Machine Learning of Noise-Resilient Quantum Circuits

Lukasz Cincio,[1] Kenneth Rudinger,[2] Mohan Sarovar,[3,*] and Patrick J. Coles[1,†]

[1] *Theoretical Division, MS 213, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA*
[2] *Quantum Computer Science, Sandia National Laboratories, Albuquerque, New Mexico 87185, USA*
[3] *Extreme-scale Data Science and Analytics, Sandia National Laboratories, Livermore, California 94550, USA*

(Received 15 July 2020; revised 30 December 2020; accepted 21 January 2021; published 15 February 2021)

Noise mitigation and reduction will be crucial for obtaining useful answers from near-term quantum computers. In this work, we present a general framework based on machine learning for reducing the impact of quantum hardware noise on quantum circuits. Our method, called noise-aware circuit learning (NACL), applies to circuits designed to compute a unitary transformation, prepare a set of quantum states, or estimate an observable of a many-qubit state. Given a task and a device model that captures information about the noise and connectivity of qubits in a device, NACL outputs an optimized circuit to accomplish this task in the presence of noise. It does so by minimizing a task-specific cost function over circuit depths and circuit structures. To demonstrate NACL, we construct circuits resilient to a fine-grained noise model derived from gate set tomography on a superconducting-circuit quantum device, for applications including quantum state overlap, quantum Fourier transform, and W-state preparation.

DOI: 10.1103/PRXQuantum.2.010324

I. INTRODUCTION

Recent years have seen a surge in quantum computer hardware development, and we now have several quantum computing platforms with tens of qubits that can be controlled and coupled with fidelities that enable execution of quantum circuits of limited depth. This has led to intense interest in formulating quantum algorithms that can be reliably executed on such devices. The challenge however is that naive compilations of nearly all nontrivial quantum algorithms require circuit depths that are currently out of reach for near-term hardware. Motivated by this challenge, in this work we study how machine learning (ML) can be applied to formulate noise-aware quantum circuits that can be executed on near-term quantum hardware to produce reliable results.

Our method is called noise-aware circuit learning (NACL), and given a suitable description of a computational task and a device model that captures the noise and constraints of a device, it outputs a native circuit that performs the task with greatest robustness to noise. NACL has several broad applications, as illustrated in Fig. 1. The task can be the compilation of a specified unitary transformation [Fig. 1(a)], the preparation of a target state from a specified input state [Fig. 1(b)], or the extraction of an observable from a many-qubit state [Fig. 1(c)]. In each case, NACL returns a circuit that is the significantly more noise resilient to the given noise model, however, as we detail below, the formulation of the machine learning problem is different in each application. Perhaps the most familiar version of NACL is that depicted in Fig. 1(a), where a specified unitary matrix is to be implemented by a circuit composed of native gates, which is usually called compilation. In this context, NACL results in noise-aware circuit compilations.

Previous work on circuit optimization for noise mitigation has largely considered the task of compilation, under restricted models of errors or imperfections. In fact, most work focuses on reducing overall circuit error by reducing the number of two-qubit gates (which tend to be more noisy than single-qubit gates), avoiding faulty qubits, reducing the number of SWAP gates required in architectures with restricted connectivity, or reducing the amount of qubit idle time and/or overall circuit depth [1–8]. These strategies incorporate very little information about errors present in a particular hardware platform. More recent work on error-aware compilation by Murali *et al.* [9] goes beyond this and includes basic calibration information [e.g., qubit T_2 times, controlled NOT (CNOT) gate error rates] to compile circuits using more reliable qubits and gates.

*mnsarov@sandia.gov
†pcoles@lanl.gov

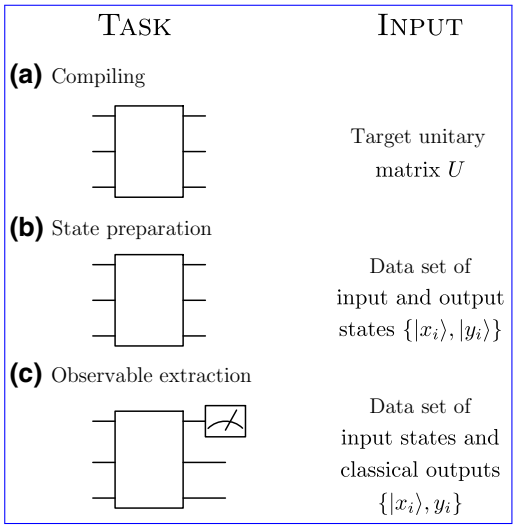

FIG. 1. Applications of NACL. (a) In compiling, the goal is to approximate an input unitary matrix U by a noise-resilient circuit that is compatible with the device constraints. (b) In state preparation, one inputs a set of N input and output states $\{|x_i\rangle, |y_i\rangle\}$, where N could be as small as 1, and the output is a noise-resilient circuit that approximately prepares the $|y_i\rangle$ states from the $|x_i\rangle$ states. (c) In observable extraction, one inputs a set of input states and classical outputs that typically correspond to local observable expectation values, $\{|x_i\rangle, y_i\}$, and the output is a noise-resilient circuit that approximately computes the outputs from any input state $|\psi\rangle$ that might or might not be in the input set.

In this work we extend this direction even further and demonstrate that one can use fine-grained error model information to increase the reliability of the outputs of quantum circuits. Incorporating detailed noise models into one's circuit optimization, as we do here, is particularly compelling at present with the advent of advanced characterization techniques like gate-set tomography [10,11]. These techniques produce fine-grained details—e.g., estimates of process matrices representing the action of imperfect quantum gates—describing the actual evolution of qubits in near-term hardware. We demonstrate that such experimentally derived noise models can be used to go beyond naive circuit compilations for several example quantum algorithms.

NACL has several additional strengths relative to existing approaches in the literature. Crucially, NACL takes a task-oriented approach to quantum circuit discovery, which implies that one does not need a starting point or example quantum circuit that already accomplishes the task. Note that traditional compilers do require such a quantum circuit to start from. Furthermore, because NACL does not start from a template circuit, the optimization is less susceptible to bias. In contrast, standard literature methods that tweak a given quantum circuit inherently bias their optimization towards solutions that look like that starting point. This means that NACL has the potential to discover more novel solutions that otherwise would not be obvious to the human mind. In addition, we will see that NACL naturally balances the trade-off between circuit depth, which leads to more expressivity, and circuit noise, which makes outputs less accurate.

Machine learning was previously applied to train parameterized quantum circuits [6,12], albeit in a noise-free setting. In addition, variational quantum algorithms (VQAs) [13–26] can also be thought of as machine learning of quantum circuits. In the Discussion (Sec. VII), we elaborate on the relationship between NACL and VQAs.

In what follows, we first present our theoretical framework (Sec. II). We then discuss a device model with experimentally determined noise parameters (Sec. III). Next, we present our implementations of NACL with this noisy device, for examples from the three different application classes shown in Fig. 1 (Secs. IV–VI). Finally, we conclude with a discussion in Sec. VII.

II. MACHINE LEARNING FRAMEWORK

A. Overview

A schematic diagram of the steps of NACL is shown in Fig. 2. There are two inputs to NACL: (1) a task, and (2) a device model. The output of NACL is an optimized quantum circuit that accomplishes the inputted task in the presence of the inputted device model. NACL may not output a globally optimal solution (this depends on details of the cost function landscape and optimization method used), but even local optima are improvements over circuit compilations that are not noise aware.

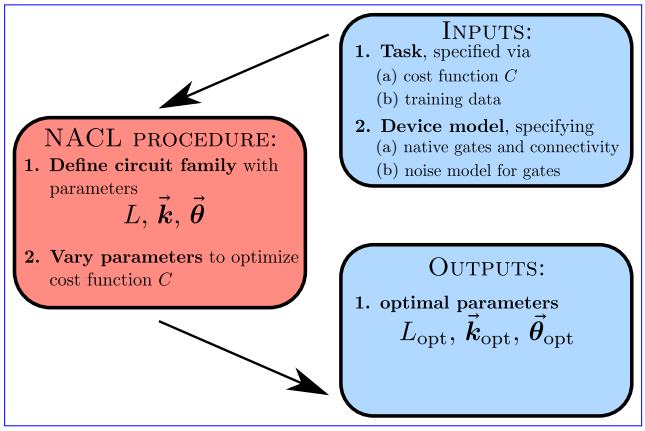

FIG. 2. Schematic diagram of NACL. Our approach takes a task and a device model as an input. The task is defined via examples in a training set and a cost function, C. That information is sufficient to find a noise-aware circuit that approximates a specified task. It is done via optimization over a set of parameters $(L, \vec{k}, \vec{\theta})$ that describe a quantum circuit. The algorithm returns parameters $(L_{\text{opt}}, \vec{k}_{\text{opt}}, \vec{\theta}_{\text{opt}})$, which represent an optimized quantum circuit that minimizes the cost function C. See the text for details.

Note that circuit depth is not an input to NACL. This is because NACL optimizes over circuit depths, and aims to find the depth that achieves the most noise resilience. In addition, an *ansatz* for the circuit is not an input, because NACL attempts to optimize over many *ansätze*. Hence, the structure of the circuit, as well as its depth, are optimized by NACL. This feature of NACL is in the spirit of task-oriented programming, where the user only needs to specify the task, and not the details of the circuit. NACL adapts the circuit structure to optimize a cost function that depends on the type of task specified. As shown in Fig. 1, there are three categories of tasks.

In what follows we provide more details on how NACL works. In Secs. II B and II C we discuss the device model and noise specification, and in Sec. II D we define the NACL cost function for each application. Finally, in Sec. II E we summarize the optimization methods used by NACL.

B. Parameterized circuit

For a given quantum hardware, we denote the native gate set or gate alphabet as $\mathcal{A} = \{A_j(\theta)\}$. Each gate A_j is either a one- or two-qubit gate and may also have an internal continuous parameter θ. As an example, the IBM Q five-qubit computer "Ourense" has the native gate alphabet

$$\begin{aligned}\mathcal{A}_{\text{Ourense}} = \{&\text{CNOT}^{12}, \text{CNOT}^{23}, \text{CNOT}^{24}, \text{CNOT}^{45},\\ &Z^1(\theta), X^1(\pi/2), Z^2(\theta), X^2(\pi/2),\\ &Z^3(\theta), X^3(\pi/2), Z^4(\theta), X^4(\pi/2),\\ &Z^5(\theta), X^5(\pi/2)\},\end{aligned} \tag{1}$$

where CNOT^{jk} is a CNOT between qubits j and k, $Z^j(\theta)$ is a rotation of angle θ about the z axis of qubit j, and $X^j(\pi/2)$ is a rotation of angle $\pi/2$ about the x axis of qubit j (also called a pulse gate).

Such a gate set is supplemented by state preparation and measurement quantum operations. These are typically fixed in most quantum computing architectures (e.g., prepare all qubits in the ground state and measure in the computational basis), and therefore there is no opportunity for optimizing over these. Therefore, we do not consider these as part of the learnable set.

We consider a generic gate sequence that defines a circuit

$$G_{\vec{\alpha}} = G_{(L,\vec{k},\vec{\theta})} = A_{k_L}(\theta_L)\cdots A_{k_2}(\theta_2)A_{k_1}(\theta_1), \tag{2}$$

where L is the number of gates, $\vec{k} = (k_1, \ldots, k_L)$ is the vector of indices describing which gates are utilized in the gate sequence, $\vec{\theta} = (\theta_1, \ldots, \theta_L)$ is the vector of continuous parameters associated with these gates, and $\vec{\alpha} = (L, \vec{k}, \vec{\theta})$ is the set of all these parameters. All parameters in $\vec{\alpha} = (L, \vec{k}, \vec{\theta})$ are optimized over in NACL.

C. Device model

An input to NACL is a device model, which captures the constraints of a device (e.g., limited connectivity) and also represents the noise in the device. We assume the device constraints and connectivity are captured by the specification of a native gate alphabet for the device, e.g., Eq. (1). Only gates that are available are listed in this specification.

The salient characteristics of noise are captured by (i) process matrices for each element of the device's native gate alphabet, and (ii) for state preparation and measurement (SPAM) noise, by quantum-classical channels that represent noisy state preparation or measurement positive operator-valued measure (POVM) elements. The assumption of a fixed process matrix for each gate in the alphabet restricts this treatment to Markovian noise. This can be relaxed by generalizing to time-dependent process matrices for each elementary gate, but we do not do this here for simplicity, and also because characterization tools capable of producing such non-Markovian representations of quantum computer operations are still in the early stages of development [27]. Similarly, in this treatment we mostly ignore the effects of crosstalk, and assume that the process matrix describing a gate operates only on the qubits the ideal gate is defined on. Properly incorporating crosstalk into the noise models that NACL considers requires advances in characterization methods [28] that we discuss later.

