Particle Track Reconstruction with Quantum Algorithms

Cenk Tüysüz[1,2,*], *Federico* Carminati[3], *Bilge* Demirköz[1], *Daniel* Dobos[4,6], *Fabio* Fracas[3], *Kristiane* Novotny[4], *Karolos* Potamianos[4,5], *Sofia* Vallecorsa[3], and *Jean-Roch* Vlimant[7]

[1]Middle East Technical University, Ankara, Turkey
[2]STB Research, Ankara, Turkey
[3]CERN, Geneva, Switzerland
[4]gluoNNet, Geneva, Switzerland
[5]DESY, Hamburg, Germany
[6]Lancaster University, Lancaster, UK
[7]California Institute of Technology, Pasadena, California, USA

Abstract. Accurate determination of particle track reconstruction parameters will be a major challenge for the High Luminosity Large Hadron Collider (HL-LHC) experiments. The expected increase in the number of simultaneous collisions at the HL-LHC and the resulting high detector occupancy will make track reconstruction algorithms extremely demanding in terms of time and computing resources. The increase in number of hits will increase the complexity of track reconstruction algorithms. In addition, the ambiguity in assigning hits to particle tracks will be increased due to the finite resolution of the detector and the physical "closeness" of the hits. Thus, the reconstruction of charged particle tracks will be a major challenge to the correct interpretation of the HL-LHC data. Most methods currently in use are based on Kalman filters which are shown to be robust and to provide good physics performance. However, they are expected to scale worse than quadratically. Designing an algorithm capable of reducing the combinatorial background at the hit level, would provide a much "cleaner" initial seed to the Kalman filter, strongly reducing the total processing time. One of the salient features of Quantum Computers is the ability to evaluate a very large number of states simultaneously, making them an ideal instrument for searches in a large parameter space. In fact, different R&D initiatives are exploring how Quantum Tracking Algorithms could leverage such capabilities. In this paper, we present our work on the implementation of a quantum-based track finding algorithm aimed at reducing combinatorial background during the initial seeding stage. We use the publicly available dataset designed for the kaggle TrackML challenge.

1 Introduction

Latest developments in Quantum Computing, significantly reduced the time needed to resolve certain problems"[1]. This resulted in a search for new methods to boost current algorithms whose computational complexity depend on the size of the dataset worse than polynomial.

The upcoming upgrade of Large Hadron Collider (LHC) at CERN to High Luminosity (HL-LHC) will increase the number of collisions. The HL-LHC upgrade will bring many

*e-mail: cenk.tuysuz@cern.ch

challenges. Particle track reconstruction is one of the challenges [2]. Current algorithms have trouble scaling up to higher collision rates. Therefore, researchers are trying new methods to tackle the problem such as the use of Graph Neural Networks[3] and Quantum Computing[4–6].

When a particle passes through a tracking detector layer, a signal called a "hit" is generated. The dataset contains precise locations of these hits and their particle identifications as labels. The challenge is to associate hits that are belonging to the same initial particle/track.

The HepTrkX team proposed a Graph Neural Network implementation for particle track reconstruction that uses the kaggle TrackML challenge dataset[3, 7]. The simulated dataset and the challenge was created by CERN scientists to invite machine learning experts to come up with novel methods to track reconstruction. Even though the Kaggle competition is concluded, the dataset is still being actively used by many researchers in the field of track reconstruction to benchmark their results.

The speed-up provided by Quantum Algorithms may play an important role in the future of track reconstruction in particle physics experiments. In this work, we present an exploratory look at the HepTrkX[8] project from a Quantum Computing perspective to evaluate the capabilities of Quantum Computing along with Deep Learning algorithms [3].

2 The Dataset and Classical Approach

The TrackML dataset consists of simulated measurements of many detector layers. The detector layers are arranged using a model layout that is common to most LHC experiments. In this layout there are several detectors that span a cylindrical geometry. All the detectors are rotated such that their center sees the collision center. In this geometry, particle beams collide on the z axis around $z = 0$ which is the center of collisions and also the center of the detector. The layout of the detectors can be seen in Figure 1.

