Optimizing Quantum Graph Neural Networks for the Particle Tracking Problem

CERN Project Report

Start Date: 21/02/2022 End Date: 25/02/2022

Team Name:

The Superpositioned States of America

Abstract:

The CERN Quantum Technology Initiative declared five quantum computing and algorithms projects of interest to the HEP community. One of these projects is particle tracking reconstruction. Particle tracking reconstruction is a fundamental research aspect for particle collider experiments, which aim to validate the elementary particle theories through the observation of particle trajectories after collisions. With the upgrade of the Large Hadron Collider (LHC) to High Luminosity (HL-LHC) at CERN in the near future (circa 2027), the rate of collisions will be further ramped up, yielding many more detector hits and a larger scale of data. Therefore, being able to efficiently identify the particle trajectories from data of a large scale and increased complexity may be the key to unveil the nature of particle physics. With the development of quantum computers on its own growth curve, untangling the information from big data via quantum computing appears to be a promising solution. Our work focused on improving the existing state-of-the-art of Quantum Graph Neural Networks (QGNNs) to solve the particle tracking reconstruction problem. Specifically, we have have resolved the vanishing/exploding gradient problem during early stages of training, introduced new ansätze for the parameterized quantum circuits (PQCs) used in the Edge/Node networks, and tested these new ansätze on a modified dataset after preprocessing. After experimenting with a variety of different circuit architectures, we have constructed two high performing circuit ansätze: "`qc102_pqc(VanBuren)`" and "`qc103_pqc(Coolidge)`". Both circuits, coupled together with an improved QGNN outperform the "`qc10_pqc`" ansätze used by CERN in the original experiment on a smaller dataset.

1 Overview

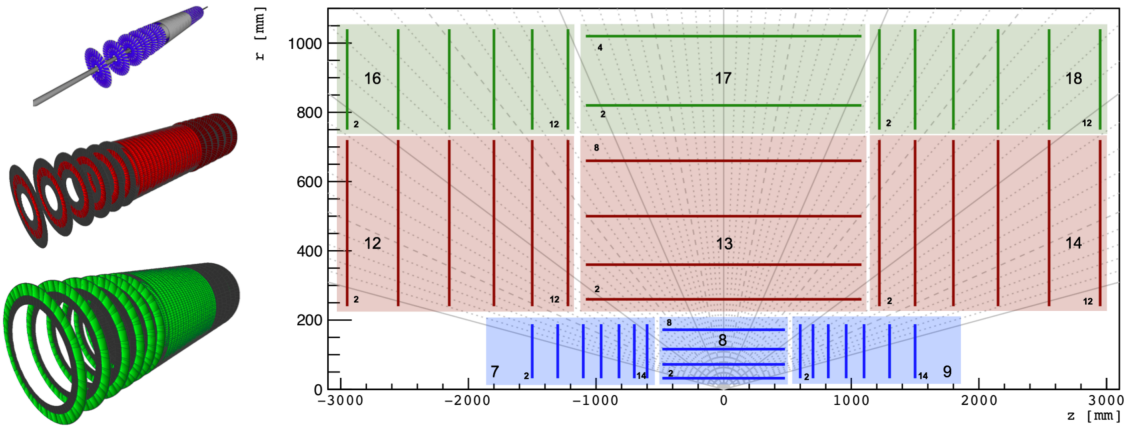

Figure 1: A scheme of the full detector layout for the virtual TrackML detector. Source: [1, 2]

1.1 Quantum graph neural networks for particle tracking

To understand the structure of subatomic particles and the laws of nature governing them, particle physicists use colliders as a research tool to accelerate particles to very high speeds and energies in order to cause collisions to happen. In each collision, particles are scattered in every direction. The detectors, also known as "trackers," that are spread around the device can measure particle properties, such as momentum, time, etc. When the particles pass through the detectors, "hits" are registered without loss of energy to the particle. The challenge comes in reconstructing the particle trajectories in 3D from the detector data. Particularly, the Large Hadron Collider (LHC) [3] at CERN is planned to be completely upgraded to the High Luminosity Large Hadron Collider (HL-LHC) by the end of 2027 [4]. The significantly rising rate of collisions coming from the HL-LHC will not only bring more high-energy physics to explore, but also subsequent challenges to reconstruct the particle tracking with much larger amounts of data. Accordingly, the search for more efficient algorithms to tackle particle tracking reconstruction becomes urgent and critical. CERN even hosted a Kaggle challenge named "TrackML" [6] in 2018, intending to encourage people of

