NISQAI: An Open-Source Library for Artificial Intelligence on NISQ Computers

Ryan LaRose,[1, 2] Yousif Almulla,[3] Nic Ezzell,[4] Joseph Iosue,[5] Arkin Tikku[6]

[1]*Department of Computational Mathematics, Science, and Engineering, Michigan State University, East Lansing, MI 48823, USA.*
[2]*Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48823, USA.*
[3]*Mathematical Institute, University of Oxford, 24-29 St. Giles, OX1 3LB, United Kingdom*
[4]*Department of Physics and Astronomy, Mississippi State University, Mississippi State, MS 39759, USA.*
[5]*Department of Physics, Massachusetts Institute of Technology, Cambridge, MA, 02139*
[6]*Department of Physics, Blackett Laboratory, Imperial College London, Prince Consort Road, London SW7 2AZ, United Kingdom*

As demonstrated by recent successes in chemistry (VQE), optimization (QAOA), and quantum compiling (QAQC), variational algorithms are emerging as a leading approach to practical implementations on NISQ computers. Quantum-assisted machine learning is an exciting but conjectural field that lacks experimentation on current hardware. We propose to research quantum neural networks (QNNs) by using a novel variational approach to enable deep learning with short-depth circuits and non-linearity between layers. Our open-source Python/Matlab code will interface with NISQ computers as an enabling framework for quantum computing and machine learning researchers to study QNNs.

CONTENTS

I. INTRODUCTION

Quantum hardware is steadily improving to meet the demands of quantum algorithms. However, it may be many years until we reach the realm of fault-tolerant quantum computing. In the meantime, quantum software is adapting to meet the constraints of noisy intermediate-scale quantum (NISQ) computers [1]. An example of such adaptation is the class of algorithms known as variational quantum algorithms [2].

Variational quantum algorithms work by by sending a parameterized unitary $U(\boldsymbol{\theta})$ from a classical computer to a quantum computer (see Fig. 1). The quantum computer executes the algorithm described by the unitary $U(\boldsymbol{\theta})$, and the classical computer samples from the output distribution $\{(z_i, p_i)\}_{i=1}^N$ of bit strings z_i which occur with frequency (probability) p_i. Based on the output recorded by the classical computer, the parameters $\boldsymbol{\theta}$ of the unitary $U(\boldsymbol{\theta})$ are then varied such as to move the sampled output toward the desired output. This iterative process repeats in a feedback loop until the desired accuracy is reached.

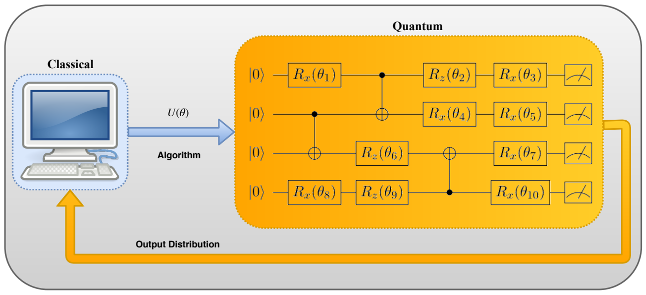

FIG. 1. The general structure of a variational quantum algorithm. A classical computer sends a quantum algorithm with parameters $\boldsymbol{\theta}$ to a quantum computer, which evaluates the algorithm. The output is then sampled by the classical computer. Based on the resulting output, the parameters $\boldsymbol{\theta}$ are varied. This process iterates until the desired output is achieved to within some specified tolerance.

Variational algorithms are useful for at least two purposes. First, they overcome the difficulty of quantum algorithm design by classically-trained minds, which is a significant stumbling block for even the best quantum algorithm engineers. Second, they are well-suited to fit the constraints of NISQ computers. NISQ constraints include limited qubit connectivity, low qubit coherence, and high noise levels in gate applications and measurements.

Despite these hardware constraints, variational algorithms have shown success in several application areas, (see Fig. 2). Although the success is celebrated for only small problem sizes, these variational algorithms are using NISQ computers to solve problems of practical importance. The notion of quantum software adapting to quantum hardware is critical to the success of quantum computing.

Application Area	Variational Algorithm
Chemistry	VQE [3]
Optimization	QAOA [4]
Error Correction	qVECTOR [5]
Autoencoding	QVAE [6]
Quantum Compiling	QAQC [7]
Factoring	VQF [8]

FIG. 2. Table showing the success of varational quantum algorithms in several application areas.

