



PAPER • OPEN ACCESS

Neural network enhanced cross entropy benchmark for monitored circuits

To cite this article: Yangrui Hu *et al* 2025 *Mach. Learn.: Sci. Technol.* **6** 045030

View the [article online](#) for updates and enhancements.

You may also like

- [Towards instance-wise calibration: local amortized diagnostics and reshaping of conditional densities \(LADaR\)](#)
Biprateep Dey, David Zhao, Brett H Andrews et al.
- [An atomic cluster expansion potential for twisted multilayer graphene](#)
Yangshuai Wang, Drake Clark, Sambit Das et al.
- [Detecting model misspecification in cosmology with scale-dependent normalizing flows](#)
Aizhan Akhmetzhanova, Carolina Cuesta-Lazaro and Siddharth Mishra-Sharma



PAPER

OPEN ACCESS

RECEIVED
4 February 2025REVISED
20 April 2025ACCEPTED FOR PUBLICATION
22 October 2025PUBLISHED
4 November 2025

Original Content from
this work may be used
under the terms of the
[Creative Commons
Attribution 4.0 licence](#).

Any further distribution
of this work must
maintain attribution to
the author(s) and the title
of the work, journal
citation and DOI.



Neural network enhanced cross entropy benchmark for monitored circuits

Yangrui Hu^{1,6,*} , Yi Hong Teoh¹ , William Witzak-Krempa^{2,3,4} and Roger G Melko^{1,5} ¹ Department of Physics and Astronomy, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada² Département de Physique, Université de Montréal, Montréal, QC H3C 3J7, Canada³ Centre de Recherches Mathématiques, Université de Montréal, Montréal, QC H3C 3J7, Canada⁴ Institut Courtois, Université de Montréal, Montréal, QC H2V 0B3, Canada⁵ Perimeter Institute for Theoretical Physics, Waterloo, Ontario N2L 2Y5, Canada⁶ Beijing Institute of Mathematical Sciences and Applications, Beijing 101408, People's Republic of China

* Author to whom any correspondence should be addressed.

E-mail: huyangrui@bimsa.cn**Keywords:** quantum simulation, quantum information with trapped ions, phase transitions, artificial neural networks

Abstract

We explore the interplay of quantum computing and machine learning to advance experimental protocols for observing measurement-induced phase transitions (MIPTs) in quantum devices. In particular, we focus on trapped ion monitored circuits and apply the cross entropy benchmark recently introduced by Li *et al* (2023 *Phys. Rev. Lett.* **130** 220404), which can mitigate the post-selection problem. By doing so, we reduce the number of projective measurements—the sample complexity—required per random circuit realization, which is a critical limiting resource in real devices. Since these projective measurement outcomes form a classical probability distribution, they are suitable for learning with a standard machine learning generative model. In this paper, we use a recurrent neural network to learn a representation of the measurement record for a native trapped-ion MIPT, and show that using this generative model can substantially reduce the number of measurements required to accurately estimate the cross entropy. This illustrates the potential of combining quantum computing and machine learning to overcome practical challenges in realizing quantum experiments.

1. Introduction

Generative models have demonstrated unprecedented abilities in scaling [1] and emergence [2]. Modern generative strategies, such as autoregressive language models, are capable of learning complex probability distributions given any number of diverse data sets. Their ability to scale and generalize makes them well-suited for applications in quantum sciences [3–5], including quantum computing design [6–9], characterization [10], and control [11–14]. Recent quantum state preparation protocols, which utilize non-unitary circuits, represent the state-of-the-art in the field. However, these protocols require knowledge of probability distributions [15] of an exponentially large number of measurement outcomes, presenting a significant challenge. Despite these challenges, this field, which is often referred to as monitored circuits, poses a rich playground for the application of generative machine learning to influence the field of quantum computing in the near future.

Monitored circuits offer an opportunity to explore new physics, including phases [16–19] and phase transitions [20–26]⁷, in systems that are not typically found in nature: a combination of quantum computers and (classical) observers [20, 35, 36]. Trapped ions have become a key experimental testing

⁷ See [27–30] for discussions from various perspectives and [31–34] for theoretical derivations and interpretations.

ground for the feasibility of monitored circuit experiments [36, 37], thanks to the precise and programmable control achievable over individual qubits [38, 39] and interactions [40] in a trapped-ion quantum computer [36, 41–44]. Notably, pure quantum states can be prepared consistently [45, 46] and single-qubit measurements with a constant rate are feasible through optical addressing [47, 48].

However, there are still a number of experimental and theoretical challenges, including the post-selection problem for examining the measurement-induced phase transition (MIPT) on a quantum computer.⁸ To address these challenges, machine learning emerges as a powerful tool. The intuition is simple and natural as follows. In general, since monitored circuits are made up of both a quantum component and a classical component, the latter is a ripe playground for exploration with machine learning. Indeed, neural networks have been used to find decoders that map measurement outcomes in monitored circuits to the reference qubit configuration [54]. In this paper, we leverage neural networks to address the sample complexity issue in the cross entropy benchmark (XEB). Let us explain it in more detail.

The XEB was applied in an experimental protocol for observing MIPTs on quantum computers in [15], where the post-selection problem is mitigated. As a price to be paid, classical simulations of the random circuits in their protocol are necessary and restrict the scalability. Indeed, in a generic setting, exponentially long classical simulations are required [15].⁹ On the other hand, generative models have the potential to scale [1], which motivates us to replace classical simulations with generative models in this work. Moreover, for non-Clifford circuits, computing the cross entropy necessitates a significantly large number of measurement runs per circuit. The number of measurement runs is also closely tied to the experiment cost, which provides another practical motivation to reduce it [55].

