ministero
dell'Università
e della Ricerca**



Italiadomani
PIANO NAZIONALE
DI RIPRESA E RESILIENZA

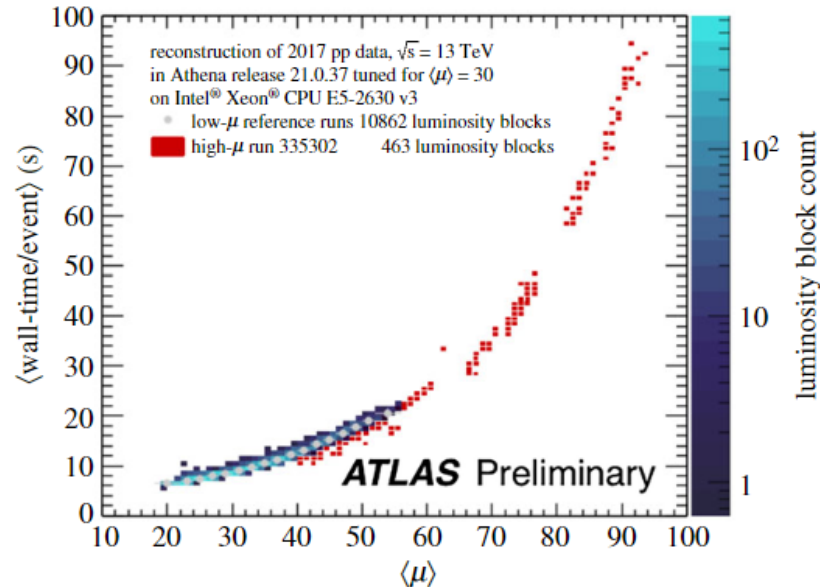
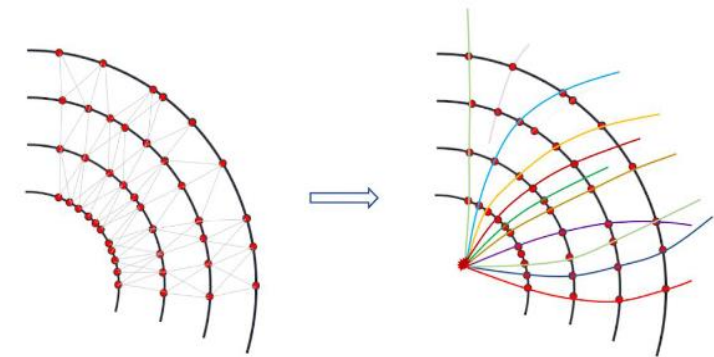


ICSC
Centro Nazionale di Ricerca in HPC,
Big Data and Quantum Computing

Track reconstruction with Graph Neural Networks

Scientific motivation

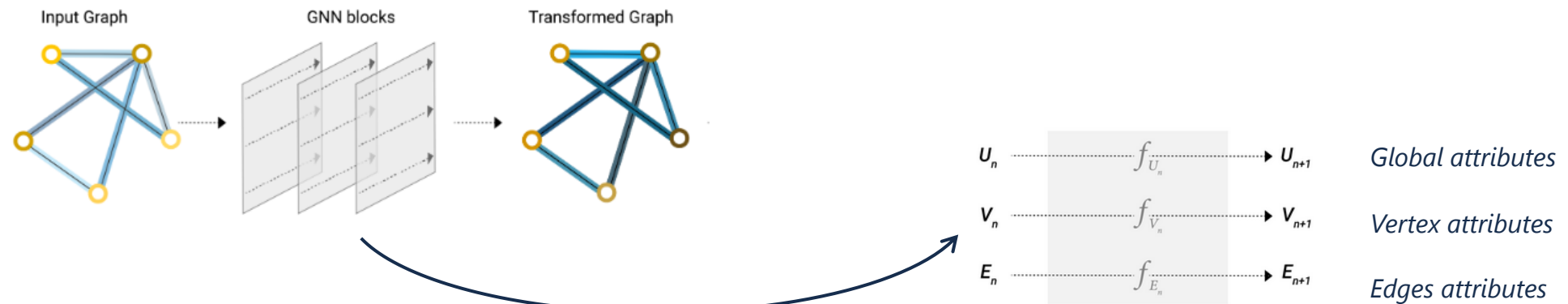
- The goal of **track reconstruction** is to assign a label to all the hits detected in an event



- With the High Luminosity LHC upgrade, the luminosity will increase
 - The events will become more and more complex
 - The wall time required for event reconstruction increases exponentially with luminosity
 - A **speedup in track reconstruction** is mandatory

Graph Neural Networks

- A possible research direction is using Machine Learning
 - A **GNN** is an optimizable transformation on all attributes of the graph (nodes, edges, global context) that preserves graph symmetries
 - Global approach in contrast with the classical local approach



- Several groups are testing this approach (e.g. [EXATrkX](#) [1] collaboration, Atlas ITK [2], LHCb [3] ...)