One area of quantum computing that needs significant adapting to NISQ hardware is quantum machine learning (QML). QML is a speculative field, and most algorithms developed for it require significant computational resources (e.g., eigenvalue solvers via phase estimation [9]) or hypothetical hardware constructs like quantum random access memory (qRAM) [10]. Nevertheless, the area of quantum machine learning, or more generally quantum artificial intelligence, is one that needs to be explored and rigorously studied.

In the rest of the paper, we propose a variational approach to implementing quantum neural networks on NISQ computers. We first briefly review the state of quantum machine learning in Section II to highlight the need for experimental testing of QML algorithms. We then present our open-source software proposal NISQAI in Section III to meet this need. We discuss our novel approach to variational quantum neural networks in Section III A, crucial steps in the development of NISQAI in Section III B, and our methodology for software development in Section III C. We discuss the novelty and necessity of our approach in Section III D, and we touch on current and future research areas crucial to the success of NISQAI in Section III E. We conclude in Section IV and briefly present developer backgrounds in Section V.

II. MACHINE LEARNING ON QUANTUM COMPUTERS

In this section we highlight the status of the field of quantum machine learning (QML) as well as recent advancements in quantum neural networks (QNNs).

A recent survey of quantum machine learning [11] reviews several algorithms in QML. The HHL algorithm [12] provides an efficient quantum approach to solving linear systems of equations, a problem that appears in many ML subroutines. Indeed, many basic linear algebra subroutines (BLAS) have been studied in the quantum context (qBLAS) [13]. Recently, several of the more standard ML algorithms have "been quantized." These include including quantum principal component analysis (qPCA) [14], quantum support vector machines (qSVM) [15], and quantum topological data analysis [16]. QML has even been explored in the context of kernel learning and feature-Hilbert spaces [17, 18]. The popularity and recent successes of deep learning [19, 20] have moved recent research toward neural network applications.

Many of the algorithms in QML are either purely theoretical or meant for future (fault-tolerant) quantum computers. Indeed, almost all QML algorithms contain "fine print" [21] that prevents practical implementations. For example, several QML algorithms are only feasible with significant restrictions on the input. Additionally, it is debated whether or not qRAM is physically realizable; hence the many "ground-breaking" algorithms based on qRAM are in question [22]. Regardless, it is clear that qRAM is not suitable for NISQ machines.

These "fine print" messages contribute largely to the lack of experimental testing of QML algorithms. With very few exceptions, none have been implemented on quantum hardware, and very few have been implemented on quantum simulators [23]. The table in Fig. 3 highlights the lack of implementation of these algorithms.

Algorithm	Simulator?	Hardware?
qSVM	✗	✗
qPCA	✓ [24]	✗ [24]
HHL	✗	✗
K-Means	✗	✗
QNN's	✓ [25]	✗

FIG. 3. Table showing the lack of testing of quantum machine learning algorithms. In the first column, the lowercase "q" in acronyms is short for "quantum." Other acronyms are standard in machine learning and/or quantum computing: specifically, PCA = principal component analysis, SVM = support vector machine, HHL = Harrow, Hasidim, & Lloyd's eponymous algorithm, and QNN = quantum neural network. The next two columns indicate if the algorithm has been tested on a quantum computer simulator (second column) or real quantum hardware (third column). Unless otherwise noted in the table, information on implementations is taken from [23].

While it is useful and appropriate to develop quantum algorithms for idealistic machines, it is more pressing to develop quantum algorithms for near-term machines. Software and hardware development should guide and complement one another. Developing software for NISQ hardware is not just useful for practical demonstrations, but also for finding new theoretical avenues. In the words of John Preskill, "The history of classical computing teaches us [...] [t]here are many examples of heuristics that were discovered experimentally, which worked better than theorists could initially explain" [1].

III. NISQAI

The main goal of NISQAI (Noisy Intermediate-Scale Quantum Artificial Intelligence) is to develop and implement practical machine learning algorithms for current

