Quantum Graph Neural Networks for High Energy Physics Analysis at the LHC

Gopal Ramesh Dahale

dahalegopal27@gmail.com, LinkedIn, Github, Gitter: @Gopal-Dahale

1 Synopsis

The LHC at CERN contains large detectors which are made up of numerous small detectors that capture the hundreds of particles produced during collisions. It's one of the most difficult tasks in High Energy Physics (HEP) to determine whether the jet particles correspond to the signal or background. Graph Neural Networks (GNNs) have recently gained popularity and shown to exhibit higher AUC scores for jet tagging [QG20]. Quantum Machine learning has shown interesting applications in HEP [Gua+20].

I aim to explore Quantum Graph Neural Networks (QGNNs) for event classification. QGNNs can leverage the power of quantum computing to perform more efficient and accurate analyses of large datasets in HEP. I will evaluate the performance of our QGNN-based approach and compare it with traditional methods for event classification in HEP. This project has the potential to significantly advance our understanding of the fundamental particles and forces that govern our universe.

2 Benefits to the Community

This project aims to initiate the development of more efficient Quantum Machine Learning (QML) algorithms in High Energy Physics (HEP). Being an open-source project, it provides opportunities for individuals to learn and contribute simultaneously. To ensure accessibility, beginner-friendly documentation will be created, and a clean code Python library will be developed for easy integration with other projects. Additionally, user-friendly notebooks will be provided for ease of use. As an individual who has learned much from open-source projects like Cirq, PennyLane and Qiskit, I view this project as my contribution to the open-source community.

3 Deliverables

The following are the required deliverables that need to be accomplished before the official end of the project on August 28, 2023:

- Using PyTorch-Geometric [FL19], implement some classical GNNs to set a baseline.
- With the help of PennyLane [al18] and Jraph [God+20], implement a hybrid quantum-classical graph neural network model for event classification. Test different data encoding schemes, including angle, amplitude encoding etc.
- Create Jupyter notebooks in the development phase and later update them to serve as a beginner-friendly documentation for contributors.

- A white-paper summarising the results obtained with quantum and classical models.

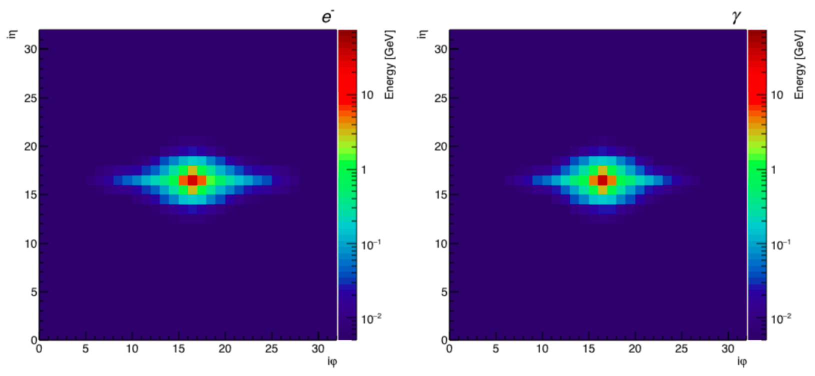
The following are additional deliverables that I plan to complete within the official deadline, if there is sufficient time. Alternatively, they will be achieved during the extended deadline of November 6, 2023, if time constraints arise:

- If any bugs are discovered in the project during the final review process, address them to ensure that the obtained results are accurate and not the result of logical errors in the code.
- Using the trained quantum models, obtain inference results on real hardware of IBM [21] and AWS [Ama20].

4 Related works

4.1 Datasets and Pre-processing

The authors of [And+18] have performed classification of e/γ showers. The image of shower is a 32 by 32 crystals from ECAL centered around the maximum-energy shower deposit. The averaged images of electron and photon showers obtained from [And+18] are shown in Figure 1



(a) Average electron shower (b) Average photon shower

Figure 1: e/γ showers averaged over 50k showers. The e shower is slightly more spread out in ϕ-in addition to being slightly asymmetric-due to bremsstrahlung effects. [And+18]