The aim of this paper is two-fold. First, we validate the XEB for MIPT in trapped ion hybrid circuits by classical simulations. Second, we propose a recurrent neural network (RNN) enhanced protocol for verifying MIPTs on quantum computers and demonstrate its effectiveness in enhancing the benchmark. The rest of this work is organized in the following manner. Section 2 provides a review of the hybrid circuit constructed using native gates to a trapped ion device and projective measurements. In section 3, we introduce and validate the XEB for observing a MIPT in the trapped ion hybrid circuit. Section 4 is devoted to the RNN model and analyzing the sample complexity. To do so, for a given circuit, we study the number of measurement runs required to achieve a cross entropy estimation with a target accuracy. We conclude with a discussion in section 5, followed by supplementary details on numerics and RNN models presented in the appendices.

2. Trapped ion monitored circuit

In this work, we consider the quantum device consisting of L trapped atomic ions and random quantum gates native to the device as in [56]. The native two-qubit entangling gates include the so-called Mølmer–Sørensen (MS) gate [57, 58] which takes the following form

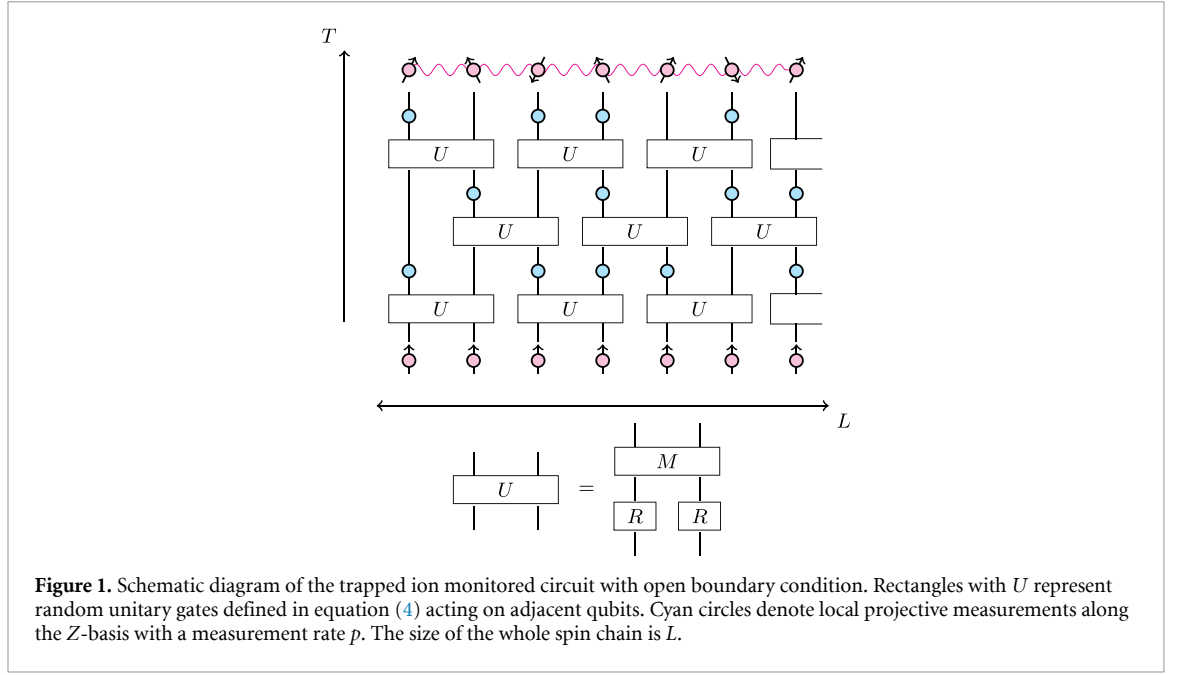
$$M_{j,k}(\theta) = \cos\theta \mathbb{I}_j \otimes \mathbb{I}_k - i \sin\theta X_j \otimes X_k, \\ = \begin{pmatrix} \cos\theta & 0 & 0 & -i\sin\theta \\ 0 & \cos\theta & -i\sin\theta & 0 \\ 0 & -i\sin\theta & \cos\theta & 0 \\ -i\sin\theta & 0 & 0 & \cos\theta \end{pmatrix}, \quad (1)$$

where j and k denote the two qubits it acts on. The above MS gate is essentially a rotation around the XX axis and can be generalized to rotations around a generic axis. Note that in conventional random quantum circuits exhibiting the MIPT, the two-qubit entangling gate is typically sampled from a specific ensemble. In contrast, although in our case randomness can be introduced by randomizing the rotation angle θ and the axis, fixing the MS gate while incorporating random single-qubit rotations is sufficient.¹⁰

⁸ With the post-selection problem, MIPT can only be observed in a quantum computer with small system sizes [49]. It has been studied extensively to address the post-selection problem via various approaches and perspectives [35–37, 50–53].

⁹ Note that for Clifford random circuits initialized with stabilizer states, this protocol is scalable [15].

