(1) Introduction The answers to many important scientific questions rely on computer simulations. Galactic evolution, weather patterns, drug discovery, and many-body physics all require high-performance computational techniques. Current computers are fundamentally limited by physical restrictions on transistor sizes—Moore's law is failing. To answer critical questions at the forefront of science, new models of computation need to be developed.

Among several promising candidates for future computing technologies (probabilistic & neuromorphic computing, tensor & graphical processing units, etc.), quantum computers (QCs) are one of the most exciting. For certain problems, algorithms for QCs (quantum algorithms/QAs) can provide exponential speedups over the best algorithms for current computers (for example, the quantum Fourier transform). With future QCs, such algorithms will enable us to compute in hours what would currently take years. However, current QCs are noisy and have not yet demonstrated clear advantage over classical computers. To do this, we must redesign existing QAs to counteract the noise of current QCs. For the long-term, we must design new QAs to expand the problem set future QCs can efficiently solve.

I propose to use machine learning (ML) to help (re)design QAs. This exciting new idea has been explored in two seminal papers [1, 2] and my own research [3]. ML can redesign existing algorithms for current/near-term QCs and design new algorithms for future QCs.

(2) Preliminary work ML has been used to design QAs for mathematical functions, a result that provides important subroutines for other QAs [1]. ML has also been used to design a QA for computing quantum state overlap [2]. In collaboration with the authors of [2], I used this result to design a new QA for matrix diagonalization (pre-print at [4]).

Additionally, I used ML to redesign several known QAs for current QCs. This work has been published in a pre-print [3] that shows, e.g., the quantum Fourier transform (QFT) redesigned to the hardware constraints of "ibmqx4" (a current five-qubit QC). Our redesigned algorithm is more robust to noise/errors on ibmqx4 than the standard QFT algorithm.

(3) Proposed work I propose to expand on my prior work to use ML to redesign QAs for near-term QCs (see 3.1) and design new QAs for future QCs (see 3.2). A quantum algorithm $\mathcal{A}$ is defined by a set of N gates $\mathcal{A} = \{G_i(k_i, \theta_i)\}_{i=1}^N$ that describe a unitary matrix. Here, G_i is the ith gate that acts on qubits indexed by k_i, and θ_i denotes internal gate parameters.

(3.1) Algorithm redesign for near-term QCs I will use a "cost-driven" method to optimally redesign QAs for current/near-term QCs. Optimal algorithms need to contain as few gates as possible (due to noise), and only a limited set of gates can be implemented on current QCs. Despite this, current QCs have been used to compute the deuteron binding energy in nuclear physics, compute ground-state energies of small molecules in chemistry, and (a result of my own research [5]) learn decision boundaries for classification problems in data science. My proposed work will increase possible problem sizes (bigger nuclei/molecules/etc.) and reduce errors by redesigning QAs to have fewer gates and added "noise-adjusting" gates.

My proposed method works as follows. Let $\mathcal{U}$ be a known QA, $\mathcal{Q}$ a current/near-term QC, and $\mathcal{Q}_G$ the set of gates that can be implemented on $\mathcal{Q}$. The goal is to design a new algorithm, $\mathcal{A}$, that produces the same output as $\mathcal{U}$ but with fewer gates. We first initialize a guess for our trainable algorithm $\mathcal{A}$ with all gates G_i from $\mathcal{Q}_G$. We then evaluate the cost $C(\mathcal{A},\mathcal{U}) = ||\mathcal{A} - \mathcal{U}||$ for a given matrix norm $||\cdot||$. The parameters $\{G_i(\theta_i), k_i\}$ in $\mathcal{A}$ are then iteratively adjusted to minimize the cost. When the cost is exactly (approximately) zero, we learn an exact (approximate) algorithm. Using similar methods in [3], I have redesigned QAs with significantly fewer gates compared to standard *quantum compiling* methods, which only locally translate individual gates into $\mathcal{Q}_G$. Additionally, by training $\mathcal{A}$ in the presence of noise, our redesigned QAs add in gates to adjust for noise. Algorithms with fewer gates and noise-adjusting gates will enable more/larger problems to be solved on near-term QCs.

My proposed work will use this method to optimally redesign QAs for simulating fermionic systems on "ibmqx5" (a current sixteen-qubit QC). This work will enable more accurate and longer simulations of Hamiltonian evolution for many chemical systems. As QCs scale to larger sizes, my proposed method could revolutionize quantum chemistry and drug discovery.

(3.2) Algorithm design for future QCs I will use a "data-driven" method to design new QAs for future QCs. The goal is to design a new QA $\mathcal{A}$ for some function f, which could be anything from computing quantum state overlap to a mathematical operation like integration. At a low-level, however, all such functions f map bit-strings to bit-strings.

My proposed method works as follows. First, we pick a number of qubits n and evaluate f (using a classical computer) on bit-strings x_j. This provides training data $y_j = f(x_j)$, where $j = 1, ..., D$. We then train over the parameters $\{G_i(\theta_i), k_i\}$ in our algorithm $\mathcal{A}$ to minimize the cost $\sum_{j=1}^{D} ||y_j - \mathcal{A}(x_j)||^2$. Then, we repeat this process for multiple n. After, we analyze the learned algorithms to gain insight from the quantum gates that compute f on multiple problem sizes. Using pattern matching or motif recognition, we can deduce a new QA.

A similar approach was used in [2] to learn a new QA for computing state overlap. My proposed work will design new QAs for computing quantum entropies of quantum states. Such QAs, currently unknown, would be extremely useful for characterizing quantum states produced by, e.g., simulation algorithms on QCs. These functions fit well into my data-driven method because they can be computed classically & scale naturally to multiple problem sizes.

(4) Challenges In both methods, numerical optimization over the parameters $\{G_i(\theta_i), k_i)\}$ is challenging for large algorithm sizes. To overcome this, we can introduce an algorithm "ansatz," or structure, to eliminate the k_i and even G_i parameters. Additionally, for the cost-driven method, we can train over "sub-algorithms" (subsets of gates) to limit the large search space while still exploring a larger space than other quantum compiling methods.

For the data-driven method, deciding which gates to include in training is an open question. Further research is needed to identify the necessary number of training data points (i.e., the D above). Methods from active learning can significantly reduce D for large algorithm sizes.

(5) Intellectual merit The idea of ML for QA design has demonstrated initial success in seminal papers [1, 2] and my own research [3]. The cost-driven method searches a more expansive space than standard quantum compiling methods to produce more optimal and more robust QAs. This method could redesign any known QA for any given QC. Methods for designing QAs for future QCs do not exist. My proposed data-driven method could produce many new QAs with applications in many fields. The recent success of this approach in [2] necessitates further study. My knowledge of both ML and QAs is ideal for these methods.

(6) Broader impact This research will significantly increase what is possible with near-term QCs and produce new algorithms for future QCs. Both will improve our understanding of the power of QCs, which have the potential to answer some of the most important scientific questions of the 21st century. Powerful QAs could impact society outside of academia as well—e.g., efficient chemistry could discover new drugs that vastly improve human health.

References [1] Phys. Rev. A 98, 032309, [2] arXiv:1803.04114, [3] arXiv:1807.00800, [4] arXiv:1810.10506, [5] github.com/QuantumAI-lib/NISQAI. (Underlined works (co-)authored by me.)