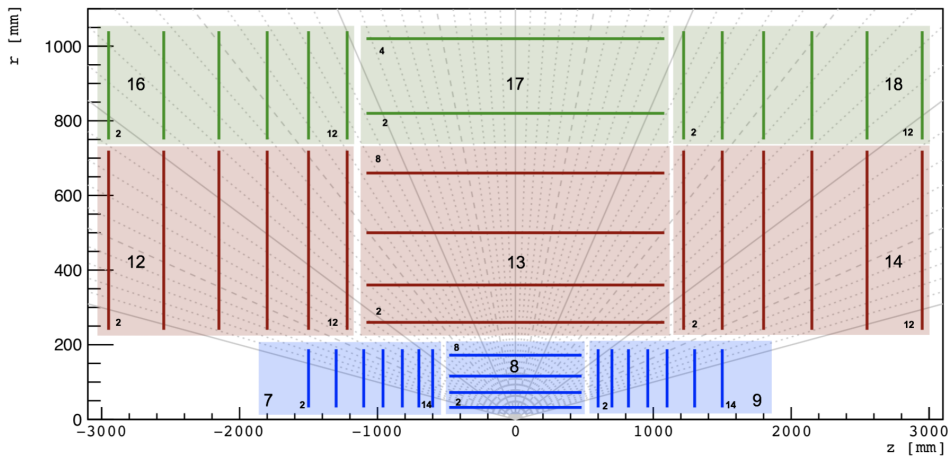

Figure 1. TrackML Detector Layout [7].

The complex layout of detectors allow better precision. However, they also increase the computational complexity as the particles might pass through both horizontal (barrel) and vertical (endcap) layers of detectors. For simplicity, the model only uses the barrel region of the detector. Therefore, the dataset contains only the layers in regions 8, 13 and 17.

The work presented in this paper uses the same preprocessing steps as those made by the HepTrkX team. This step is used to create an initial graph and labels from the TrackML dataset which contain particle momentum and spatial coordinates of detector measurements which are called "hits" and particle identification numbers.

The initial graph is created by connecting all logically possible combinations of hits and then applying loose selection criteria in order to decrease connectivity of the initial graph. This selection prevent connections between far detectors and acute angles which are physically not possible. The cuts used are given in Table 1.

Table 1. Cuts applied to TrackML dataset for preprocessing.

$\|p_T\|$	$> 1GeV$
$\Delta\phi$	< 0.0006
z_0	$< 100mm$
η	$[-5, 5]$

The coordinate definitions are as follows. ϕ is the angle along the transverse plane (xy plane) and $|p_T|$ is the magnitude of momentum along the same plane. η is the psuedorapidity which is a parameter widely used in particle physics as a measure of the azimuthal angle with respect to the beam axis.

The graphs created using the TrackML dataset are further divided into 8 in ϕ direction and into 2 in z direction to reduce the size of the graphs for a single event. A single event contains $\sim 8k$ hits requiring tremendous amounts of memory. By dividing a graph in its symmetry axes to 16, the same operations can be applied with less memory. This becomes handy when using limited memory systems to train the model.

As a first step, 100 events from the TrackML dataset are used to create 1600 subgraphs using the selection defined in Table 1. The reprocessed dataset is created using 1.0% of the TrackML dataset. The size of the dataset is kept intentionally small to reduce simulation times in order to speed up prototyping. Figure 2 shows an initial subgraph after preprocessing. Histograms showing the distribution of hits from the preprocessed data in cylindrical coordinates are given in Figure 3.