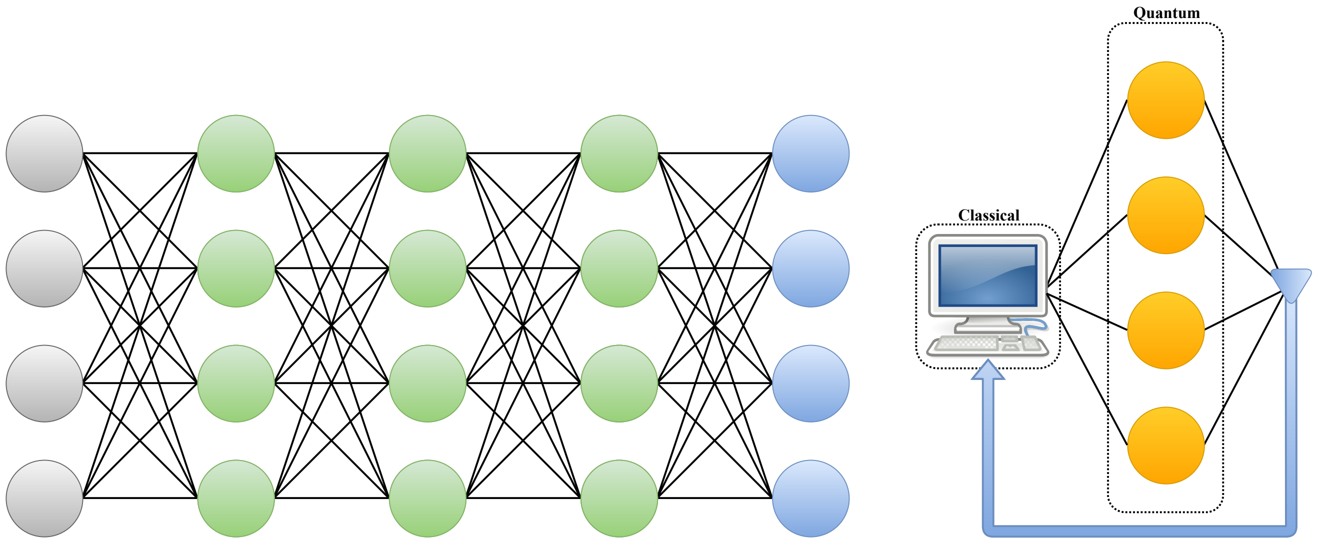

FIG. 4. LEFT: A schematic diagram of a deep neural network. Circles represent neurons and lines between nodes represent connections between neurons. The input layer is represented by gray nodes, hidden layers are represented by green nodes, and the output layer is represented by blue nodes. RIGHT: A schematic diagram of NISQAI's novel approach to implementing deep neural networks on NISQ computers. To overcome issues of limited circuit depth and inability to implement nonlinearity between layers, we implement each layer of the neural net successively in a variational manner. Input data is sent to the quantum computer from the classical computer, and the layer is evaluated on the quantum computer. The output of the quantum circuit is then sampled and processed by the classical computer, which sends this data as input to the next layer.

and near-term quantum computers. Our initial direction and proposal for the Unitary Fund is to develop and implement quantum neural networks (QNNs). This implementation will be done on real quantum hardware using a novel variational approach developed by the authors. In what follows, we provide details on this approach.

A. Variational Quantum Neural Networks

The NISQAI library will make QNNs amenable to current and near-term quantum computers. Inspired by the success of variational quantum algorithms in several application areas (Fig. 2), we propose a hybrid quantum-classical variational approach to neural networks. A high-level diagram of this approach is shown in Fig. 4. Here, the classical computer (CPU) sends a parameterized algorithm to the quantum computer (QPU). The QPU evaluates *a single layer* of the neural network in the form of a parameterized quantum algorithm. The output of this layer is produced by measurements in the quantum circuit. This output is sent to the CPU, processed, and sent back to the QPU as input to the next layer of the network. This process repeats in an iterative fashion until all layers in the neural network have been evaluated.

Our approach has several significant benefits that could lead to practical QNN implementations. These benefits include:

1. Ability to implement non-linear activation functions between layers.
2. Short-depth circuits amenable to NISQ computers.
3. Ability to implement deep neural networks (i.e., neural networks with many layers) on NISQ computers.
4. A built-in method for training QNNs.

We now elaborate in more detail on these points. First, a fundamental difficulty of QNN's is implementing non-linearity between layers of the network. While this feature is crucial for classical neural network success [20], it is fundamentally unrealizable in quantum circuits because evolution is unitary [26]. However, our approach enables such non-linearity in two ways. First, it is enabled by measurements, an idea which is not unique to our proposal [27, 28]. Second, it is enabled by classical processing between layers, an idea which is unique to our proposal.

In addition to enabling non-linearity, restricting the quantum computer to evaluating a single layer of the neural network at a time also achieves short-depth circuits. This is a crucial aspect to any successful implementation on NISQ computers. For NISQAI, this makes results more accurate and QNN behavior more reliable.

Another aspect of our approach is that we can achieve deep neural networks without being limited by the constraint of short-depth circuits. This again results from evaluating a single layer at a time on the quantum computer. The depth of the neural network will still be limited by the difficulty of training the network over many layers. This is a consideration in the classical case as well, however, and is not unique to QNNs.