[1] X. Ju, D. Murnane, Calafiura P. et al. Performance of a geometric deep learning pipeline for HL-LHC particle tracking. *Eur. Phys. J. C* 81, 876 (2021). <https://doi.org/10.1140/epjc/s10052-021-09675-8>

[2] S. Caillou, P. Calafiura, et al.. ATLAS ITk Track Reconstruction with a GNN-based pipeline. *Connecting The Dots*, May 2022, Princeton, United States

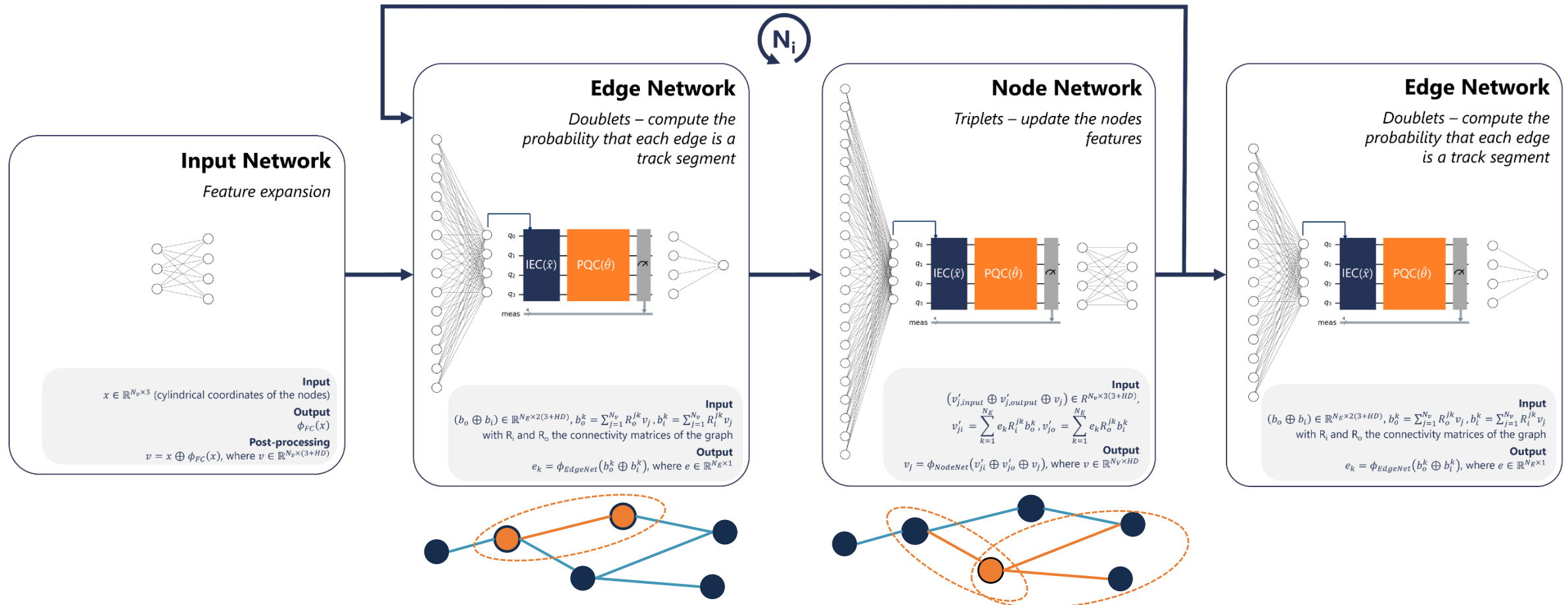
[3] A. Correia, et al. Graph Neural Network-Based Track Finding in the LHCb Vertex Detector, *arXiv:2407.12119*



Could the GNN
benefit from
Quantum Computing?