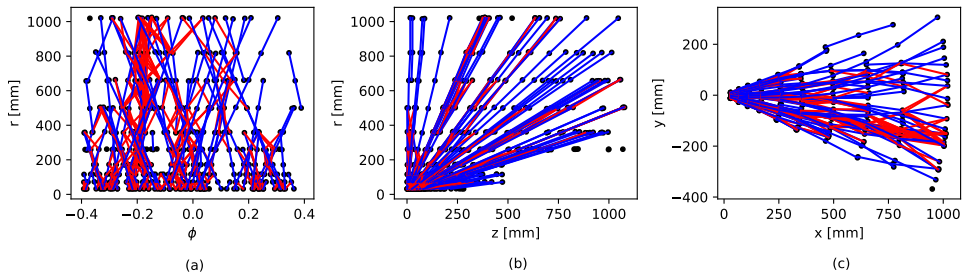

Figure 2. 1 of 16 subgraphs created from a single event. (a,b) are subgraphs in cylindrical coordinates and (c) is a subgraph in Cartesian coordinates. Red represents Ground Truth, while Blue shows Fake edges created using loose cuts.

The HepTrkX team proposed a GNN (Graph Neural Network) to perform segment classification. The model consists of 3 types of networks. The first one is an Input Network which takes a graph and maps it to higher dimensions. The second one is an Edge Network

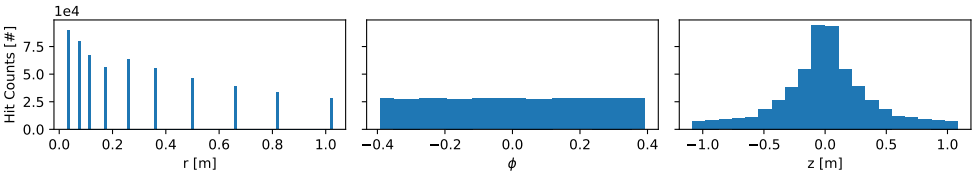

Figure 3. Histogram of hits crated using 1600 subgraphs in cylindrical coordinates. The distribution of r and z can be explained by referring to the geometry of the detector in Figure 1. The distribution in ϕ is uniform as expected since the geometry is symmetric along the transverse plane.

which takes node information from all edges in a graph and computes the edge information. The last network is a Node Network which computes hidden node features by looking at neighbouring nodes of each node. Edge and Node Networks are applied recursively after the execution of the Input Network. The model which is tested using the same TrackML dataset showed excellent performance. The model scores are 99.5% purity, 98.7% efficiency, and overall accuracy of 99.5% with 0.5 threshold [3]. Readers can refer to [9] for definitions of metrics.

3 Quantum Circuits as Neural Networks

The increase in computational power in the last two decades allowed scientists to deploy efficient machine learning models. Neural Networks use the immense power of classical computing to represent hidden features in data using millions of artificial neurons. However, even with today's computational power some tasks still take a considerable amount of time as the complexity of the tasks increases.

Quantum computing allows using entanglement, enabling the introduction of correlations that are not classically available to the neural network models. Additionally, the models can be extended to higher dimensions much faster as the dimension of the Hilbert Space is 2^n, with n being the number of qubits.

Quantum circuits have been previously shown to perform binary classification. Many Quantum circuit models in literature were considered [10, 11]. In this work, hierarchical quantum circuits have been selected to replace Neural Network layers, due to their high accuracy and robustness against noise [10].

The constructed models are implemented using Tensorflow [9] and Pennylane [12]. Pennylane is an automatic differentiation tool for quantum circuits and is used to calculate the gradients [12]. Readers can also refer to [13] for a more theoretical view.

4 Quantum Computing Integration

Transforming a well-performing Graph Neural Network to a Quantum Computing structure requires many modifications. To go step by step, this work only replaces the Edge Network of HepTrkX and does not use the Input and Node Network for simplicity. Therefore, the network only takes spatial information of nodes and computes the probability of an edge being true or fake. The original HepTrkX GNN model and the model in this work can be seen in Figure 4.