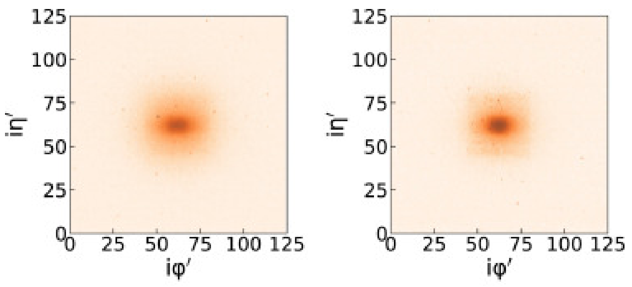
The paper [And+18] uses models like CNN, LSTM and FCN to perfrom shower classification and achieves a maximum ROC AUC score of 0.807. This will also become a baseline for the quantum GNN models that I will implementing.

Jet classification of quarks and gluons were performed in the paper [And+20]. Figure 2 shows the various subdetector image overlays averaged over test set of about 70k jets for each class. The maximum ROC AUC score they achieve with Tracks + ECAL + HCAL channel is 0.807. This will serve as baseline for the quantum GNN models.

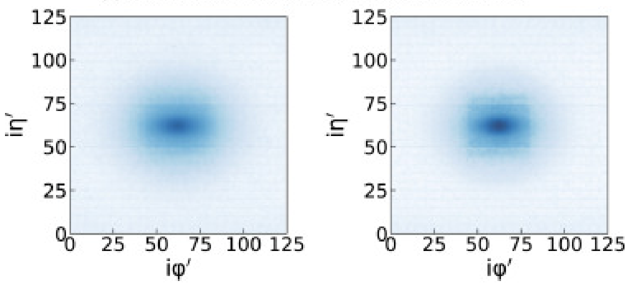
The pre-processing of data involves the selection of non-zero hit locations to obtain the x and y coordinates. Then we concatenate these coordinates with the hit energy at that location. This will result in a sample data point point of shape $N \times 3$, where N is the total non-zero hits.

4.1.1 Graph representation of events

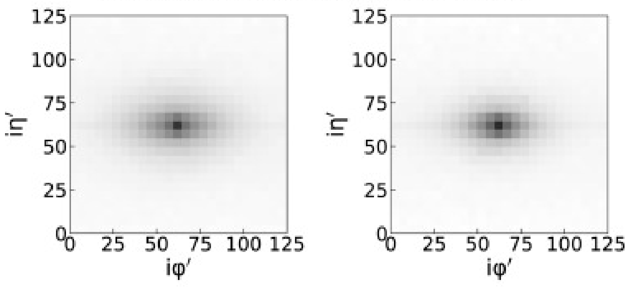
Since the images are sparse, I will employ the strategy described in [HDG21]. A graph is represented by $G = (V, E, A)$, where V refers to the vertices that make up the graph,



(a) Tracks overlay. Left: gluon jet, Right: quark jet.



(b) ECAL overlay. Left: gluon jet, Right: quark jet.



(c) HCAL overlay. Left: gluon jet, Right: quark jet.

Figure 2: Jet-view image overlays split by subdetector: tracks 2(a), ECAL 2(b), and HCAL 2(c) over 70k jets each, all in log-scale. Gluon jets appear on the left, quark jets on the right. Gluon jet showers are visibly more dispersed in all channels. Note the presence of horizontal bands in the ECAL 2(b) overlays, highlighting the image fidelity to capture the energy leakage across the ECAL barrel-endcap boundary. Image resolution: 125 $\times$ 125. [And+20]

E represents the edges that connect these vertices, and A is an adjacency matrix of size $N \times N$ that describes the relationship between the nodes, where N represents the total number of nodes in the graph. Each vertex in V is associated with a set of features that describe it.

In our scenario, consider the particle hits in a detector to be interconnected nodes within a graph. Therefore, each event is described by a set of nodes, with each node having features that include its $x - y$ location in 2D space and its energy. Then, use the k-nearest neighbors algorithm in 2D space to connect nodes, such that each node p is connected by an edge to k-nodes that are closest in terms of Euclidean distance given by $\sqrt{(x - x_i)^2 + (y - y_i)^2}$, where x_i and y_i refer to the coordinates of the node.

