Quantum-Enhanced Image Classification with Graph Neural Networks

Yeray Cordero Carrasco[1]

[1]Computer Science Student, Escola d'Enginyeria, Universitat Autònoma de Barcelona

Keywords:
Graph Neural Networks
Quantum Algorithms
Quantum Computing
Quantum GNN
Quantum Machine Learning

Abstract The abstract discusses Quantum Graph Neural Networks (QGNNs) and their potential to revolutionize quantum machine learning by efficiently addressing complex graph-based problems. QGNNs leverage quantum states and operations for graph analysis. The optimization of QGNNs is a multifaceted challenge involving hardware and algorithmic aspects. Hardware advancements in quantum computing, including larger and more stable quantum processors, are crucial for maximizing QGNN capabilities. Algorithmic optimization entails designing quantum circuits, gate optimization, and parameter tuning to improve QGNN performance.[1] Hybrid quantum-classical approaches, combining classical neural networks with quantum components, offer new avenues for optimization. As quantum computing technologies mature and optimization techniques for QGNNs evolve, we can expect transformative breakthroughs in solving real-world problems involving graph-structured data.[2]

 Submitted: 08th October 2023 as the BSc Final Project Initial Report.

1. Objectives

Our project focuses on enhancing Quantum Graph Neural Network (QGNN) optimization for image classification, as introduced on [3], through a combination of generic and specific objectives:

1.1 Generic Objectives

1. **Contributing to Innovation**: Drive innovation in the field of image classification and graph optimization by exploring new approaches and techniques.
2. **Disseminate Findings and Insights**: Actively share the findings, insights, and outcomes of the project with the scientific community and stakeholders.

1.2 Specific Objectives

1. **Accuracy Analysis**: Enhance the accuracy of image classification tasks compared to baseline models by optimizing QGNN architectures. Similar to [4].
2. **Training Time Analysis**: Develop and apply innovative techniques that reduce the training time of QGNN models while maintaining or surpassing existing performance benchmarks.
3. **Enhance Robustness to Noise**: Improve the adaptability and robustness of QGNNs, enabling them to proficiently analyze diverse image data, even when confronted with noise or inherently unstructured inputs. This was explored in several previous papers, as in [5].
4. **Improve Interpretability**: Devote efforts to make QGNN-based image classification models more interpretable and explainable. See [6].

These objectives serve as the guiding framework for our project, ensuring a clear focus on specific, measurable, achievable, relevant, and time-bound goals in the optimization of QGNNs for image classification while contributing to innovation and knowledge dissemination.

2. Software and Tools

In our pursuit of exploring the scope of our project, we rely on a combination of classical and quantum software tools. At the forefront of our quantum endeavors, we have selected Qiskit as our primary quantum framework. Qiskit, developed by IBM, is a comprehensive and open-source software development kit for quantum computing.

Qiskit provides a versatile set of tools, libraries, and APIs that facilitate quantum circuit design, execution, and optimization. It offers a range of quantum programming fatures. Qiskit's ease of use and extensive documentation make it an ideal choice for quantum algorithm development.

2.1 Classical Machine Learning Libraries

While quantum computing plays a pivotal role in our project, we also employ classical machine learning libraries to support data preprocessing, feature engineering, and evaluation of quantum machine learning models. Libraries such as scikit-learn, TensorFlow, and PyTorch are invaluable tools in our toolkit.

Scikit-learn provides a rich set of machine learning algorithms, including support vector machines and neural networks, which can be used for classical baseline models and data preprocessing tasks. TensorFlow and PyTorch are widely recognized deep learning frameworks that allow us to build and train classical neural networks for comparison with their quantum counterparts.

These classical machine learning libraries complement our quantum efforts, enabling us to assess the effectiveness of quantum algorithms against well-established classical approaches.

2.2 Additional Tools

In addition to Qiskit and classical machine learning libraries, we utilize a range of data analysis and visualization tools. Jupyter notebooks serve as our primary environment for code development, experimentation, and documentation, allowing us to seamlessly integrate quantum and classical components

of our project. We also employ data visualization libraries such as Matplotlib and Seaborn to create informative plots and visual representations of our results.

Overall, the combination of Qiskit, classical machine learning libraries, and data analysis tools equips us with a robust software stack to explore and evaluate Quantum Algorithms for Image Classification effectively.