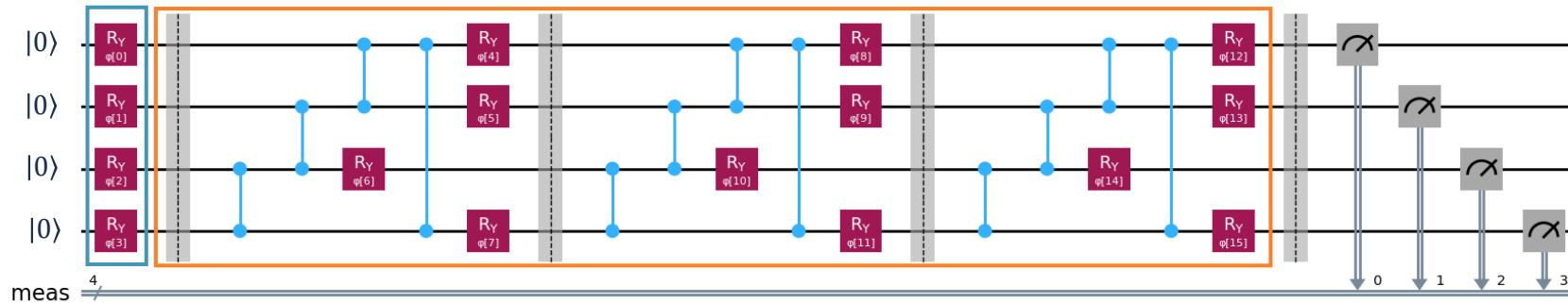
Quantum computing and GNN

C. Tüysüz in 2021 proposed a **hybrid approach** [4] to study a possible quantum advantage

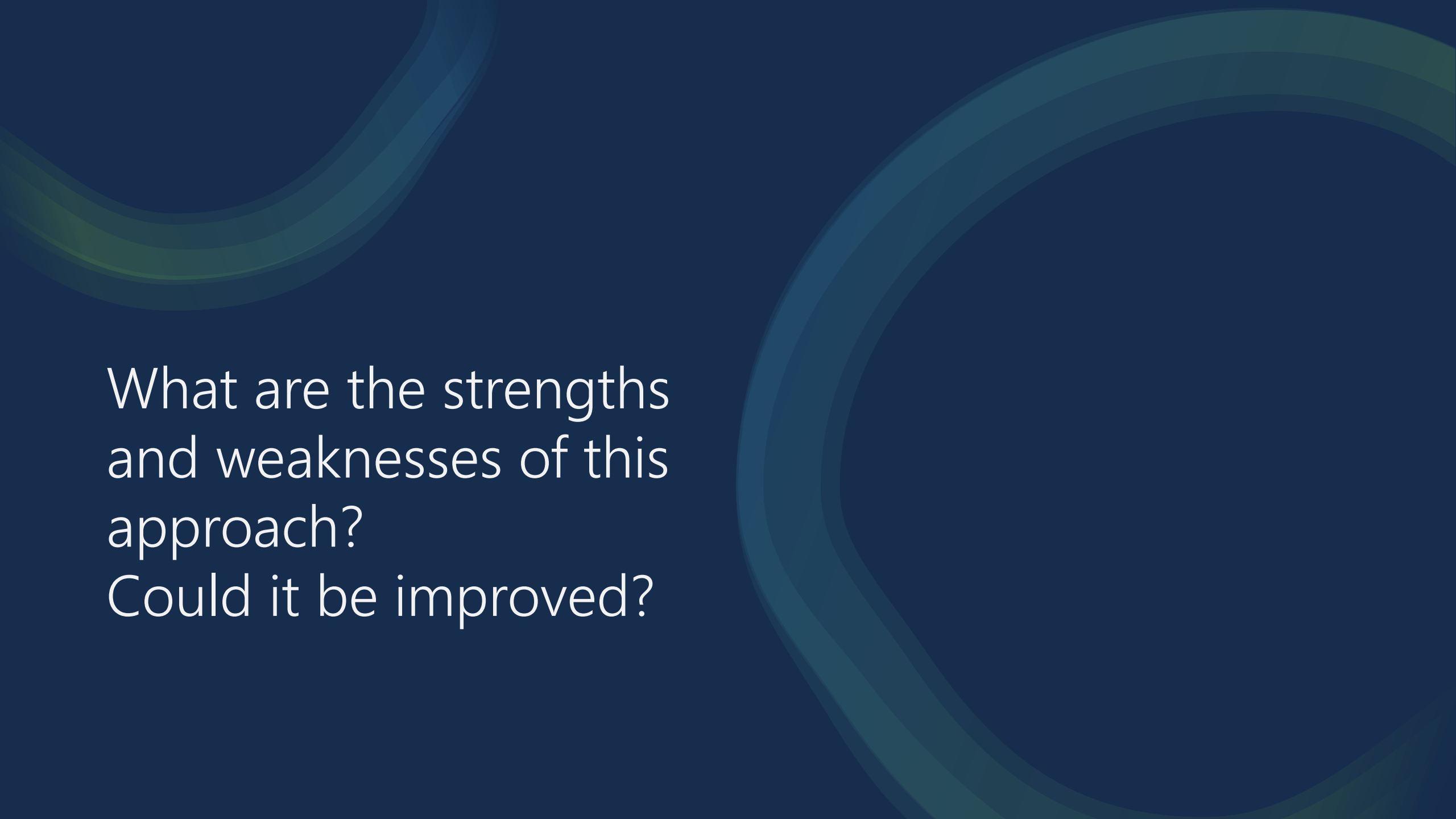


[4] Tüysüz C. et al. Hybrid quantum classical graph neural networks for particle track reconstruction. Quantum Mach. Intell. 3, 29 (2021). <https://doi.org/10.1007/s42484-021-00055-9>

The quantum circuit



- The quantum layer consists of:
 - An **Information Encoding Circuit** (IEC)
 - stores classical data into quantum states using angle encoding
 - A **Parametrized Quantum Circuit** (PQC)
 - rotates the input states in the Hilbert space depending on the angle parameters of the gates
