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

Gopal Ramesh Dahale
dahalegopal27@gmail.com
gopald@iitbhilai.ac.in
+91 8425930185
LinkedIn
Github
Gitter handle: @Gopal-Dahale

Synopsis

Determining whether an image of a jet particle corresponds to signals or background signals is one of the many challenges faced in High Energy Physics. CNNs have been effective against jet particle images as well for classification purposes [9] [10]. Quantum computing is promising in this regard and as the QML field is evolving, this project aims to understand and implement QCNN and gain some enhancement.

The goal of this study is to show the capabilities of QML especially QCNN for classifying the HEP image datasets. QCNN can be completely quantum or can be a hybrid with classical. The aim is to implement both. We will use quantum variational classification instead of the final FC classical layers in the quantum setting. This will give more depth about the quantum power that can be used in the near term future.

Benefits to the Community

The implementation and the research that will be done throughout this project will be a starting step towards more efficient QML algorithms in HEP. As this is an open-source project, people can contribute and learn at the same time. The documentation will be beginner-friendly and a python library will be created for directly importing into other projects with clean code. Notebooks will be made available for ease of use. Since I have learned quite from open source projects like TensorFlow, Cirq, Qiskit etc., this project will be my contribution to the open-source community.

Deliverables

Below I summarize the required deliverables that will be achieved before the official end (12 Sept 2022).

1. Implementation of some chosen classical ML models as a baseline using TensorFlow.
2. Implementation of QCNN for fully quantum and hybrid architectures including different data encoding schemes (angle, amplitude encoding etc.). Jupyter notebooks will be used as a testing environment but the final python modules will have a .py extension. These notebooks will be then extended as a form of documentation for beginner contributors.
3. Comparison of classical ML baseline models with QCNN and documenting the results.
4. A report which will be an analysis of both the works (fully quantum and hybrid).
5. Documentation of all code (comments, markdown, notebooks, tutorials etc.).

Below I summarize optional deliverables which I plan to achieve within the official deadline (12 Sept 2022) if time permits or can be achieved during the extended deadline (21 November 2022) if time does not permit.

1. Fix any bugs if found in the project during a final review done by me. This makes sure that the results obtained are correct and not due to some logical error in the code.
2. Comparision of existing open-source QCNNs with this work for a scope of improvement.

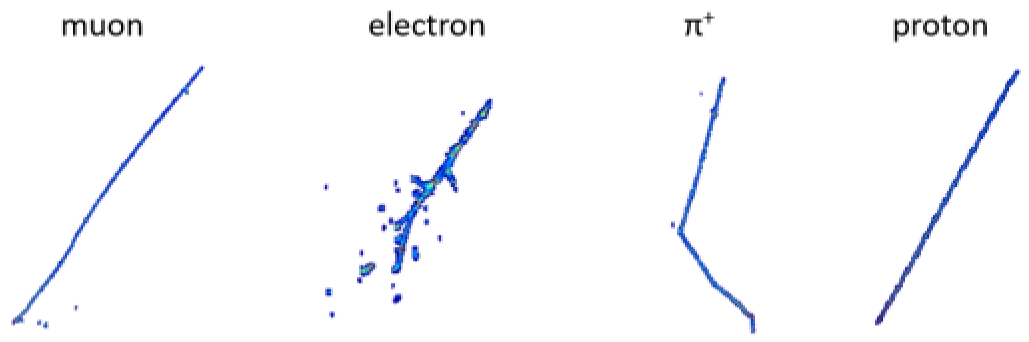
Related works

Related Datasets

I believe that using datasets that previous QCNN works have used will be a good choice as it will also help in the comparison of the results. I choose to consider the dataset used by the authors of Quantum Convolutional Neural Networks for High Energy Physics Data Analysis [1]. They have achieved a testing accuracy as follows:

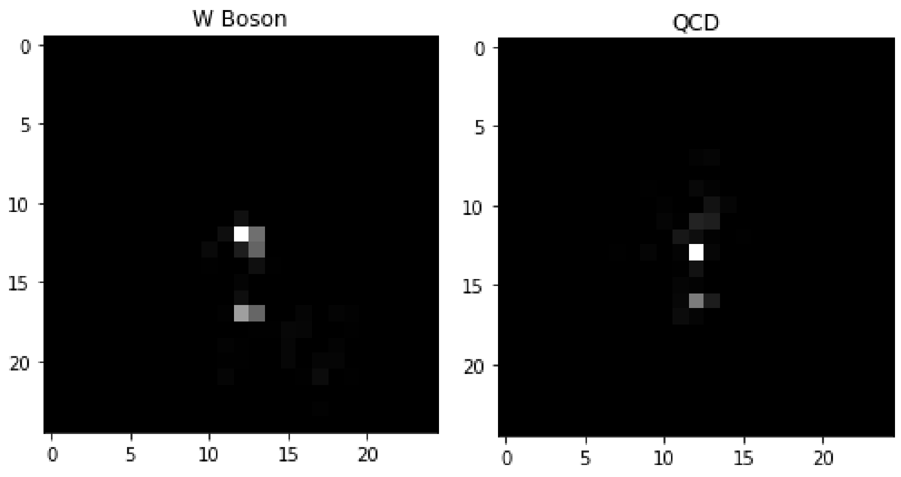
Muon vs Electron	92.5 %
Muon vs Charged Pion	97.5 %
Muon vs Proton	97.5 %

These will be the baseline for our QCNN model.



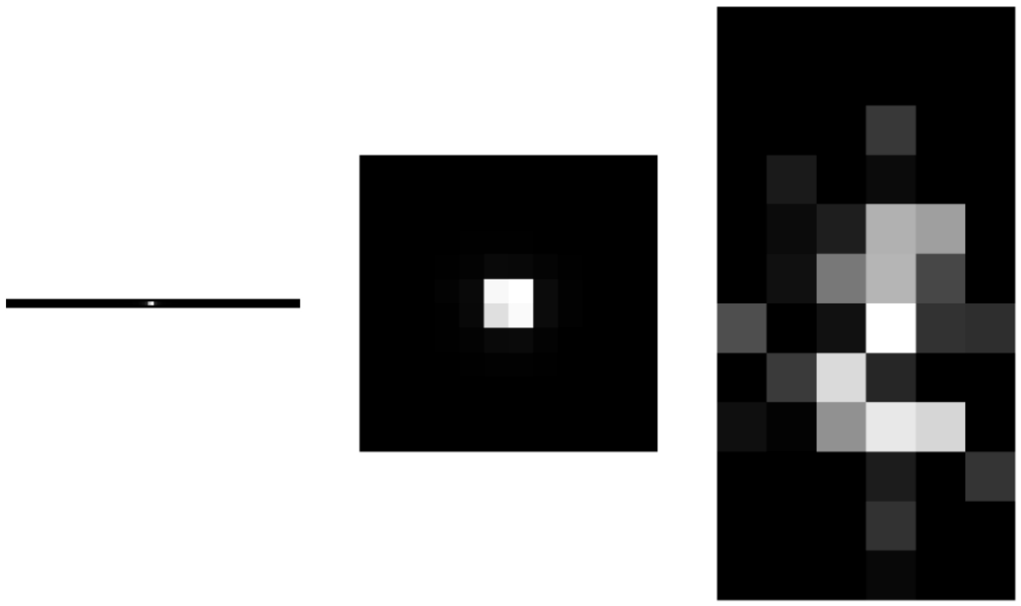
Simulated particle activities (μ +, e −, π +, p) in a LArTPC detector [1]

Public Datasets of LHC by CERN are a good source for testing the model. One of the datasets is of jet images (W boson and QCD) used for GAN which has images of size 25x25. An example is shown below