Given this paradigm for representing noisy quantum operations, each gate in the alphabet $\mathcal{A}$ has an associated process matrix that accounts for the local noise occurring during that gate. Note that even the identity gate may have a nontrivial process matrix, for example, due to relaxation during idling.

Mathematically speaking, the noise model provides a map from a parameterized circuit $G_{\vec{\alpha}}$ to a parameterized quantum channel $\mathcal{E}_{\vec{\alpha}}$:

$$G_{\vec{\alpha}} \xrightarrow{\text{noise model}} \mathcal{E}_{\vec{\alpha}}. \tag{3}$$

Here, $\mathcal{E}_{\vec{\alpha}}$ is a completely positive trace-preserving (CPTP) map that represents the action of $G_{\vec{\alpha}}$ in a noisy environment.

Specifically, when the noise model is given in the form of process matrices for gates, one can do the following. Let $\mathcal{A} = \{A_j(\theta)\}$ denote the gate alphabet associated with the noiseless gates. In the presence of noise, this gate alphabet becomes a set of quantum channels, $\bar{\mathcal{A}} = \{\bar{A}_j(\theta)\}$, where we note that $\bar{A}_j(\theta)$ now denotes a quantum channel. Now suppose that $G_{\vec{\alpha}}$ is given by $G_{\vec{\alpha}} = A_{k_L}(\theta_L)\cdots A_{k_2}(\theta_2)A_{k_1}(\theta_1)$. Then the simplest way to incorporate the noise model would be to replace each A_{k_i} with $\bar{A}_{k_i}$; i.e., to transform $G_{\vec{\alpha}}$ into a sequence of quantum channels:

$$\mathcal{E}_{\vec{\alpha}} = \bar{A}_{k_L}(\theta_L) \circ \cdots \circ \bar{A}_{k_2}(\theta_2) \circ \bar{A}_{k_1}(\theta_1). \tag{4}$$

However, it is important to note that this formula for $\mathcal{E}_{\vec{\alpha}}$ only accounts for the nontrivial gates that were in the original circuit $G_{\vec{\alpha}}$. However, in practice, identity gates will occur with noise due to, e.g., thermal relaxation. Therefore, care must be taken with respect to identity gates, and we discuss this next.

1. Parallelization

The object we are optimizing over, the circuit in Eq. (2), needs to be modified in the presence of imperfect idle operations. In this case, the sensible thing to do is to perform as many gates in parallel as possible, but the description of a circuit as a sequence of gates, as in Eq. (2), is incomplete because it does not capture which gates can be performed in parallel. In other words, in the presence of imperfect idle operations we cannot simply think of $G_{\vec{\alpha}}$ as a linear sequence of gates; we have to map $G_{\vec{\alpha}}$ to a two-dimensional circuit diagram, in space and time.

Abstractly, we can rewrite

$$G_{\vec{\alpha}}^{\mathrm{Par}} = G_{\vec{\alpha}} = U_{\vec{\alpha},M} \cdots U_{\vec{\alpha},2} U_{\vec{\alpha},1}. \tag{5}$$

Here, each $U_{\vec{\alpha},j}$ represents a layer of gates that can be parallelized. Specifically, we take the circuit proposed in $G_{\vec{\alpha}}$ and compress it using simple circuit rules to minimize idling of qubits. For example, an $X(\pi/2)$ rotation that occurs on the target qubit after a CNOT gate can be moved to before the CNOT gate because their actions on the target qubit commute. In this manner, each gate in $G_{\vec{\alpha}}$ is moved to as early a time as possible without changing the unitary being implemented by $G_{\vec{\alpha}}$. This naturally defines the circuit layers and subsequently $G_{\vec{\alpha}}^{\mathrm{Par}}$. Even though the reordering does not change the overall unitary, whenever we write $G_{\vec{\alpha}}$ in the form in Eq. (5) we denote it as $G_{\vec{\alpha}}^{\mathrm{Par}}$. An important aspect of the optimization in NACL is to numerically find the parallelized representation, $G_{\vec{\alpha}}^{\mathrm{Par}}$, that yields the minimum error in the cost functions detailed below.

Once $G_{\vec{\alpha}}$ is rewritten in the form of $G_{\vec{\alpha}}^{\mathrm{Par}}$, we can then account for noise by replacing each gate in $G_{\vec{\alpha}}^{\mathrm{Par}}$ by the quantum channel that represents its noisy implementation. For example, if a circuit layer in $G_{\vec{\alpha}}^{\mathrm{Par}}$ on a five-qubit processor happens to be

$$U_{\vec{\alpha},j} = Z^1(\theta) \otimes \mathrm{CNOT}^{23} \otimes I^4 \otimes X^5, \tag{6}$$

where the superscript indicates which qubit the gates are operating on [and $Z(\theta)$ is a rotation around the Z axis, CNOT is a CNOT gate, I is the identity, and X is a $\pi/2$ rotation around the X axis]. This layer would be replaced by

$$\bar{U}_{\vec{\alpha},j} = \bar{Z}^1(\theta) \otimes \overline{\mathrm{CNOT}}^{23} \otimes \bar{I}^4 \otimes \bar{X}^5, \tag{7}$$

where the quantities with bars above them are the quantum channels representing those gates. Then the overall noisy circuit corresponding to $G_{\vec{\alpha}}^{\mathrm{Par}}$ is written as

$$\mathcal{E}_{\vec{\alpha}}^{\mathrm{Par}} = \bar{U}_{\vec{\alpha},M} \circ \cdots \circ \bar{U}_{\vec{\alpha},2} \circ \bar{U}_{\vec{\alpha},1}. \tag{8}$$

It is important to note that NACL uses $\mathcal{E}_{\vec{\alpha}}^{\mathrm{Par}}$ rather than $\mathcal{E}_{\vec{\alpha}}$ as the overall noisy channel associated with $G_{\vec{\alpha}}$.

Note that this procedure of parallelizing and incorporating the noise model that we have outlined is valid because our noise models do not account for crosstalk effects. If crosstalk is significant then this strategy of maximizing parallelization might not be optimal since performing many gates in parallel may lead to more noise. Moreover, in the presence of significant crosstalk, capturing processor noise using quantum channels for each of its gates is probably insufficient. Instead, one would need to characterize each possible layer (there are an exponential number of these) since the operation on a qubit due to application of a gate could depend on what is done to any other qubit in the computer at the same time. We discuss how to extend NACL in the presence of crosstalk in the Sec. VII.

D. Cost functions

In this subsection, we construct the cost functions that are minimized by NACL in each of the application classes outlined in Fig. 1.

1. Preliminaries

We first define some relevant quantities. Let $F(\rho,\sigma) = (\mathrm{Tr}\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}})^2$ be the fidelity between two states ρ and σ. For a given pure input state $|\psi\rangle$, we can denote the fidelity of the output states under quantum channels $\mathcal{E}$ and $\mathcal{F}$ as

$$F(\mathcal{E},\mathcal{F},|\psi\rangle) := F(\mathcal{E}(|\psi\rangle\langle\psi|),\, \mathcal{F}(|\psi\rangle\langle\psi|)). \tag{9}$$

We are interested in the case where $\mathcal{F}$ corresponds to a unitary process $\mathcal{U}$, in which case we have

$$F(\mathcal{E},\mathcal{U},|\psi\rangle) = \mathrm{Tr}[\mathcal{E}(|\psi\rangle\langle\psi|)\mathcal{U}(|\psi\rangle\langle\psi|)]. \tag{10}$$

Furthermore, we can define the average process fidelity as

$$\begin{aligned}\overline{F}(\mathcal{E},\mathcal{U}) &= \int d\psi F(\mathcal{E},\mathcal{U},|\psi\rangle) \\ &= \int d\psi\, \mathrm{Tr}[\mathcal{E}(|\psi\rangle\langle\psi|)\mathcal{U}(|\psi\rangle\langle\psi|)],\end{aligned} \tag{11}$$

with the integral taken over the Haar measure.

2. Observable extraction

The first class of applications involves estimating an observable given input state or a set of input states. An example of this is computing the overlap of two quantum states (discussed in Sec. IV). In this application, the output

of the circuit is a classical number (the observable expectation, which in practice is estimated by many executions of the circuit), denoted $f(x)$, and the input, denoted $|x\rangle$, is a quantum state (or classical data encoded in a quantum state). Hence, we want to construct a circuit that computes the function $|x\rangle \rightarrow f(x)$, We classically generate a training data set of the form

$$T = \{(|x^{(i)}\rangle, f(x^{(i)}))\}_{i=1}^{N}. \tag{12}$$

In general, the amount of training data required could scale exponentially in the problem size (i.e., number of qubits), since the data must be general enough to cover the space of possible inputs.

Recall that the parameters $\vec{\alpha}$ define a circuit $G_{\vec{\alpha}}$, which in turn defines a noisy quantum channel $\mathcal{E}_{\vec{\alpha}}^{\mathrm{Par}}$. For this quantum channel, let $y_{\vec{\alpha}}^{(i)}$ denote the output of the circuit (i.e., the expectation of the observable of interest) when the input is $|x^{(i)}\rangle$. Then we define the cost function as

$$C_{\mathrm{OE}}(\vec{\alpha}) = \frac{1}{N}\sum_{i=1}^{N}[f(x^{(i)}) - y_{\vec{\alpha}}^{(i)}]^2. \tag{13}$$

The cost quantifies the discrepancy between the desired output $f(x^{(i)})$ and the true output $y_{\vec{\alpha}}^{(i)}$, averaged over all training data points.

3. State preparation

A second class of applications outlined in Fig. 1 is state preparation. Here, the input is a quantum state or, more generally, a set of quantum states $\{|x^{(i)}\rangle\}_{i=1}^{N}$. The task is then to construct a circuit U that prepares the output states $\{|y^{(i)}\rangle = U|x^{(i)}\rangle\}_{i=1}^{N}$ from these input states. In other words, one wishes to learn a unitary U that accomplishes the desired state preparation task on the training data, $\{|x^{(i)}\rangle, |y^{(i)}\rangle\}_{i=1}^{N}$. Note that this is an underconstrained problem since in the state preparation application $N \ll 2^n$, where U is an n-qubit unitary. In this case, we use the cost function

$$C_{\mathrm{SP}}(\vec{\alpha}) = 1 - \frac{1}{N}\sum_{i=1}^{N} F(\mathcal{E}_{\vec{\alpha}}^{\mathrm{Par}}, \mathcal{U}, |x^{(i)}\rangle), \tag{14}$$

where $\mathcal{U}(\cdot) \equiv U(\cdot)U^{\dagger}$. This is the infidelity between the state prepared by $\mathcal{E}_{\vec{\alpha}}^{\mathrm{Par}}$ and the target state $|y^{(i)}\rangle$, averaged over the training data points. A typical scenario is when there is a single input and output state ($N = 1$), as we consider below in Sec. V.

4. Compilation

Finally, we consider the application of compiling a target unitary, U, into a set of native gates. The action of U on all possible input quantum states must be reproduced. This is a more challenging task than constructing a state-preparation circuit, since one must consider the action on all states rather than on just one state or a small set of states.

Let $\mathcal{U}(\cdot) \equiv U(\cdot)U^{\dagger}$ denote the quantum channel associated with U. Then we define the cost function for compiling as

$$C_{\mathrm{UC}}(\vec{\alpha}) = 1 - \overline{F}(\mathcal{E}_{\vec{\alpha}}^{\mathrm{Par}}, \mathcal{U}). \tag{15}$$

Note that this is analogous to Eq. (14) with the discrete average replaced by a continuous average (i.e., integral with Haar measure). The average $\bar{F}$ can be computed in various ways. Most elegantly, the average process fidelity is related to the entanglement fidelity F_e, via [29–31]

$$\overline{F}(\mathcal{E}_{\vec{\alpha}}, \mathcal{U}) = \frac{dF_e(\mathcal{U}^{\dagger} \circ \mathcal{E}_{\vec{\alpha}}^{\mathrm{Par}}) + 1}{d+1}, \tag{16}$$

where $F_e(\mathcal{E}) = \langle\phi|\mathcal{I} \otimes \mathcal{E}(|\phi\rangle\langle\phi|)|\phi\rangle = F(|\phi\rangle\langle\phi|, \mathcal{E}(|\phi\rangle\langle\phi|))$, with $|\phi\rangle = \sum_j |j\rangle|j\rangle/\sqrt{d}$ being a maximally entangled state, and $d = 2^n$ being the Hilbert-space dimension. Therefore, we can compute the compilation cost function by computing $F(|\phi\rangle\langle\phi|, \mathcal{I} \otimes \mathcal{U}^{\dagger} \circ \mathcal{E}_{\vec{\alpha}}^{\mathrm{Par}}(|\phi\rangle\langle\phi|))$. From the machine learning perspective, the training data set in this case just consists of a pair of states $\{|\phi\rangle, (\mathbb{1} \otimes U)|\phi\rangle\}$. However, this approach requires a computation in a doubled space of dimension 2^{2n}.

