

Predictive Models from Quantum Computer Benchmarks

Daniel Hothem^{*§}, Jordan Hines^{*†}, Karthik Nataraj[‡], Robin Blume-Kohout^{*} and Timothy Proctor^{*}

^{*}Quantum Performance Laboratory, Sandia National Laboratories, Livermore, CA 94550, USA

[†]Department of Physics, University of California, Berkeley, CA 94720

[‡]Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305

[§]Email: dhothem@sandia.gov

Abstract—Holistic benchmarks for quantum computers are essential for testing and summarizing the performance of quantum hardware. However, holistic benchmarks—such as algorithmic or randomized benchmarks—typically do not predict a processor’s performance on circuits outside the benchmark’s necessarily very limited set of test circuits. In this paper, we introduce a general framework for building predictive models from benchmarking data using *capability models*. Capability models can be fit to many kinds of benchmarking data and used for a variety of predictive tasks. We demonstrate this flexibility with two case studies. In the first case study, we predict circuit (i) process fidelities and (ii) success probabilities by fitting *error rates models* to two kinds of volumetric benchmarking data. Error rates models are simple, yet versatile capability models which assign effective error rates to individual gates, or more general circuit components. In the second case study, we construct a capability model for predicting circuit success probabilities by applying transfer learning to ResNet50, a neural network trained for image classification. Our case studies use data from cloud-accessible quantum computers and simulations of noisy quantum computers.

I. INTRODUCTION

Quantum processors have rapidly grown over the past decade, but hardware errors (i.e., noise) limit their computational capabilities. The errors in one- or two-qubit systems can be studied in detail using tomographic methods [1], but contemporary processors suffer from complex errors (e.g., crosstalk) that are challenging to fully characterize [2]. This has led to the proliferation of holistic benchmarks that aim to quantify the overall impact of errors on a processor’s performance [3], [4], [5], [6]. Holistic benchmarks run suites of test circuits and use the data to compute metrics that summarize a processor’s performance, e.g., mean gate infidelities [7], [8], [9], the quantum volume [3], volumetric benchmarking plots [4], or capability regions [5]. However, while many holistic benchmarks offer useful summaries of a processor’s performance, most benchmarks do not make *predictions* about how well other circuits or algorithms will execute. Any performance predictions from the results of a holistic benchmark have been typically obtained using informal and *ad hoc* extrapolations (e.g., see [6]).

In this work we introduce a framework for constructing predictive models from benchmarking data, and we provide two case studies demonstrating how to do so. Our general framework (see Section II) is based on *capability models*, which builds on the concept of a processor’s capability

function [10]. A capability function captures how well a processor can run circuits—by formalizing the concept of a circuit’s “success probability” using a performance metric such as process fidelity—and a capability model is simply a parameterized model for a capability function. As we explain herein, most holistic benchmarks can be interpreted as probing a capability function, and so the parameters of a capability model can be fit to benchmarking data. Capability models are flexible, and we provide two case studies to demonstrate the promise of this approach.

Our first case study (see Section III) introduces *error rates models* (ERMs), which are flexible, scalable, and interpretable capability models that can be designed to predict a variety of figures of merit. ERMs generalize and formalize the widely-used idea of representing the errors in each of a processor’s gates with a generic error process (global depolarization) that has a single parameter—the gate’s error rate. Fitting ERMs to benchmarking data summarizes the data in terms of *effective* error rates and enables predictions for how other circuits will perform. Our second case study (see Section IV) creates a capability model for circuit success probabilities from a pre-trained neural network. We apply transfer learning [11] to ResNet50 [12], an image classifier, to create a capability model that predicts circuit performance on a simulated noisy quantum computer. This complements recent work that has used custom (rather than pre-trained) neural networks to predict a variety of circuit performance metrics [10], [13], [14], [15], [16].

II. PREDICTIVE MODELS FROM BENCHMARKS

A. Benchmarks and capability functions

We begin with a brief overview of holistic quantum computer benchmarks, and we explain how most benchmarking data can be interpreted as estimates of a *capability function* [10]. Holistic benchmarks typically consist of: (i) selecting some set of circuits $\{c\}$, e.g., via sampling from some distribution; (ii) running these circuits (or some closely related circuits) on the processor being tested; (iii) for each circuit $c \in \{c\}$, computing from the data a single number $\hat{s}(c)$ that is an estimate of a metric $s(c)$ (such as fidelity) that quantifies how well the processor can run c ; and (iv) computing one or more summary statistics from the data $\{\hat{s}(c)\}_{c \in \{c\}}$. We refer to $s(c)$ as the benchmark’s *capability function* [10].