 - generates entanglement between the qubits
 - Measurement of the final state
- The PQC parameters are trained to minimize the global loss function



What are the strengths
and weaknesses of this
approach?
Could it be improved?



With a complete characterization of QGNN for particle tracking it is possible to study and implement improvements

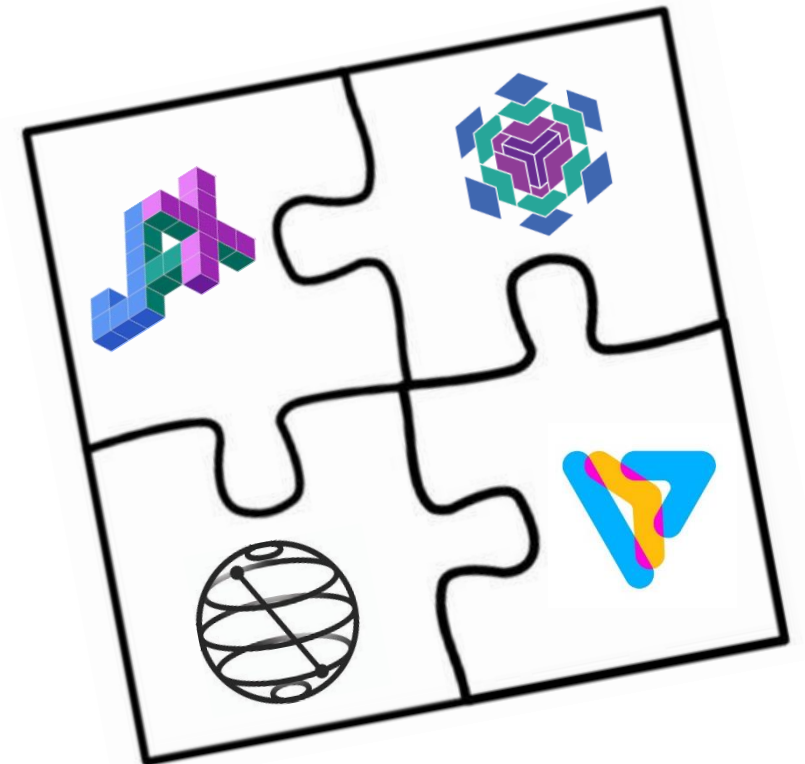


- **Implement** the network
- Find an appropriate **dataset** to work with
- Study the model for **increasing pileup values**
 - Testing the model on noiseless and noisy **simulator**
 - Running on real IBM **quantum hardware***