¹⁰ This sufficiency was demonstrated exclusively through numerical methods in [56]. However, [59] developed a mathematical framework to study the efficiency of the random circuits. It provides a theoretical foundation for random circuit models such that the entangling gates are fixed and the randomness only comes from single-qubit gates. In particular, they considered a relevant circuit model called the ‘ironed gadget’, which consists of single-qubit Haar-random gates and a fixed 2-qubit unitary gate. They systematically studied the speed of convergence for various choices of the fixed 2-qubit gate. It would be interesting to apply this framework in the (trapped ion) hybrid circuit design and benchmarking.



In what follows, the rotation angle in equation (1) is fixed as $\theta = \pi/4$. For the random single-qubit rotations R_j

$$R_j(\theta_j, \varphi_j) = \begin{pmatrix} \cos \frac{\theta_j}{2} & -ie^{-i\varphi_j} \sin \frac{\theta_j}{2} \\ -ie^{i\varphi_j} \sin \frac{\theta_j}{2} & \cos \frac{\theta_j}{2} \end{pmatrix}, \quad (2)$$

we set $\theta_j = \pi/2$ and each φ_j is randomly chosen from one of the following three angles

$$\varphi_j \in \left\{ 0, \frac{\pi}{4}, \frac{\pi}{2} \right\}. \quad (3)$$

Note that $R(\pi/2, \varphi = 0)$ and $R(\pi/2, \varphi = \pi/2)$ correspond to 90-degree rotations about the x - and y -axes, respectively. In contrast, $R(\pi/2, \varphi = \pi/4)$ is not a Clifford gate. The random unitaries acting on two adjacent qubits can be expressed collectively as follows

$$U_{i,i+1} = M_{i,i+1} \left(\frac{\pi}{4} \right) R_i \left(\frac{\pi}{2}, \varphi_i \right) R_{i+1} \left(\frac{\pi}{2}, \varphi_{i+1} \right). \quad (4)$$

Finally, the complete monitored circuit under study is constructed in a brick-layer fashion as shown in figure 1. After each layer of unitaries, we perform single-qubit projective measurements along the Z -basis with probability p for each qubit.

As examined in [56], random trapped ion hybrid circuits with a circuit depth of $T = 4L$ yield steady states, characterized by the convergence of the half-chain entanglement entropy at $T = 4L$. Furthermore, by analyzing the dynamical behavior of entanglement entropy across varying measurement rates p , [56] identified a MIPT in this system. The critical measurement rate was determined to be $p_c \approx 0.17 \pm 0.02$.

To obtain stable numerical results for the XEB in the next section, we introduce additional encoding layers following the procedure outlined in [15]; the circuit with local measurements (figure 1) will be referred to as the bulk. Starting with a pure initial product state, we apply encoding layers consisting solely of random unitaries defined in equation (4) in the same manner as depicted in figure 1. Subsequently, the bulk circuit is applied. With additional encoding layers, the evolution of the half-chain entanglement entropy as a function of time (or circuit depth) is presented in appendix A. In what follows, we set the encoding and bulk circuit depths to $T_{\text{encoding}} = T_{\text{bulk}} = 2L$.

Note that this family of random circuits is fully specified by the arrangement of measurements and the choice of unitary gates. Thus, in what follows, we represent a circuit C using its corresponding set of φ_i 's and the positions of measurements s_i 's, as follows:

$$C = \left\{ \left\{ \varphi_i^{(t)} \right\}_{i=1, \dots, L}^{t=1, \dots, T}; s_1, s_2, \dots, s_N \right\}, \quad (5)$$

where $T = T_{\text{encoding}} + T_{\text{bulk}}$ and $s_i = (q_i, t_i)$ represents the measurement site, i.e. specifies the qubit q_i measured at time step t_i . Here, N denotes the total number of measurements in the circuit, which varies across different realizations. The sequence $\{s_i = (q_i, t_i)\}$ is ordered by iterating through the spin chain at each time step, repeating the process for subsequent time steps.

3. XEB

The idea of the XEB for studying the MIPT [15] is based on the fact that in the volume law phase, the ‘coding’ property of the system ensures bulk measurements extract little information about the initial state. This is due to strong scrambling in the low- p phase. Conversely, in the area law (high- p) phase, some information about the initial state propagates into the measurement outcomes, and to some extent one is able to distinguish different initial states from bulk measurements. Thus the MIPT can be probed by comparing the probability distributions formed by measurement outcomes in circuits with different initial states as follows.

First, let us set up the notation: For the circuit with an initial state ρ , its measurement outcomes $\vec{m} = \{m_1, m_2, \dots, m_N\}$ follow a probability distribution and we denote it as $p_{\vec{m}}^{\rho}$. Now we consider a circuit C and input two different initial states, ρ and σ . The difference between their corresponding probability distributions $p_{\vec{m}}^{\rho}$ and $p_{\vec{m}}^{\sigma}$ can be quantified using the cross entropy χ_C defined in equation (6). The circuit averaged cross entropy $\chi = \mathbb{E}_C \chi_C$ serves as an order parameter for the MIPT. In the volume law phase ($p < p_c$), $\chi = 1$, indicates that based on measurement outcomes one is not able to distinguish different initial states. In contrast, in the area law phase ($p > p_c$), χ takes on a constant value less than 1, signifying that information about the initial states is accessible from the bulk measurements.