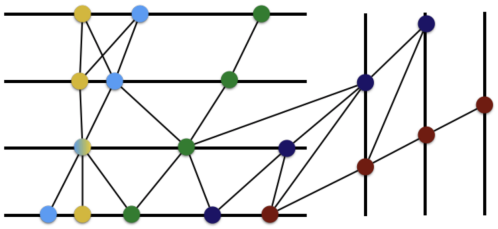

Figure 2: A graph representation of track hit data. Source: [5]

different academic/professional backgrounds to try and solve this particle tracking reconstruction problem. The TrackML Challenge dataset contains 10000 events to simulate the HL-LHC conditions [1]. Each event has $O(10^5)$ space-point hits, corresponding to $O(10^4)$ particles. With such an exponentially large amount of data, it is conceivable to seek out solutions that may exist from Maching Learning (ML) or Deep Learning (DL) approaches. The idea to implement Quantum Graph Neural Networks (QGNNs) for the particle tracking problem is as follows: Firstly, in the TrackML detector layout (Fig. 1), the geometry of the detected "hits" distributed in real space is layer-by-layer (Fig. 2). The hits can be the "nodes" while the traveling path between hits (from one layer to the next layer) can be the "edges" if we borrow the language from

Graph Neural Networks (GNNs). Therefore, applying GNNs to the measured particle data appears to be a promising way to reconstruct the particle trajectories [5]. Secondly, with the rapid development and growth of quantum computing (QC), a fertile area of research is Quantum Machine Learning (QML) where qubits can be used directly in a neural network. By replacing each node of the GNN structure as a qubit and constructing the corresponding quantum circuits, we can combine both DL and QC into the QGNN approach to attack the particle tracking reconstruction problem.

In prior work [2], the authors mentioned problems with training times and vanishing gradients. We decided during this short period of time that we would look into these issues.

1.2 Vanishing/Exploding Gradient Problem

A known issue of recurrent neural networks is the vanishing gradient problem and the quantum graph neural network used in this work is no exception. Training an RNN requires using back-propagation through time (BPTT), which means that you choose a number of time steps N, and you unroll your network so that it becomes a feed-forward neural network (FFNN) with N duplicates of the original network.

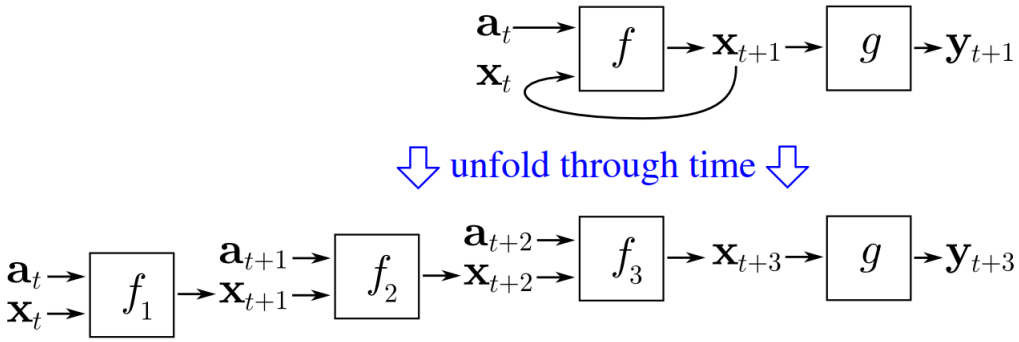

Figure 3: An unrolled recurrent neural network. Source: [7]

So training an RNN with BPTT is equivalent to unrolling the RNN into a deep FFNN and proceeding to calculate the gradients as you normally would with a FFNN (backpropagation). However, we soon face difficulties with this approach. It becomes unwieldy to compute the gradients of the early layers of the unrolled RNN as N scales. This is due to the nature of BPTT, where early layers' gradients are the products of all succeeding terms and thus are prone to vanishing or exploding gradients.