Alternative approaches to computing $\bar{F}$ that trade this greater memory complexity for greater time complexity (but can be easily parallelized) are (i) to approximate the Haar average with a sample average over a set of states that form a 2-design, or (ii) to use Nielsen's formula in terms of Pauli operators $\{\sigma_i\}_{i=1}^{d^2}$ [31],

$$\overline{F}(\mathcal{E}_{\vec{\alpha}}, \mathcal{U}) = \frac{1}{d^2(d+1)}\left\{\sum_{i=1}^{d^2} \mathrm{Tr}[U\sigma_i U^{\dagger}\mathcal{E}(\sigma_i)] + d^2\right\}.$$

From the machine learning perspective, for (i), the training data set corresponds to the sampled 2-design and the action of the ideal channel on these, $\{|\phi_i\rangle, U|\phi_i\rangle\}$, and for (ii), the training data set corresponds to the Pauli operators and the action of the ideal channel on these, $\{\sigma_i, U\sigma_i U^{\dagger}\}$.

E. Machine learning algorithms

In this section we describe machine learning methods that we use to find quantum circuits. The general idea behind our approach is the principle of task-oriented programming. The method should work with minimal information about the quantum algorithm, like the system size, number of ancilla qubits, measurement type, or details about the target quantum computer on which the circuit will be run. With that minimal input, our method is designed to find the best performing circuit that achieves the initially specified task.

The task is defined by the choice of training data and the form of a cost function. Here, the choice depends on the particular application. The general rule is that the training data exemplifies the action of the algorithm that needs to be found. A generic example of training data is given in Eq. (12). There, the training set is generated by a function f that encodes an initially specified task. In Sec. IV, we use that framework to construct a training set for the quantum algorithm for computing state overlap. In this case, the function f in Eq. (12) takes the form of a trace and the training data set is defined as

$$T = \{((\rho_i, \sigma_i), \mathrm{Tr}(\rho_i \sigma_i))\}_{i=1}^{N}. \tag{17}$$

The size of the training data set is expected to grow exponentially with the system size for many applications due to the exponential size of the Hilbert space. Nevertheless, there are applications that do not require exponentially large training data sets. One such example is (possibly multiple) state preparation discussed in Sec. V. In the present work we use $N = 15$ training data points in the observable extraction example in Sec. IV. Other examples considered in the paper are trivial from the training data construction point of view. The choice of data in the training set affects the discovered circuit, as we discuss in Sec. IV. In most applications one chooses the data to be as representative for a given task as possible. However, with our approach, there is also an interesting alternative to specialize the quantum algorithm by restricting the type of input data.

The choice of the cost function depends on the application. It is typically defined to measure the discrepancy between the action of the current circuit under optimization and the expected action. The value of the cost function should reach zero if and only if the circuit reproduces the specified task exactly. In Sec. II D we give details on the choice of the cost function for specific tasks considered in the paper.

We use regression machine learning algorithms to learn the relationship between inputs and outputs specified in the training data. In other words, we want to learn how to implement function f in Eq. (12) [e.g., the trace in Eq. (17)] as a quantum circuit. It is done by minimizing the cost function over the space of quantum circuits. This space is described by a set of parameters $\vec{\alpha} = (L, \vec{k}, \vec{\theta})$, where L is the depth of the circuit, $\vec{k}$ is a set of discrete parameters describing the layout of the circuit, and the $\vec{\theta}$ are continuous parameters that span some of the quantum gates, as detailed in Sec. II B.

We stress that the way to include information about the noise model in our machine learning algorithm is to evaluate the cost function in a noisy simulator that implements that noise model. This includes creating a device model described with certain parallelization rules, as described in Sec. II C.

The optimization methods play an important role in our approach. The methods we describe here are general and can be applied to any type of cost function. In particular, they are applicable to the cost functions associated with the applications discussed in Sec. II D.

The space in which the optimization takes place is large and has a complicated form. In our method we are optimizing over circuits composed of gates taken from a particular alphabet. The circuit is described by two kinds of parameters, discrete and continuous. The discrete parameters $\vec{k}$ define the circuit's layout. That is, they specify what type of gate is acting on a given qubit, at a given time during the evaluation of the circuit. The continuous parameters $\vec{\theta}$ span all gates that contain at least one variational parameter. In the example of an alphabet derived from the IBM Q Ourense device in Eq. (1), only Z rotations contain a continuous parameter.

The optimization is an iterative procedure in which every iteration is organized in two nested loops. In the inner loop, the optimizer deals with continuous parameters with a fixed circuit layout $\vec{k}$. Changes to the structure of the circuit are introduced in the outer loop. The optimization over continuous parameters $\vec{\theta}$ is straightforward. Once the structure parameters $\vec{k}$ are fixed, the cost function depends on at most L continuous parameters θ_i. We use adaptive mesh–based, gradient-free, unconstrained (the cost function is invariant under $\theta_i \to \theta_i + 2\pi$) methods [32] to find a minimum of the cost function $C_{\vec{k}} = C_{\vec{k}}(\vec{\theta})$.

When the minimum c of $C_{\vec{k}} = C_{\vec{k}}(\vec{\theta})$ is found, the optimizer switches to the outer loop and makes a change in the structural parameters $\vec{k}$. In this part of the procedure the optimizer is testing small, random updates to the structure of the circuit. Those updates include gate shuffling, gate removal, as well as inserting new gates in the form of resolutions of identity (one-qubit and two-qubit ones). This way, the number of gates L in the circuit is variable and reaches an optimal (noise-dependent) value during the optimization; see below for a more detailed discussion. The circuit is also periodically compiled using simple, standard techniques. Here, we check for gate cancelations and simplification that arise from commutation relations. In principle, we could leave that task to machine learning as well but we find that doing it explicitly speeds up the learning process. After new structural parameters $\vec{k}'$ are identified, the optimizer enters the inner loop and varies continuous parameters $\vec{\theta}$ to find a new minimum c' of a cost function $C_{\vec{k}'} = C_{\vec{k}'}(\vec{\theta})$. Finally, the optimizer makes a decision whether or not the old circuit structure $\vec{k}$ should be replaced by the new one $\vec{k}'$. Here we follow the simulated annealing approach and accept the change if $c' < c$. The change is rejected if $c' > c$ with probability exponentially increasing in $c' - c$.

The above describes one iteration of the optimization algorithm. The iterations are repeated until convergence

of the cost function is observed. The optimization is also restarted multiple times to detect possible local minima.

Finally, let us mention an important feature of the optimization approach. As stated above, random structure updates done in the outer loop involve identity insertion and gate removal. Because the cost function is evaluated in the presence of noise, this procedure can sometimes lead to a larger value of the cost function (this is not possible with a noiseless simulator). Thanks to that, the optimization algorithm automatically finds the optimal length L of the circuit for a specified error model. Other machine learning approaches that are not noise aware must be artificially biased towards short circuits. In contrast, our approach automatically finds a balance between deep, expressive but noisy circuits and shallow, less noisy ones.

III. NOISE MODEL

We demonstrate NACL in the following sections using a fine-grained noise model derived from one- and two-qubit gate-set tomography (GST) [10,11,33] experiments run on the five-qubit IBM Q Ourense superconducting qubit device. We emphasize that we are not claiming to capture the full behavior of this device; this cannot be done with just one- and two-qubit GST, and we need to make some assumptions about device behavior. The most important physical effects we are ignoring in this noise model are: (i) nonuniformity across the device, since we use one-qubit GST results on qubit 0 and two-qubit GST on the qubit pair 0-1 to infer process matrices for all qubits on the device, and (ii) since we do not characterize spectator qubits, we do not capture any crosstalk effects.

One-qubit GST on qubit 0 of the Ourense device yields estimated one-qubit process matrices representing channels associated to the principal native gates on the device, $X(\pi/2)$ (or the "pulse" gate), and I, the single-qubit idle operation. The other single-qubit gate used in this device is $Z(\theta)$, but this is performed virtually in software (through a phase shift of future single-qubit gates) and so we assume that it takes no time and is implemented perfectly. We also use the process matrices estimated by single-qubit GST for $|0\rangle$ state preparation and single-qubit measurement POVM elements for representing these operations. Then two-qubit GST on qubits 0 and 1 is used to extract a process matrix for the CNOT gate. All the estimated process matrices and their figures of merit are presented in Appendix B. Note that, although these process matrices are gauge dependent [10], we only use them to simulate circuit output distributions, which are manifestly gauge independent.

We assume that the layout and connectivity of the qubits are the same as for the IBM Q Ourense device, and these are outlined in Fig. 3. This connectivity and the process matrices described above together define our *device model*.

Note that we only performed GST on qubits 0 and 1 for simplicity, and assume that the resulting process matrices describe the same gates on other qubits also. This assumption could easily be relaxed at the expense of more GST experiments on all the qubits in the device.

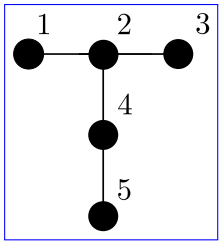

FIG. 3. Qubit layout and connectivity for device modeled in the noise model used to demonstrate NACL. This layout is inspired by the IBM Q Ourense device, and the lines indicate the qubits that can participate in CNOT coupling gates.

In the following sections, we demonstrate NACL with a device model composed of the connectivity information for the IBM Ouresnse device and the above process matrices obtained through GST. All cost functions will be evaluated through simulation of circuits based on this device model.

IV. IMPLEMENTATION FOR OBSERVABLE EXTRACTION

The observable extraction application we focus on is state overlap estimation, where the task is to estimate the overlap between two input states ρ and σ, i.e., estimate $\mathrm{Tr}(\rho\sigma)$. The standard way to achieve this is to apply a controlled-SWAP operation conditioned on an ancilla qubit, and then measure an expectation of an observable on the ancilla. We consider the case where ρ and σ are single-qubit states, and decompose the textbook SWAP-based circuit for overlap estimation into a standard gate set in Fig. 4.

For evaluation under the noise model, we first compile the textbook circuit in Fig. 4 into the native gate set composed of CNOT gates, as well as $X(\pi/2)$ and $Z(\theta)$ rotations. Given the connectivity of the device, Fig. 3, we map the input qubits to qubits 2 and 3, and the ancilla qubit to qubit 1. This is the most favorable mapping since in this case the minimal number (2) of CNOT gates in Fig. 4 needs to be decomposed to account for the lack of device connectivity.

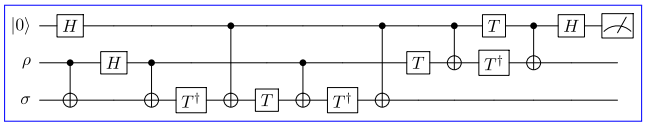

FIG. 4. The textbook SWAP test–based circuit for state overlap estimation when the input states (ρ, σ) are single-qubit states. It is obtained by decomposing the SWAP operation into a standard universal gate set.

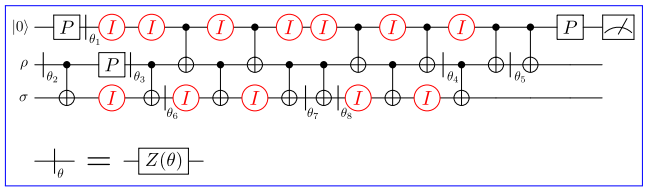

FIG. 5. The form of the textbook SWAP test–based overlap estimation circuit, shown in Fig. 4, when decomposed into the native gates in our device model. Here P denotes the pulse gate, or $X(\pi/2)$ rotation, and I is an idle timestep. The vertical lines denote $Z(\theta)$ rotations that are done virtually and therefore take no time. This notation helps with visualizing which gates can be performed in parallel. Values of θ_n are shown in Appendix A.

There are other mappings that result in similar requirements for CNOT decomposition. We iterate over all of them and select the decomposition that gives the smallest error (as measured by the value of the cost function evaluated in the presence of noise).

The decomposed circuit is shown in Fig. 5. In this figure we show identity gates, or periods where a qubit is idle, in red. This circuit has been compressed and made as parallel as possible (using simplifications afforded by simple commutation relations and circuit identities); however, the remaining idle periods cannot be compressed away. We assume that $X(\pi/2)$ rotations (denoted by P in the figure) take the same amount of time as a CNOT gate for simplicity.