Examples of benchmarks that can be described as above include many RB methods [7], [8], [9], the quantum volume benchmark [3], cross-entropy benchmarking [17], volumetric benchmarks [4], and many algorithmic benchmarks [6]. For example, many RB methods consist of running randomly sampled circuits where each circuit’s overall action is to map the standard input state $|00\dots\rangle$ to some computational basis state $|x\rangle$. Therefore, an RB circuit is a *definite outcome* circuit, meaning that it produces one particular bit string x if it is run without error. In RB, each circuit’s success probability—i.e., the probability the correct bit string is returned—is estimated from data. So this benchmark’s capability function $s(c)$ is success probability.

B. Capability models

Holistic benchmarks summarize their data using one or more statistics or plots—e.g., see the volumetric benchmarking plot of Fig. 1—but they typically do not make predictions. We propose fitting parameterized models to benchmarking data to construct predictive models from benchmarks. Our framework is based on a particular kind of parameterized model that we call a *capability model*.

Definition 1. Let s be a capability function $s : \mathcal{C} \rightarrow \mathbb{R}$ defined over a set of circuits \mathcal{C} . A capability model for s is a parameterized function $\epsilon : \mathcal{C} \rightarrow \mathbb{R}$ used to approximate s .

Capability models do *not* predict the outcome distribution of a circuit, unlike many error models for quantum processors (such as those constructed via gate set tomography [1]). Instead capability models predict only how accurately the processor implements circuits, as quantified by a capability function s . A capability model can be any type of parameterized function (e.g., a neural network) and its parameters do not need to correspond to the rates of physical errors (e.g., gate over-rotation angles etc.).

We propose building predictive models from benchmarks as follows:

- (1) Run a benchmark that generates data $\{\hat{s}(c)\}$ for a set of circuits $\{c\}$, where $\hat{s}(c)$ is an estimate of some capability function $s(c)$.
- (2) Select a capability model ϵ for capability function s .
- (3) Fit the parameters of ϵ to the data $\{\hat{s}(c)\}$.

We will explore two kinds of capability models in this work: ERMs (see Section III) and pre-trained neural networks (see Section IV).

III. CASE STUDY 1: ERROR RATE MODELS

In this case study, we introduce error rates models (ERMs) and demonstrate their utility by fitting them to benchmarking data.

A. Theory

ERMs generalize and formalize two simple concepts: (i) represent the error in each of a processor’s quantum logic operations by a single error rate, and (ii) predict a circuit’s

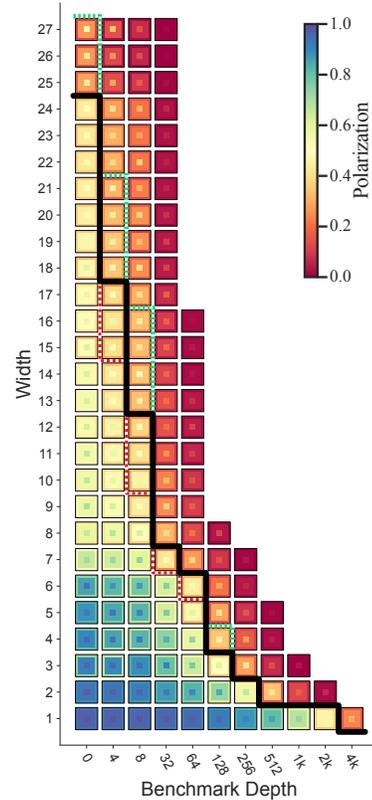


Fig. 1. **Volumetric benchmarks are not predictive.** A volumetric benchmarking plot [4] summarizing data from randomized mirror circuits (RMCs) run on `ibmq_montreal`. For each circuit width and benchmark depth (see definition in Ref. [5]), the concentric squares show the maximum (inner square), mean (middle square), and minimum (outer square) of the estimated polarizations for all the circuits of that shape that were run. The polarization (s_{pol}) of the output distribution of a definite outcome n -qubit circuit c is simply $s_{\text{pol}} = (s_{\text{sp}} - 1/2^n)/(1 - 1/2^n)$ where s_{sp} is c ’s success probability [5]. Frontiers (green, black, and red lines) show the circuit shapes at which these three statistics drop below the threshold value of $1/e$. Volumetric plots summarize a processor’s observed performance on some (necessarily small) set of benchmarking circuits $\{c\}$, but they make no predictions. In this work, we propose fitting parameterized models to benchmarking data like this, enabling predictions of the processor’s performance on other circuits.

failure (or success) rate by a simple function (e.g., the sum) of the error rates of all the operations in the circuit.