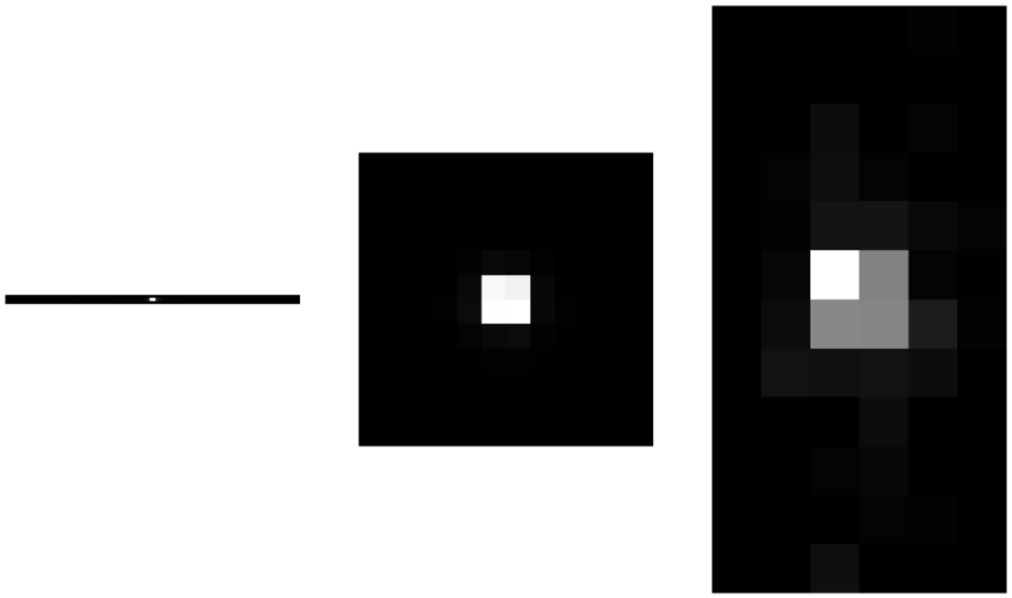


Jet Images (W Boson and QCD)

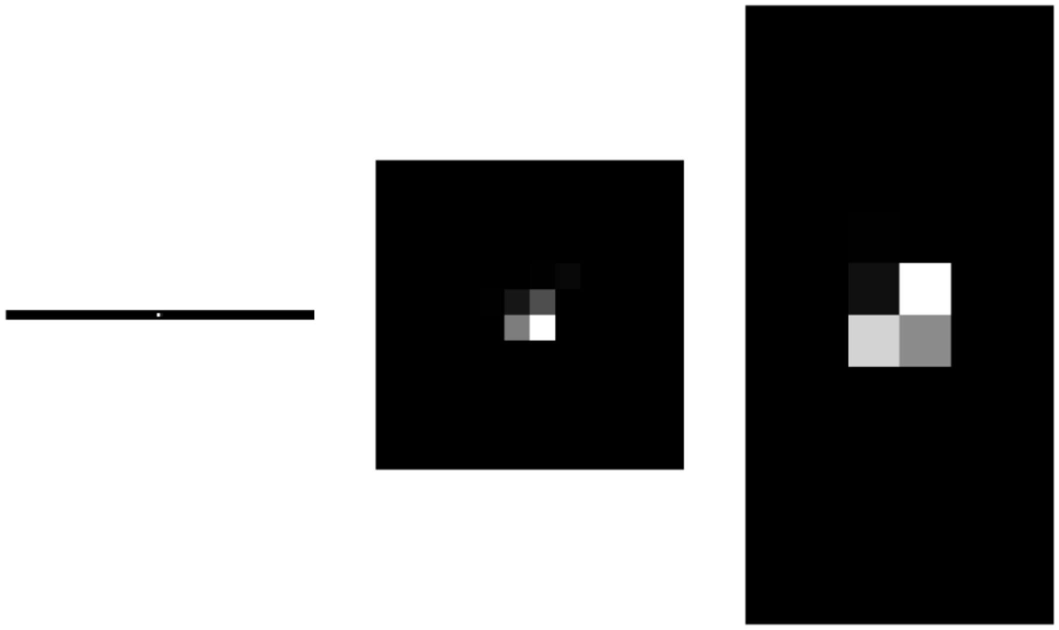
Another Dataset is Electromagnetic Calorimeter Shower images which collection of 100,000 calorimeter showers corresponding to the particles e+, gamma, and p+. Sample images are shown below:



EC shower for e+ for layers 0,1 and 2



EC shower for gamma for layers 0,1 and 2



EC showers for p+ for layers 0,1 and 2

For a final check for the generality of the QCNN model, we can use the MNIST handwritten digits database.