Lastly, we note that a means for training the neural network is built-in to our approach due to the hybrid quantum-classical variational design. For this, we can leverage the large body of research developed in the classical domain, e.g. [29, 30]. Additionally, there is a growing body of research on optimization for variational quantum algorithms, e.g. [31], that we can utilize as well.

We strongly feel that our approach to QNNs has high probability for success on current and near-term quantum hardware. However, we acknowledge that we have not solved every issue in practical QNN implementations. For example, we do not know the optimal structure for the quantum circuit in each layer. This point, along with other points of consideration discussed in Section III E, comprise the primary questions we seek to answer during our research/implementations. In the next two subsections, we highlight specific development sub-goals and software-methodologies of NISQAI.

B. Development Steps for NISQAI

In this section we present step-by-step goals that we seek to achieve in the development of NISQAI. Ideally, we would like NISQAI to be a hub for open-source software devoted to practical QML algorithms on NISQ computers. While we have initial ideas for future QML directions, we restrict our focus to the more reasonable goal of QNN development and implementation for this proposal. Along these lines, an outline of the steps we wish to achieve in our development is included below:

1. Successfully implement a QNN using the approach outlined in Fig. 4, first focusing on quantum simulator implementations. Address the following questions:
    (a) What is the optimal structure of the quantum circuit for implementing a neural network layer?
    (b) How many layers are necessary for learning?
    (c) At what circuit size/depth does the optimization problem become infeasible?
    (d) Can we see any demonstrable advantage to evaluating layers on a quantum computer?
2. Test the QNN developed on the simulator on current quantum hardware. Address the following questions:
    (a) How well does the quantum hardware perform relative to the quantum simulator?
    (b) Can we expect accurate learning when implementing on NISQ computers?
    (c) At what size quantum computer, if any, do we expect to see a quantum advantage?
3. Use the QNN to perform learning on canonical benchmarking problems for machine learning techniques (e.g., MNIST data) and comparatively evaluate the performance. Address the following:
    (a) How well does the QNN perform relative to standard classical methods?
4. Release the software to the quantum computing and machine learning communities to provide an open-source tool for further research and discovery.
    (a) Provide tutorials (Jupyter notebooks, etc.) of NISQAI capabilities and features.
    (b) (Possibly) provide a release paper.
    (c) Provide example cases of what we found was successful and what we found was unsuccessful.
    (d) Better understand the capabilities of quantum neural networks and quantum-assisted machine learning in general.

C. Software Methodology & Features of NISQAI

The growing trend of open-source software for quantum computing is critical to the success of the field. Open-source quantum software promotes reproducible results and expands the field by providing a free framework for any individual. In this section we highlight some of our core design principles for NISQAI to achieve these effects.

NISQAI will be free and open-source under the Apache 2 license. NISQAI will be developed in Python to interface with current quantum computers, most notably those of Rigetti. In addition, we will provide a Matlab interface for all of our Python code. Our motivation for this is that we wish to provide a tool for both quantum computing and machine learning researchers, and Matlab is a major language in the machine learning community.

Following the trends of quantum software developers [32–34], we would like to publish a release (arXiv) paper when NISQAI is released. In addition to increasing the visibility of NISQAI, this could help bring in developers to contribute to future versions of NISQAI. It could also be a step towards achieving our long-term goal of having NISQAI be a hub for near-term quantum artificial intelligence applications.

D. Why NISQAI is Different

There are now nearly one hundred open-source quantum software efforts [35]. In this section, we highlight the differences and advantages of NISQAI relative to other software and papers in the field of quantum neural networks.

Perhaps the first effort towards numerical experimentation of QNNs occurred in Ref. [36]. As this work is nearly twenty years old, no implementation was performed on real quantum hardware. The authors did find some numerical evidence of certain advantages in classically-simulated QNNs over classical neural networks, however.

Since this paper, much of the field of QML has turned away from QNNs and focused on theoretical algorithms for idealistic quantum computers, as highlighted in Section II. There have been a few recent exceptions, however, which we now discuss.

Reference [37] provides a comprehensive overview of the field of quantum artificial intelligence and provides several algorithms for quantum neural networks. These include "Baqprop," the quantum analogue of backpropagation, as well as other quantum algorithms for training neural networks. While this work is novel and comprehensive, almost none of the algorithms are amenable to near-term hardware. NISQAI separates itself from this work due to it's practical utility on current and near-term quantum computers.