Definition 2. An error rates model E , for a capability function $s : \mathcal{C} \rightarrow \mathbb{R}$, is a capability model defined by the tuple $(\mathcal{X}, \mathcal{E}, \mathcal{N}, f)$ where:

- \mathcal{X} is a set of quantum logic operations, called basis elements;
- $\mathcal{E} = \{\epsilon_x \in \mathbb{R} \mid x \in \mathcal{X}\}$ are the model’s parameters, called error rates;
- $\mathcal{N} : \mathcal{C} \rightarrow \mathbb{N}^{|\mathcal{X}|}$ counts how many times each basis element $x \in \mathcal{X}$ appears in a circuit, defined by a rule \mathcal{R} for decomposing any circuit $c \in \mathcal{C}$ into the basis elements \mathcal{X} ;

- and $f : \mathbb{N}^{|\mathcal{X}|} \rightarrow \mathbb{R}$ is an \mathcal{E} -parameterized function that is composed with \mathcal{N} to compute the model’s prediction, i.e., $E(c) = f(\mathcal{N}(c))$.

ERMs are simple and flexible. For example, an ERM’s basis elements can contain a diverse set of circuit components—such as gates, circuit layers, or even large subroutines used in algorithms. ERMs are also scalable, i.e., it is feasible to fit ERMs to many-qubit data.

We now introduce a class of ERMs that are based on *global depolarization*, which is a simple and widely-used model for errors in quantum gates [5], [9], [10], [18]. Consider a circuit set \mathcal{C} and corresponding basis element set \mathcal{X} that contains (i) circuit sub-components (e.g., one- and two-qubit gates) that ideally implement unitary evolutions, and (ii) readout operations that only occur at the end of a circuit (so, e.g., \mathcal{X} does not include mid-circuit measurements). Now model the imperfect operation of each $x \in \mathcal{X}$ in an n -qubit circuit by the perfect operation preceded by an n -qubit depolarizing channel¹ (\mathcal{D}_{ϵ_x}) with process infidelity ϵ_x , i.e.,

$$\mathcal{D}_{\epsilon_x}[\rho] = \gamma_x \rho + [1 - \gamma_x] \mathbb{I}/2^n, \quad (1)$$

where $\gamma_x = \gamma(F_x)$, $F_x = 1 - \epsilon_x$ is the *process fidelity* with which operation x is implemented, and

$$\gamma(F) = \frac{4^n F - 1}{4^n - 1}, \quad (2)$$

is a rescaling of process fidelity called *process polarization*. Because n -qubit depolarizing channels commute with each other and every n -qubit unitary superoperator, this model predicts that the superoperator implemented when executing a circuit c is simply

$$\Lambda(c) = \left(\prod_{x \in \mathcal{R}(c)} \mathcal{D}_{\epsilon_x} \right) \mathcal{U}(c) = \left(\prod_{x \in \mathcal{X}} \mathcal{D}_{\epsilon_x}^{N_x(c)} \right) \mathcal{U}(c), \quad (3)$$

where $\mathcal{U}(c)$ is the superoperator representation of the unitary ideally implemented by c , and $N_x(c)$ is the number of instances of basis element x in c . Given a capability function s (e.g., process fidelity) and a basis element set \mathcal{X} , this model for c ’s superoperator implies an ERM for s . We now demonstrate ERMs based on Eq. (3).

B. Example 1: Predicting process fidelities

Most benchmarks run a set of circuits $\{c\}$ and then evaluate performance on each circuit c by the difference between the observed and ideal output distributions of c (quantified using, e.g., cross-entropy [17], classical fidelity [6], or heavy output probability [3]). However, stricter measures of performance can be obtained by quantifying the difference between the actual and ideal quantum processes implemented when running c [19], using, e.g., process fidelity (F)—or, equivalently, process polarization [Eq. (2)]. Benchmarks whose data $\{\hat{s}(c)\}$ consists

¹For example, if \mathcal{X} contains one- and two-qubit gates, the error channel for a 3-qubit circuit layer containing a one-qubit gate (x_1) and a two-qubit gate (x_2) is the product of two 3-qubit depolarizing channels with infidelities ϵ_{x_1} and ϵ_{x_2} . Because n -qubit depolarizing channels commute the order in which the error channels are applied is irrelevant.