The quantum graph neural network proposed in existing literature uses a similar recurrent architecture where each layer/step of message passing can be thought of as a time step, and thus faces the vanishing gradient problem as the number of cycles of message passing increase. Our project aimed to provide a solution to the vanishing gradient problem faced by QGNNs. In a later section, we consider how we draw from the progress of RNNs in classical machine learning to introduce a novel solution to help with the training of QGNNs.

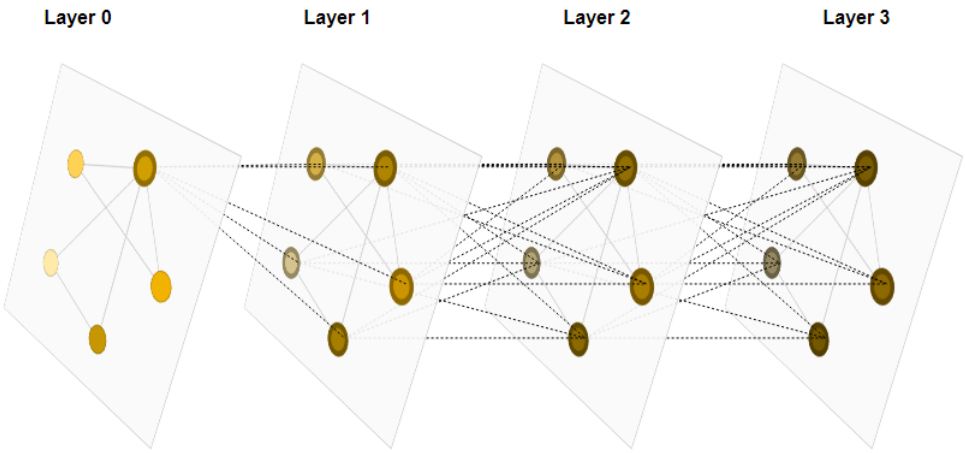

Figure 4: The information of the highlighted initial node propagates through the graph network with each layer of message passing (propagation). Source: [8]

2 Data Analysis & preprocessing

The raw data used in our work is from the TrackML dataset [1], which contains 10000 events. Each event refers to each launched collision happening at $z = 0$ (after the particle beams propagate along the z-axis), which involves roughly $8k$ hits. The dataset includes the spatial coordinates (and also velocities, momentums...etc.) of each particle from each hit. Our task is to convert these datasets to graph datasets for QGNN implementation.

In the original QGNN paper [2], the authors mention that training time was on the order of a week even with a reduced dataset (100 events out of 10000 events). For this project, we unfortunately didn't have a full week, so it was necessary to find a set of criteria to decrease the amount of data we processed. The idea was that the results we obtained would offer insight into how an *even smaller* dataset might extrapolate to the overall, larger dataset.

2.1 Analysis of the Original Data

The original (unpreprocessed) data as in Fig. 5 is hard to distinguish visually. The "true" paths and "fake" paths are across each other densely, prohibiting us to analyze well. The meanings of the physical quantities are explained as follows: $r = \sqrt{x^2 + y^2}$ is the transverse radius in the $(x - y)$ plane. $\phi \in [-\pi, \pi]$ is the azimuthal angle in the $(x-y)$ plane, with $\phi = 0$ denoting the x-axis. The pseudo-rapidity $\eta = \log(\tan(\theta/2))$ quantifies the similarity along the z-axis, with the polar angle $\theta \in [0, \pi]$ measured from the z-axis. For example, $\eta = \pm\infty$ is very close to $\pm z$ direction, while $\eta = 0$ is vertical to z-axis. Due to the cylindrical symmetry of the particle beams collisions along the z-axis, we can divide the space along the z-axis by two and the transverse space ($x - y$ plane) by N sections ($N = 8$ in all the path graphs in Sec. 2). In fact, the condensed paths in Fig. 5 are even just from one of the 16 equal divisions in the spatial space. So it was essential to preprocess the data.

2.2 Data preprocessing

Before we preprocessed our data, we decided to see how the preprocessed data would look like under certain selection constraints in the original paper [2]. Fig. 6 is one of the subfigures of their preprocessed data, which demonstrates more true edges than fake edges, thus making the model training data biased.