Next, we consider ML-based circuit implementations that *do not* consider noise. Using techniques developed in Ref. [6], which attempt to find exact implementations that consist of as few gates as possible, we perform training without the noise model (but with the connectivity restrictions of the device). The training data set size consists of 15 pairs of randomly generated single-qubit states and their computed overlap. The resulting circuit for overlap estimation and its compiled version are shown in Fig. 6. In the absence of the noise model there is no penalty for the circuit to contain identity gates, and so the resulting circuit has a lot of them.

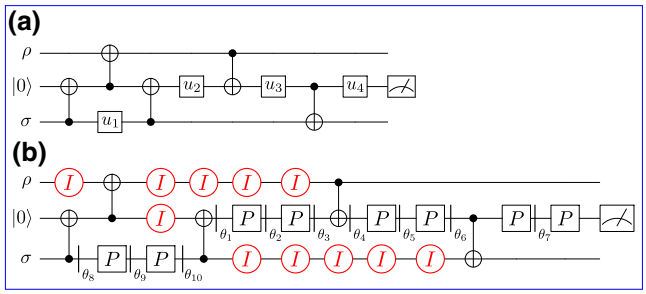

FIG. 6. (a) Machine learned circuit found without considering the noise model. (b) The circuit decomposed into the native gates in the device model. The notation is the same as in Fig. 5. Values of θ_n are shown in Appendix A.

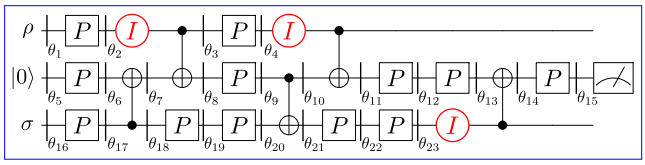

FIG. 7. Machine learned circuit found by NACL incorporating the noise model. The notation is the same as in Fig. 5. Values of θ_n are shown in Appendix A.

Finally, we apply NACL to this problem and formulate the cost function using circuit simulation with the noise model described in Sec. III. The training data set size consists again of 15 pairs of randomly generated single-qubit states and their computed overlap. The algorithm works directly with the native gate set, and so no subsequent decomposition is necessary. The circuit found by NACL is shown in Fig. 7. Two features of the NACL circuit immediately stand out. First, since we have taken into account the noise associated with idling qubits, the circuit contains very few idles. Second, NACL makes interesting use of $Z(\theta)$ gates—these are error free, take no time, and also increase the expressiveness of a circuit—and, consequently, NACL seems to maximize their use [especially compared the noise unaware ML circuit in Fig. 6, which does not distinguish $Z(\theta)$ gates from other gates, and therefore does not use them more frequently]. This liberal use of $Z(\theta)$ gates most likely also leads to the shorter depth circuit. It should be stressed that these features are not built into the algorithm but result from the optimization and represent the best found balance between the number of gates and the noise induced by their action.

In the following, we compare the performance of the three circuits described above. We generate a validation data set—1000 pairs of new random one-qubit, mixed states $\{\rho_j, \sigma_j\}$—and apply the three circuits to estimate the overlap between each pair (the circuits are simulated under the noise model). For simplicity, we label the textbook circuit (Fig. 5) $\mathcal{A}_1$, the noise unaware, standard ML circuit (Fig. 6) $\mathcal{A}_2$, and the result of NACL (Fig. 7) $\mathcal{A}_3$. In Fig. 8(a) we compare the errors of all three circuits, defined as the absolute value of the difference between the exact overlap $\mathrm{Tr}(\rho_j \sigma_j)$ and its estimate computed with the given circuit:

$$\mathrm{error}_{j,\mathcal{A}_i} = |\mathrm{Tr}(\rho_j \sigma_j) - \langle \sigma_z \rangle_{\mathcal{A}_i}|. \tag{18}$$

Here $\langle \sigma_z \rangle_{\mathcal{A}_i}$ is the expectation value of the σ_z operator on the measured qubit at the end of circuit $\mathcal{A}_i$. The data are sorted such that the error of $\mathcal{A}_1$ is increasing with sample index, j. In Fig. 8(a) we show that noise-aware ML-generated circuit gives the best overlap estimate for most of the state pairs.

In the inset of Fig. 8(a) we show the difference between the error of the textbook circuit and both ML circuits (these

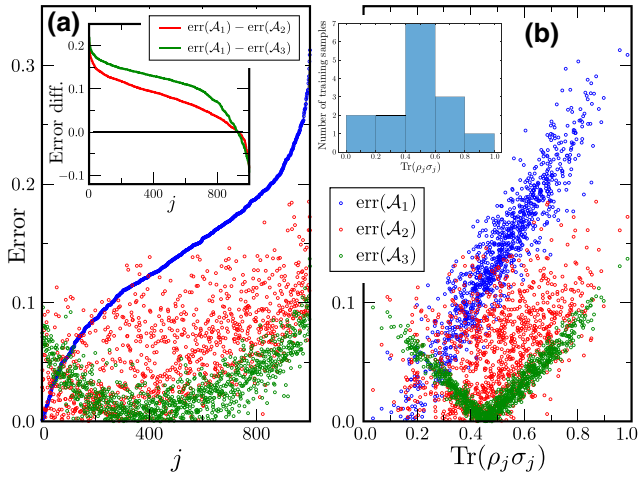

FIG. 8. (a) A comparison of the error in computing state overlap [as quantified by Eq. (18)] for each of the validation samples for the three circuits: textbook ($\mathcal{A}_1$, blue), noise unaware ML ($\mathcal{A}_2$, red), and NACL ($\mathcal{A}_3$, green). The x-axis indexes pairs of states in the validation data set. The inset shows differences in error. (b) Overlap estimation error for the three circuits as a function of the exact value of the overlap $\mathrm{Tr}(\rho_j\sigma_j)$. The inset shows a histogram of exact overlaps for the validation data set.

sets of data points are both independently ordered according to decreasing error difference). The ML circuit is better than the textbook one if the value shown in the inset is positive. We can see that this is indeed the case for over 90% of cases, with the NACL circuit also outperforming the regular ML circuit in these cases. For further analysis, we look at the same data in Fig. 8(b), but this time with the errors plotted against exact overlap of the 1000 samples in the validation data set. It can be seen that the error of $\mathcal{A}_1$ generally decreases with the exact overlap. In addition, the error of $\mathcal{A}_3$ (NACL) shows nonmonotonic behavior with exact overlap, achieving its minimum around exact overlap of 0.5 and increasing for larger and smaller overlaps. This behavior of the NACL error can be explained by the specifics of the training method and the type of cost function that was used. NACL tries to minimize average error [see Eq. (13)], and examining a histogram of overlaps in the training sample [inset of Fig. 8(b)] we see that these overlaps are concentrated between 0.4 and 0.5. Therefore, NACL optimizes the average-case cost function by performing best on input state pairs that have overlaps around this value. An interesting observation is that there can be a correlation between the structure of a circuit and the overlaps it can best estimate.

Finally, we can explain why the textbook circuit outperforms NACL in regions of low exact overlap as a combination of two factors: (i) as mentioned above, NACL minimizes average error, and the contribution to this from training samples with small overlap is small; hence, it sacrifices performance on small overlap states to get better performance on states with larger overlap, and (ii) the other factor that results in the textbook circuit performing well for small exact overlap samples is accidental; namely, that the overlap is estimated by measuring $\langle\sigma_z\rangle$ on the ancilla, and this quantity tends to 0 with circuit length (since the stochastic noise in the gates dampens this polarization). The output of $\mathcal{A}_1$ is small due to noise, and thus is *accidentally* close to the correct answer for small overlap states.

We note that the uneven behavior of NACL with exact overlap of input states can be easily modified by (i) modifying the training data set to have uniformly distributed overlaps, and (ii) modifying the cost function to be a worst-case measure of performance instead of average case and/or a function of relative error as opposed to absolute error with the exact overlap.

V. IMPLEMENTATION FOR STATE PREPARATION

For the state preparation application, we focus on preparing W states of n qubits, i.e.,

$$|W_n\rangle = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}|i\rangle, \tag{19}$$

where $|i\rangle$ is the state where qubit i is $|1\rangle$ and all other qubits are in state $|0\rangle$. W states are multipartite entangled states that are robust against loss and can be used for multipartite cryptographic protocols and for teleportation [34]. As far as we are aware, the circuits generated in Cruz *et al.* [35] are the most efficient circuits for W-state generation, and we use these circuits as our base-case "textbook" circuits to compare against.

In the following we study the preparation of W states for $n = 4, 5$.

A. Four-qubit W-state preparation

The textbook circuit for preparing $|W_4\rangle$ is shown in Fig. 9(a). It is obtained by following the general procedure given in Ref. [35]. This circuit will be applied to the first four qubits in the device shown in Fig. 3. The performance of the textbook circuit and the NACL circuit will depend on the subset of qubits on which we prepare the state. However, in realistic situations, one will not be given that freedom since the state preparation is usually only one step in a larger quantum circuit, which imposes constraints on the choice of qubits. We select qubits 1–4 to show how NACL can optimize circuits on devices with restricted connectivity.

The one-qubit gate, depicted as $G(p)$ in Fig. 9(a), is defined as

$$G(p) = \begin{pmatrix} \sqrt{p} & \sqrt{1-p} \\ \sqrt{1-p} & -\sqrt{p} \end{pmatrix}. \tag{20}$$

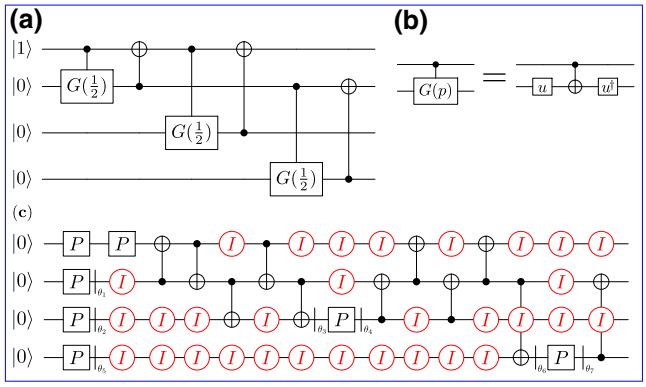

FIG. 9. (a) Textbook circuit for preparing $|W_4\rangle$. (b) Decomposition of a control-$G(p)$ gate into CNOT and one-qubit gates u and $u^\dagger$, where $u = e^{-iY\alpha}$ and $\alpha = \arcsin(\sqrt{p})/2$. (c) Compilation of the textbook circuit shown in (a) into the first four qubits of the device model in Fig. 3. The notation is the same as in Fig. 5. The values of angles θ_j are given in Appendix A.

Note that this is a slightly different definition than that given in Ref. [35]. The above definition of $G(p)$ leads to the same state that is prepared with the circuit shown in Fig. 9(a), but allows for more efficient decomposition of the control-$G(p)$ gate into CNOT and one-qubit gates.

The circuit shown in Fig. 9(a) must be compiled into the native gate set in the device model. The W state is invariant under permutation of qubits, and so one can relabel the qubits in the circuit shown in Fig. 9(a) if this is advantageous for compilation. To find the optimal compilation of the textbook circuit, we check all possible permutations of qubits. All permutations lead to a compilation in which at least two CNOT gates are not compatible with device connectivity and need to be decomposed further. We evaluate each permutation by simulation (with the noise model) under the corresponding compiled circuit and compute the fidelity of the output with the exact $|W_4\rangle$ state. The permutation that gives the highest fidelity is simply $[1, 2, 3, 4]$ (there are however other permutations that lead to the same fidelity), and the corresponding compiled circuit is shown in Fig. 9(b). We find that this textbook circuit produces $|W_4\rangle$ with fidelity 0.671 under the noise model.

The circuit produced by NACL for preparing $|W_4\rangle$ is shown in Fig. 10. Since the task here is to prepare one state from one other state, the training data set and validation data set are the same, and just consist of one pair $\{|0\rangle^{\otimes 4}, |W_4\rangle\}$; the first element is the input state and the second is the ideal output state. This NACL circuit outputs a state under the noise model with a fidelity of 0.8894 to the exact state. This is a reduction in error (as measured by $1 - F$, where F is fidelity) by a factor of 3 as compared with the best-known textbook circuit.