3. Work Calendarization

In this section, we introduce the Gantt chart, a visual representation of the project timeline and task scheduling. The chart outlines the start and end dates for various project tasks, along with their respective durations. This detailed calendarization in Figure 1 serves as a roadmap for the project's progress from September 2023 to February 2024.

The tasks have been carefully organized to encompass activities related to quantum computing, machine learning, and optimization. This calendarization ensures that project milestones are met in a systematic and efficient manner, facilitating effective project management and tracking of progress.

4. Initial Tests

Our initial tests were based on the Qiskit Machine Learning tutorials website. In this section, we introduce a summary of each of the tutorials we have tested and learned during these first weeks.

4.1 Quantum Neural Networks

Classical neural networks are a model inspired by the human brain and consist of interconnected nodes organized in layers. These networks learn by adjusting parameters through techniques like machine learning or deep learning. Quantum Machine Learning (QML) aims to combine classical neural networks with quantum circuits. This process can be divided into three main components:

1. **Data Loading:** Data is transformed into a quantum format using a feature map. This step converts classical data into quantum data suitable for analysis.
2. **Data Processing:** An Ansatz, which is a quantum circuit, is used to recompute weights at different stages of training. It plays a crucial role in the learning process.
3. **Measurement:** The final measurement, or its classical equivalent, corresponds to the output neurons in a classical neural network.

In addition to this, there are two key components to consider: **EstimatorQNN**, which focuses on evaluating quantum observables, and **SamplerQNN**, which is based on collecting samples from measurements of quantum circuits. These tools enable the configuration of a QNN with capabilities for both the forward and backward pass, along with gradient computation.

4.2 Neural Network Classifier and Regressor

In this section, we explore the utilization of neural network classifiers and regressors in a quantum context. The following topics are addressed:

4.2.1 Classification

1. **Classification with EstimatorQNN:** Employ an EstimatorQNN for binary classification within a NeuralNetworkClassifier. We provide an illustrative example of a quantum circuit configuration tailored for this purpose.
2. **Classification with SamplerQNN:** Utilization of a SamplerQNN for classification within a NeuralNetworkClassifier. The SamplerQNN yields multidimensional probability vectors, facilitating classification into multiple classes.
3. **Variational Quantum Classifier (VQC):** A special variant of the NeuralNetworkClassifier that leverages a SamplerQNN for classification. Cross-entropy loss is employed for training.

4.2.2 Regression

1. **Regression with EstimatorQNN:** Regression using an EstimatorQNN, which yields continuous values instead of probabilities.
2. **Variational Quantum Regressor (VQR)** Similar to VQC.

4.3 Training a Quantum Model on a Real Dataset

We explored training classical and quantum machine learning models using the Iris flower dataset. We start by loading the Iris dataset, performing data exploration, and normalizing the data.

For classical modeling, we train a Support Vector Machine (SVC) model on the dataset, and it demonstrates good performance.

In the quantum realm, we introduce a Variational Quantum Classifier (VQC) for classification tasks. We train and adapt the VQC to the Iris dataset.

Additionally, we experiment with feature reduction techniques, specifically Principal Component Analysis (PCA), to reduce the number of features from four to two.

We also compare the performance of classical and quantum models using reduced features.

In conclusion, the classical model, particularly the Support Vector Machine (SVC), achieves the highest accuracy. While the Variational Quantum Classifier (VQC) shows promise, it does not surpass the classical model's accuracy. The choice of feature reduction and the type of ansatz in the VQC impact its performance. In our tutorial, the VQC with the EfficientSU2 ansatz outperforms the VQC with RealAmplitudes in the two-feature configuration.

Overall, classical models remain a solid and mature choice for various classification tasks, even when compared to quantum models. The decision between classical and quantum models should consider specific problem requirements, available resources, and dataset complexity.