Reference [25] studies QNNs via variational quantum algorithms similar to the proposal of NISQAI. This work performs numerical simulations on a quantum simulator to show that learning with QNNs is possible. In particular, they perform learning on the standard MNIST training set, an exciting and notable landmark for quantum artificial intelligence. However, this work is significantly restricted due the inability to implement nonlinear activation functions between neural network layers, a crucial feature to the success of classical neural networks [19]. This is because the entire QNN is evaluated as one quantum circuit. NISQAI will significantly build on the preliminary results of this study. Indeed, the approach taken by this reference can be seen as a subset of the approach of NISQAI with only one layer.

Reference [28] also explores QNN training and implementation. Similar to NISQAI, this work introduces nonlinearity via measurements in the quantum circuit. However, the implementations only measure one qubit at a time. This severely restricts the set of nonlinear functions that can be performed with this method. Additionally, deep neural networks are not feasible on near-term computers because the quantum computer evaluates *every* layer of the network with one circuit. While the authors indeed numerically study the circuit, this is done only through a quantum simulator and not a quantum computer. Lastly, the numerical study is restricted to "quantum data" only, not classical data. The novel approach of NISQAI can achieve all of these features.

In short, NISQAI separates itself from the majority of the field of QML due to its emphasis on near-term applications. While NISQAI shares some similarity with recent QNN papers, our approach is novel and will significantly build on these works. Another feature of NISQAI not contained in these works is the emphasis on providing an open-source tool for quantum computing and machine learning researchers.

As mentioned, open-source QML software packages exist. For example, the Quantum Machine Learning Toolkit (QMLT) is an excellent resource provided by researchers at Xanadu [38]. However, this software is centered around machine learning for quantum algorithms, not quantum algorithms for machine learning. NISQAI will provide a novel tool for the purpose of quantum algorithms for machine learning, namely quantum neural network implementations. This tool will be free, open-source, and available to both the quantum computing and machine learning communities. We will provide a platform on which practical QML can form a solid foundation.

Lastly, we remark that this section is meant to highlight the novel approach of NISQAI and explain why we think it will work. Our goal is not to compete with other researchers but rather to organize the collective study of quantum artificial intelligence into a unified open-source framework. The synthesis of ideas is largely beneficial in scientific endeavors, and our approach could combine with others. For instance, rather than implementing a single layer of the neural network on a quantum computer, multiple layers can be implemented in an approach similar to [28] by performing measurements on a subset of the qubits. Then, the subsequent output data can be processed classically, sent back to the quantum computer to process subsequent layers, and so on.

E. Research Areas

We believe that our unique approach in NISQAI will be successful for QNN implementations for the reasons listed in Section III B. However, as mentioned in this same section, we acknowledge that there are several further points to research. In what follows, we enumerate these and elaborate on the challenges as well as our initial ideas.

1. *The input problem.*

    - The problem of inputting data into a quantum computer appears in many areas, especially quantum machine learning. Indeed, the idea of qRAM was designed as an oracle to supersede this issue for classical data. Since NISQAI evaluates a single layer at a time on the quantum computer, the data from the previous layer needs to be input to the next layer at every step?

    - It is easy to input a bit string $\boldsymbol{z} = z_1 \cdots z_n \in \{0,1\}^n$ by performing the circuit $X^{\boldsymbol{z}} = X_1^{z_1} \cdots X_n^{Z_n}$. Will this be sufficient to characterize the output of the previous layer? Or will we need a complete description of the wavefunction to see any quantum advantage.

    - More sophisticated and universal circuits for state preparation can be found in Refs. [39, 40].

2. *The output problem.*

    - Associated to the input problem is the output problem. The exponential size of quantum state vectors poses an issue when it comes to reading out classical information from a quantum computation. In general, an exponentially large number of samples is required in order to obtain the output distribution of a quantum circuit with arbitrary precision. In order for a quantum algorithm to be of any practical use, it is necessary for this problem to be circumvented.
    - Like state preparation, we also need to perform measurements at the end of every layer of the QNN. We can ask the same question for state preparation: is a complete description of the wavefunction needed to see a quantum advantage?

3. *The optimization/training problem.*

    - Variational algorithms at large depths necessarily introduce a hard optimization problem for classical computers. Optimization algorithms have been studied slightly in [2], but only for a particular instance of VQE. Recent results [41] show that gradient-based optimization strategies are futile for a large class of random circuit ansatz on non-trivial depths. A more recent numerical study [31] benchmarks several classical optimizers–namely Nelder-Mead, quasi-Newton, and analytical gradient–on QAOA-type circuits.
    - The same problems in variational algorithm optimization occur analogously in neural network training [30]. We can draw inspiration from these resources in our work.
    - A numerical study of the best classical optimization/training method is likely required in our development of NISQAI.