Data preprocessing

Since the maximum number of qubits allowed on Cirq is 20, we need some strategy for preprocessing the data. As done in the MNIST classification tutorial by TensorFlow quantum, we can rescale images. I have tried this approach in task 3b but this does not preserve the features well. Another approach is max-pooling which also ends up doing the same.

As PCA worked well in task 3b without affecting the variance much and also the authors of Layerwise learning for quantum neural networks and Quantum-enhanced supervised learning with variational quantum circuits use PCA in their work. The variance of the dataset after PCA depends more on the dataset and can be fine-tuned during the development phase. One of the problems in using PCA is that if the number of sample points in the dataset is too small then the model won't train enough. This has happened with me in task 3 but as the dataset that I have mentioned above is large enough (10,000 data points), this won't be an issue.

After PCA, we may need to normalize the dataset depending on the data encoding scheme used. As done in the MNIST classification tutorial, they use a threshold value (0.5) to decide whether to put a NOT gate on a qubit for a particular image. Amplitude encoding[5] is one way to encode data as it encodes a very large dataset using only the log_2(N) number of qubits (N is the length of a feature vector).

One of the widely used data encoding schemes in QML is angle encoding [5] which rotates the qubit by a certain amount of angle (eg. (-pi, pi)). Since I will use PCA, angle encoding works well. We can try both the methods (amplitude and angle) and give results accordingly. We can try both the methods (amplitude and angle) and give results accordingly.

Convolutional Algorithm and classification

For this work I will, we will use different circuit ansatz. In Quantum Convolutional Neural Networks for High Energy Physics Data Analysis [1], the authors have used an angle encoding scheme followed by variational which consists of CNOT gates and quantum measurement. They used VQC as a convolutional kernel (filter). Authors of Hybrid quantum-classical Convolutional Neural Networks [6] enhance the feature mapping process. Cong, Choi and Lukin of Quantum Convolutional Neural Networks [7] use a threshold for applying a NOT gate on a qubit and then train using one qubit and two-qubit unitaries. These works mentioned have performed are at par with classical CNN or even better. I prefer a QCNN circuit that performs a measurement operation in the intermediate as shown by Franken and Georgiev [8].

I will compare the two architectures. One with a classical fully connected layer at the end for making predictions and the other with a VQC (non-classical). This offers two new perspectives and gives a broader view. This helps in deciding which type of architecture should be used under different circumstances. I will use the VQC as proposed by Rodney Osodo as am I have experimenting with it quite a lot lately.

We know that the problem of exponentially vanishing gradients (barren plateaus) exists as discussed in [2] and [3]. Choosing an optimizer is therefore necessary for avoiding barren plateaus. I think using the Adam optimizer will be a good strategy as discussed in [4].

Proposed work

The classical ML models baselines that I will compare with the QCNN are classical CNN, feed-forward neural network and SVM. These will be implemented using TensorFlow in Python.

Models will be trained on datasets of jet particle images and electromagnetic calorimeter showers dataset and LArTPC detector dataset [1]. I will also use the MNIST dataset in the end for general evaluation purposes. This will not be limited to binary classification.

For Data preprocessing, check whether PCA suits the dataset. If this is the case then we can feed it into VQC as well as the QCNN and assess the performances in both cases.

Use convolution operation with trainable circuits ansatz and then used VQC for classification (quantum) and use fully-connected classical layers for classification (hybrid). The actual details about the convolution ansatz can change as per the mentors' wisdom and guidance. Using the proposed ansatz in the previous section is my strategy.

Adam optimizer will be used. If it fails then other optimization methods will be used. Hyperparameter tuning (epochs, learning rate, count of convolutional layers, batch size etc.) will be tuned during the development phase.

I will use the loss curves and accuracy curves as metrics for evaluating the results. ROC-AUC will be used for gaining more fine details.