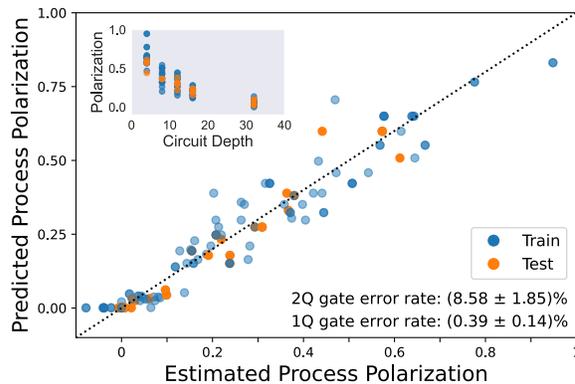


Fig. 2. **Predicting circuit process fidelities.** The predictions of a two-parameter ERM for `ibmq_kolkata` that was fit to benchmarking data consisting of estimated process polarizations (a rescaling of process fidelity) for 150 random circuits. The ERM was fit to training data (120 circuits, blue points) and assessed on holdout test data (30 circuits, orange points). The data is summarized in the inset, which shows estimated process polarization versus circuit depth. The fit values for the model’s parameters (lower right)—an error rate for one-qubit gates and an error rate for two-qubit gates—are effective error rates that summarize the data.

of estimates of process polarization (or fidelity) for a circuit set $\{c\}$ can be constructed using mirror circuit fidelity estimation (MCFE) [19]. MCFE is an efficient method for estimating the process fidelity of a circuit c by embedding c within a variety of other circuits.

We now demonstrate fitting ERMs based on the global depolarizing model [Eq. (3)] to benchmarking data that has process polarization as its capability function. The model of Eq. (3) implies that the process polarization of a circuit c is

$$E(c) = \prod_{x \in \mathcal{X}} \gamma_x^{N_x(c)}. \quad (4)$$

We fit this capability model to data from `ibmq_kolkata` that consists of estimated process polarizations for 150 12-qubit random circuits (these estimates were obtained using MCFE). The inset of Fig. 2 summarizes the data. We fit a simple ERM, with two basis elements—a one-qubit gate and a two-qubit gate—and corresponding error rates.

Figure 2 shows the predictions of the best-fit ERM (fit with a least squares objective function). We find effective one- and two-qubit error rates of $\epsilon_1 = (0.39 \pm 0.14)\%$ and $\epsilon_2 = (8.5 \pm 1.8)\%$, respectively. These best-fit error rates are approximately eight times larger than the mean of IBM’s reported one- and two-qubit gate error rates for the twelve qubits used in this experiment ($\epsilon_1 = 0.05\%$ and $\epsilon_2 = 1.1\%$) and our best-fit error rates are a better description of the data: the mean absolute error $\delta_{\text{abs}}(E) = \mathbb{E}_c |E(c) - \hat{s}(c)|$ of our two-parameter ERM with best-fit error rates is 3.0% whereas if we use IBM’s error rates it is 11.8%.

C. Example 2: Predicting circuit success probabilities

Many benchmarks (e.g., RB) run definite outcome circuits and estimate these circuits’ success probabilities—the probability that the circuit returns its single correct bit string.