4.2 Classical GNNs for classification

GNNs can be used in three different ways for making predictions: node level, edge level and global level. GNNs have been used at global level to perform jet classification. They have also been used in particle track reconstruction and secondary vertex reconstruction. [SBV20] surveys the applications of GNNs in HEP. In this work, I will focus on global level. There exists popular architectures of the traditional GNNs inlcuding the Graph Convolutional Network (GCN), Graph Attention Network (GAT), Graph SAGE etc. I here describe a general formalism of GNN and in Section 4.3 and extend it to the quantum case.

One way to depict a graph is by using $G = (u, V, E)$, where there are N_v vertices and N_e edges. The u element refers to the attributes of the entire graph. The nodes or vertices are represented by $V = v_i$ for $i = 1 : N_v$, where v_i indicates the attributes of the i^{th} node. The edges are represented by $E = (e_k, r_k, s_k)$ for $k = 1 : N_e$, where e_k denotes the attributes of the k^{th} edge, and r_k and s_k are the indices of the two nodes (receiver and sender) that are linked by the k^{th} edge.

The processing stages are as follows:

$$e'_k = \phi^e(e_k, v_{r_k}, v_{s_k}, u) \qquad \bar{e}_i{}' = \rho^{e\to v}(E'_i) \qquad \text{Edge Block} \tag{1}$$

$$v'_i = \phi^v(\bar{e}_i{}', v_i, u) \qquad \bar{e}' = \rho^{e\to u}(E') \qquad \text{Node Block} \tag{2}$$

$$u' = \phi^u(\bar{e}', \bar{v}', u) \qquad \bar{v}' = \rho^{v\to u}(V') \qquad \text{Global Block} \tag{3}$$

A GN block contains 6 internal functions: 3 update functions (ϕ^e, ϕ^v, ϕ^u) and 3 aggregation functions $(\rho^{e\to v}, \rho^{e\to u}, \rho^{v\to u}$. The Graph Network formalism is not a model architecture in itself, as it does not specify the exact functions to be used.

The edge block calculates an output value, e'_k, for each edge and aggregates them based on their corresponding receiving node, $\bar{e}_i{}'$, where E'_i is the set of edges linked to the i^{th} node. The vertex block computes an output value, v'_i, for each node. Both the edge- and node-level outputs are then combined in the global block by aggregation. The GN's output is the collection of all the edge-, node-, and graph-level outputs, represented as $G' = (u', V', E')$.

(ϕ^e, ϕ^v, ϕ^u) are typically simple trainable neural network and $(\rho^{e\to v}, \rho^{e\to u}, \rho^{v\to u}$ are implemented as element wise mean, sum or maximum.

4.3 Quantum GNNs for classification

Quantum GNNs are relatively new (compared to Quantum NNs and CNNs) and are in research phase. [Gui19] proposes a Quantum Graph Recurrent Neural Networks (QGRNN) and Quantum Graph Convolutional Neural Networks (QGCNN) for learning hamiltonian dynamics of quantum systems. [Bee21] describes training with graph-structured quantum data. [Tüy+21] explores a hybrid-quantum classical graph neural network for particle tracks reconstruction. The QGNN I propose is similar to [Tüy+21] but will have 2 parts: Input and Node network. In the next sections I briefly describe how they work.

4.3.1 Input Network

This network increases the dimension of input data using a classical neural network. This works as an embedding network. The input size will be $N_v \times f$, where N_v is the number of nodes and f is the number of features corresponding to each node. The output size will be $N_v \times N_d$ where N_d is the hidden dimension.

4.3.2 Node Network

The node network builds upon the classical node block (Equation 2) where ϕ^v is replaced with a hybrid quantum classical neural network. Figure 3 describes the architecture.