Careful inspection of the circuit in Fig. 10 reveals an interesting feature. In certain circumstances, it is more

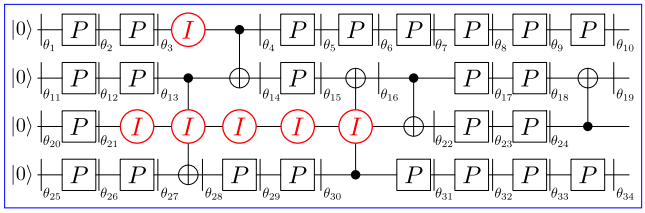

FIG. 10. Circuit that prepares $|W_4\rangle$ found by NACL. The notation is the same as in Fig. 5. Angles θ_j are specified in Appendix A.

beneficial (from the point of minimizing the cost function; infidelity in this case) to have a long sequence of gates that are not compiled into an equivalent transformation with a shorter sequence. An example is the final 13 gates [including $Z(\theta)$ gates in this count] applied to qubit 1. It is possible to implement the resulting transformation with a shorter sequence of gates, but doing so would mean that the qubit sits idle for the remaining time while the operations on the other qubits complete. Apparently, this incurs a greater cost than the longer sequence (the pulse gates are fairly high-quality gates for this device and, in fact, have a smaller infidelity than the idle operations; see Appendix B). We thus observe a feature that resembles dynamical decoupling or a dynamically corrected gate for this final transformation of qubit 1. We have reasonable confidence that this feature is not a numerical artifact or local optimum because we also independently optimized just that subcircuit (i.e., keep the rest of the circuit fixed and optimized just the last six clock cycles of qubit 1 under the same cost function that evaluates the error on the four-qubit output state), and could not find a better sequence. Note that this feature is "emergent." Dynamical gate correction techniques are not coded in the search algorithm and yet NACL effectively used them in the optimized solution. It a way, those techniques are "discovered" via cost optimization. We also point out that this feature of preferring longer sequences to idles is not general—one cannot replace every sequence of idles with a sequence of pulses and $Z(\theta)$ rotations and lower the error. For example, qubit 3 sits idle over five clock cycles and this achieves the minimum cost function even when we attempt to reoptimize just that subsequence of gates. This feature demonstrates the ability of NACL to find circuit implementations that optimize performance in highly nontrivial ways that incorporate an interplay between the computational task (encoded in the cost function) and the device model.

B. Five-qubit *W*-state preparation

We also study the preparation of $|W_5\rangle$ since this task requires the use of all qubits on the device in Fig. 3. Again, we follow the prescription in Cruz *et al.* [35] to arrive at the best textbook circuit for preparing $|W_5\rangle$ in Fig.

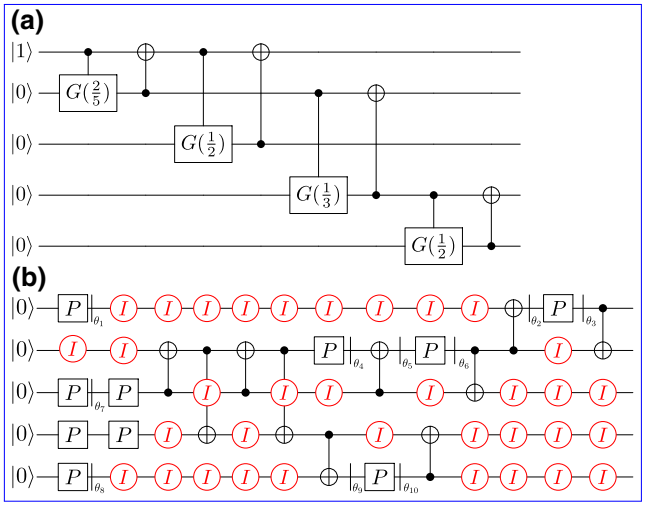

FIG. 11. (a) Circuit for preparing $|W_5\rangle$ obtained by following the construction given in Ref. [35]. The first controlled-$G(p)$ gate can be simplified, as the first qubit is initialized in $|1\rangle$. This allows for a shorter compilation. (b) Its best compiled version achieved by a proper permutation of qubits. The notation is the same as in Fig. 5. Angles θ_j are given in Appendix A.

11. The compilation of this circuit onto the device under study is not trivial since we can arbitrarily permute the qubits. Every permutation will result in a potentially different decomposition of CNOT gates, given the constrained connectivity of the device. We check all 120 qubit permutations and find that the circuit compilation shown in Fig. 11(b) gives the smallest value of the cost function when evaluated under the noise model. We find this optimal permutation to be $[4, 3, 5, 2, 1]$. Under this permutation, only one CNOT gate [the second gate from the left in Fig. 11(a)] needs to be decomposed due to the lack of connectivity. The circuit in Fig. 11(b) achieves the fidelity of 0.675.

NACL found the circuit presented in Fig. 12 for $|W_5\rangle$ state preparation. Again, NACL finds a circuit that is much more compact than the textbook one. It uses fewer CNOT gates, requires less idling of qubits, and uses the error-free $Z(\theta)$ gates liberally. The circuit produces an output state with fidelity $F = 0.837$ with the ideal $|W_5\rangle$ state. That is, the error (as measured by $1 - F$) is reduced by a factor of 2 as compared to the textbook circuit.

VI. IMPLEMENTATION FOR CIRCUIT COMPILATION

For the circuit compilation application, we consider the problem of compiling the quantum Fourier transform (QFT), which is a paradigmatic building block that is used in many quantum algorithms [36]. In the following we consider implementing a three-qubit QFT.

A textbook circuit for implementing a QFT on three qubits is shown in Fig. 13(a). We consider implementing this on qubits 1, 2, and 3 in the device shown in Fig. 3.

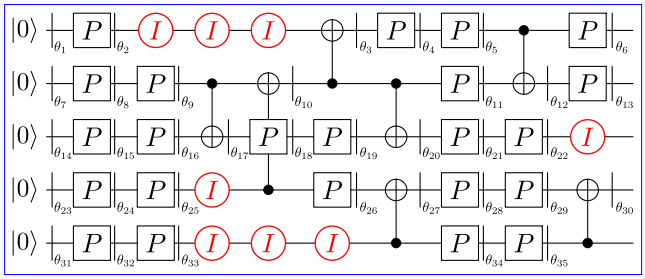

FIG. 12. The circuit that approximates preparation of $|W_5\rangle$ found by NACL. The notation is the same as in Fig. 5. Angles θ_j are given in Appendix A.

We first need to decompose the controlled-$Z(\theta)$ rotations. Every controlled $Z(\theta)$ is decomposed using two CNOT gates [37]. This decomposition leads to two CNOT gates between qubits 1 and 3. Since these qubits are not directly connected, these CNOT gates need to be decomposed further. The result of this compilation procedure is shown in Fig. 13(b). This compilation leads to a very sparse circuit with many (incompressible) idle gates, which has negative impact on the quality of the final result.

The circuit constructed via NACL is shown in Fig. 14. We use NACL with the cost function defined in Eq. (15) with the average process fidelity computed via Eq. (16). The circuit has shorter depth than the compiled textbook circuit, and does not contain a single idle gate (as compared with 18 for the textbook circuit). It also contains more error-free $Z(\theta)$ rotations, enhancing the expressiveness of the circuit.

To compare the performance of the two compiled circuits for QFT, we select 1000 random pure states $|\Psi_j\rangle$ and evaluate each circuit on those states. The error metric we use is the infidelity between the ideal QFT output and the circuit output; $1 - \mathrm{Tr}(\rho_j |\Psi_j^{\mathrm{ex}}\rangle\langle\Psi_j^{\mathrm{ex}}|)$, where $|\Psi_j^{\mathrm{ex}}\rangle$ is the result of the exact evaluation of QFT on $|\Psi_j\rangle$. Our results are summarized in Fig. 15. We also compare our

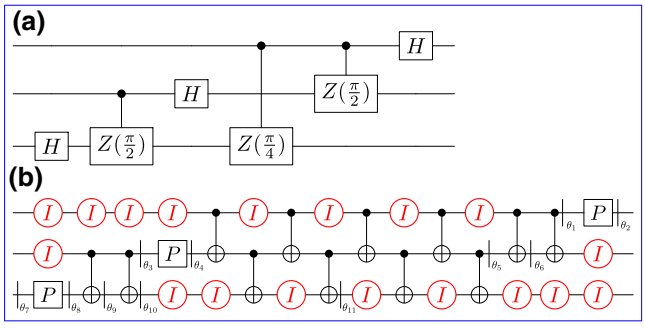

FIG. 13. (a) A textbook circuit for performing QFT on three qubits. (b) A compilation of the circuit in (a) into the native gate set in the device model we are simulating. The compilation has to take into account the fact that qubits 1 and 3 are not directly connected. Angles θ_j are specified in Appendix A.

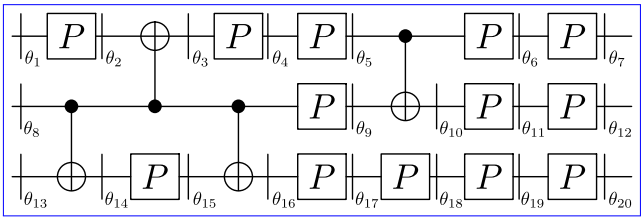

FIG. 14. Circuit performing QFT found by NACL. The notation is the same as in Fig. 5. Angles θ_j are given in Appendix A.

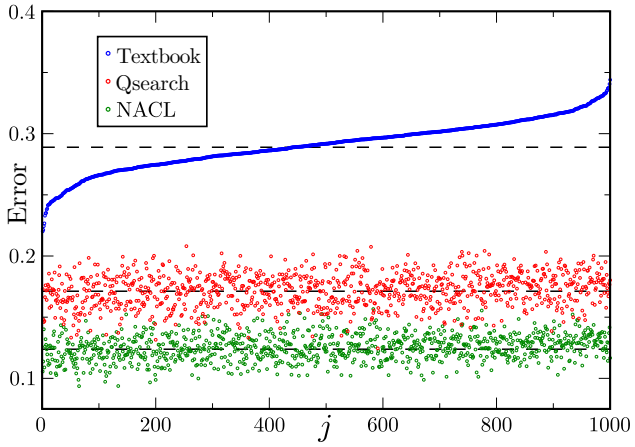

FIG. 15. Performance of textbook, qsearch-generated, and NACL-generated circuits to evaluate QFT. The figure shows error (as defined in the text) for 1000 randomly generated pure states. The NACL-generated circuit performs much better than the textbook one on all considered states. The circuit found by NACL also has a lower error than that generated by qsearch.

results with qsearch [38], a recently proposed technique for circuit compilation. Qsearch is typically used as an exact compilation method, but it can be used with a finite precision δ. Qsearch results shown in Fig. 15 are obtained by running it with various values of δ and selecting the circuit that gives the smallest error as defined by Eq. (16). For easier comparison, the states $|\Psi_j\rangle$ are ordered such that the error of the textbook circuit (represented by the blue line) increases with the state index j. The NACL-generated circuit performed better than the textbook one on all considered states. It also outperforms the circuit found by qsearch even after minimizing over compilation precision δ. Since the validation data set is composed of random pure input states, the average infidelity (over these input states) is related to the entanglement infidelity of the channel defined by the noisy circuit [see Eq. (16)], which is an input-state-independent measure of the quality of a channel (or circuit implementation). We use this relation to validate our error metric defined over randomly sampled input states. In Fig. 15 the dotted lines show $1 - [dF_e(\mathcal{U}^\dagger \circ \mathcal{E}) + 1]/(d+1)$, where $d = 8$, $\mathcal{U}$ is the channel corresponding to the ideal circuit implementation, $\mathcal{E}$ is the channel corresponding to the noisy circuit implementation, and F_e is the entangled fidelity defined in Sec. II D. These lines correspond well to the sample averages of our infidelity error metric. We find that NACL reduced the average infidelity of a textbook circuit from 0.289 to 0.124, that is, by 57%. NACL also reduced the error by a factor of 1.4 as compared to qsearch. Another observation is that the performance of the textbook circuit varies more significantly with input state than for the NACL-generated circuit.

VII. DISCUSSION AND CONCLUSIONS

We have introduced the framework of NACL, whereby the circuit implementation of a quantum algorithm is formulated by machine learning and optimization based on a cost function that captures the goal of the algorithm and a device model that captures the connectivity and noise in the device that executes the circuit. We have shown that this framework can be applied to all of the common tasks in quantum computing—observable (or mean-value) extraction, state preparation, and circuit compilation—and have demonstrated through examples the types of performance improvements that can be obtained through NACL. For the examples considered here, NACL produces reductions in error rates (suitably defined for the different tasks) by factors of 2 to 3, when compared to textbook circuits for the same tasks.