Milestones and Deadlines

Review Period (20 April - 20 May)

Try more hands-on with TensorFlow quantum and Cirq. Although I have developed some experience with it during the tasks, understanding it in more depth will help later. Also, implement fully connected classical layers for hybrid QCNN models in TensorFlow. Implementing the VQC as discussed in the proposed work section will be a helpful exercise for me and will also save time during the coding period.

Community Bonding Period (20 May - 12 June)

Week 1 (20 May - 26 May)

Obtain feedback on the proposal from mentors. This will help me in deciding what should be the next steps. On a high level, I will discuss with mentors the choices that require any revision like the deliverables or timeline. After this, we will discuss in-depth more about the basics like datasets and preprocessing steps etc. and what coding practices to follow. This will help in fulfilling the targets on time and will create a smooth coding period.

Week 2 (27 May - 2 June)

Create a GitHub repository for all the work required for this project. Setup Jupyter notebook and python environment properly to eliminate any possibility of library versions error and other similar errors. Gathering all the required datasets and cleaning them. Performing exploratory data analysis and preprocessing steps. Try a basic classification using just VQC after the preprocessing step. Record all observations and share a report with the mentors.

Week 3 (3 June - 12 June)

Spend some time learning about ML4SCI's processes, code of conduct etc. Interacting with senior developers and researchers. This will help me understand research in quantum machine learning and quantum computing in general in more depth and will be my stepping stone toward a PhD in Quantum Computing.

Coding Period (13 June- 12 Sept)

Most of the coding will be done in Jupyter notebooks. After satisfactory results, I will convert them to python files. The Jupyter notebooks will serve the documentation purpose.

Week 1 (13 June - 19 June)

Implementation of classical ML models that I have discussed in the proposed work section. Training them on the cleaned datasets and tabulating the results. These will be the baselines for QCNN.

Week 2 (20 June - 26 June)
Implement the QCNN circuit ansatzes that I have discussed in the proposed work section. Training them on a subset of datasets to achieve some satisfactory results. Recording these observations.

Till Phase I (27 June - 25 July)
Till Phase I ends, the idea is to work on each QCNN ansatz week by week. Each week, train each ansatz with VQC and fully connected layers (classical). Hyperparameter tuning will be done each week. The observations will be recorded. The performance of models will be discussed with mentors each week. I have not segregated the work of Phase I in weeks as this work is susceptible to the performance of each QCNN model.

Before Phase I ends:
Resolve any bugs if present in the code. Review the whole code properly so that no logical error is present which might affect the results.

Evaluations (25 July - 29 July)
Create a report with all results mentioned done till phase I. Present an analysis of full quantum and hybrid models.

Till Phase II (27 July - 28 August)
After Phase I, we have results of QCNN models and we are ready to compare them with classical models. We will have three categories. QCNN (fully quantum), QCNN (hybrid) and Classical ML models. We will do a comparison of these models. We might want to check the count of trainable parameters for each of them as we want to be on the same scale for comparing them. I will need some assistance on this with mentors.

The observations will be tabulated and will be visualized using plotting libraries in python. Recording observations include but are not limited to datasets, count of qubits, count of layers, accuracy (train and test), loss etc.

Same as Phase I, resolve any bugs if present in the code. Review the whole code properly so that no logical error is present which might affect the results.

The final week (29 August - 4 Sept)

I have spared this week to finalize the documentation. Creating proper markdowns and beginner-friendly notebooks (used from the coding period). This will be like Qiskit's textbook. This will help newbies to learn and contribute as well.

Evaluations (5 Sept - 12 Sept)

Final report will be created which will include an abstract, introduction and revised proposal. All the reports that are made from the start will be combined here. The QCNN model and its details in depth will be explained. The tabulated results will be shown. Concluding remarks and future work will be discussed. Future work includes using different optimizers (apart from Adam) and Optional deliverables (if remained uncompleted during the coding phase).

Biographical Information

I am pursuing my BTech. electrical engineering student with a specialization in computer science at the Indian Institute of Technology, Bhilai and currently I am in the final semester. Although I am from electrical engineering, I am more interested in computer science, especially quantum computing and quantum physics. I have a strong interest in the research domain and recently have been researching high-performance memory optimal CNNs on GPUs.