The cross entropy for a given circuit C is defined as follows

$$\chi_C = \frac{\sum_{\vec{m}} p_{\vec{m}}^{\rho} p_{\vec{m}}^{\sigma}}{\sum_{\vec{m}} p_{\vec{m}}^{\sigma} p_{\vec{m}}^{\sigma}} = \frac{\langle p_{\vec{m}}^{\sigma} \rangle_{\rho}}{\langle p_{\vec{m}}^{\sigma} \rangle_{\sigma}}, \quad (6)$$

where in the first equality, the sums run over all possible measurement outcome $\vec{m}$'s. For a generic circuit, the total number of possible outcomes grows exponentially with the system size and measurement rate, scaling as 2^{pLT} . Due to this exponential growth, directly evaluating the cross entropy is computationally infeasible. Instead, we approximate the numerator and denominator independently using the law of large numbers [15]. Namely,

$$\begin{aligned} \langle p_{\vec{m}}^{\sigma} \rangle_{\rho} &= \lim_{M \rightarrow \infty} \frac{1}{M} \sum_{j=1, \vec{m}_j \sim p_{\vec{m}}^{\rho}}^M p_{\vec{m}_j}^{\sigma}, \\ \langle p_{\vec{m}}^{\sigma} \rangle_{\sigma} &= \lim_{M \rightarrow \infty} \frac{1}{M} \sum_{k=1, \vec{m}_k \sim p_{\vec{m}}^{\sigma}}^M p_{\vec{m}_k}^{\sigma}, \end{aligned} \quad (7)$$

where $\vec{m}_j \sim p_{\vec{m}}^{\rho}$ means that the outcome $\vec{m}_j$ is sampled from the probability distribution $p_{\vec{m}}^{\rho}$.

With the help of classical simulations, [15] proposed an experimental protocol for observing MIPT on a quantum device. The procedure is outlined as follows. (1) Run the ρ -circuit: initialize a quantum computer with the state ρ and execute the random circuit. Record the circuit parameters C defined in equation (5) and measurement outcomes $\vec{m}$. (2) Simulate the σ -circuit: on a classical computer, simulate the circuit C with a different initial state σ , replacing the projective measurements with their corresponding projector operators applied based on $\vec{m}$ (without normalization). Thus the probability $p_{\vec{m}}^{\sigma}$ for obtaining measurement result $\vec{m}$ in the σ -circuit is simply the trace of the final state density matrix. For pure states, this corresponds to the norm of the final state. In what follows, we consider¹¹

$$\rho = |+\rangle^{\otimes L}, \quad \sigma = |0\rangle^{\otimes L}. \quad (8)$$

(3) Compute χ_C : perform M measurement runs on the quantum computer using the same circuit C . Compute the numerator in equation (6) as the mean of $p_{\vec{m}}^{\sigma}$ across all runs. Similarly, calculate the denominator in equation (6) and use these results to determine the cross entropy χ_C for the circuit C . (4) Average over circuits: repeat the procedure for M_C sampled ρ -circuits and compute the average of χ_C across these circuits.

¹¹ These initial states can be prepared with high fidelities in trapped-ion systems using optical pumping [60], with fidelity > 99.9% [39], and single-qubit rotation gates with fidelity > 99.9999% [61] applied globally.

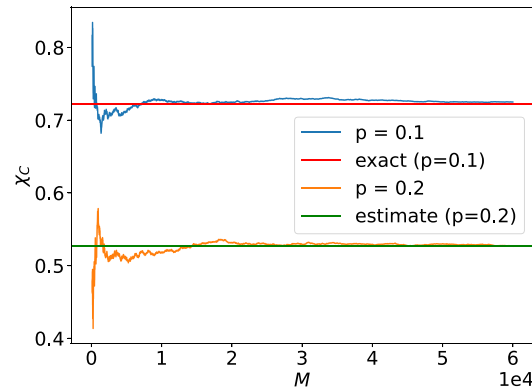


Figure 2. For two given circuits with $L = 8$ and different p 's, the evolution of the cross entropy χ_C estimated using equation (7) as a function of the number of measurement runs M . For $p = 0.1$, the exact value of χ_C is plotted in red. For $p = 0.2$, the estimation of χ_C at $M = 6 \times 10^3$ is plotted in green for reference.

This protocol is scalable for stabilizer circuits [15], while for generic non-stabilizer circuits, the scalability is restricted by classical simulations. Additionally, the sample complexity—namely, the number of measurement runs M required to obtain an accurate estimation of the cross entropy as defined in [55]—is relatively unexplored. In this work, we take the first step toward addressing it. As shown in equations (6) and (7), the XEB relies solely on measurement outcomes and their associated probabilities. This feature naturally aligns with the framework of language models, as we will see in the next section. Before delving into the implementation and advantages of using language models, in the rest of this section, we validate the XEB for the hybrid circuit of interest introduced in section 2 using classical simulations.