In the TrackML detector layout (Fig. 2), there are three kinds of barrel detectors expressed in different colors arranged layer-by-layer horizontally and vertically. In the original paper [2], they focus on the centralized 10 layers (volumns 8, 13 and 17 in Fig. 1). They even applied the following criteria to deduce

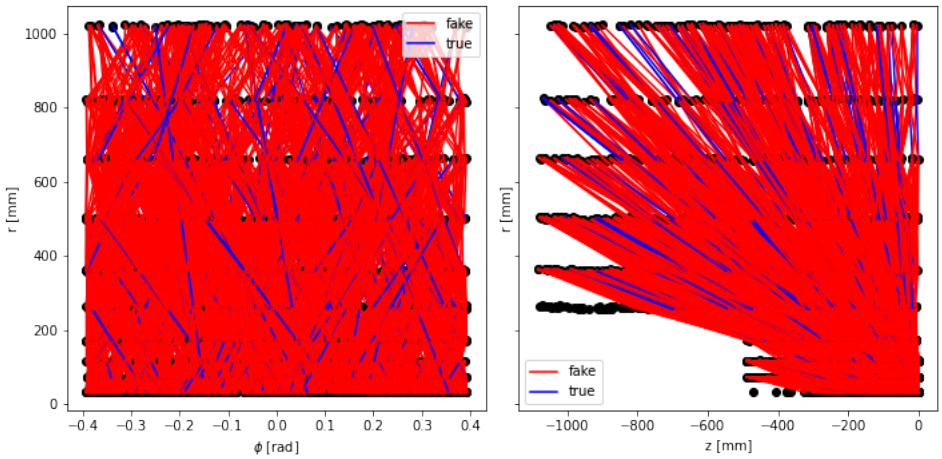

Figure 5: One of the subfigures of the unpreprocessed data from one event.

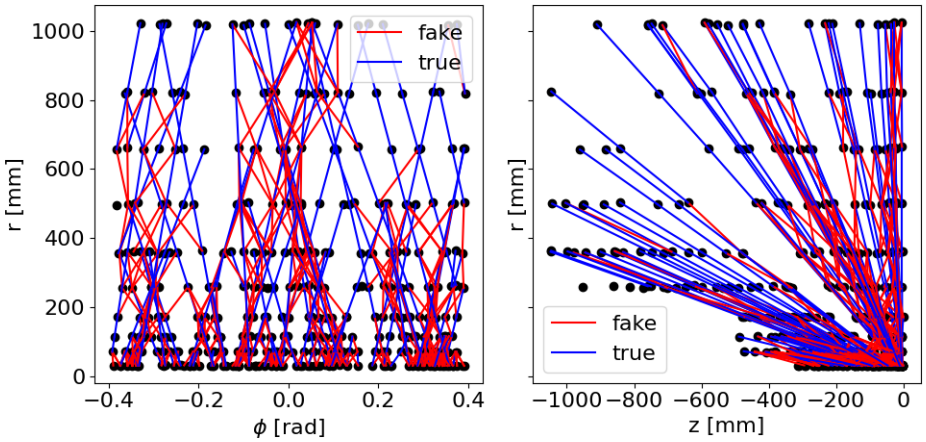

Figure 6: The preprocessed data from the original QGNN paper [2, 9]. One of 16 subgraphs created from a single event. Source: [9]

the amount of data and eliminate illogical edges in the graph, as the following table. Note that $|p_T|$ is the

$\lvert p_T \rvert$	> 1 GeV
$\Delta\phi$	< 0.0006
z_0	< 100 mm
η	$[-5, 5]$
ϕ	$[-\pi, \pi]$

magnitude of momentum along the radius direction and z_0 is the maximum cutoff of z-coordinate.