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

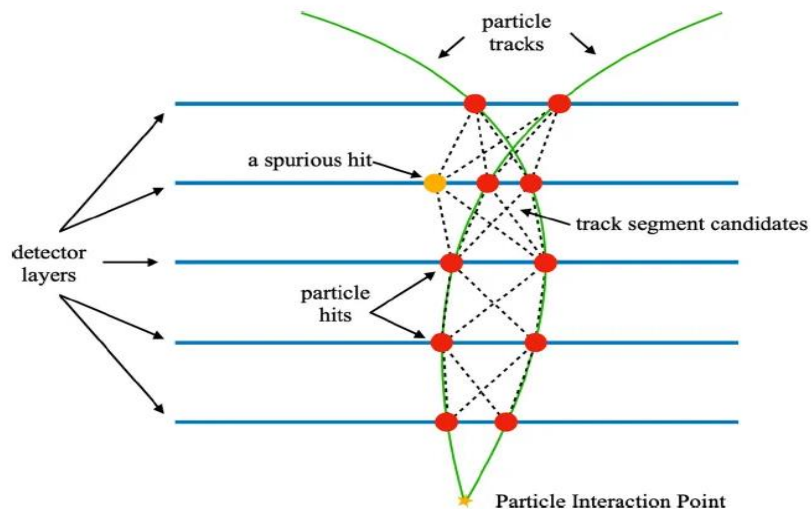
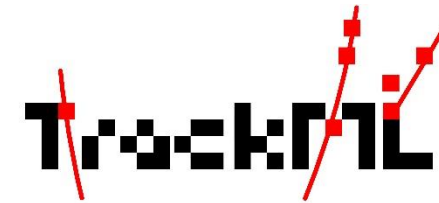
QGNN implementation

- We have implemented different versions of the QGNN with different frameworks:
 - *Tensorflow-quantum* + *Cirq* (from Tüysüz's work)
 - abandoned cause software deprecation
 - *Torch* + *Qiskit*
 - not usable due to very long training time (few days instead of few hours)
 - *Jax* + *PennyLane* + *Qiskit*
 - Data is stored in *Jax* format
 - The Neural Network is defined in *Flax*
 - Quantum circuits are implemented in *PennyLane*
 - The backend for the training is the IBM *Qiskit-aer* simulator
 - The quantum hardware is *IBM-Osaka*



Dataset and preprocessing

- Goal of preprocessing:
 - select events with pileup 10, 50, 100, 150 and 200
 - prepare the data to feed the model
- We use the [TrackML Challenge dataset](#)
 - Collection of thousands of simulated events with average pileup 200
 - Each event is a set of hits, so we need to build the associated graph structure



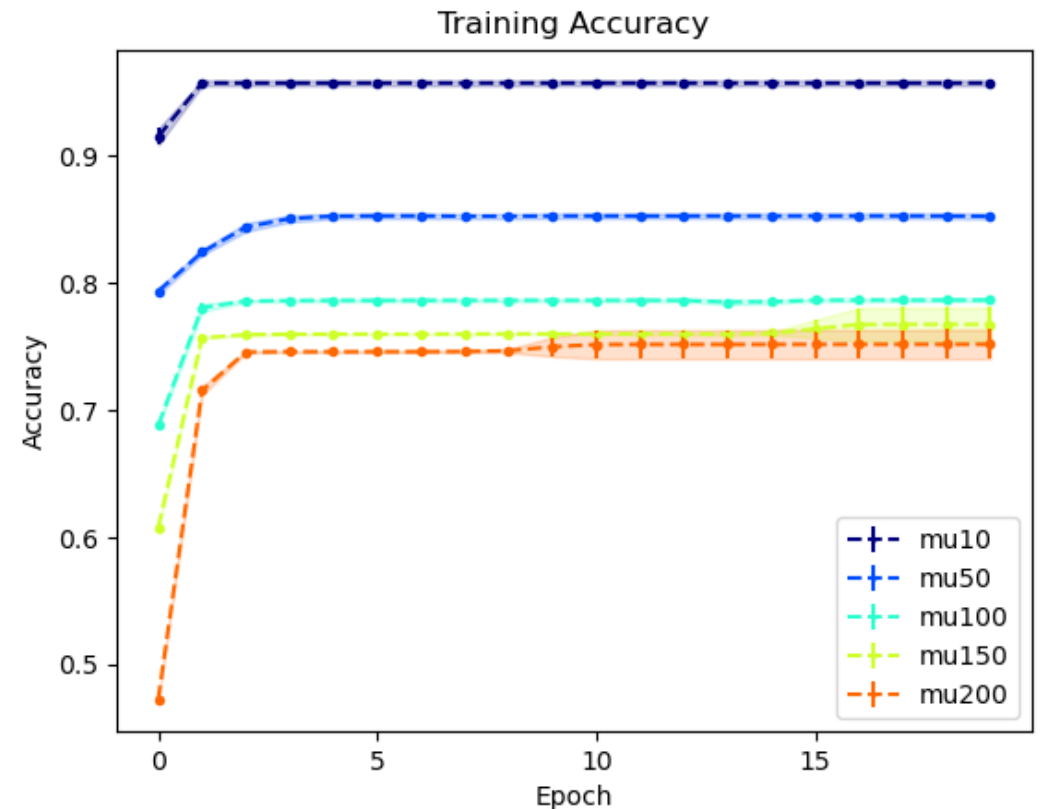
- An event is coded as a graph where:
 - **Nodes** are hits in a detector layer
 - **Edges** are track segments
 - Connections between hits in adjacent layers can be seen as candidate edges
- The network should learn to recognize true and fake edges