4. *The circuit for each layer.*

    - Universal quantum circuits are known for any given number of qubits (e.g., [42]), but these generally take a number of gates that scales exponentially in the number of qubits.
    - Recent variational algorithm implementations have seen success by enforcing an ansatz on the structure of the unitary comprising the circuit [3–8] and then optimizing over internal gate parameters.
    - Does the circuit implementing each layer need to be a universal quantum circuit (i.e., one that can realize an arbitrary unitary)? There likely exists suitable ansatz for NISQ devices that is tailored towards evaluating a single layer in a QNN. What is the form of this ansatz?

5. *How deep is deep?*

    - How many layers will be required to learn a given task of interest?
    - This point will likely vary depending on each training set we consider in the development of NISQAI.

6. *The advantage problem.*

    - Will quantum neural networks provide any computational or practical advantages over their classical counterparts?

The last question is of fundamental importance to all of quantum computing [43]. The development of NISQAI will help to answer it. We note that there are several avenues in which NISQAI could demonstrate quantum advantage. For example, NISQAI could demonstrate advantage in computational runtime, size of the network (height & width), or accuracy/generalized learning performance.

IV. CLOSING REMARKS

The existence of a quantum advantage for machine learning is an exciting yet elusive question [44]. We will provide an easy-to-use tool for QNN implementations that will help scientists develop intuition for the utility of QML. Such applications have helped shine light on the efficacy of certain algorithms in the past, such as the simplex algorithm. Although few theorists thought this could be successful [1], it proved enormously so.

Applications on NISQ devices will help launch the field of quantum machine learning into a practical domain. We strongly believe that artificial intelligence needs to be rigorously studied on NISQ computers. By implementing neural networks on current quantum technology, we allow for the possibility of discovering new algorithms and new uses for NISQ computers. NISQAI will bring new minds into the fields of quantum computing and artificial intelligence by bridging the domain expertise gap between them. Our work will investigate the pressing question of achieving quantum advantage in neural network applications.

V. DEVELOPERS

In this section, we briefly introduce the developers of NISQAI. **Ryan LaRose** is a doctoral student at Michigan State University pursuing dual PhD's in physics and computational mathematics, science, and engineering (CMSE). His dissertation research is centered around

machine learning for variational quantum algorithms as well as quantum-assisted machine learning. In the summer of 2018, he was a student fellowship guest at Los Alamos National Laboratory in the first-ever Quantum Computing Summer School.

Yousif Almulla holds a an Honors B.Sc. in Math and Physics from Oregon State University. He is presently pursuing a M.Sc. in Mathematics and the Foundations of Computer Science at the University of Oxford. In the summer of 2018, he was a student fellowship guest at Los Alamos National Laboratory in the first-ever Quantum Computing Summer School.

Nic Ezzell is an undergraduate student of physics and mathematics at Mississippi State University with a solid background in Python programming. His research interests focus on near-term applications of D-Wave's quantum annealing hardware. In the summer of 2018, he was an intern at Oak Ridge National Laboratory as a part of the Science Undergraduate Laboratory Internships (SULI) program.

Joseph Iosue is a rising senior physics major and computer science minor at MIT. He has a strong coding background and has performed research on novel variational quantum algorithms. In the summer of 2018, he was a student fellowship guest at Los Alamos National Laboratory as part of the first-ever Quantum Computing summer school.

Arkin Tikku is a recent physics graduate of Imperial College London and has worked on holographic quantum error correcting codes under Fernando Pastawski. In the summer of 2018, he worked on variational quantum algorithms as a student fellowship guest at Los Alamos National Laboratory as part of the first-ever Quantum Computing summer school.

Note: NISQAI has no affiliation with Los Alamos National Laboratory nor Oak Ridge National Laboratory. The ideas in this proposal are solely those of the authors and borrow no intellectual property from either institution.

[1] John Preskill. Quantum computing in the NISQ era and beyond. URL `http://arxiv.org/abs/1801.00862`.

[2] Jarrod R. McClean, Jonathan Romero, Ryan Babbush, and AlÃąn Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. 18(2):023023, . ISSN 1367-2630. doi:10.1088/1367-2630/18/2/023023. URL `http://arxiv.org/abs/1509.04279`.

[3] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, AlÃąn Aspuru-Guzik, and Jeremy L. O'Brien. A variational eigenvalue solver on a quantum processor. 5(1). ISSN 2041-1723. doi:10.1038/ncomms5213. URL `http://arxiv.org/abs/1304.3061`.

[4] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. URL `http://arxiv.org/abs/1411.4028`.

[5] Peter D. Johnson, Jonathan Romero, Jonathan Olson, Yudong Cao, and AlÃąn Aspuru-Guzik. QVECTOR: an algorithm for device-tailored quantum error correction. URL `http://arxiv.org/abs/1711.02249`.