Following the criteria of the original QGNN paper [2], we intended to preprocess our data such that the particle trajectories were sparse and distinguishable, but producing more fake paths than true paths. In this way, we could generate datasets that would potentially be harder to train. We only kept the first 8 layers (volumes 8 and 13). The settings we use to generate the graphs like Fig. 7 were:

$\lvert p_T \rvert$	> 1 GeV
$\Delta\phi$	< 0.0006
z_0	< 700 mm
η	$[-3, 3]$
ϕ	$[-3\pi/4, 3\pi/4]$

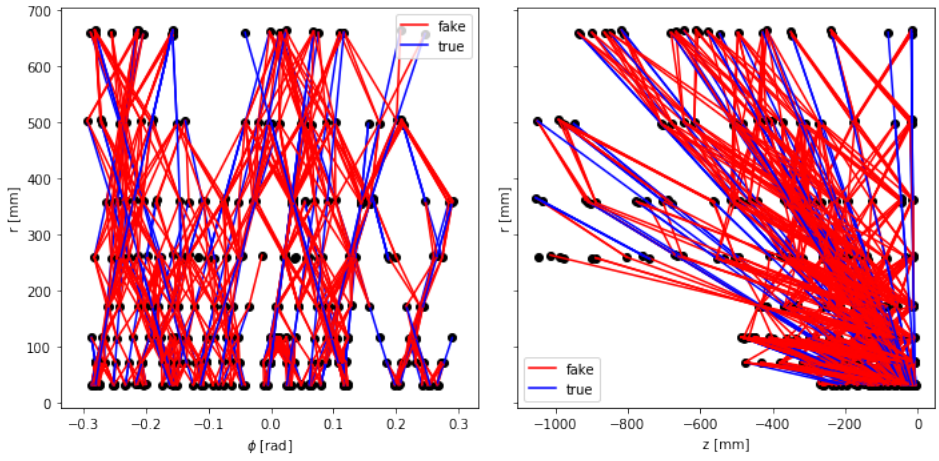

Figure 7: The information of the highlighted initial node propagates through the graph network with each layer of message passing (propagation). Source: [8]

3 Solutions to the Vanishing Gradient Problem

3.1 Activation function improvements

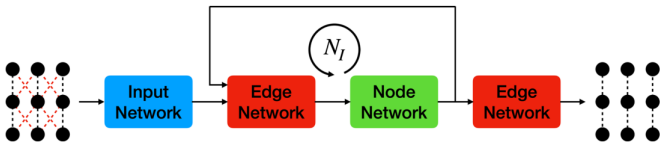

Figure 8: The Quantum Graph Neural Network model used in this work. Source: [2]

Parsing through the existing QGNN architecture, our team noticed the prevalence of sigmoid activation functions throughout all layers of the network. Naturally, we wondered why. After speaking with the lead author of the original paper, we realized that it was simply used to meet the parameter constraints of quantum gates, $[-\pi, \pi]$. Further, due to the 2π periodicity of these quantum gates, the quantum parameters must be constrained within $[0, \pi]$. Such a constraint is effectively achieved by applying a π factor to the output of each sigmoid. So we understood that the motivation behind the use of sigmoid activation functions throughout the network was to adhere to the $[0, 1]$ bounds.

However, it is well understood in classical machine learning literature that sigmoid activations in between layers can often lead to vanishing gradients.

$Sigmoid(x) = \frac{1}{1+e^{-x}}$

As you can see, the gradient for the sigmoid function will saturate and when using the chain rule, it will shrink. By contrast, the derivative for the rectified linear unit (ReLU) is always 1 or 0. The argument for using ReLU activation functions within the hidden layers is made even stronger when considering that ReLU remains the most widely used activation function in classical graph neural networks.

So, motivated by the appeal of ReLU, our group probed the question of whether it is possible to adapt a ReLU function to the QGNN such that one can both benefit from the rich gradients that ReLU provides while conforming to the $[0, 1]$ output bounds that are required for all classical layers that feed into a parameterized quantum circuit. In this project, we proved the affirmative. We substituted all of the sigmoid activation functions except for the edge network outputs with ReLU activations that are paired with a rescaling layer which is used to re-scale all tensor values to be between $[0, 1]$.

By employing this technique, the QGNN was immune to vanishing gradients for up to 50 more layers in comparison to the original architecture based upon the sigmoid activation.

3.2 Circuit Ansätze Analysis

Our team consulted the lead author of the original QGNN paper [2] about resolving the vanishing/exploding gradient problem. After confirming that this was an issue, we utilized some recent work from Google [10] to diagnose the gradient variance in different circuit architectures. After running existing circuits through an adapted version of their work, we confirmed that less connected circuit architectures such as TTN and MPS are more robust to the vanishing gradient problem, which is known among QML researchers.