Training the QGNN

- We have trained the network on a local simulator to perform scalability tests
 - 35 training graphs and 10 validation graphs of pileup 10, 50, 100, 150 and 200
- Accuracy is higher with lower pileup

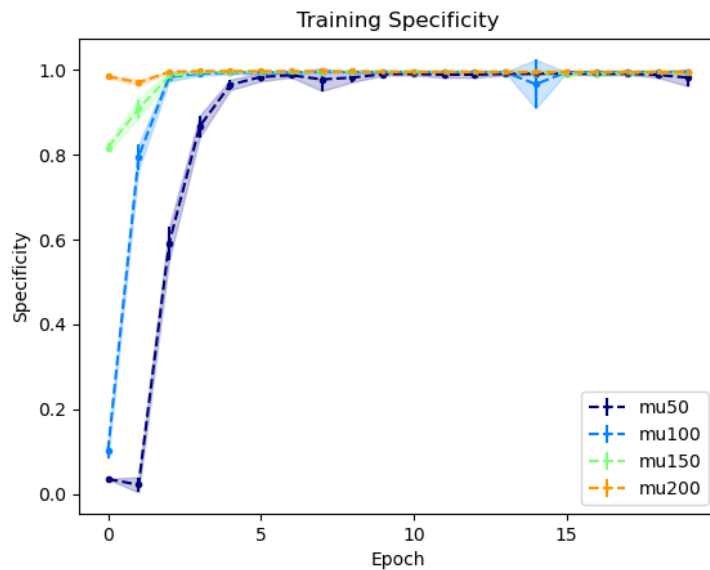
$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

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

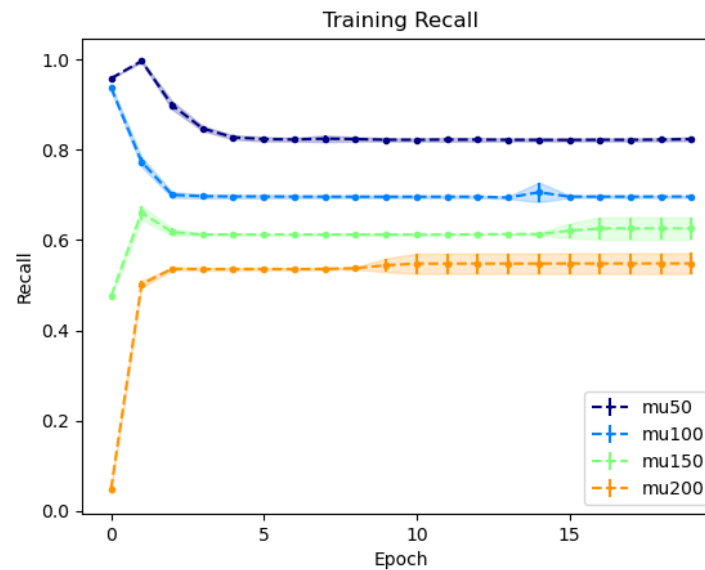


Training the QGNN

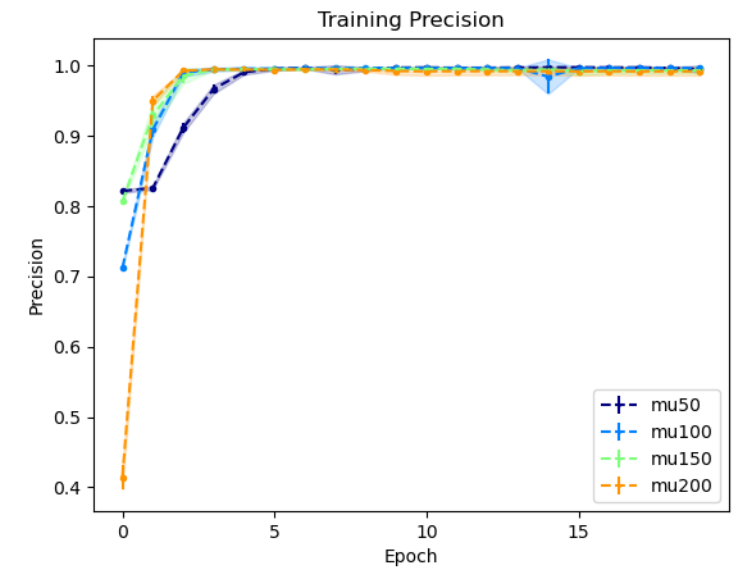
- Other metrics show that the QGNN can correctly recognize fake edges, but struggles with true edge classification