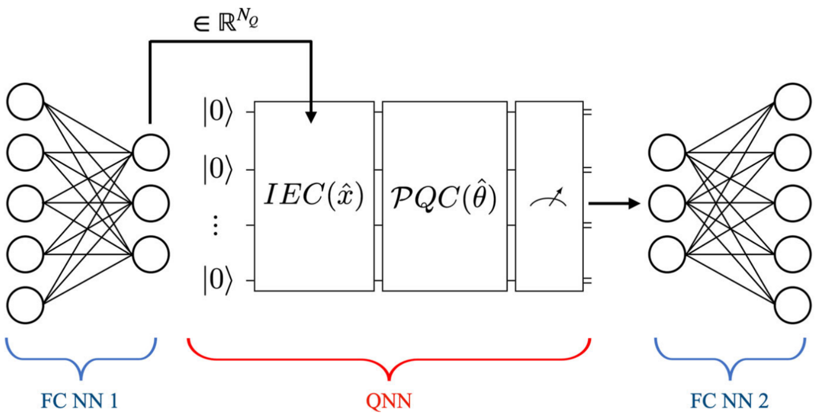


Figure 3: The HNN architecture comprises a classical fully connected Neural Network (FC NN) layer with sigmoid activation as the initial input. The output of this layer is processed by the information encoding circuit (IEC) in the QNN module. The encoded states then undergo transformations using the parametrized quantum circuit (PQC). Finally, the QNN output is computed as the expectation values of the measured qubits. The results of the qubit measurements are integrated using a final FC NN layer with sigmoid activation. The network's input and output dimensions vary based on the type of network. [Tüy+21]

4.3.3 Information Encoding Circuit

The authors of [Tüy+21] uses Angle embedding circuit. This is a simple data embedding scheme but scales with the number of features. Due to the hybrid network architecture, this encoding with suitable as we can control the qubits count.

I want to test out different encoding schemes like Amplitude Encoding to understand their impact on the metrics. To make it more concrete, I want to remove the fully connected and make it a fully quantum model to ensure that the number of input features to the quantum circuit doesn't shoot up, I will use a Data-Reuploading circuit. This is described in the Section 4.3.4

4.3.4 Choice of Ansatz

The ansatz or the Parameterized Quantum Circuit (PQC) is the trainable circuit. [Tüy+21] uses different architectures including Matrix product state (MPS) [Wik], Tree Tensor Networks (TTN) [Hik+23] etc. I will try to use them too for with the proposed datasets.

I propose using a Data-Reuploading circuit (DRC) [Pér+20]. This is shown is Figure 4 and 5. **With DRC, we can control the number of number of input and output features and make the network fully quantum**. This project introduces a novel research contribution in HEP image data analysis using a fully quantum model, which has not been explored previously in QGNN-based methods.

4.3.5 Optimizer and Loss function

The issue of barren plateaus, where gradients exponentially vanish, is a known problem discussed in previous works [Mcc+18] [Cer+21]. To avoid this problem, it is important to

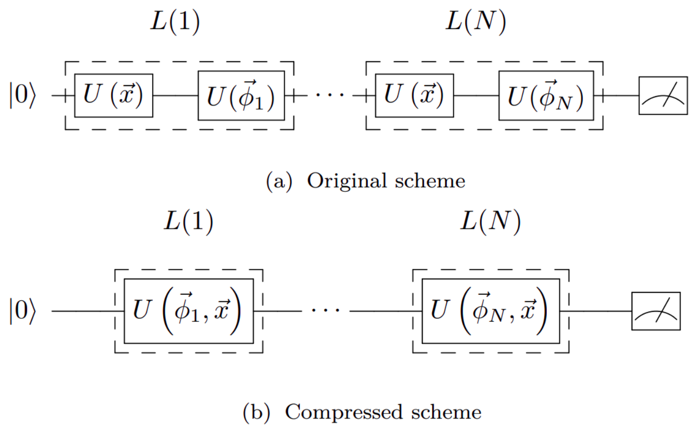


Figure 4: Data reuploading on single qubit. The quantum circuit is partitioned into layer gates, denoted as $L(i)$, which serve as the building blocks for the classifier. In the upper circuit, each layer is comprised of a $U(x)$ gate that loads the data and a parameterized unitary gate $U(\phi)$. We repeat this building block N times and ultimately evaluate a cost function that is linked to the similarity between the final state of the circuit and the target state for its class. The ϕ_i parameters can be adjusted to minimize this cost function. In the bottom circuit, a single unitary gate can incorporate both data and tunable parameters. [Pér+20]