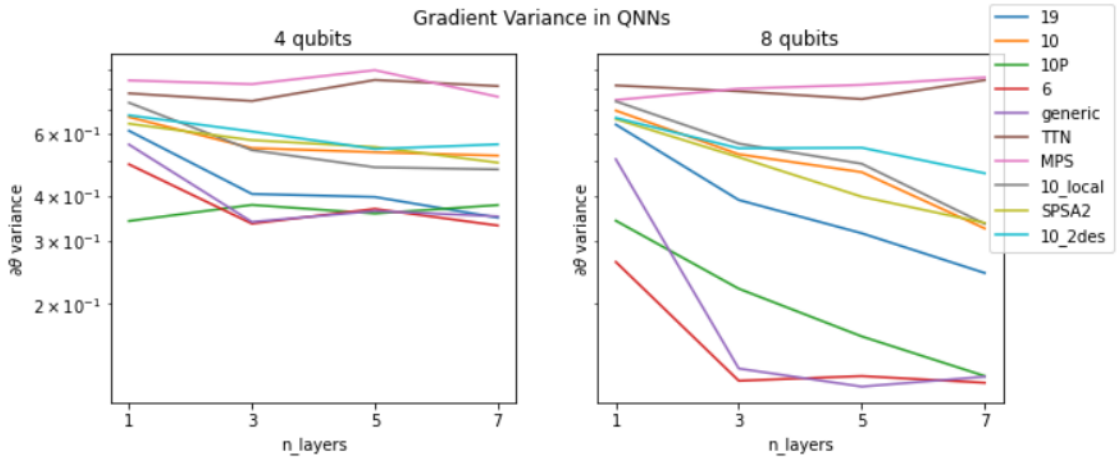

Figure 9: Prevalence of the Vanishing Gradient in the Various Circuit Ansätze. Source: [10]

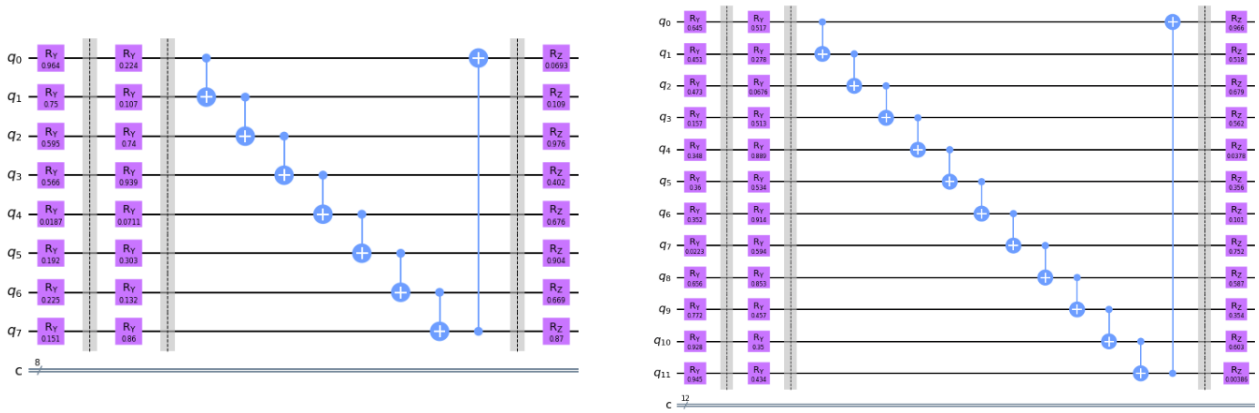

Figure 10: 102 Van Buren Ansatz on 8 qubits and 12 qubits before measurement.