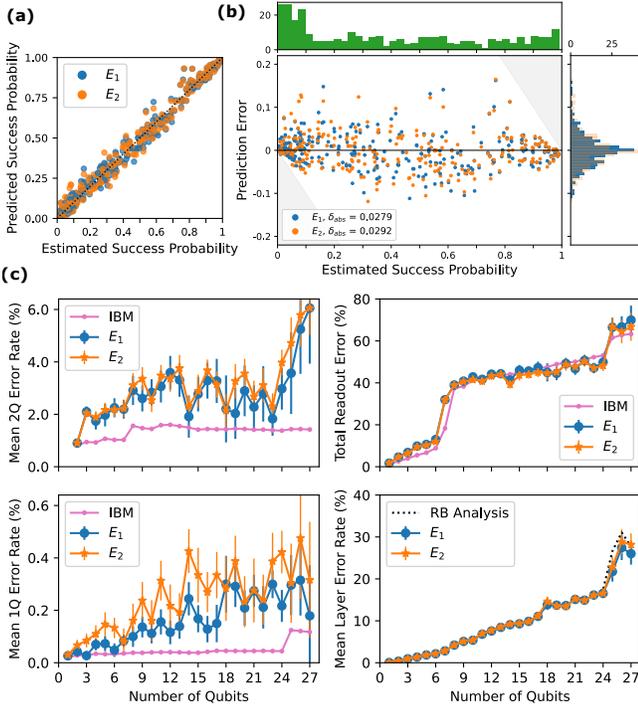


Fig. 3. **Effective error rates for `ibmq_montreal`.** We fit ERMs to 1-27 qubit random circuit data from `ibmq_montreal`. (a) The predictions for holdout test data of two ERMs E_1 (blue) and E_2 (orange) (details in the main text) versus each circuit c 's success probability $\hat{s}(c)$. Both models have moderate prediction accuracy, as summarized by (b) prediction errors, i.e., $\delta(c) = E(c) - \hat{s}(c)$ (see legend for the mean absolute error). (c) Both ERMs are parameterized by error rates that depend on circuit width. Here we show each model's average two-qubit gate error rate (top left), average one-qubit gate error rate (lower left) and total readout error (upper right) versus the circuit width (number of qubits) alongside IBM's reported error rates. IBM's average gate error rates fail to account for increasing crosstalk errors as circuit width increases, and our gate error rates are *effective error rates* that better describe the data. Each ERM's error rates can be used to compute a mean error rate ($\bar{\epsilon}_w$) for a random w -qubit circuit layer (lower right). We observe close agreement between each ERM's estimate for $\bar{\epsilon}_w$ and an independent estimate extracted from the data by a conventional RB analysis (at each w , we fit the success probabilities versus circuit depth to an exponential). All uncertainties are 1σ and are computed using a bootstrap.

The capability model for the success probability of an n -qubit definite outcome circuit c implied by Eq. (3) is simply

$$E(c) = \left(1 - \frac{1}{2^n}\right) \prod_{x \in \mathcal{X}} \gamma_x^{\mathcal{R}(c)_x} + \frac{1}{2^n}. \quad (5)$$

We demonstrate this family of capability models using data from randomized mirror circuits (RMCs) [8] run on `ibmq_montreal`, a 27-qubit system. We ran approximately 3000 circuits with varied circuit widths and depth. For each RMC we estimate its success probability.² The data is summarized in Fig. 1. For each width w , circuits were run on a single connected set of w qubits (Γ^w) and these sets were

²RMCs data is often analyzed using so-called *adjusted* success probabilities defined in Refs. [8], [9], but we do not this here for conceptual simplicity.

nested ($\Gamma^w \subset \Gamma^{w+1}$). We fit two ERMs (E_1 and E_2) with different basis elements \mathcal{X} . To investigate crosstalk errors in `ibmq_montreal` using ERMs we fit models in which each gate's error rate is indexed by circuit width. We describe this in terms of sub-ERMs: Each ERM E_i consists of 27 different sub-ERMs (E_i^w) where E_i^w makes predictions for (and is fit to the data from) circuits of width w . Each sub-ERM in E_1 is a three-parameter model, with generic one-qubit gate, two-qubit gate, and readout basis elements. In contrast, each sub-ERM in E_2 models one-qubit gates (two-qubit gates) on different qubits (qubit pairs) with independent error rates.

Figure 3 shows (a) the predictions and (b) prediction errors, of best-fit E_1 and E_2 capability models. These models were fit to training data (90% of the circuits) using maximum likelihood estimation. In Figure 3 (a-b) we evaluate the models using holdout test data (the remaining 10% of the circuits). Both models have moderately low prediction error—the mean absolute errors on the test data are $\delta_{\text{abs}}(E_1) \approx 2.8\%$ and $\delta_{\text{abs}}(E_2) \approx 2.9\%$. The additional parameters of E_2 therefore do not improve prediction accuracy on the test data. The fit error rates—the parameters of E_1 and E_2 —are summarized in Fig. 3 (c). The average one-qubit gate and two-qubit gate error rates of E_1 and E_2 (which depend on circuit width w) are in close agreement. Both one- and two-qubit gate error rates generally increase with w , becoming significantly larger than the average gate error rates reported by IBM [compare the blue and orange points with the pink points in the left column of Fig. 3 (c)]. This discrepancy suggests large crosstalk errors, as IBM's gate error rates do not include the effects of crosstalk (they are estimated using one- and two-qubit RB). Our gate error rates are *effective error rates* that better describe the data and include the impact of crosstalk. In contrast, our total readout error estimates are consistent with IBM's reported readout error rates.