In general, NACL produces shorter depth circuits that minimize the impact of stochastic noise sources. However, as demonstrated through the examples considered here, NACL can automatically derive known noise-suppression concepts such as dynamical decoupling and apply these in contexts where they are useful (as defined by the cost function). It also naturally outputs circuits that incorporate commonsense strategies such as minimizing the number of noisy idle gates and maximizing the use of ideal gates, such as error-free $Z(\theta)$ rotations. NACL can incorporate much more fine-grained information about the device than other circuit compilation techniques—e.g., in the demonstrations presented here we have used process matrices derived from gate set tomography of real hardware to approximately model noise on this device. Such process matrices can capture effects ignored by effective noise models, such as coherent noise and nonunital processes such as relaxation.

We note that we have also executed NACL with an error model derived from trapped-ion physics (see Appendix C for details), to validate that the technique can be used with a variety of noise model specifications. The results are very similar to those presented above, although there are some simplifications due to an assumption of full connectivity in the device (which is realistic for small trapped-ion platforms).

The noise models currently compatible with NACL do not include crosstalk effects. Although these can be incorporated for small devices using the approach outlined in this paper, incorporating crosstalk in a scalable manner is complicated. The heart of the issue is how to model crosstalk in a scalable manner [28]. In the presence of crosstalk, the natural description of operations on a quantum computer is not in terms of gates, but in terms of *layers*, which capture what is done to each qubit in the device in a given clock cycle. This is because the precise operation performed on a qubit could, in principle, depend on what is performed on any other qubit in the device. Therefore, the first extension of NACL required to capture crosstalk is to optimize circuits in terms of layers as opposed to gate sequences. Moreover, one has to also consider whether it is realistic to develop quantum channels representing noisy implementation of any circuit layer. Firstly, there are an exponential (in n, the number of qubits) number of possible layers to characterize, and secondly, one needs to perform n-qubit process tomography in order to get quantum channels for each layer. This last task is obviously impossible for large n, and therefore one has to develop more approximate techniques to describe noisy implementations of layers. One approach around these issues is to patch together quantum channels derived from one-, two-, and three-qubit tomography to get an approximate description of a circuit layer, similar to what is demonstrated in Govia *et al.* [39]. This would model a physically important subclass type of crosstalk errors [28]. Another approach is to forego full tomographic information about error processes and instead use effective noise models or error rates that contain information about crosstalk; see, e.g., Ref. [40] for an example of how such partial information can be used to model crosstalk errors. Of course, one is trading off NACL prediction accuracy when approximate noise models are used. Future work will look at incorporating these more complex noise effects into the NACL circuit learning framework.

An important issue to consider is how to scale NACL to develop noise-resilient circuits for larger devices. In Appendix D, we present the empirical running times for all the examples presented in the paper. The complexity of circuit simulation under a noise model and the complexity of optimization over the circuit parameters increase exponentially with the number of qubits. As a consequence, the current NACL approach could be used as is to optimize circuits on up to about 8–10 qubits. With code optimization and parallelization, this could be extended to circuits on 12–14 qubits. Such machine learned noise resilient circuits could be useful for increasing the performance of small modular elements of larger circuit applications; e.g., magic state distillation circuits. However, we can also outline a strategy for extending NACL beyond such use cases. The strategy applies when one is already given a circuit compilation for a computational task. Perhaps this is a compilation derived using theoretical decompositions or some other efficient method. Then one can sample a subcircuit from this circuit. This subcircuit defines an ideal unitary and one can use NACL to find best approximations to this unitary under the given device model. This sampling can be repeated for multiple subcircuits. However, note that this strategy does not guarantee any optimality properties for the circuit derived from combining these individually optimized subcircuits. Studying the potential of this strategy for scaling up the NACL framework is left as future work.

Related to scalability is the connection between NACL and VQAs. An alternative to evaluating the NACL cost functions in Sec. II D by simulating a parameterized quantum circuit on a classical computer is to evaluate them by executing the parameterized circuits on quantum hardware directly in the spirit of VQAs. In addition to the obvious advantage of scalability, this hardware-enabled approach has the advantage of capturing the noise model exactly (and does not require any noise modeling). However, for certain applications (e.g., compiling and state preparation [19,20]), the NACL cost function requires comparing against the *ideal* target circuit outputs. In a VQA setting, any preparation of the targets would also be noisy, and therefore one cannot exactly evaluate the required cost functions. Whether it is possible to sufficiently approximate the cost functions with noisy hardware is an open problem [41], and if this were possible, it would make hardware-enabled NACL realistic.

We note that NACL typically outputs approximations of the task that is specified. This is because of two reasons: (i) in the presence of typical noise models the best one can do is approximate an ideal unitary map, and (ii) NACL provides no guarantee of finding global minima of the cost functions, which are typically extremely nonlinear. Therefore, even if the noise model is benign enough that the global minimum or minima correspond(s) to ideal implementations of the target unitary, NACL will most likely find a local minima that is an approximation of the target unitary. However, as empirically demonstrated in this paper, NACL output is often far superior to textbook derived circuits, or even circuits optimized using other compilation techniques.

Modern optimization and machine learning methods will be critical for deriving computational use from near-term quantum devices. Motivated by this, we have developed the NACL framework as a way to utilize detailed noise characterization information to build noise-resilient circuits for near-term quantum computing applications, and we have outlined promising directions for extending this framework. Our NACL method can be combined with (and hence is complementary to) other approaches to error mitigation that have recently been proposed [42–46]. Hence, NACL is a novel primitive that will play an important role in the quest for quantum advantage.

ACKNOWLEDGMENTS

The authors would like to thank Tim Proctor and Andrew Baczewski for useful comments on a draft of this work. The research presented in this article is supported by the Laboratory Directed Research and Development program of Los Alamos National Laboratory under project number 20180628ECR for the noise-free machine learning approach and project number 20190065DR for the machine learning approach in the presence of noise. P.J.C. also acknowledges support from the LANL ASC Beyond Moore's Law project. This work is also supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under the Quantum Computing Application Teams (QCAT) program. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under Contract No. DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

APPENDIX A: NUMERICAL VALUES OF ROTATION ANGLES

In Table I we list the angles θ_n that define the $Z(\theta)$ gates in all the circuits presented in the main text.

APPENDIX B: NOISE MODEL PROCESS MATRICES

In this appendix we list the process matrices and SPAM elements derived from GST experiments that define our error model for the five-qubit device we demonstrate NACL on. These process matrices are completely positive trace-preserving estimates of the corresponding operations. [We note that, in order to estimate these process matrices, GST required that we also estimate the process matrix corresponding to the $Y(\pi/2)$ operation. We omit that estimate here as our device model does not include the $Y(\pi/2)$ gate in the native gate set.] All process matrices are given in the Pauli basis (i.e., they are "Pauli transfer matrices") while the SPAM operations are given in the "standard" representation. Because of throughput constraints only "short" GST circuits (i.e., circuits for linear-inversion GST [10]) are used; each circuit is repeated 1024 times.

We have

$$I = \begin{pmatrix} 1.0000 & -0.0000 & 0.0000 & -0.0000 \\ 0.0042 & 0.9943 & -0.0064 & 0.0178 \\ -0.0033 & 0.0120 & 0.9962 & 0.0186 \\ 0.0029 & -0.0182 & -0.0167 & 0.9928 \end{pmatrix},$$

$$X(\pi/2) = \begin{pmatrix} 1.0000 & 0.0000 & 0.0000 & -0.0000 \\ 0.0007 & 0.9988 & -0.0050 & -0.0055 \\ -0.0010 & -0.0060 & 0.0167 & -0.9980 \\ -0.0017 & 0.0065 & 0.9979 & 0.0176 \end{pmatrix},$$

$$P_0 = \begin{pmatrix} 0.9997 & -0.0006 \\ 0.0055 & 0.0231 \end{pmatrix},$$

$$P_1 = \begin{pmatrix} 0.0003 & 0.0006 \\ -0.0055 & 0.9769 \end{pmatrix},$$

$$\rho_0 = \begin{pmatrix} 0.9903 & 0 \\ 0 & 0.0097 \end{pmatrix}.$$

Here, P_0 and P_1 are the imperfect POVM effects for projections onto the $|0\rangle$ and $|1\rangle$ states, respectively, and ρ_0 is the density matrix for the single-qubit imperfect state preparation. Finally,

CNOT

$$= \begin{pmatrix} 1.000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.012 & 0.973 & 0.016 & 0.005 & 0.005 & -0.002 & 0.012 & -0.004 & -0.002 & 0.003 & -0.004 & 0.002 & -0.010 & 0.008 & 0.015 & -0.001 \\ 0.001 & -0.009 & 0.004 & -0.003 & -0.002 & 0.000 & -0.023 & 0.001 & -0.006 & -0.001 & -0.007 & 0.003 & 0.005 & -0.019 & 0.974 & 0.003 \\ 0.002 & 0.006 & 0.000 & 0.003 & -0.005 & -0.001 & 0.002 & -0.021 & -0.010 & 0.001 & 0.003 & -0.010 & -0.001 & -0.007 & 0.004 & 0.983 \\ 0.002 & 0.001 & 0.012 & -0.008 & 0.015 & 0.964 & 0.017 & 0.004 & 0.001 & 0.020 & -0.018 & 0.003 & 0.048 & 0.020 & -0.002 & -0.004 \\ 0.002 & -0.001 & 0.004 & 0.002 & 0.980 & 0.004 & -0.002 & -0.009 & 0.018 & 0.001 & -0.005 & 0.012 & 0.021 & 0.042 & 0.002 & 0.005 \\ -0.002 & -0.003 & 0.041 & 0.002 & -0.009 & 0.001 & 0.005 & -0.018 & -0.005 & -0.002 & 0.003 & 0.977 & 0.014 & -0.003 & 0.000 & 0.012 \\ -0.003 & -0.006 & -0.002 & 0.045 & -0.006 & 0.019 & 0.015 & 0.006 & -0.002 & 0.022 & -0.968 & -0.001 & -0.006 & 0.001 & -0.008 & 0.005 \\ 0.001 & 0.007 & -0.004 & 0.001 & 0.000 & -0.019 & 0.017 & -0.001 & 0.011 & 0.966 & 0.019 & 0.003 & 0.012 & 0.009 & -0.002 & -0.005 \\ 0.001 & 0.008 & 0.004 & -0.001 & -0.021 & -0.000 & 0.002 & -0.011 & 0.981 & 0.004 & -0.001 & -0.005 & 0.014 & 0.004 & 0.002 & 0.010 \\ -0.001 & -0.005 & 0.007 & -0.002 & 0.005 & 0.005 & -0.003 & -0.975 & -0.011 & 0.002 & 0.007 & -0.020 & -0.003 & -0.002 & 0.008 & -0.023 \\ -0.002 & -0.012 & 0.004 & 0.006 & 0.003 & -0.021 & 0.967 & 0.001 & -0.005 & 0.017 & 0.016 & 0.007 & 0.003 & 0.004 & 0.021 & 0.004 \\ -0.002 & -0.003 & -0.001 & 0.001 & -0.021 & -0.035 & -0.008 & -0.001 & -0.010 & -0.006 & 0.001 & -0.006 & 0.987 & 0.002 & 0.001 & -0.000 \\ -0.008 & 0.006 & 0.012 & -0.001 & -0.043 & -0.020 & -0.003 & 0.003 & -0.010 & -0.009 & 0.003 & 0.008 & 0.011 & 0.970 & 0.016 & 0.007 \\ 0.005 & -0.018 & 0.973 & 0.003 & -0.004 & -0.009 & 0.002 & 0.008 & 0.002 & 0.005 & -0.001 & -0.039 & -0.004 & -0.007 & 0.005 & -0.005 \\ 0.000 & -0.007 & 0.005 & 0.982 & 0.005 & 0.002 & -0.008 & 0.003 & 0.003 & -0.009 & 0.040 & 0.002 & 0.002 & 0.005 & 0.001 & 0.001 \end{pmatrix}.$$

TABLE I. Angles (in radians) defining the $Z(\theta)$ gates in each of the circuits presented in the main text.