[6] Amir Khoshaman, Walter Vinci, Brandon Denis, Evgeny Andriyash, and Mohammad H. Amin. Quantum variational autoencoder. *arXiv:1802.05779 [quant-ph, stat]*, Feb 2018. URL `http://arxiv.org/abs/1802.05779`. arXiv: 1802.05779.

[7] Sumeet Khatri, Ryan LaRose, Alexander Poremba, Lukasz Cincio, Andrew T. Sornborger, and Patrick J. Coles. Quantum assisted quantum compiling. URL `http://arxiv.org/abs/1807.00800`.

[8] Eric R. Anschuetz, Jonathan P. Olson, AlÃąn Aspuru-Guzik, and Yudong Cao. Variational quantum factoring. *arXiv:1808.08927 [quant-ph]*, Aug 2018. URL `http://arxiv.org/abs/1808.08927`. arXiv: 1808.08927.

[9] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 10th anniversary ed edition. ISBN 978-1-107-00217-3.

[10] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. 100(16). ISSN 0031-9007, 1079-7114. doi:10.1103/PhysRevLett.100.160501. URL `https://link.aps.org/doi/10.1103/PhysRevLett.100.160501`.

[11] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195âĂŞ202, Sep 2017. ISSN 0028-0836, 1476-4687. doi:10.1038/nature23474. arXiv: 1611.09347.

[12] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for solving linear systems of equations. *Physical Review Letters*, 103 (15), Oct 2009. ISSN 0031-9007, 1079-7114. doi:10.1103/PhysRevLett.103.150502. URL `http://arxiv.org/abs/0811.3171`. arXiv: 0811.3171.

[13] Fernando G. S. L. Brandao and Krysta Svore. Quantum speed-ups for semidefinite programming. *arXiv:1609.05537 [quant-ph]*, Sep 2016. URL `http://arxiv.org/abs/1609.05537`. arXiv: 1609.05537.

[14] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. 10(9):631–633. ISSN 1745-2473, 1745-2481. doi:10.1038/nphys3029. URL `http://arxiv.org/abs/1307.0401`.

[15] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. 113(13). ISSN 0031-9007, 1079-7114. doi:10.1103/PhysRevLett.113.130503. URL `http://arxiv.org/abs/1307.0471`.

[16] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. Quantum algorithms for topological and geometric analysis of big data. *arXiv:1408.3106 [quant-ph]*, Aug 2014. URL `http://arxiv.org/abs/1408.3106`. arXiv: 1408.3106.

[17] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. URL `http://arxiv.org/abs/1803.07128`.

[18] Vojtech Havlicek, Antonio D. CÃşrcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum enhanced feature spaces. *arXiv:1804.11326 [quant-ph, stat]*, Apr 2018. URL `http://arxiv.org/abs/1804.11326`. arXiv: 1804.11326.

[19] Juergen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85âĂŞ117, Jan 2015. ISSN 08936080. doi:10.1016/j.neunet.2014.09.003. arXiv: 1404.7828.

[20] Henry W. Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223âĂŞ1247, Sep 2017. ISSN 0022-4715, 1572-9613. doi:10.1007/s10955-017-1836-5. arXiv: 1608.08225.

[21] Scott Aaronson. Read the fine print. 11:3.

[22] Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-OâĂŹConnor, Michele Mosca, and Priyaa Varshinee Srinivasan. On the robustness of bucket brigade quantum ram. *New Journal of Physics*, 17(12):123010, Dec 2015. ISSN 1367-2630. doi: 10.1088/1367-2630/17/12/123010. arXiv: 1502.03450.

[23] Peter Wittek. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Academic Press. ISBN 978-0-12-800953-6.

[24] Patrick J. Coles, Stephan Eidenbenz, Scott Pakin, Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, Nathan Lemons, Shizeng Lin, Andrey Lokhov, Alexander Malyzhenkov, David Mascarenas, Susan Mniszewski, Balu Nadiga, Dan O'Malley, Diane Oyen, Lakshman Prasad, Randy Roberts, Phil Romero, Nandakishore Santhi, Nikolai Sinitsyn, Pieter Swart, Marc Vuffray, Jim Wendelberger, Boram Yoon, Richard Zamora, and Wei Zhu. Quantum algorithm implementations for beginners. URL `http://arxiv.org/abs/1804.03719`.

[25] Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors. *arXiv:1802.06002 [quant-ph]*, Feb 2018. URL `http://arxiv.org/abs/1802.06002`. arXiv: 1802.06002.