IV. CASE STUDY 2: TRANSFER LEARNING WITH RESNET50

In our second case study, we create a capability model by applying transfer learning to ResNet50, a pre-trained neural network.

A. Neural network capability models

Neural networks are highly-expressive parameterized models that are general-purpose function approximators [20]. Recent work has explored using custom neural networks as capability models with promising results [10], [13], [15], [16]. However, training and tuning *de novo* neural network capability models is costly—it can be computationally intensive and can require large amounts of data. One approach to reducing the cost of creating neural network capability models is *transfer learning*. Transfer learning is a broad set of techniques designed to modify pre-trained neural networks for use on a new task [11]. Transfer learning can be particularly valuable when training data for the new task is scarce.

B. Capability models from ResNet50

To explore the feasibility of creating capability models from pre-trained neural networks, we used transfer learning

techniques to create a capability model from ResNet50 [12]. ResNet50 is a large pre-trained image classifier that consists of 48 convolutional layers, an average pooling layer, a max pooling layer, and a 1000-unit classification layer [12]. We modified ResNet50 to create a capability model by replacing ResNet50’s final 1000-unit classification layer with a single-unit dense layer that has a sigmoid activation function (so its predictions are within $[0, 1]$). Only the weights in the final single-unit dense layer are learnable parameters, i.e., all the weights in the layers from ResNet50 are fixed (frozen).

C. Encoding circuits for ResNet50

Inputting a circuit c into ResNet50 requires a representation $I(c)$ of c that is compatible with ResNet50. As an image classifier, ResNet50 processes three-dimensional tensors. We therefore input circuits into ResNet50 by modifying the three-dimensional tensor encoding $I(c)$ of circuit introduced in Ref. [10]. This encoding represents a $w \times d$ circuit c for an n -qubit device as an $n \times d_{\max}$ color image where d_{\max} is the depth of the deepest circuit in the dataset, i.e., $I(c)$ is an $n \times d_{\max} \times h$ tensor where h is the number of “color” channels ($h = 10$ for the encoding of Ref. [10]). The color channels store information about which gate is performed on each qubit in each layer of the circuit (as well as some limited information about each qubit’s error sensitivity). See Ref. [10] for details on the circuit encoding $I(c)$.

ResNet50 accepts tensors with three channels, so we must map $I(c)$ to a three-channel image $I(c)$. Our mapping consists of simply reshaping $I(c)$ (after embedding it within a larger tensor, if necessary). This reshaping destroys some locality information encoded within $I(c)$, and more principled or trainable $I(c) \rightarrow I(c)$ mappings are possible.

D. Demonstration on simulated data

To train and test our ResNet50-based capability model we used simulated data from a publicly-accessible repository [21]. The data consists of success probabilities for RMCs for a 5-qubit processor. The circuits ranged in width from 1 to 5 qubits and in depth from 3 to 1825 layers. The circuits were simulated with a stochastic Pauli errors model (detailed in Section IV A of Ref. [10]). We used training, validation, and test datasets containing 996, 664, and 1494 circuits, respectively.

We trained our model for up to 150 epochs, using binary cross-entropy (BCE) on the training data as the loss metric. Throughout training, only the weights in the final single-unit dense layer were updated (all the weights in ResNet50 were frozen). Loss on the validation dataset was monitored, training was stopped after five epochs of increasing validation loss, and the final model used the weights that minimized the validation loss (this occurred after the 144th epoch). Figure 4 (b) shows the training history.

The trained ResNet50-based capability model (E_{ResNet50}) has moderate prediction accuracy on the test data. Figure 4 (a) shows the prediction error of E_{ResNet50} for every circuit c in the test dataset, for which $\delta_{\text{abs}}(E_{\text{ResNet50}}) = 2.48\%$. This demonstrates the feasibility of using transfer learning