n	θ_n in Fig. 5	θ_n in Fig. 6(b)	θ_n in Fig. 7	θ_n in Fig. 9(c)	θ_n in Fig. 10	θ_n in Fig. 11(b)	θ_n in Fig. 12	θ_n in Fig. 13(b)	θ_n in Fig. 14
1	3.926 991	4.729 459	1.249 703	1.570 796	6.261 032	0.785 398	0.000 110	2.748 894	5.528124
2	1.570 796	2.356 455	2.410 073	0.785 398	1.581 678	2.356 194	3.107 730	1.570 796	1.603 007
3	1.570 796	6.271 231	5.503 714	2.356 194	0.618 247	3.141 593	0.775 046	2.356 194	3.180 240
4	0.785 398	0.012 770	0.172 928	3.141 593	6.210 505	0.615 480	1.385 048	1.570 796	1.588 121
5	5.497 787	5.497 958	0.106 332	0.785 398	3.155 136	2.526 113	3.218 745	0.785 398	5.982 926
6	5.497 787	0.017 911	0.022 432	2.356 194	3.088 771	3.141 593	6.184 576	5.497 787	1.517 815
7	0.785 398	0.785 692	1.621 729	3.141 593	3.127 992	1.369 438	0.000 214	1.570 796	3.174 252
8	5.497 787	4.713 347	6.267 293		2.708 279	0.785 398	0.725 692	2.356 194	2.380 614
9		2.355 849	3.672 289		1.477 670	2.356 194	0.895 856	5.497 787	3.909 589
10		4.711 034	0.132 619		2.327 048	3.141 593	0.149 143	0.392 699	6.271 246
11			5.697 289		0.012 648		2.289 056	5.890 486	0.006 795
12			3.141 953		0.876 330		3.142 930		3.903 039
13			4.364 565		0.444 937		4.665 748		4.728 925
14			0.964 557		4.781 066		0.000 281		3.157 351
15			6.037 635		5.429 627		0.614 126		2.383 873
16			5.975 455		2.827 826		6.176 411		0.022 179
17			0.159 144		3.101 400		3.698 677		3.135 448
18			6.194 334		0.505 015		1.349 278		5.062 797
19			1.518 005		1.752 444		5.651 896		3.119 667
20			2.570 119		0.077 924		0.048 880		0.821 506
21			2.836 344		2.324 862		1.604 027		
22			3.171 423		5.525 191		4.721 183		
23			4.005 286		1.711 153		0.000 189		
24					0.113 893		1.563 388		
25					0.040 828		3.165 257		
26					0.863 436		0.025 552		
27					6.210 510		3.165 166		
28					4.981 590		1.566 059		
29					2.622 501		0.897 177		
30					0.536 382		5.028 318		
31					2.882 208		6.282 094		
32					0.148 144		2.400 830		
33					2.916 385		3.127 614		
34					5.971 181		2.312 352		
35							5.047 211		

We also list in Table II various error metrics for these noisy operators (as compared to ideal operators). "Infidelity" for gate operations is taken to be average gate infidelity, i.e., $1-\bar{F}$, where $\bar{F}$ is the average gate fidelity (with respect to the desired target operation), as defined in Eq. (11). For SPAM operations, we simply use state infidelity, i.e.,

$$1 - F(\rho,\sigma) = 1 - \left(\mathrm{Tr}\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}\right)^2. \tag{B1}$$

The half-diamond distance, denoted $\epsilon_\diamond$, is defined as

$$\epsilon_\diamond(A,B) = \tfrac{1}{2}\|A-B\|_\diamond = \tfrac{1}{2}\sup_\rho \|(A\otimes \mathbb{1}_d[\rho]) - (B\otimes \mathbb{1}_d[\rho])\|_1, \tag{B2}$$

where $\|\cdot\|_1$ is the trace norm, sup is taken over all density matrices of dimension d^2, and $d = \dim A = \dim B$.

The average gate infidelity may be thought of as, averaged over the Haar measure, the infidelity of a state that has passed through the gate's channel; the diamond distance

TABLE II. Error metrics for noisy operations (compared to ideal operations) used in our device model input to NACL. For gate operations, entanglement infidelity and diamond distance are presented, while for SPAM operations, only state infidelity is used.

Gate label	Infidelity	Half-diamond distance
I	2.8×10^{-3}	1.7×10^{-2}
$X(\pi/2)$	8.8×10^{-4}	1.1×10^{-2}
CNOT	1.9×10^{-2}	5.0×10^{-2}
ρ_0	9.7×10^{-3}	...
P_0	2.0×10^{-3}	...
P_1	2.3×10^{-2}	...

may be thought of as a worst-case error rate. The average gate infidelity is quadratically more sensitive to stochastic error than the unitary error, while the diamond distance is equally sensitive to both classes of errors [47].

APPENDIX C: NOISE MODEL FOR A TRAPPED-ION QUANTUM COMPUTER

In addition to the noise model presented in the main text, we also ran NACL using an additional noise model, that is, an effective model formed from error metrics derived from a near-term trapped-ion quantum computer. We adapt the coarse-grained error maps used to model errors during execution of a common trapped-ion gate set developed by Trout *et al.* [48]. In particular, the native gates in the processor are assumed to be $X(\theta), Y(\theta), Z(\theta)$, and $XX(\theta) \equiv e^{i\theta X\otimes X}$, where the first three are single-qubit rotations about the three orthogonal axes and the last is an arbitrary angle Molmer-Sørensøn interaction between two qubits. The quantum channels representing the noisy versions of each of these gates are given by

$$\begin{aligned}
\mathcal{E}_X(\theta) &= \mathcal{D}(p_d)\circ\mathcal{W}(p_{\text{dep}})\circ\mathcal{R}_X(p_\alpha)\circ\mathcal{U}_X(\theta),\\
\mathcal{E}_Y(\theta) &= \mathcal{D}(p_d)\circ\mathcal{W}(p_{\text{dep}})\circ\mathcal{R}_Y(p_\alpha)\circ\mathcal{U}_Y(\theta),\\
\mathcal{E}_Z(\theta) &= \mathcal{D}(p_d)\circ\mathcal{W}(p_{\text{dep}})\circ\mathcal{R}_Z(p_\alpha)\circ\mathcal{U}_Z(\theta),\\
\mathcal{E}_{XX}(\theta) &= [\mathcal{D}_1(p_{\text{d},1})\otimes\mathcal{D}_2(p_{\text{d},2})]\\
&\quad\circ[\mathcal{W}_1(p_{\text{dep}})\otimes\mathcal{W}_1(p_{\text{dep}})]\\
&\quad\circ\mathcal{H}(p_{\text{xx}})\circ\mathcal{H}(p_h)\circ\mathcal{U}_{XX}(\theta).
\end{aligned}$$

Here, $\mathcal{U}_k(\theta)$ represents an ideal rotation about axis k [e.g., $\mathcal{U}_X(\theta)\rho = e^{-i\theta X}\rho e^{i\theta X}$], $\mathcal{R}_k(p_\alpha)$ represents the effects of rotation angle imprecision about axis k [e.g., $\mathcal{R}_X(p_\alpha)\rho = (1-p_\alpha)\rho + p_\alpha X\rho X$)], $\mathcal{W}(p_{\text{dep}})$ is a depolarizing channel [i.e., $\mathcal{W}(p_{\text{dep}})\rho = (1-p_{\text{dep}})\rho + p_{\text{dep}}I$], $\mathcal{D}(p_d)$ is a dephasing channel [i.e., $\mathcal{D}(p_d)\rho = (1-p_d)\rho + Z\rho Z$], and, finally, $\mathcal{H}(p)\rho = (1-p)\rho + XX\rho XX$ is a two-qubit channel that represents the effects of an imprecise rotation (when $p = p_{xx}$) or the effects of ion heating (when $p = p_h$). The subscripts on any of these channels (in the case of the two-qubit operation) denotes action on that qubit.

In addition to these imperfect gates, we model SPAM errors by following an ideal ground-state preparation with a depolarizing channel, and by preceding ideal single-qubit measurement POVM effects by a depolarizing channel, i.e.,

$$\begin{aligned}
\langle\langle 0| &\to \langle\langle 0|\mathcal{D}(p_{\text{dep}}),\\
|i\rangle\rangle &\to \mathcal{D}(p_{\text{dep}})|i\rangle\rangle \quad \text{for } i = 0, 1,
\end{aligned}$$

where we have notated state preparation and measurement effects as Hilbert-Schmidt vectors. Finally, in order to capture noise during idle cycles, all idles are modeled as a depolarizing channel $\mathcal{D}(p_{\text{idle}})$.

This effective noise model captures many of the nonidealities in typical ion trap quantum computing architectures. However, note that, under this model, there are no connectivity restrictions and it is possible to perform a two-qubit gate between any two qubits. In the following computations we use the error rates

$$\begin{aligned}
p_d &= 1.5\times10^{-4},\\
p_{\text{dep}} &= 8\times10^{-4},\\
p_{d,1} &= p_{d,2} = 7.5\times10^{-4},\\
p_\alpha &= 1\times10^{-4},\\
p_{xx} &= 1\times10^{-3},\\
p_h &= 1.25\times10^{-3},\\
p_{\text{idle}} &= 8\times10^{-4}.
\end{aligned}$$

APPENDIX D: NACL TRAINING TIMES

In this appendix we discuss computational resources that are needed to compile circuits with NACL. NACL is initialized with a random quantum circuit. That circuit is evolved in time such that it minimizes a properly defined cost function. This aspect of our approach together with the nonconvex optimization landscape with many local minima makes it hard to define "compilation time" or "time to solution." Furthermore, the optimization could be terminated before reaching the global minimum of the cost function. As we show here, such imperfect compilations are still better than other compilations in many cases.

Instead of working with a particular definition of the "compilation time," we analyze how quickly NACL can reduce the cost and find circuits that perform better than textbook circuits. This gives more information about performance of the algorithm than a single number that characterizes the compilation time. Our results are summarized in Fig. 16, which shows all the use cases that we consider in the main text. In Fig. 16(a) we show how the quality of the compilation increases with time. Here (as well as in other panels) time indicates wall-clock time, measured in seconds. Compilation quality is measured as $1 - C_{\text{OE}}$, where C_{OE} is the observable extraction cost function defined in Eq. (13). We use the same training data set as that considered in the main text. The results suggest that NACL quickly finds an algorithm that outperforms the textbook one. It took NACL only 15 s to find a circuit that has a lower cost function than the textbook one. Further improvements however are more costly and reaching a global minimum took more than 20 h. As expected, one is dealing with diminishing returns in running longer optimizations, as the algorithm that reached 98% of the best achievable quality was found only after 26 s.

The above cost analysis is representative for other applications. In Fig. 16(b) we show results for three-qubit QFT

considered in Sec. VI. The plot displays how the average fidelity of the channel $\mathcal{E}$ defined by the noisy circuit improves as a function of time. Here, the fidelity is defined by $[dF_e(\mathcal{U}^\dagger \circ \mathcal{E}) + 1]/(d+1)$, where $d = 8$ for a three-qubit example, F_e is the entanglement fidelity defined in Sec. II D, and $\mathcal{U}$ is the ideal channel implementing three-qubit QFT. Again, we find that NACL quickly surpasses the textbook solution but takes significantly more time to find the global optimum. In Figs. 16(c) and (d) we present results for W-state preparation on 4 and 5 qubits, respectively. Here, the compilation quality is defined as a fidelity between the noisy state and exact W state. As expected, compiling bigger circuits is more expensive due to exponential scaling of quantum simulation. Nevertheless, NACL manages to surpass the best textbook algorithm within 110 s and 37 min for four-qubit and five-qubit W-state preparation, respectively.

In Table III we combine compilation times (as defined by the time to beat the textbook algorithm and the time to reach 98% of the best NACL result) for various applications of NACL. We use a 16-core desktop computer to generate the data shown in Table III and Fig. 16.

TABLE III. Compilation times for all NACL applications considered in the main text. The data presented in the table are extracted from Fig. 16 and show the time needed to surpass the textbook algorithm and time to reach 98% of the best circuit found by NACL.