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



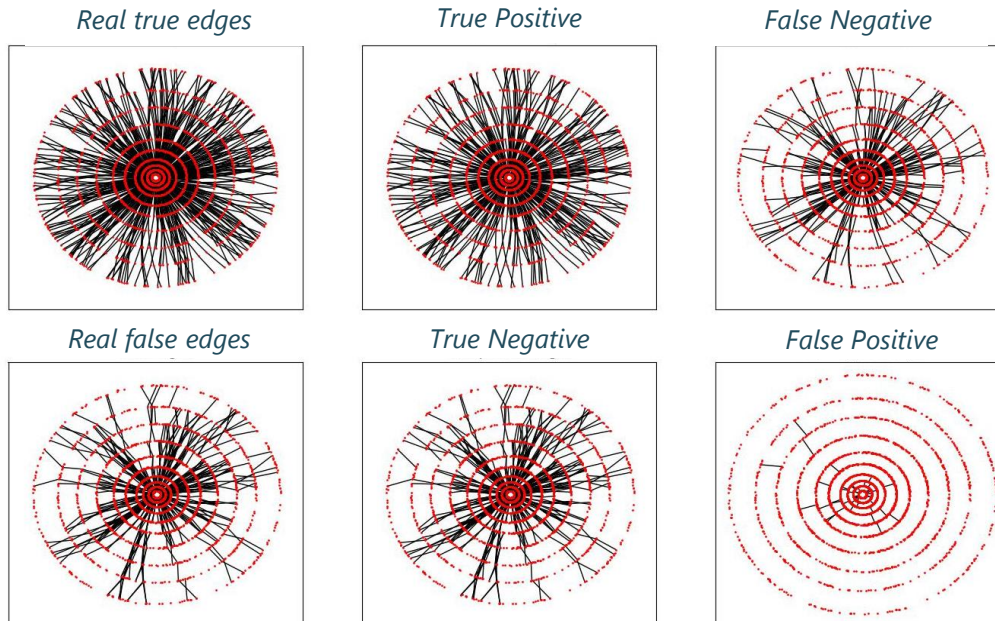
$$recall = \frac{TP}{TP + FN}$$



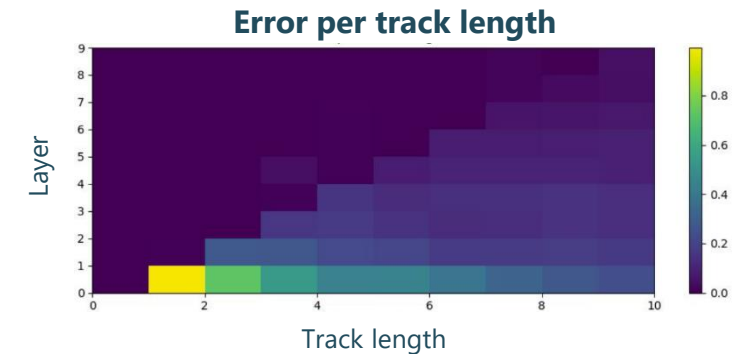
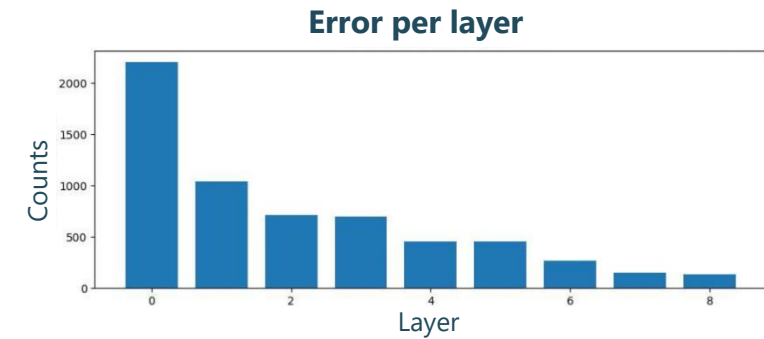
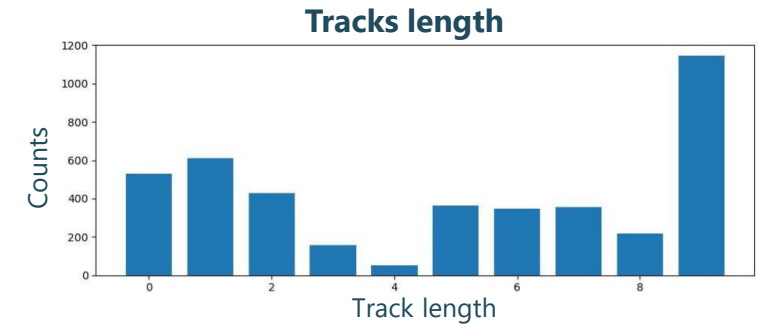
$$precision = \frac{TP}{TP + FP}$$