[26] Note1. Note that continuous variable quantum computers, such as those based on photonic architectures, can achieve such nonlinear activation functions, but here we restrict our discussion to qubit-based computers.

[27] M. Schuld, I. Sinayskiy, and F. Petruccione. The quest for a Quantum Neural Network. *Quantum Information Processing*, 13:2567, August 2014. doi:10.1007/s11128-014-0809-8.

[28] Hongxiang Chen, Leonard Wossnig, Simone Severini, Hartmut Neven, and Masoud Mohseni. Universal discriminative quantum neural networks. *arXiv:1805.08654 [quant-ph, stat]*, May 2018. URL `http://arxiv.org/abs/1805.08654`. arXiv: 1805.08654.

[29] Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Gradient-based hyperparameter optimization through reversible learning. *arXiv:1502.03492 [cs, stat]*, Feb 2015. URL `http://arxiv.org/abs/1502.03492`. arXiv: 1502.03492.

[30] Will Grathwohl, Elliot Creager, Seyed Kamyar Seyed Ghasemipour, and Richard Zemel. Gradient-based optimization of neural netowrk architecture. page 6, 2018.

[31] Gian Giacomo Guerreschi and Mikhail Smelyanskiy. Practical optimization for hybrid quantum-classical algorithms. *arXiv:1701.01450 [quant-ph]*, Jan 2017. URL `http://arxiv.org/abs/1701.01450`. arXiv: 1701.01450.

[32] Jarrod R. McClean, Ian D. Kivlichan, Kevin J. Sung, Damian S. Steiger, Yudong Cao, Chengyu Dai, E. Schuyler Fried, Craig Gidney, Brendan Gimby, Thomas HÃďner, Tarini Hardikar, VojtÄŻch HavlÃŋÄŊek, Cupjin Huang, Zhang Jiang, Matthew Neeley, Thomas O'Brien, Isil Ozfidan, Maxwell D. Radin, Jhonathan Romero, Nicholas Rubin, Nicolas P. D. Sawaya, Kanav Setia, Sukin Sim, Mark Steudtner, Wei Sun, Fang Zhang, and Ryan Babbush. OpenFermion: The electronic structure package for quantum computers. . URL `http://arxiv.org/abs/1710.07629`.

[33] Damian S. Steiger, Thomas HÃďner, and Matthias Troyer. ProjectQ: An open source software framework for quantum computing. 2:49. ISSN 2521-327X. doi:10.22331/q-2018-01-31-49. URL `http://arxiv.org/abs/1612.08091`.

[34] qiskit-core: Quantum information science kit for writing quantum computing experiments, programs, and applications. URL `https://github.com/QISKit/qiskit-core`. original-date: 2017-03-03T17:02:42Z.

[35] Note2. Mark Fingerhuth et al, Open-Source Quantum Software Projects, accessed September 5, 2018.

[36] Ajit Narayanan and Tammy Menneer. Quantum artificial neural network architectures and components. *Information Sciences*, page 25, 2000.

[37] Guillaume Verdon, Jason Pye, and Michael Broughton. A universal training algorithm for quantum deep learning. URL `http://arxiv.org/abs/1806.09729`.

[38] Note3. AS Christensen, FA Faber, B Huang, LA Bratholm, A Tkatchenko, KR Muller, OA von Lilienfeld (2017) "QML: A Python Toolkit for Quantum Machine Learning" https://github.com/qmlcode/qml.

[39] Martin Plesch and ÄŇaslav Brukner. Quantum-state preparation with universal gate decompositions. *Physical Review A*, 83(3), Mar 2011. ISSN 1050-2947, 1094-1622. doi:10.1103/PhysRevA.83.032302. URL `http://arxiv.org/abs/1003.5760`. arXiv: 1003.5760.

[40] Vivek V Shende, Stephen S Bullock, and Igor L Markov. Synthesis of quantum logic circuits. page 19.

[41] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. . URL `http://arxiv.org/abs/1803.11173`.

[42] Debajyoti Bera, Stephen Fenner, Frederic Green, and Steve Homer. *Efficient Universal Quantum Circuits*, volume 5609, page 418âĂŞ428. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-02881-6. doi:10.1007/978-3-642-02882-3_42. URL `http://link.springer.com/10.1007/978-3-642-02882-3_42`.

[43] Rigetti Computing. Introducing rigetti quantum cloud services, Sep 2018. URL `https://medium.com/rigetti/introducing-rigetti-quantum-cloud-services-c6005729768c`.

[44] E. Tang. A quantum-inspired classical algorithm for recommendation systems. *ArXiv e-prints*, July 2018.