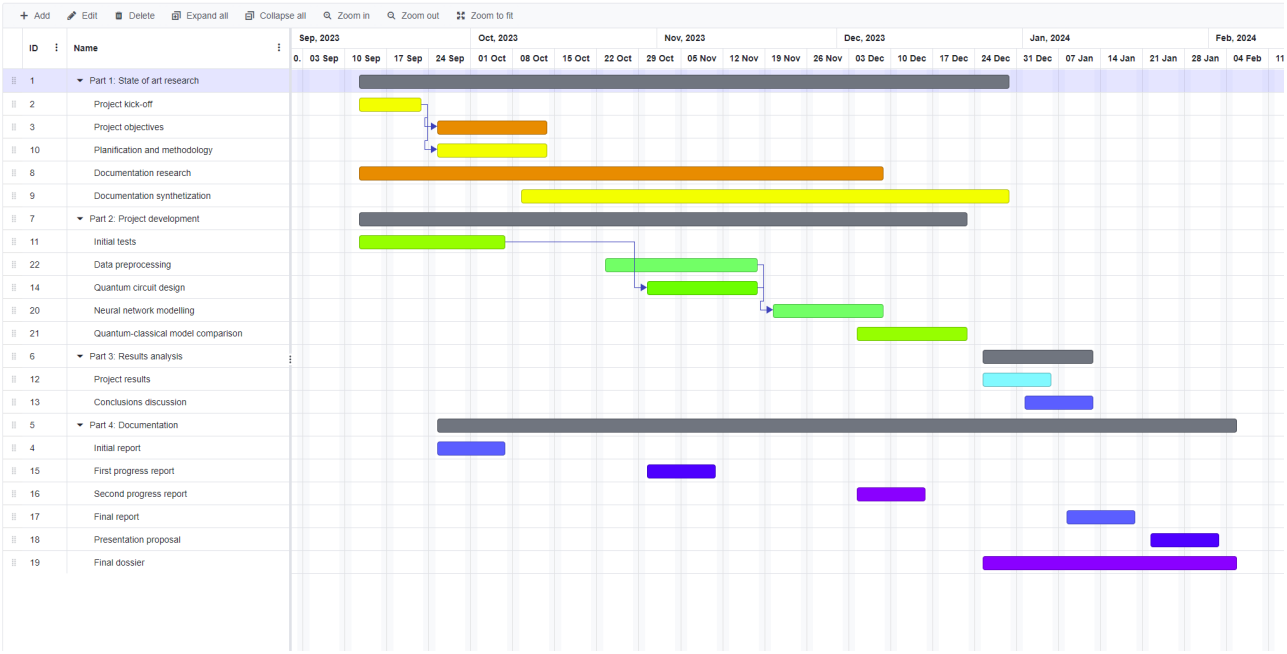
4.4 Quantum Kernel Machine Learning

This section introduces the concept of kernel methods and their application in supervised learning algorithms such as Support Vector Machines (SVM), as seen in [7]. We explore how kernels can map data to a higher-dimensional space and then discuss the following key points:

We delve into the mathematical foundations of kernel functions and elucidate how they are employed to map data into a high-dimensional feature space. The concept of quantum kernels is presented, showcasing their use of quantum mappings to execute the "kernel trick" in machine learning.

We illustrate how quantum kernels can be utilized in combination with SVM and the QSVC algorithm for classification

FIGURE 1. Gantt chart calendarization



* This Gantt chart is presented for the purpose of project calendarization, illustrating the timeline and scheduling of tasks and activities within the project.

tasks. We also provide an overview of how quantum kernels can be employed for data clustering, including the utilization of spectral clustering algorithms.

A demonstration how quantum kernel-based Principal Component Analysis (PCA) can transform a dataset, simplifying the classification process is also studied.

In summary, this section highlights the significance of kernel methods in quantum machine learning, offering insights into their applications for classification, clustering, and dimensionality reduction in a quantum context.

4.5 PyTorch qGAN Implementation

A Quantum Generative Adversarial Network (qGAN) is a hybrid algorithm that combines a quantum generator (a parameterized quantum circuit) and a classical discriminator (a neural network) to learn the underlying probability distribution of a training dataset. Both components are trained in alternating steps to enable the generator to learn to produce probabilities similar to those of the real dataset. This section is based on [8].

In this section, two key neural networks are defined:

- A quantum generator, implemented as a parametrized quantum circuit (ansatz).
- A classical discriminator, implemented as a PyTorch neural network.

There is an explanation on the configuration of the training loop, loss functions, optimizers, and providing a visualization function for both the generator and discriminator training processes.

The training process is detailed, performed through a loop where the generator and discriminator are trained in alternating steps. This process occurs for a specific number of epochs.