Algorithm	Time to beat the textbook algorithm	Time to achieve 98% of the best NACL
Observable extraction	15 s	26 s
Three-qubit QFT	28 s	23 min
Four-qubit QFT	110 s	1.3 h
Five-qubit QFT	37 min	12 h

While the NACL compilation times presented in Table III and Fig. 16 are generally too long for just-in-time compilers of quantum circuits, we stress that this is not how we envision NACL will be used in practice. Instead, NACL will be an offline utility that will precompile noise-resistant circuits for common subroutines. We envision that NACL will be used to create libraries of commonly used algorithmic routines for specific devices. We argue that in the

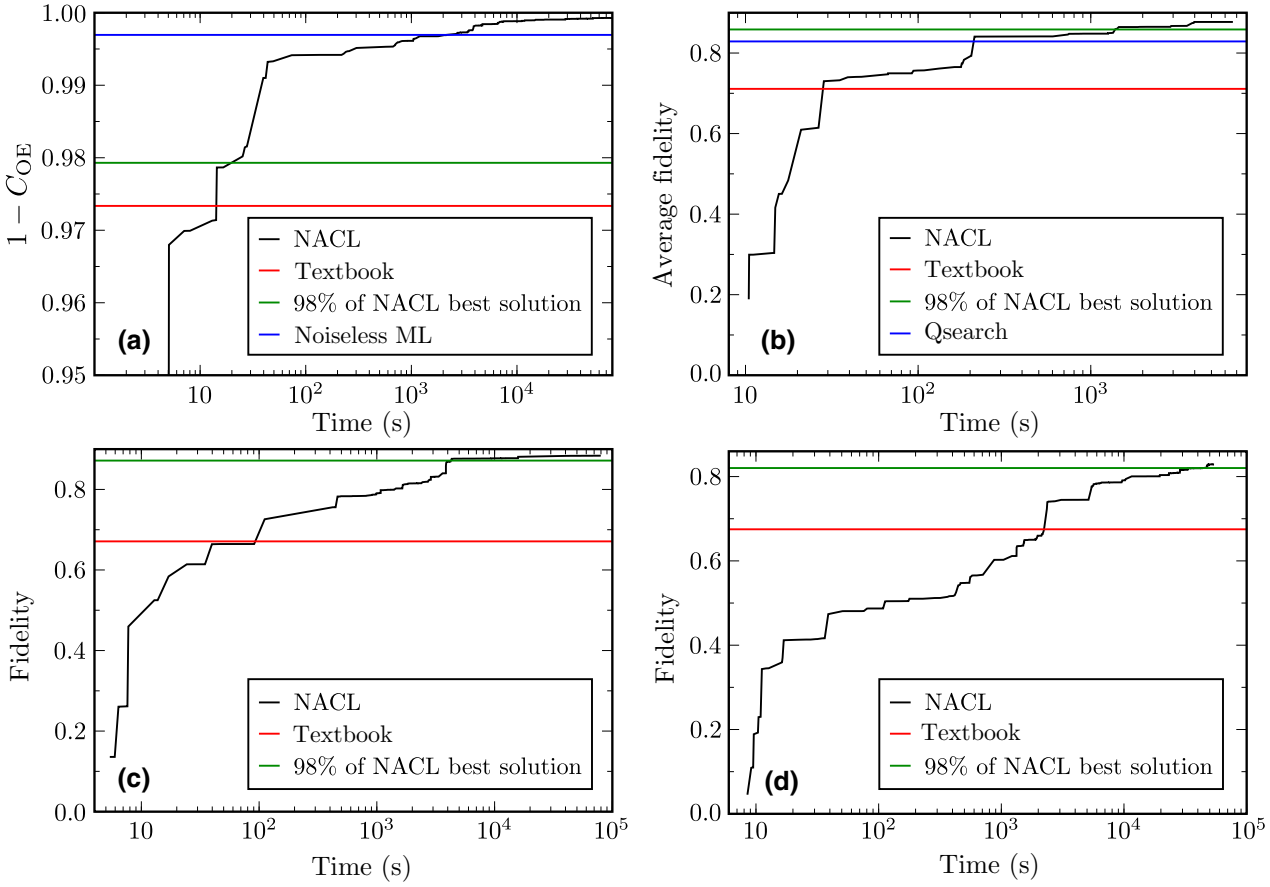

FIG. 16. NACL performance. Panels show compilation quality as a function of the wall-clock time. The definition of compilation quality depends on a particular task; see the text for details. Panels (a)–(d) show NACL performance as applied to observable extraction, three-qubit QFT compilation, and four-qubit and five-qubit W-state preparation, respectively.

noisy intermediate-scale quantum era such a capability is more useful than fast just-in-time compilers.

[1] Robert R. Tucci, Qubiter Algorithm Modification, Expressing Unstructured Unitary Matrices with Fewer CNOTs, arXiv:quant-ph/0411027 (2004).

[2] Ali JavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T. Chong, and Margaret Martonosi, in *Proceedings of the 11th ACM Conference on Computing Frontiers - CF '14* (ACM Press, Cagliari, Italy, 2014), p. 1.

[3] Dmitri Maslov, Basic circuit compilation techniques for an ion-trap quantum machine, New J. Phys. **19**, 023035 (2017).

[4] Davide Venturelli, Minh Do, Eleanor Rieffel, and Jeremy Frank, in *Proc. of IJCAI* (2017), p. 4440.

[5] Prakash Murali, Ali Javadi-Abhari, Frederic T. Chong, and Margaret Martonosi, Formal constraint-based compilation for noisy intermediate-scale quantum systems, Microprocess. Microsyst. **66**, 102 (2019).

[6] L. Cincio, Y. Subaşı, A. T. Sornborger, and P. J. Coles, Learning the quantum algorithm for state overlap, New J. Phys. **20**, 113022 (2018).

[7] Swamit S. Tannu and Moinuddin K. Qureshi, in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (ACM, Providence RI, USA, 2019), p. 987.

[8] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, Will Simmons, Alec Edgington, and Ross Duncan, t—ket⟩ : A retargetable compiler for NISQ devices, Quantum Sci. Technol. **6**, 014003 (2020).

[9] Prakash Murali, Jonathan M. Baker, Ali Javadi Abhari, Frederic T. Chong, and Margaret Martonosi, in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (ACM, 2019), p. 1015.

[10] Robin Blume-Kohout, John King Gamble, Erik Nielsen, Kenneth Rudinger, Jonathan Mizrahi, Kevin Fortier, and Peter Maunz, Demonstration of qubit operations below a rigorous fault tolerance threshold with gate set tomography, Nat. Commun. **8**, 1 (2017).

[11] PyGSTi. A python implementation of Gate Set Tomography, https://www.pygsti.info/.

[12] Kwok Ho Wan, Oscar Dahlsten, Hlér Kristjánsson, Robert Gardner, and M. S. Kim, Quantum generalisation of feedforward neural networks, npj Quantum Inf. **3**, 36 (2017).

[13] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'brien, A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. **5**, 4213 (2014).

[14] Jarrod R. McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New J. Phys. **18**, 023023 (2016).

[15] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann, A quantum approximate optimization algorithm, arXiv:1411.4028 (2014).

[16] J. Romero, J. P. Olson, and A. Aspuru-Guzik, Quantum autoencoders for efficient compression of quantum data, Quantum Sci. Technol. **2**, 045001 (2017).

[17] Ying Li and Simon C. Benjamin, Efficient Variational Quantum Simulator Incorporating Active Error Minimization, Phys. Rev. X **7**, 021050 (2017).

[18] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii, Quantum circuit learning, Phys. Rev. A **98**, 032309 (2018).

[19] Sumeet Khatri, Ryan LaRose, Alexander Poremba, Lukasz Cincio, Andrew T. Sornborger, and Patrick J. Coles, Quantum-assisted quantum compiling, Quantum **3**, 140 (2019).

[20] Tyson Jones and Simon C. Benjamin, Quantum compilation and circuit optimisation via energy dissipation, arXiv:1811.03147 (2018).

[21] Ryan LaRose, Arkin Tikku, Étude O'Neel-Judy, Lukasz Cincio, and Patrick J. Coles, Variational quantum state diagonalization, npj Quantum Inf. **5**, 57 (2019).

[22] Andrew Arrasmith, Lukasz Cincio, Andrew T. Sornborger, Wojciech H. Zurek, and Patrick J. Coles, Variational consistent histories as a hybrid algorithm for quantum foundations, Nat. Commun. **10**, 3438 (2019).

[23] Marco Cerezo, Alexander Poremba, Lukasz Cincio, and Patrick J. Coles, Variational quantum fidelity estimation, Quantum **4**, 248 (2020).

[24] Carlos Bravo-Prieto, Ryan LaRose, M. Cerezo, Yigit Subasi, Lukasz Cincio, and Patrick J. Coles, Variational quantum linear solver: A hybrid algorithm for linear systems, arXiv:1909.05820 (2019).

[25] Cristina Cirstoiu, Zoe Holmes, Joseph Iosue, Lukasz Cincio, Patrick J. Coles, and Andrew Sornborger, Variational fast forwarding for quantum simulation beyond the coherence time, npj Quantum Inf. **6**, 82 (2020).

[26] M. Cerezo, Kunal Sharma, Andrew Arrasmith, and Patrick J. Coles, Variational quantum state eigensolver, arXiv:2004.01372 (2020).

[27] Timothy Proctor, Melissa Revelle, Erik Nielsen, Kenneth Rudinger, Daniel Lobser, Peter Maunz, Robin Blume-Kohout, and Kevin Young, Detecting, tracking, and eliminating drift in quantum information processors, Nat. Commun. **11**, 5396 (2020).

[28] Mohan Sarovar, Timothy Proctor, Kenneth Rudinger, Kevin Young, Erik Nielsen, and Robin Blume-Kohout, Detecting crosstalk errors in quantum information processors, Quantum **4**, 321 (2020).

[29] Benjamin Schumacher, Sending entanglement through noisy quantum channels, Phys. Rev. A **54**, 2614 (1996).

[30] Michał Horodecki, Paweł Horodecki, and Ryszard Horodecki, General teleportation channel, singlet fraction, and quasidistillation, Phys. Rev. A **60**, 1888 (1999).

[31] Michael A. Nielsen, A simple formula for the average gate fidelity of a quantum dynamical operation, Phys. Lett. A **303**, 249 (2002).

[32] Mark A. Abramson, Charles Audet, John E. Dennis, Jr., and Sébastien Le Digabel, OrthoMADS: A deterministic MADS instance with orthogonal directions, SIAM J. Optim. **20**, 948 (2009).

[33] Erik Nielsen, Kenneth Rudinger, Timothy Proctor, Antonio Russo, Kevin Young, and Robin Blume-Kohout, Quantum Sci. Technol. **5**, 044002 (2020).

[34] Jaewoo Joo, Young-Jai Park, Sangchul Oh, and Jaewan Kim, Quantum teleportation via a *W* state, New J. Phys. **5**, 136 (2003).

[35] Diogo Cruz, Romain Fournier, Fabien Gremion, Alix Jeannerot, Kenichi Komagata, Tara Tosic, Jarla Thiesbrummel, Chun Lam Chan, Nicolas Macris, Marc-André Dupertuis, *et al.*, Efficient quantum algorithms for ghz and w states, and implementation on the ibm quantum computer, Adv. Quantum Technol. **2**, 1900015 (2019).

[36] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, Cambridge, England, 2010).

[37] Andrew W. Cross, Lev S. Bishop, John A. Smolin, and Jay M. Gambetta, Open Quantum Assembly Language, arXiv:1707.03429 [quant-ph] (2017).

[38] M. G. Davis, E. Smith, A. Tudor, K. Sen, I. Siddiqi, and C. Iancu, in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)* (2020), p. 223.

[39] L. C. G. Govia, G. J. Ribeill, D. Ristè, M. Ware, and H. Krovi, Bootstrapping quantum process tomography via a perturbative ansatz, Nat. Commun. **11**, 1084 (2020).

[40] Prakash Murali, David C. Mckay, Margaret Martonosi, and Ali Javadi-Abhari, in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20 (Association for Computing Machinery, New York, NY, USA, 2020), p. 1001.

[41] Kunal Sharma, Sumeet Khatri, Marco Cerezo, and Patrick Coles, Noise resilience of variational quantum compiling, New J. Phys. **22**, 043006 (2020).

[42] Kristan Temme, Sergey Bravyi, and Jay M. Gambetta, Error Mitigation for Short-Depth Quantum Circuits, Phys. Rev. Lett. **119**, 180509 (2017).

[43] Abhinav Kandala, Kristan Temme, Antonio D. Córcoles, Antonio Mezzacapo, Jerry M. Chow, and Jay M. Gambetta, Error mitigation extends the computational reach of a noisy quantum processor, Nature **567**, 491 (2019).

[44] Piotr Czarnik, Andrew Arrasmith, Patrick J. Coles, and Lukasz Cincio, Error mitigation with clifford quantum-circuit data, arXiv:2005.10189 (2020).

[45] Armands Strikis, Dayue Qin, Yanzhu Chen, Simon C. Benjamin, and Ying Li, Learning-based quantum error mitigation, arXiv:2005.07601 (2020).

[46] Alexander Zlokapa and Alexandru Gheorghiu, A deep learning model for noise prediction on near-term quantum devices, arXiv:2005.10811 (2020).

[47] Yuval R. Sanders, Joel J. Wallman, and Barry C. Sanders, Bounding quantum gate error rate based on reported average fidelity, New J. Phys. **18**, 012002 (2015).

[48] Colin J. Trout, Muyuan Li, Mauricio Gutierrez, Yukai Wu, Sheng-Tao Wang, Luming Duan, and Kenneth R. Brown, Simulating the performance of a distance-3 surface code in a linear ion trap, New J. Phys. **20**, 043038 (2018).