Note that in stabilizer circuits, when both initial states are stabilizer states, χ_C can be computed exactly since equation (7) has a closed form. Effectively, this corresponds to having an infinite number of measurement runs, $M = \infty$. However, for the trapped ion hybrid circuits, which are non-stabilizer circuits, it is necessary to first study the behavior of the cross entropy χ_C for a given circuit as a function of the number of measurement runs M . To do so, we consider a small system size $L = 8$ and simulate two circuits generated with measurement rates $p = 0.1$ and $p = 0.2$ respectively. The corresponding number of projective measurements is $N = 12$ for $p = 0.1$ and $N = 24$ for $p = 0.2$. For the $p = 0.1$ circuit, the relatively small N allows us to compute the exact value of χ_C by summing over all $2^N = 4096$ possible measurement outcomes as in equation (6). This exact value will serve as a benchmark to evaluate the accuracy of the estimations in the next section. For the $p = 0.2$ circuit, we focus solely on the trend of χ_C . As shown in figure 2, in both cases, χ_C initially fluctuates before gradually converging to a constant value as M increases. In the following simulations, we set $M = 5 \times 10^3$ to compute the cross entropy χ_C for each circuit using equation (7). This value is selected based on convergence trends: despite being relatively small (to save computational resources), it provides a reliable estimation.¹²

We next simulate hybrid circuits with various system sizes L and compute the cross entropy χ following the procedure outlined above. In particular, for each system size, we simulate $M_C = 100$ circuits, and for each circuit configuration, we sample $M = 5 \times 10^3$ measurement outcomes for both ρ - and σ -circuits. The numerical results are shown in figure 3, which perform similar behavior as that of Haar random circuits for small system sizes examined in [15]. Ideally, in the limit where M , M_C , and L approach infinity, χ would behave as a step function of the measurement rate p : $\chi = 1$ for $p < p_c$ and dropping to a constant at the critical point p_c . However, due to finite-size effects, the $p - \chi$ curves for different L 's are expected to exhibit a downward slope while intersecting at the critical point, which is indeed observed in figure 3. This crossing behavior confirms that the XEB is effective for trapped ion hybrid circuits and the critical measurement rate is estimated to be $p_c \approx 0.15$ based on the plot, which is consistent with [56].

The codes for the classical simulations that generate random circuit realizations, compute the cross entropy, and sample the measurement outcomes are available on [GitHub](#).

¹² We have also verified the χ_C - M dependence for $L = 10$ and $L = 16$ and various values of p . In all cases, $M = 5 \times 10^3$ is an appropriate choice.

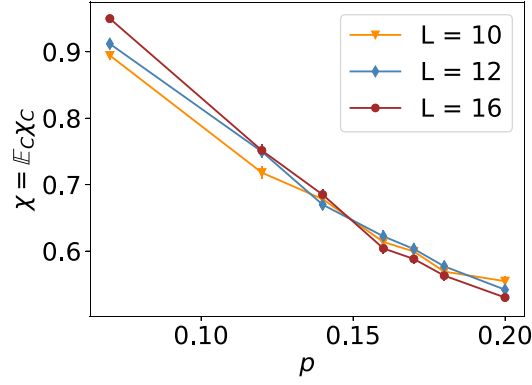


Figure 3. Numerical results of the circuit-averaged cross entropy χ for trapped ion hybrid circuits with initial states $\rho = |+\rangle^{\otimes L}$ and $\sigma = |0\rangle^{\otimes L}$. For each circuit, we estimate the cross entropy χ_C following equation (7) and using $M = 5 \times 10^3$ measurement runs. This value of M is selected based on convergence trends. Then we average over $M_C = 100$ circuits to estimate χ . Error bars represent one standard error of the cross entropy expectation value, averaged over 100 independent circuits.

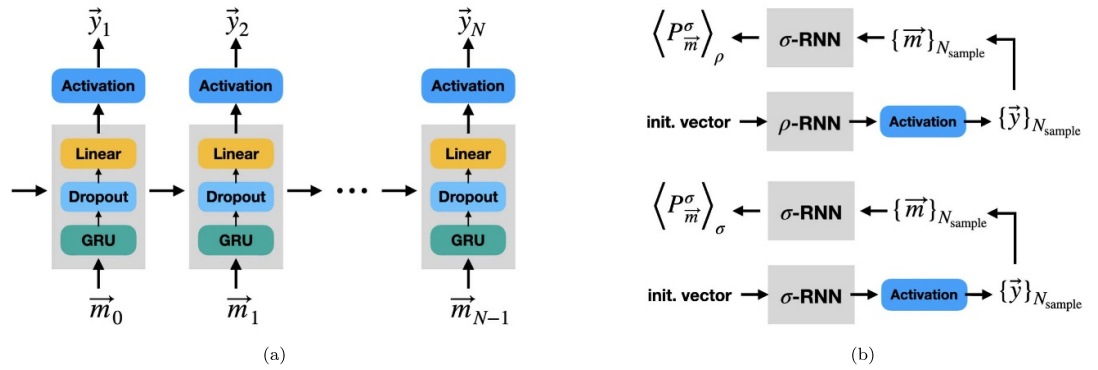


Figure 4. (a): A schematic diagram of the RNN model used in this work. The gray-shaded block represents the RNN cell, which processes a sequence of inputs $\{\vec{m}\}$, where each input vector has a dimension of $N_i = 2$. Here $\vec{m}_0 = (0, 0)$ is an initial vector to kick off the model. Within the RNN cell, the GRU layer generates a hidden state with a dimension of N_h . After passing through a Softmax activation layer, the output vector $\vec{y}_i$ (dimension $N_o = 2$) corresponds to the conditional probability distribution, as described in equation (9). (b): The RNN estimation procedure for the numerator and denominator in equation (6). The autoregressive property of the model yields a recursive sampling process as follows. An initial zero vector $\vec{m}_0$ is fed into the RNN, producing the conditional probability $\vec{y}_1$ for the first site. Based on this probability, a configuration is sampled at the first site. This outcome is then used to sample the second site based on $\vec{y}_2$, and so on.