I know various programming languages like **Python, C/C++, Javascript** etc. Much time is spent around things related to machine learning and quantum computing. Recently I took a course on **Quantum symmetric key cryptanalysis** by Dr Dhiman Saha (PhD IIT Kharagpur) and this made my interest in quantum computing raise to the next level.

Within the course, I have studied and implemented various quantum algorithms and key recovery attacks on cyphers like simplified AES and SIMON.

Earlier I used to study quantum computing on my own but didn't realise its applications in the real world. The course taught me its applications in cryptography. Out of curiosity I started studying QML in December 2021 and have developed a sound understanding of it.

I am currently a part of OpenLake, which is a club aimed at promoting open source in the world and also a part of Developer Student Clubs. Through this, I have contributed to and mentored open source projects. Our team organized open source hackathons and winter of code (similar to GSoC) where I have mentored a Quiz-App project.

I have studied Quantum computing and machine learning a lot in the previous months. I have sound knowledge of Qiskit, ProjectQ, QSim, libQuantum and QuEST. Regarding the classical part, I have taken Andrew Ng's course on Coursera and have strong fundamentals. I have utilized frameworks such as TensorFlow, PyTorch, Pytorch Lightning and Scikit-Learn to build and train machine learning models and deploy them as well. As I have interned as a software engineer in multiple startups based on machine learning (Newzera, MeetAI) I know programming ethics and best practices and the importance of time management.

Note: The link for tasks is here.

References

1. Quantum convolutional neural networks for high energy physics data analysis. Samuel Yen-Chi Chen (Brookhaven), Tzu-Chieh Wei (YITP, Stony Brook), Chao Zhang (Brookhaven), Haiwang Yu (Brookhaven), Shinjae Yoo (Brookhaven). e-Print: 2012.12177 [cs.LG]. DOI: 10.1103/PhysRevResearch.4.013231 (publication). Published in: Phys.Rev.Res. 4 (2022) 1, 013231
2. McClean, J.R., Boixo, S., Smelyanskiy, V.N. *et al.* Barren plateaus in quantum neural network training landscapes. *Nat Commun* 9, 4812 (2018). https://doi.org/10.1038/s41467-018-07090-4
3. Cerezo, M., Sone, A., Volkoff, T. *et al.* Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nat Commun* 12, 1791 (2021). https://doi.org/10.1038/s41467-021-21728-w
4. Avoiding local minima in variational quantum eigensolvers with the natural gradient optimizer David Wierichs, Christian Gogolin, and Michael Kastoryano Phys. Rev. Research **2**, 043246 – Published 17 November 2020
5. Maria Schuld and Francesco Petruccione, *Supervised Learning with Quantum Computers*, Springer 2018, doi:10.1007/978-3-319-96424-9.
6. Liu, J., Lim, K.H., Wood, K.L. *et al.* Hybrid quantum-classical convolutional neural networks. *Sci. China Phys. Mech. Astron.* 64, 290311 (2021). https://doi.org/10.1007/s11433-021-1734-3
7. Cong, I., Choi, S. & Lukin, M.D. Quantum convolutional neural networks. *Nat. Phys.* 15, 1273–1278 (2019). https://doi.org/10.1038/s41567-019-0648-8
8. Franken, Lukas and Bogdan Georgiev. “Explorations in Quantum Neural Networks with Intermediate Measurements.” *ESANN* (2020).
9. Deep neural network for pixel-level electromagnetic particle identification in the MicroBooNE liquid argon time projection chamber C. Adams *et al.* (MicroBooNE Collaboration) Phys. Rev. D **99**, 092001 – Published 7 May 2019
10. Context-enriched identification of particles with a convolutional network for neutrino events F. Psihas, E. Niner, M. Groh, R. Murphy, A. Aurisano, A. Himmel, K. Lang, M. D. Messier, A. Radovic, and A. Sousa Phys. Rev. D **100**, 073005 – Published 14 October 2019