The Tree Tensor Network (TTN) is chosen among the hierarchical quantum classifiers as the quantum circuit to replace the neural network layer. The first attempt is made with no

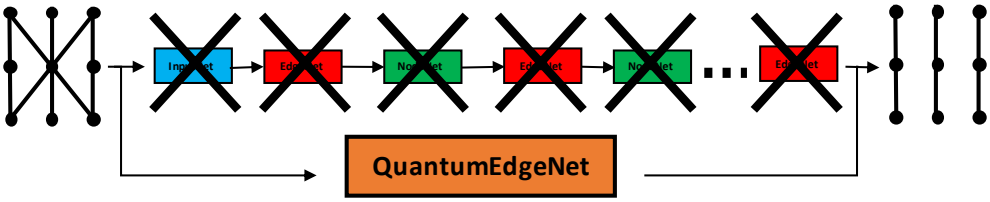

Figure 4. The HepTrkX GNN structure and the QuantumEdgeNet model used in this work.

hidden dimensions to test the capabilities of the structure. In later work, hidden dimensions will be included.

The Quantum Edge Network is applied to all edges one by one. All edges contain 2 node information which consists of 3 spatial coordinates each summing up to 6 data points for every edge. The coordinate information is encoded in 6 qubits, mapping each one to $0 - 2\pi$. This encoding method is chosen for its simplicity as other methods introduce additional computational complexities to use less qubits negating the gain in the number of qubits [14]. The data points are used as inputs to R_y rotation gates to encode the information as the angle between $|0\rangle$ and $|1\rangle$ states.

$$R_y(\theta)\,|0\rangle = \cos(\theta/2)\,|0\rangle + \sin(\theta/2)\,|1\rangle \tag{1}$$

After encoding the input in qubits, the TTN circuit is applied. The TTN circuit contains R_y and $CNOT$ gates. R_y gates starts with random parameters to be tuned later and R_y gates rotate the state according to the parameter's value. The $CNOT$ gate is used to introduce correlation between qubits so that their values are not independent. At the end of the circuit, there is a measurement. The input encoding layer and TTN circuit structure plotted using Qiskit [15] and matplotlib [16] can be seen in Figure 5.

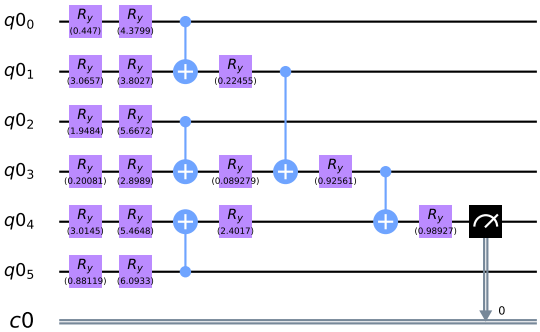

Figure 5. 6 Qubit Tree Tensor Network (TTN) representation of the Quantum Edge Network.

The quantum circuit is required to be run multiple times as the result of a single measurement is either 0 or 1. Therefore, the circuit is run 1000 times and the average of the outputs is used to determine the probability of an edge being true or fake. The number of circuit runs, in general called *shots*, should be selected carefully and the best value depends on a trade off between error rate and run time.

The network is trained over 2 epochs. The subgraphs are divided randomly into training and test sets with a 9:1 ratio. The model is trained using stochastic gradient descent and weighted binary cross entropy [9]. Training performance of the model can be seen in Figure 6.

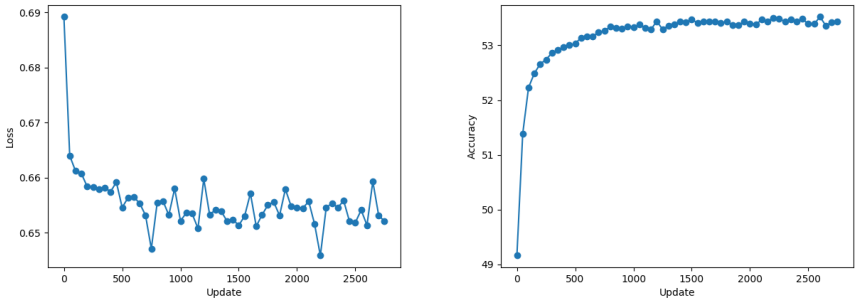

Figure 6. Training Loss (on the left) and Validation Accuracy (on the right) of the TTN in 2 full epochs. (1 epoch = 1440 updates)

The results in Figure 6 show that the model can learn features of the graph data. The accuracy achieved is considerably small, but this is mainly due to over simplification of the model. The network showcased here is a proof of principle prototype of a complete Quantum Graph Neural Network structure.