4. RNN implementation

One core aspect of language models is their ability to process sequences of text or data in a probabilistic manner. Given the preceding context, these models generate the next token in a sequence based on a learned conditional probability distribution. Modern architectures for sequence processing, such as RNNs and transformers, have been studied extensively on their applicability to representing quantum states [5, 6, 62–65]. In this work, we apply RNNs to learn and infer the true probability distributions for both ρ - and σ -circuits using the measurement outcomes generated via classical simulations in section 3. Specifically, we focus on the same two circuits with $L = 8$ simulated to obtain results in figure 2. In what follows, the estimation method described below equation (7) is referred to as the histogram approach.

Below we first introduce the RNN model for learning probability distributions in section 4.1 and then demonstrate its implementation in the XEB in section 4.2. Finally, in section 4.3, we analyze the data complexity by comparing the number of measurement runs required for accurate estimation of χ_C using the histogram approach versus RNNs.

4.1. Vanilla RNNs for probability distributions

We consider a similar architecture for simple (vanilla) RNNs as in [62]. The schematic diagram of this model is presented in figure 4(a).

The input data is a one-hot encoding of the measurement outcomes $\vec{m} = \{m_1, m_2, \dots, m_N\}$ in the monitored circuit. Here, N is the total number of measurements in a given circuit C , as defined in

equation (5), and each measurement outcome m_i takes a value of either 0 or 1 for qubits. Through one-hot encoding, 0 and 1 are mapped to $(1,0), (0,1)$ respectively, making the input dimension $N_i = 2$. We use the vector notation $\vec{m}_i$ to denote the one-hot encoding of m_i . As shown by the gray-shaded block in figure 4(a), the building block of this model, called the RNN cell, comprises the following: (1) a gated recurrent unit (GRU) layer¹³ that generates a hidden state with dimension N_h ; (2) a dropout layer for regularization to prevent overfitting; (3) a linear layer that maps hidden states to output logits with a dimension of $N_o = 2$. Subsequently, a Softmax activation layer converts the logit into $\vec{y}_i$, corresponding to the conditional probability distribution of the outcomes at site i . Namely, $\vec{y}_i$ is a two-dimensional vector with non-negative real components that sum to one. Given the preceding measurement record $\{m_1, \dots, m_{i-1}\}$, the conditional probability of obtaining m_i at site i reads

$$P(m_i | m_1, m_2, \dots, m_{i-1}) = \vec{y}_i \cdot \vec{m}_i, \quad (9)$$

where $\cdot$ is the inner product of two 2-dimensional vectors. The probability of observing the full configuration $\vec{m}$ is the production of a sequence of these conditional probabilities. Namely,

$$P(m_1, m_2, \dots, m_N) = \prod_{i=1}^N \vec{y}_i \cdot \vec{m}_i. \quad (10)$$

To learn the probability distribution $p_{\vec{m}}$, we train the model using a set of outcome configurations $\{\vec{m}\}$ sampled from $p_{\vec{m}}$. We employ the Negative Log Likelihood as the loss function

$$\mathcal{L} = -\frac{1}{N} \text{avg}_{\{\vec{m}\}} \log P(\vec{m}), \quad (11)$$

where the loss is averaged over both the training batches and the measurement sites. This loss function evaluates how closely the model's predicted probability distribution matches the true one. The source codes for vanilla RNN models are available on [GitHub](#).

4.2. RNN-enhanced protocol

The model introduced above is capable of learning any classical probability distribution. In the context of the XEB, RNNs can play the role of a decoder, mapping the measurement outcomes in a monitored circuit C initialized with two different states to its corresponding cross entropy χ_C . The RNN-enhanced experimental protocol is outlined as follows.

1. Run ρ - and σ -circuits: Sample a random circuit C with a given system size L and measurement rate p . Execute this circuit C independently with initial states ρ and σ on quantum computers, recording the measurement outcomes for both ρ - and σ -circuits.
2. Train ρ - and σ -RNNs: Use the respective sets of measurement outcomes to train two separate vanilla RNNs. The first, denoted as ρ -RNN, learns the probability distribution $p_{\vec{m}}^\rho$, while the second, called σ -RNN, learns $p_{\vec{m}}^\sigma$.
3. Compute χ_C : With the trained RNNs, sample N_{sample} outcomes $\{\vec{m}\}_{N_{\text{sample}}}$ from ρ -RNN and feed them into σ -RNN. The mean of the output probabilities, defined in equation (10), provides an estimate of the numerator in equation (6). Similarly, calculate the denominator in equation (6) and obtain χ_C . This estimation procedure is depicted in figure 4(b).
4. Average the cross entropy over circuits.

This protocol replaces the classical simulations in [15] with trained RNNs. Consequently, the scalability of this approach hinges on the scalability of the RNNs, which will be discussed in section 5. In the following, we demonstrate the advantages of this protocol by analyzing the data complexity.

4.3. Training results and data complexity

As guiding examples, we revisit the two circuits with system size $L = 8$ and measurement rates $p = 0.1$ and $p = 0.2$ studied in section 3. For different sizes of the training data, we independently train separate RNNs and tune their model parameters. The final RNN model parameters used in this study are provided in appendix B.1 and training costs are discussed in appendix B.3.

¹³ A GRU [66] layer employs gating mechanisms, including the reset and update gates, to selectively retain and update information from preceding inputs. It is the key component that enables the autoregressive feature in this RNN.