Finally, we were introduced to Cumulative Density Functions (CDF) as a visualization tool to interpret the results and assess the performance of the qGAN model.

In summary, this section provides an overview of Quantum Generative Adversarial Networks (qGANs), their components, training setup, and the training process, along with the use of Cumulative Density Functions for result visualization.

4.6 Torch Connector and Hybrid QNNs

In this section, we introduce the TorchConnector class in Qiskit and demonstrate how it enables a seamless integration of any Quantum Neural Network (QNN) from Qiskit Machine Learning into a PyTorch workflow. The TorchConnector takes a Qiskit Quantum Neural Network and makes it available as a PyTorch module. The resulting module can be effortlessly incorporated into classical PyTorch architectures and trained jointly without additional considerations. This allows for the development and testing of new hybrid quantum-classical machine learning architectures.

We address two types of problems, one involving simple classification and regression, utilizing the automatic differentiation engine of PyTorch (torch.autograd). Specifically, we solve two classifications: one with PyTorch and EstimatorQNN and another with PyTorch and SamplerQNN. Additionally, we perform regression using PyTorch and SamplerQNN.

We delve into more complex problem types, including classification with the MNIST dataset and hybrid QNN architectures.

In summary, TorchConnector streamlines the integration of Qiskit Quantum Neural Networks into PyTorch workflows, enabling the combined utilization of the advantages of both technologies. It facilitates classification and regression tasks with quantum neural networks, followed by their seamless incorporation into broader PyTorch models. Furthermore, it allows for the joint training of hybrid quantum-classical models, which can prove beneficial for specific machine learning applications.

4.7 Pegasos Quantum Support Vector Machines

In the early stages of our project, we have acquired valuable insights into the implementation of a Quantum Support Vector Classifier (QSVC). However, it is important to note that QSVC is not the sole algorithm employing Quantum Kernels.

Another noteworthy algorithm within our study is the Pegasos Quantum Support Vector Classifier.[9]

The Pegasos Quantum Support Vector Classifier serves as an alternative to the dual optimization approach found in the scikit-learn library. This algorithm harnesses the power of the kernel trick, offering the advantage of training complexity that remains independent of the size of the training dataset. Notably, it is expected to outperform the conventional QSVC when dealing with large training datasets. Moreover, it can be employed as a viable replacement for QSVC through appropriate hyperparameterization.

The implementation of PegasosQSVC seamlessly integrates with scikit-learn interfaces and adheres to a conventional training methodology. In the constructor, we specify crucial algorithmic parameters, including the regularization hyperparameter denoted as C, along with various other steps required for optimization. Subsequently, the training process involves feeding the algorithm with training features and labels through the 'fit' method. This method not only trains the model but also returns a fitted classifier, ready for evaluation. Finally, we assess the model's performance by scoring it against test features and labels.

This robust framework not only enhances the versatility of projects but also streamlines the implementation of quantum support vector classifiers, paving the way for efficient and scalable solutions.

4.8 Quantum Kernel Alignment

Another technique we have learned is Quantum Kernel Alignment (QKA), which iteratively adapts a parameterized quantum kernel to a dataset while converging to the maximum SVM margin. We learned how to train using the Quantum Kernel Alignment technique by selecting the kernel loss function as input to the QuantumKernelTrainer from Qiskit.[10]

4.9 Saving and loading models

We also acquired the knowledge of saving and loading Qiskit machine learning models. The ability to save a model is of paramount importance, especially when a significant amount of time is invested in training a model on real hardware. Additionally, we learned how to resume training of a previously saved model.

4.10 Neural Networks Effective Dimension

Both classical and quantum machine learning models share a common goal: being adept at generalizing, i.e., learning insights from data and applying them to unseen data. Finding a suitable metric to assess this ability is a non-trivial endeavor. The global effective dimension serves as a valuable indicator of how well a particular model can perform on new data.[11] In addition, the local effective dimension is introduced as a novel capacity measure that constrains the generalization error of machine learning models.[12]

The key distinction between global (EffectiveDimension class) and local effective dimension (LocalEffectiveDimension class) lies not in how they are computed, but in the nature of the parameter space they analyze. The global effective dimension encompasses the entire parameter space of the model and is calculated from a multitude of parameter (weight) sets. Conversely, the local effective dimension centers on how well the trained model can generalize to new data and how expressive it can be. As a result, the local effective dimension is computed from a single set of weight samples, typically obtained during training. While this distinction may seem subtle from a practical implementation perspective, it holds considerable conceptual significance.