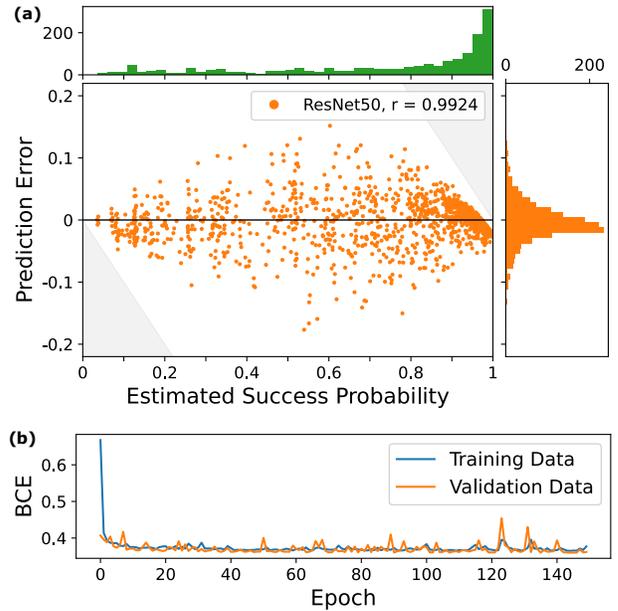


Fig. 4. **Predicting circuit success probabilities using Resnet50.** We used transfer learning to create a capability model from ResNet50. (a) The prediction error on test data of our ResNet50-based capability model versus the true success probabilities (main plot), a histogram of the success probabilities (upper plot), and a histogram of prediction error (right plot). We observe moderate prediction accuracy on the test data ($\delta_{\text{abs}} = 2.48\%$). (b) Training and validation loss versus training epoch. The fluctuations in both the training and validation loss could be caused by a high learning rate ($\alpha = .0001$).

to create capability models from large neural networks that have been trained for different prediction tasks. However, E_{ResNet50} ’s prediction accuracy is worse than has been obtained using custom *de novo* CNNs trained on data from this dataset (c.f., Ref. [10] with $\delta_{\text{abs}} \approx 0.8\%$). This suggests that transfer learning is likely to prove most useful when neural networks that have been designed and trained as capability models need to make *out-of-distribution* (OOD) predictions. Two particularly relevant examples of OOD prediction tasks are: (1) training on data from one circuit family (e.g., random circuits) and then making predicting for another circuit family (e.g., algorithmic circuits), and (2) training on data from one processor (real or simulated) and then making predictions for a difference processor. Transfer learning has the potential to greatly improve OOD predictions by fine-tuning a pre-trained network using a small amount of new OOD training data.

V. CONCLUSIONS

In this paper we have proposed a general framework for building predictive models from quantum computer benchmarking data. Our framework consists of fitting a *capability model* to benchmarking data, and it can be applied to data from a wide range of benchmarks—including RB, cross-entropy benchmarking, algorithmic benchmarks, volumetric benchmarks, and the quantum volume benchmark. Capability models encompass a broad range of parameterized models,

and we explored two interesting classes of capability model: ERM (error rates models) and neural networks.

ERMs are simple, flexible, interpretable, and scalable models that make accurate predictions when stochastic errors dominate. But even when a best-fit ERM's predictions have low accuracy, ERM's are still a powerful tool, because a best-fit ERM's parameters summarize the data in terms of effective error rates. In contrast, neural network capability models have the potential to be accurate in the presence of complex and poorly-understood errors [10] but do not have interpretable parameters. In this work, we applied transfer learning to create a capability model from ResNet50, a large, pre-trained CNN. Although we obtained lower prediction accuracy than with bespoke models (see Ref. [10]), our results demonstrate the promise and feasibility of a transfer learning approach to capability learning.

DATA AND CODE AVAILABILITY

The data and code used in the project will be available at Ref. [22]. In the meantime, please email the corresponding author.

ACKNOWLEDGEMENTS

This material was funded in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Quantum Testbed Pathfinder Program, and by the Laboratory Directed Research and Development program at Sandia National Laboratories. T.P. acknowledges support from an Office of Advanced Scientific Computing Research Early Career Award. Sandia National Laboratories is a multi-program 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 DE-NA-0003525. All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of the U.S. Department of Energy, or the U.S. Government. We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.

REFERENCES

- [1] E. Nielsen, J. Gamble, K. Rudinger, T. Scholten, K. Young, and R. Blume-Kohout, "Gate set tomography," *Quantum*, vol. 5, p. 557, 2021. [Online]. Available: <https://quantum-journal.org/papers/q-2021-10-05-557/>
- [2] M. Sarovar, T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, "Detecting crosstalk errors in quantum information processors," *Quantum*, vol. 4, p. 321, 2020. [Online]. Available: <https://quantum-journal.org/papers/q-2020-09-11-321/>
- [3] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, "Validating quantum computers using randomized model circuits," *Phys. Rev. A*, vol. 100, p. 032328, 2019. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevA.100.032328>
- [4] R. Blume-Kohout and K. Young, "A volumetric framework for quantum computer benchmarks," *Quantum*, vol. 4, p. 362, 2020. [Online]. Available: <https://quantum-journal.org/papers/q-2021-10-05-557/>
- [5] T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, "Measuring the capabilities of quantum computers," *Nature Phys.*, vol. 18, no. 1, p. 75, 2021. [Online]. Available: <https://www.nature.com/articles/s41567-021-01409-7>
- [6] T. Lubinski, S. Johri, P. Varosy, J. Coleman, L. Zhao, J. Necaie, C. H. Baldwin, K. Mayer, and T. Proctor, "Application-oriented performance benchmarks for quantum computing," *IEEE Transactions on Quantum Engineering*, vol. 4, pp. 1–32, 2023. [Online]. Available: <https://ieeexplore.ieee.org/document/10061574>