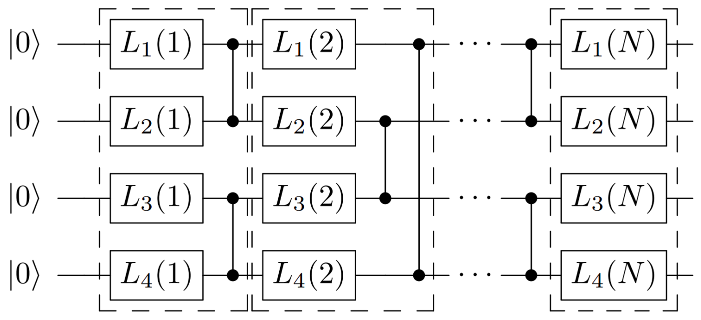


Figure 5: The quantum classifier circuits have four qubits. The circuit has entanglement and each layer includes a simultaneous rotation and two simultaneous CZ gates. The order of CZ gates alternates between different sets of qubits in each layer. The last layer has no CZ gates. When the number of layers is fixed, the number of parameters to be optimized is four times the number needed for a single-qubit classifier. [Pér+20]

carefully select an appropriate optimizer. The Adam optimizer has been suggested as a good strategy to mitigate the issue, as noted in [WGK20].

I also want to try using multiple optimizers [Ten] to use quantum natural gradient descent [Sto+20]. The classical neural network will be optimized with the classical optimizers eg. Adam.

The loss function will be the binary cross-entropy function, where y_i is the ground truth and $\hat{y_i}$ is the predicted label. Equation 4 describes the loss function.

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i log(1-\hat{y_i}) + (1-\hat{y_i})log(\hat{y_i}) \tag{4}$$

5 Proposed work

Based on the approaches described in Section 4.3, I propose the following strategy.

- Implement the classical GNN models to compare with the QGNNs. These models will be implemented using PyTorch-Geometric [FL19].
- The datasets used for training the models include Electron Photon showers [And+18] and Quark Gluon jets [And+20].
- The datasets will be preprocessed for suitable conversion of images to graphs. This can be fed into classical as well as hybrid quantum-classical neural networks.
- The DRC ansatz will be used along with baselines obtained from MPS and TTN ansatzes and fully-connected classical layers for hybrid classification. I will benchmark by varying the number of qubits and layers. The details may change based on mentor guidance.
- The Adam optimizer will be the first choice, but other optimization methods may be used if necessary as described in Section 4.3.5. Hyperparameters such as epochs, learning rate, count of ansatz layers, and batch size will be tuned during the development phase.
- Loss curves and accuracy curves will be used as metrics for evaluating the results, with ROC-AUC used to gain more detailed insights.

6 Milestones and Deadlines

6.1 Review Period (5 April - 4 May)

To gain more practical experience, I plan to work more extensively with Jraph[God+20] and PennyLane[al18]. Even though I have some experience with PennyLane, Jraph as a library is new to me and delving deeper into these frameworks will be beneficial for future work.

6.2 Community Bonding Period (4 April - 28 May)

6.2.1 Week 1

I plan to receive feedback from my mentors on the proposal to make informed decisions on the next steps. I will engage with the mentors to discuss any revisions that need to be made to the deliverables or timeline. After that, we will have more detailed discussions about topics such as datasets, preprocessing steps, and coding practices. This will ensure that the project's goals are met on time and that the coding period proceeds smoothly.

6.2.2 Week 2

Set up a GitHub repository to store all the necessary work for the project. Ensure that the Jupyter notebook and Python environment are correctly set up to avoid any errors caused by library versions or similar issues. Collect all necessary datasets and clean them. Conduct exploratory data analysis and perform preprocessing steps. Create a basic classical GNN. Document all observations and share a report with the mentors.

6.2.3 Week 3

Allocate some time to learn about the processes, code of conduct, and other related matters of ML4SCI. Communicate with experienced developers, researchers and fellow participants to gain a better understanding of research in quantum machine learning and quantum computing in general. This will enable me to delve deeper into the subject matter and serve as a foundation for my pursuit of a PhD in Quantum Computing.

6.3 Coding Period (29 May - 28 August)

The majority of the coding work will be carried out in Jupyter notebooks. Once satisfactory results are obtained, I will convert the notebooks into Python library. The Jupyter notebooks will be used primarily for documentation purposes.