5 Future Work

The work presented here is the first step towards a Quantum Graph Neural Network that can classify tracks with high precision and accuracy. The final structure will include;

- The Node Network as another quantum circuits to learn features among neighbouring nodes.
- Recursive iterations of Node and Edge Networks to pass the information through nodes
- An input layer to include hidden layers, which do learn in a large Hilbert Space.

The up-to-date model and all source codes can be accessed through [17].

6 Conclusion

In this work, we show that it is possible to implement Quantum Graph Neural Network based approaches for the track reconstruction problem. Although, the current model is not complete, it shows promising preliminary results towards a quantum circuit based algorithm that can be run using a universal quantum computer. This work only uses simulations and does not consider practical applicability at the moment.

7 Acknowledgments

Part of this work was conducted at "*iBanks*", the AI GPU cluster at Caltech. We acknowledge NVIDIA, SuperMicro and the Kavli Foundation for their support of "*iBanks*". This work was partially supported by Turkish Atomic Energy Authority (TAEK) (Grant No: 2017TAEKCERN-A5.H6.F2.15). Cenk Tüysüz thanks Oral Okan and Egemen Sert from STB for their valuable discussions.

References

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J.C. Bardin, R. Barends, R. Biswas, S. Boixo, F.G.S.L. Brandao, D.A. Buell et al., Nature **574** (2019), arXiv:1910.11333

[2] G. Apollinari, O. Bruening, T. Nakamoto, L. Rossi, *High luminosity large hadron collider hl-lhc* (2017), arXiv:1705.08830

[3] S. Farrell, P. Calafiura, M. Mudigonda, Prabhat, D. Anderson, J.R. Vlimant, S. Zheng, J. Bendavid, M. Spiropulu, G. Cerati et al. (2018), arXiv:1810.06111

[4] I. Shapoval, P. Calafiura (2019), arXiv:1902.00498

[5] F. Bapst, W. Bhimji, P. Calafiura, H. Gray, W. Lavrijsen, L. Linder (2019), arXiv:1902.08324

[6] A. Zlokapa, A. Anand, J.R. Vlimant, J.M. Duarte, J. Job, D. Lidar, M. Spiropulu (2019), arXiv:1908.04475

[7] S. Amrouche, L. Basara, P. Calafiura, V. Estrade, S. Farrell, D.R. Ferreira, L. Finnie, N. Finnie, C. Germain, V.V. Gligorov et al. (2019), arXiv:1904.06778

[8] https://heptrkx.github.io/

[9] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin et al., *TensorFlow: Large-scale machine learning on heterogeneous systems* (2015), software available from tensorflow.org, https://www.tensorflow.org/

[10] E. Grant, M. Benedetti, S. Cao, A. Hallam, J. Lockhart, V. Stojevic, A.G. Green, S. Severini, npj Quantum Information **4**, 17 (2018)

[11] A.S. Bhatia, M.K. Saggi, A. Kumar, S. Jain, Neural Computation **31**, 1499 (2019), arXiv:1905.01426v1

[12] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, C. Blank, K. McKiernan, N. Killoran, pp. 1–12 (2018), arXiv:1811.04968

[13] E. Farhi, H. Neven (2018), arXiv:1802.06002

[14] S. Aaronson, Nat. Phys. **11**, 291 (2015)

[15] H. Abraham, I.Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, G. Alexandrowics, E. Arbel, A. Asfaw, C. Azaustre, AzizNgoueya, P. Barkoutsos et al., *Qiskit: An open-source framework for quantum computing* (2019)

[16] J.D. Hunter, Computing in Science & Engineering **9**, 90 (2007)

[17] https://github.com/cnktysz/HepTrkX-quantum/