4.11 Quantum Convolutional Neural Network

We also embarked on the journey of implementing a Quantum Convolutional Neural Network (QCNN) using Qiskit. In this endeavor, we modeled both the convolutional layers and pooling layers using quantum circuits. Our goal was to create a QCNN capable of differentiating horizontal and vertical lines from a pixelated image.[13]

QCNNs exhibit behavior analogous to classical Convolutional Neural Networks (CNNs). Initially, we encode our pixelated image into a quantum circuit using a designated feature map, such as Qiskit's ZFeatureMap or ZZFeatureMap, among others available in the circuit library.

Once the image is encoded, we introduce alternating convolutional and pooling layers, as elaborated upon in the following section. The application of these alternating layers serves to reduce the dimensionality of our circuit until we are left with a single qubit. Subsequently, we can classify our input image by measuring the output of this sole remaining qubit.

The Quantum Convolutional Layer consists of a series of two-qubit unitary operators, which recognize and establish relationships between the qubits within our circuit.

In the context of the Quantum Pooling Layer, we cannot employ the same dimension-reduction techniques as used classically to decrease the number of qubits in our circuit. Instead, we achieve dimensionality reduction by performing operations on each qubit until reaching a specific point, at which we disregard certain qubits in a particular layer. These layers, where we cease operations on specific qubits, are referred to as our 'pooling layers'.

Notably, within the QCNN, each layer comprises parameterized circuits. This implies that we can modify our output result by adjusting the parameters associated with each layer. During the training of our QCNN, these parameters are tuned to minimize the loss function, thereby enhancing the model's performance.

4.12 Quantum Autoencoders

Our journey of exploration also led us to the fascinating realm of Quantum Autoencoders. These are circuits designed to compress a quantum state onto a smaller number of qubits while retaining essential information from the original state. We delved into the architecture of a Quantum Autoencoder, comprehending how to design and train such a system effectively for information compression and encoding.[14]

Much like their classical counterparts, Quantum Autoencoders aim to reduce the dimensionality of the input to the neural network, specifically dealing with a quantum state.

Quantum Autoencoders have versatile applications, including but not limited to the following:

- **Digital Compression**: These can encode information into a smaller number of qubits, which proves highly advantageous for near-term quantum devices. Smaller qubit systems are less susceptible to noise, enhancing the robustness of quantum computations.
- **Denoising**: Quantum Autoencoders excel at extracting relevant features from initial quantum states or encoded data while effectively filtering out unwanted noise, contributing to data quality improvement.

- **Quantum Chemistry**: In the domain of quantum chemistry, Quantum Autoencoders serve as a valuable ansatz for various systems, such as the Hubbard Model. This model is frequently employed to describe electron-electron interactions in molecules, shedding light on complex chemical phenomena.

Our exploration of Quantum Autoencoders unveiled their potential and versatility in quantum computing applications.

5. Conclusion

In this initial report, we have outlined the objectives, methodologies, and early findings of our project on Quantum Machine Learning on image classification. Our investigation into Quantum Graph Neural Networks (QGNNs) and related quantum machine learning techniques has revealed several key insights:

1. **Potential for Quantum Enhancement**: Quantum algorithms hold immense potential for enhancing image classification tasks. Leveraging quantum states and operations, these models can potentially outperform classical counterparts in terms of accuracy and efficiency.
2. **Hybrid Approaches**: Combining quantum and classical machine learning techniques, as demonstrated in our project, can lead to synergistic improvements in performance. Quantum components can enhance the capabilities of classical models, especially in tasks involving large and complex datasets.
3. **Versatile Quantum Tools**: Our exploration of quantum tools and libraries, such as Qiskit, has demonstrated their versatility and accessibility. Quantum computing is becoming increasingly accessible for researchers and developers, paving the way for more innovative applications.
4. **Future Directions**: As we progress with our project, we anticipate further advancements in Quantum Image Classification. This includes fine-tuning Quantum Neural Network architectures, exploring novel quantum kernels, and enhancing the interpretability of quantum models.
5. **Knowledge Dissemination**: We emphasize the importance of sharing our findings and insights with the scientific community and stakeholders. Knowledge dissemination is crucial for fostering collaboration and driving innovation in the field.