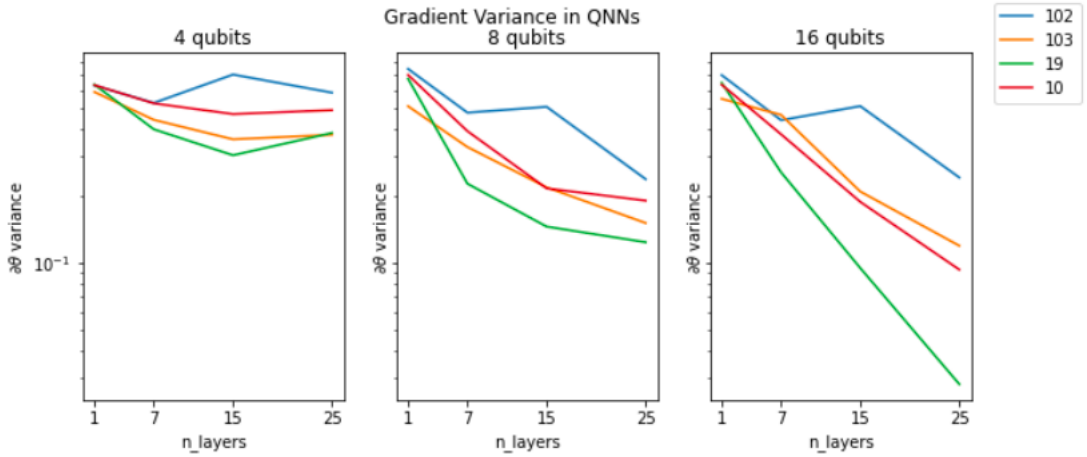

Figure 11: 102 (Van Buren) and 103 (Coolidge) vs 10PQC and 19PQC in terms of Vanishing Gradient Prevalence

Once confirming the existing ansätze it then allowed us to move on to evaluating variance of new ansätze we chose to experiment with. While having higher variance in the circuit architecture seems to be a necessary condition towards better training it is not a sufficient one. This was already known in the prior work re: TTN and MPS – these have high variance, but also less expressibility. We found that the new circuits outperform existing ones in terms of performance and have less vanishing gradients.

4 Results

In summary, the CERN Quantum Technology Initiative declared five quantum computing and algorithms projects of interest to the HEP community, which can be of benefit to humanity. The ambitions of CERN and the HEP community are large and it is reasonable that quantum computing has been folded into that large ambition. We chose to focus our efforts on assisting one of these projects, namely particle tracking reconstruction via QGNNs. Throughout the week we have performed a holistic diagnosis of the problem and came up with promising results. Our new Quantum Graph Neural Network together with the "`qc102_pqc(VanBuren)`" ansätze outperformed the best circuit in the repository "`qc10_pqc`" in the precision metric by 6%, which in the context of the newly generated dataset gives a good indication that our process would scale well to the larger dataset.

References

[1] S. Amrouche et al., *The Tracking Machine Learning challenge : Accuracy phase*, arXiv:1904.06778 [hep-ex, physics:physics] (2020), pp. 231–264. http://arxiv.org/abs/1904.06778. arXiv:1904.06778, doi:10.1007/978-3-030-29135-8_9.

[2] C. Tüysüz et al., *A Quantum Graph Neural Network Approach to Particle Track Reconstruction*, arXiv:2007.06868 [quant-ph] (2020). http://arxiv.org/abs/2007.06868. arXiv:2007.06868, doi:10.5281/zenodo.4088474.

[3] *The Large Hadron Collider*. https://home.cern/science/accelerators/large-hadron-collider.

[4] *High-Luminosity LHC*. https://home.cern/science/accelerators/high-luminosity-lhc.

[5] S. Farrell et al., *Novel deep learning methods for track reconstruction*, arXiv:1810.06111 [hep-ex, physics:physics] (2018). http://arxiv.org/abs/1810.06111. arXiv:1810.06111.

[6] *TrackML particle tracking challenge*. https://sites.google.com/site/trackmlparticle/.

[7] *Backpropagation through time*, Wikipedia (2021). https://en.wikipedia.org/w/index.php?title=Backpropagation_through_time&oldid=1051436108.

[8] B. Sanchez-Lengeling, E. Reif, A. Pearce and A. B. Wiltschko, *A Gentle Introduction to Graph Neural Networks*, Distill **6** (2021), p. e33. https://distill.pub/2021/gnn-intro. doi:10.23915/distill.00033.

[9] C. Tüysüz et al., *Performance of Particle Tracking Using a Quantum Graph Neural Network*, arXiv:2012.01379 [quant-ph] (2021). http://arxiv.org/abs/2012.01379. arXiv:2012.01379.

[10] *High-Luminosity LHC*. https://www.tensorflow.org/quantum/tutorials/barren_plateaus.