Where the network goes wrong

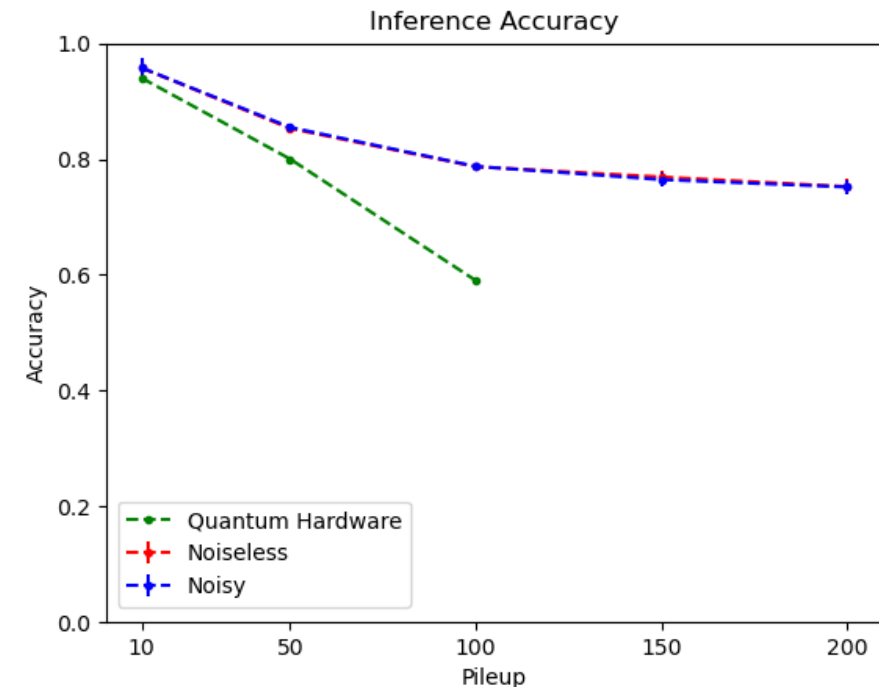
- Most of the errors occur in the innermost layers of the detector
 - In layers 0-1 we find the vast majority of the combinatorial for the track segment candidates



A visual confusion matrix of an event



- We have run the tests on 10 graphs with different backends:
 - **Noiseless simulators** by Qiskit and PennyLane
 - **Noisy Qiskit Aer** simulator
 - **IBM's quantum hardware*** (IBM_Osaka)
- What we can observe:
 - There is no significant difference between the results for **noiseless** and **noisy** simulated values, the two curves are overlapping
 - **Quantum hardware** accuracy decreases quickly



* Test set reduced due to issues in QPU time and resources availability



Prospects and conclusion

Critical points and prospects

	Critical points	Prospects
Dataset	The dataset is too complex for the QGNN architecture	<ul style="list-style-type: none">• Quantum: simplify and use toy models• HEP: use different preprocessing filters
Classical GNN	The model is simplified wrt the state of the art	Align with ExaTrkX or other GNN architectures
Quantum concepts	<ul style="list-style-type: none">• In the NISQ era, iterations are a relevant bottleneck• Computing time on simulators increases exponentially with the number of qubits	Focus more on superposition and entanglement (e.g. do not use circuits that manage information edge by edge)

Conclusion

- We have successfully implemented a QGNN model
 - We **trained** and performed **inferences** on **simulators** and **real quantum hardware**
 - The training time on simulators **is reasonable**, which is mandatory for future studies
- Using quantum hardware for this type of problems is still complicated due to QPU time and resource availability
- Tracking is not a **low-hanging fruit** application for quantum machine learning
 - We are working on different approaches to tackle this challenge

our repo is on INFN's gitlab!

<https://baltig.infn.it/quantum-fe/qgnn-tracking>



Thank you for
your attention!