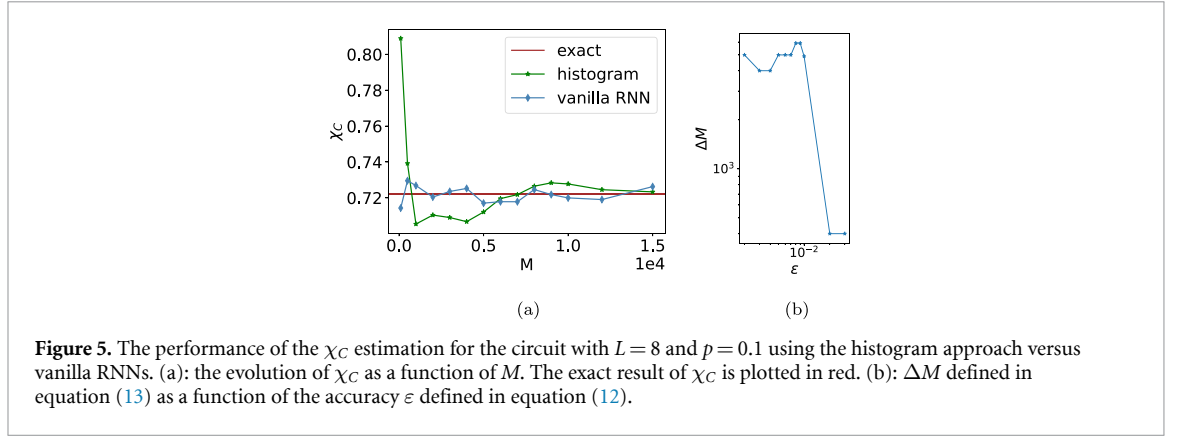


Figure 5. The performance of the χ_C estimation for the circuit with $L = 8$ and $p = 0.1$ using the histogram approach versus vanilla RNNs. (a): the evolution of χ_C as a function of M . The exact result of χ_C is plotted in red. (b): ΔM defined in equation (13) as a function of the accuracy ε defined in equation (12).

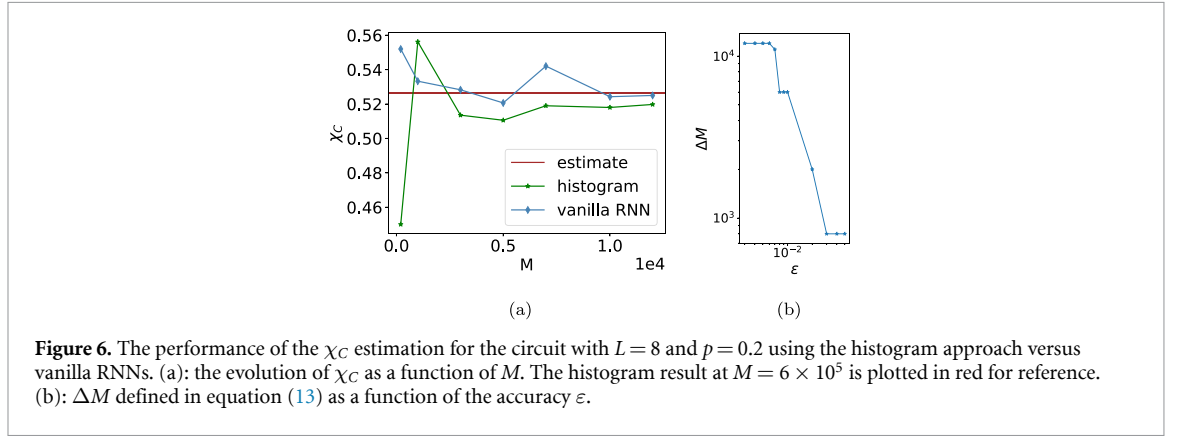


Figure 6. The performance of the χ_C estimation for the circuit with $L = 8$ and $p = 0.2$ using the histogram approach versus vanilla RNNs. (a): the evolution of χ_C as a function of M . The histogram result at $M = 6 \times 10^5$ is plotted in red for reference. (b): ΔM defined in equation (13) as a function of the accuracy ε .

Before diving into the data complexity analysis, we validate the idea of using RNNs to approximate the true distribution from its fuzzy estimate, which is the frequency count of the measurement outcomes. To do so, in appendix B.2, we compare probability distributions formed by trained RNN models and the raw data. Indeed, the trained RNNs capture the distributions formed by their training data.

To analyze the data complexity, we compare the evolution of estimated χ_C as a function of the number of measurement runs M using the histogram approach versus RNNs. For each value of M , the same set of measurement outcomes, generated via classical simulations, is used to train the RNNs. The results for $p = 0.1$ and $p = 0.2$ are shown in figures 5(a) and 6(a) respectively.¹⁴ In both cases, the values of χ_C obtained using the RNN and histogram approaches fluctuate around a constant value at which they converge. The RNN-enhanced protocol exhibits smaller fluctuations, highlighting its advantages, which we will quantify shortly.