- [7] E. Magesan, J. M. Gambetta, and J. Emerson, "Scalable and robust randomized benchmarking of quantum processes," *Phys. Rev. Lett.*, vol. 106, p. 180504, 2011. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevLett.106.180504>
- [8] T. Proctor, S. Seritan, K. Rudinger, E. Nielsen, R. Blume-Kohout, and K. Young, "Scalable randomized benchmarking of quantum computers using mirror circuits," *Phys. Rev. Lett.*, vol. 129, p. 150502, 2022. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevLett.129.150502>
- [9] J. Hines, M. Lu, R. K. Naik, A. Hashim, J.-L. Ville, B. Mitchell, J. M. Kriekebaum, D. I. Santiago, S. Seritan, E. Nielsen, R. Blume-Kohout, K. Young, I. Siddiqi, B. Whaley, and T. Proctor, "Demonstrating scalable randomized benchmarking of universal gate sets," 2022, unpublished manuscript. [Online]. Available: <http://arxiv.org/abs/2207.07272>
- [10] D. Hothem, T. Catanach, K. Young, and T. Proctor, "Learning a quantum computer's capability using convolutional neural networks," 2023, unpublished manuscript. [Online]. Available: <https://arxiv.org/abs/2304.10650>
- [11] K. Weiss, T. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J Big Data*, vol. 3, no. 9, 2016. [Online]. Available: <https://doi.org/10.1186/s40537-016-0043-6>
- [12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [13] N. Elsayed Amer, W. Gomaa, K. Kimura, K. Ueda, and A. El-Mahdy, "On the learnability of quantum state fidelity," *EPJ Quantum Technology*, vol. 9, p. 31, 2022. [Online]. Available: <https://epjquantumtechnology.springeropen.com/articles/10.1140/epjqt/s40507-022-00149-8#citeas>
- [14] J. Liu and H. Zhou, "Reliability modeling of NISQ-era quantum computers," in *Proceedings of the 2020 IEEE International Symposium on Workload Characterization*, 2020, pp. 94–105.
- [15] A. Vadali, R. Kshirsagar, P. Shyamsundar, and G. Perdue, "Quantum circuit fidelity estimation using machine learning," 2022, unpublished manuscript. [Online]. Available: <https://arxiv.org/pdf/2212.00677.pdf>
- [16] H. Wang, P. Liu, J. Cheng, Z. Liang, J. Gu, Z. Li, Y. Ding, W. Jiang, Y. Shi, X. Qian, D. Pan, F. Chong, and S. Han, "Quest: Graph transformer for quantum circuit reliability estimation," in *Proceedings of the 39th International Conference on Computer-Aided Design*. ACM Press, 2022. [Online]. Available: <https://arxiv.org/pdf/2210.16724.pdf>
- [17] S. Boxio, S. Isakov, V. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. Bremner, J. Martinis, and H. Neven, "Characterizing quantum supremacy in near-term devices," *Nature Phys.*, vol. 14, pp. 595–600, 2018. [Online]. Available: <https://www.nature.com/articles/s41567-018-0124-x>
- [18] IBM Quantum, <https://quantum-computing.ibm.com>, 2021.
- [19] T. Proctor, S. Seritan, E. Nielsen, K. Rudinger, K. Young, R. Blume-Kohout, and M. Sarovar, "Establishing trust in quantum computations," 2022, unpublished manuscript. [Online]. Available: <http://arxiv.org/abs/2204.07568>
- [20] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, p. 359, 1989. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0893608089900208>
- [21] D. Hothem, T. Catanach, K. Young, and T. Proctor, <https://doi.org/10.5281/zenodo.7829489>, accessed: 2023-04-12.
- [22] D. Hothem, J. Hines, K. Nataraj, R. Blume-Kohout, and T. Proctor, <https://doi.org/10.5281/zenodo.todo>, published: TBD.