6.3.1 Week 1-2

The classical GNNs discussed in the proposal will be implemented and trained on the cleaned datasets. The results will be tabulated and used as baselines for comparison with the QGNNs.

6.3.2 Phase I (till 10 July)

During Phase I, I plan to focus on creating a working model of QGNN as described in Section 4.3. Once it is done, I can continue working on each QGNN ansatz followed by DRC weekly. The goal is to train each ansatz each week while also tuning the hyperparameters.

One of the goals is to get rid of the fully connected layers and use a DRC with tunable parameters being the count of qubits and the number of layers. Observations will be recorded and the performance of the models will be discussed with mentors weekly. I have not divided the work of Phase I into specific weeks because the progress of each QGNN model can be unpredictable.

Correct any errors in the code and conduct a thorough review of the entire code to ensure that there are no logical errors that could impact the results.

6.3.3 Evaluations (10 July - 14 July)

Produce a comprehensive report that encompasses all the results achieved during Phase I.

6.3.4 Phase II (till 21 August)

After completing Phase I, the results of the QGNN models are ready, and we can compare them with classical models. We will divide them into three categories: fully quantum QGNN, hybrid QGNN, and classical GNN models. To compare them, we need to make sure they have the same count of trainable parameters, and we will seek guidance from mentors for this. We will record observations, including but not limited to datasets, qubit count, layer count, train and test AUC, and loss. We will tabulate the observations and visualize them using plotting libraries in python. We will also resolve any bugs in the code and review it to ensure there are no logical errors that might affect the results.

In the final week, I allocated time to complete the documentation by creating markdowns and beginner-friendly notebooks. The aim is to deploy the notebooks in the form of a website, which will be helpful for newcomers to learn and contribute to the project.

6.3.5 Evaluations (21 August - 28 August)

The final report will comprise an abstract, an introduction, and a refined proposal. It will incorporate all the reports that were produced from the beginning of the project. The report will have a detailed explanation of the QGNN model along with its intricacies. The results obtained during the experimentation will be presented in the form of tables. A conclusion and future scope of work will be discussed. The future scope will include exploring different optimizers apart from Adam and accomplishing optional deliverables that may/may not complete during the coding phase.

7 Biographical Information

I graduated from the Indian Institute of Technology, Bhilai [Ind] in the field of Electrical Engineering with Specialization in Computer Science. I have been in the field of quantum computing for a couple of years. I have experience in open-source contributions and research in quantum computing.

I participated in the Qiskit Advocate Mentorship Program in Fall 22 [Qis], where I contributed to the design of Tensor networks and their performance analysis over different parameters. I also investigated tensor network architectures in quantum circuits and tested various data encoding schemes.

I was previously a participant in Google Summer of Code with this org, ML4SCI, where I investigated the potential of Quantum Convolutional Neural Networks for the classification of particle images from high-energy physics events. These open-source programs have well acquainted me with how research is done.

Recently I participated in QHack 2023 [Xan] organized by PennyLane where I enhanced VQE for carbon capture applications. I have used hardware to test my algorithm and summarized my findings. Later I extended it for larger molecules (Metal-Organic Frameworks) using Density Matrix Embedding Theory (DMET) and submitted it to Deloitte's Quantum Climate Challenge 2023 [Del] and ranked in the Top 5.

Note: The Tasks can be found here.

References

[al18] Ville Bergholm et al. "PennyLane: Automatic differentiation of hybrid quantum-classical computations" (2018). URL: `https://docs.pennylane.ai/en/stable/#how-to-cite`.

[And+18] M. Andrews et al. "End-to-End Event Classification of High-Energy Physics Data". *J. Phys. Conf. Ser.* 1085.4 (2018), p. 042022. DOI: `10.1088/1742-6596/1085/4/042022`.

[Mcc+18] Jarrod Mcclean et al. "Barren plateaus in quantum neural network training landscapes". *Nature Communications* 9 (Nov. 2018). DOI: `10.1038/s41467-018-07090-4`.

[FL19] Matthias Fey and Jan E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". *ICLR Workshop on Representation Learning on Graphs and Manifolds.* 2019.

[Gui19] Trevor McCourt Guillaume Verdon. "Quantum Graph Neural Networks" (2019). DOI: `10.48550/arXiv.1909.12264`.