In conclusion, our project represents an exciting journey into the intersection of quantum computing and image classification. As quantum technologies continue to evolve and mature, we are optimistic about the transformative potential of Quantum Graph Neural Networks and their applications in solving real-world image classification problems. This initial report serves as a foundation for our ongoing research and development efforts in this promising field.

CODE AVAILABILITY

The full code used during all these first steps of the project can be found on our GitHub public repository (https://github.com/adriend1102/QML).

References

[1] Skolik, Andrea, Michele Cattelan, Sheir Yarkoni, Thomas Bäck, and Vedran Dunjko: *Equivariant quantum circuits for learning on weighted graphs*. npj Quantum Information 2023 9:1, 9:1–15, May 2023, ISSN 2056-6387. https://www.nature.com/articles/s41534-023-00710-y.

[2] Verdon, Guillaume, Trevor Mccourt, Enxhell Luzhnica, Vikash Singh, Stefan Leichenauer, and Jack Hidary: *Quantum graph neural networks*.

[3] Zheng, Jin, Qing Gao, and Yanxuan Lu: *Quantum graph convolutional neural networks*. Chinese Control Conference, CCC, 2021-July:6335–6340, July 2021, ISSN 21612927. https://arxiv.org/abs/2107.03257v1.

[4] Innan, Nouhaila, Abhishek Sawaika, Ashim Dhor, Siddhant Dutta, Sairupa Thota, Husayn Gokal, Nandan Patel, Muhammad Al Zafar Khan, Ioannis Theodonis, and Mohamed Bennai: *Financial fraud detection using quantum graph neural networks*. September 2023. https://arxiv.org/abs/2309.01127v1.

[5] Zhou, Kaixiong, Zhenyu Zhang, Shengyuan Chen, Tianlong Chen, Xiao Huang, Zhangyang Wang, and Xia Hu: *Quangcn: Noise-adaptive training for robust quantum graph convolutional networks*. November 2022. https://arxiv.org/abs/2211.07379v1.

[6] Aktar, Shamminuj, Andreas Bärtschi, Abdel Hameed A. Badawy, Diane Oyen, and Stephan Eidenbenz: *Predicting expressibility of parameterized quantum circuits using graph neural network*. September 2023. https://arxiv.org/abs/2309.06975v1.

[7] Havlicek, Vojtech, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta: *Supervised learning with quantum enhanced feature spaces*. 2018.

[8] Zoufal, Christa, Aurélien Lucchi, and Stefan Woerner: *Quantum generative adversarial networks for learning and loading random distributions*. npj Quantum Information 2019 5:1, 5:1–9, November 2019, ISSN 2056-6387. https://www.nature.com/articles/s41534-019-0223-2.

[9] Shalev-Shwartz, Shai, Yoram Singer, Nathan Srebro, and Andrew Cotter: *Pegasos: Primal estimated sub-gradient solver for svm*. http://www.kernel-machines.org.

[10] Glick, Jennifer R., Tanvi P. Gujarati, Antonio D. Corcoles, Youngseok Kim, Abhinav Kandala, Jay M. Gambetta, and Kristan Temme: *Covariant quantum kernels for data with group structure*. May 2021. https://arxiv.org/abs/2105.03406v2.

[11] Abbas, Amira, David Sutter, Christa Zoufal, Aurelien Lucchi, Alessio Figalli, and Stefan Woerner: *The power of quantum neural networks*. 2020.

[12] Abbas, Amira, David Sutter, Alessio Figalli, and Stefan Woerner: *Effective dimension of machine learning models*. December 2021. https://arxiv.org/abs/2112.04807v1.

[13] Cong, Iris, Soonwon Choi, and Mikhail D. Lukin: *Quantum convolutional neural networks*. Nature Physics 2019 15:12, 15:1273–1278, August 2019, ISSN 1745-2481. https://www.nature.com/articles/s41567-019-0648-8.

[14] Romero, Jonathan, Jonathan P. Olson, and Alan Aspuru-Guzik: *Quantum autoencoders for efficient compression of quantum data*. Quantum Science and Technology, 2, December 2016. http://arxiv.org/abs/1612.02806http://dx.doi.org/10.1088/2058-9565/aa8072.