[Ama20] Amazon Web Services. *Amazon Braket.* 2020. URL: `https://aws.amazon.com/braket/`.

[And+20] M. Andrews et al. "End-to-end jet classification of quarks and gluons with the CMS Open Data". *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 977 (2020), p. 164304. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2020.164304. URL: https://www.sciencedirect.com/science/article/pii/S0168900220307002.

[God+20] Jonathan Godwin* et al. *Jraph: A library for graph neural networks in jax.* Version 0.0.1.dev. 2020. URL: http://github.com/deepmind/jraph.

[Gua+20] Wen Guan et al. "Quantum machine learning in high energy physics". *Machine Learning: Science and Technology* 2 (Oct. 2020). DOI: 10.1088/2632-2153/abc17d.

[Pér+20] Adrián Pérez-Salinas et al. "Data re-uploading for a universal quantum classifier". *Quantum* 4 (Feb. 2020), p. 226. DOI: 10.22331/q-2020-02-06-226.

[QG20] Huilin Qu and Loukas Gouskos. "Jet tagging via particle clouds". *Phys. Rev. D* 101 (5 Mar. 2020), p. 056019. DOI: 10.1103/PhysRevD.101.056019. URL: https://link.aps.org/doi/10.1103/PhysRevD.101.056019.

[SBV20] Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. "Graph Neural Networks in Particle Physics" (July 2020). DOI: 10.1088/2632-2153/abbf9a. arXiv: 2007.13681 [hep-ex].

[Sto+20] James Stokes et al. "Quantum Natural Gradient". *Quantum* 4 (May 2020), p. 269. ISSN: 2521-327X. DOI: 10.22331/q-2020-05-25-269. URL: https://doi.org/10.22331/q-2020-05-25-269.

[WGK20] David Wierichs, Christian Gogolin, and Michael Kastoryano. "Avoiding local minima in variational quantum eigensolvers with the natural gradient optimizer". *Phys. Rev. Res.* 2 (4 Nov. 2020), p. 043246. DOI: 10.1103/PhysRevResearch.2.043246. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.2.043246.

[Bee21] Khosla M. Beer K. "Quantum machine learning of graph-structured data" (2021). DOI: 10.48550/arXiv.2103.10837.

[Cer+21] M. Cerezo et al. "Cost function dependent barren plateaus in shallow parametrized quantum circuits". *Nature Communications* 12 (Mar. 2021). DOI: 10.1038/s41467-021-21728-w.

[HDG21] Ali Hariri, Darya Dyachkova, and Sergei Gleyzer. "Graph Variational Autoencoder for Detector Reconstruction and Fast Simulation in High-Energy Physics". *EPJ Web of Conferences* 251 (Jan. 2021), p. 03051. DOI: 10.1051/epjconf/202125103051.

[21] *IBM Quantum*. 2021. URL: https://quantum-computing.ibm.com/.

[Tüy+21] Cenk Tüysüz et al. "Hybrid quantum classical graph neural networks for particle track reconstruction". *Quantum Machine Intelligence* 3 (2021).

[Hik+23] Toshiya Hikihara et al. "Automatic structural optimization of tree tensor networks". *Phys. Rev. Res.* 5 (1 Jan. 2023), p. 013031. DOI: 10.1103/PhysRevResearch.5.013031. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.5.013031.

[Del] Deloitte. *Deloitte's Quantum Climate Challenge 2023.* URL: https://app.ekipa.de/challenges/deloitte-quantum-23/brief.

[Ind] Bhilai Indian Institute of Technology. *Indian Institute of Technology, Bhilai.* URL: https://www.iitbhilai.ac.in/.

[Qis] Qiskit. *Qiskit Advocate Mentorship Program in Fall 22*. URL: https://github.com/qiskit-advocate/qamp-fall-22/issues/28.

[Ten] TensorFlow. *tfa.optimizers.MultiOptimizer*. URL: https://www.tensorflow.org/addons/api_docs/python/tfa/optimizers/MultiOptimizer.

[Wik] Wikipedia. *Matrix Product State*. URL: https://en.wikipedia.org/wiki/Matrix_product_state.

[Xan] Xanadu. *QHack*. URL: https://qhack.ai/.