Recall that the total number of measurements N is $N = 12$ for $p = 0.1$ and $N = 24$ for $p = 0.2$. For the $p = 0.1$ circuit, we have computed the exact value of χ_C , enabling us to define the accuracy ε of the estimation as the difference between the exact and estimated values:

$$\varepsilon := |\chi_C^{\text{exact}} - \chi_C|. \quad (12)$$

To assess the performance of each estimation approach, we determine the minimum number of measurement runs $M_{\min}$ required to achieve a target accuracy ε .¹⁵ The improvement offered by the RNN-enhanced protocol is quantified by the relative reduction in measurement runs:

$$\Delta M := M_{\min}^{\text{histogram}} - M_{\min}^{\text{RNN}}, \quad (13)$$

which is analyzed for various accuracy targets ε in figure 5(b). For the $p = 0.2$ circuit, where calculating the exact value of χ_C is infeasible, we treat the steady value of χ_C at $M = 6 \times 10^5$ as the exact value

¹⁴ Although we focus on two example circuits here, similar results can be obtained for other circuit realizations. In particular, we examined two additional circuit realizations with $L = 8$, $p = 0.1$, and $N = 12$, and considered training data with sizes of $M = 10^4$ and $M = 2 \times 10^3$. It was found that the corresponding trained RNNs all yield $\varepsilon \in \mathcal{O}(10^{-3})$, consistent with the results obtained from the example circuit presented in the main text.

¹⁵ This is an analog of the sample complexity defined in [55].

for the purpose of evaluating accuracy. The result is shown in figure 6(b). In both cases (figures 5(b) and 6(b)), ΔM is always positive, indicating that the RNN-enhanced protocol outperforms the histogram approach. In particular, it significantly reduces the number of measurement runs required to achieve a given accuracy. For high accuracy (i.e. small ε), the improvement is substantial, with reductions on the order of 10^3 for $p=0.1$ and 10^4 for $p=0.2$. As the accuracy requirement decreases (i.e. ε increases), ΔM also drops. This aligns with intuitions: When accuracy is relaxed, both $M_{\min}^{\text{histogram}}$ and $M_{\min}^{\text{RNN}}$ are reduced, resulting in a smaller difference between them. Namely, the stricter the accuracy requirement, the greater the advantage of the RNN-enhanced protocol over the histogram approach.

5. Discussion

In this work, we examine the XEB for a MIPT in trapped ion monitored circuits. We propose a RNN strategy to enhance the XEB signal obtained in a data-limited setting, with the goal of exploring how generative machine learning models can assist in quantum computer benchmarking and control. We will close with several remarks and natural routes for future investigations.

When evaluating the computational efficiency of a machine learning protocol, two primary aspects are typically considered: computational resources and sample complexity. In this work, our focus has been on the latter, which is crucial in the case of MIPTs, since the calculation of the XEB requires knowledge of the probability distribution underlying the measurement record. We have demonstrated that using an RNN-based generative model can improve the data efficiency of the XEB calculation, as compared to using a parameter-free distribution simply obtained from the data frequency counts. In this paper, we have focussed on a demonstration using vanilla RNNs on two specific circuits with simplified parameter settings. A more systematic assessment of the RNN-enhanced protocol in terms of sample complexity is left for future work. For instance, further exploration of the learnability [67] of the distributions generated by such monitored quantum circuits would be an exciting direction.

Like in many machine learning strategies, the power of generative models lies in their broad applicability, scalability, and ability to generalize well to unseen data. The scalability of our RNN-enhanced protocol depends on extending the generative models to handle a larger number of qubits N . Among generative models, autoregressive language models like the RNN are particularly promising in this regard due to their demonstrated abilities to scale. Notably, the current state-of-the-art language model is the attention-based transformer [64, 68–73]. Further investigation is necessary to evaluate different autoregressive models and assess their efficiency in the context of our problem.

Regarding potential generalizations in the model architecture, one natural direction would be to incorporate the circuit parameters as an encoder, enabling the machine learning model to learn an entire family of circuits rather than being restricted to a single probability distribution. For instance, the current input data is a flattened sequence of measurement outcomes without any information about the measurement positions. To better capture spatial and temporal correlations, as in [54], one could adopt the input as an $L \times T$ matrix that records the measurement configuration at each site: entries of 0 indicate no measurement, while ± 1 correspond to the measurement outcomes.

Note that in this work we use synthetic data to train the RNN models. As a next step toward benchmarking quantum devices, it would be interesting to introduce noise models in the simulations. One natural starting point could be incorporating a natural noise model for the trapped ion devices discussed in [56]. Additionally, conducting simulations with larger system sizes and training the models on real experimental data will be key to demonstrating the effectiveness of our protocol in the future.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: <https://github.com/yhu1996/RNN-enhanced-XEB> [74].

Acknowledgments

Numerical simulations and machine learning in this work were enabled in part by support provided by the Digital Research Alliance of Canada (alliancecan.ca). The learning results were produced using the Pytorch package [75]. YH is supported by an Alliance Quantum Program Grant funded by NSERC, for the project entitled ‘A new generation of hardware efficient superconducting qubits’. YH thanks Zi-Wen Liu for the valuable discussions. WWK is supported by a Grant from the Fondation Courtois, a Chair of the Institut Courtois, a Discovery Grant from NSERC, and a Canada Research Chair. RGM is supported by NSERC and the Perimeter Institute for Theoretical Physics. Research at the Perimeter Institute is

supported by the Government of Canada through the Department of Innovation, Science and Industry Canada and by the Province of Ontario through the Ministry of Colleges and Universities.

Appendix A. Benchmark for the circuit depth

In this appendix, we study the evolution of the half-chain von Neumann entropy to determine the time (or equivalently the circuit depths) required for reaching steady states in the encoding-bulk circuits described in section 2.

The von Neumann entropy for a subsystem A is defined as follows

$$S_A = -\text{Tr}(\rho_A \log \rho_A), \quad (\text{A1})$$

where $\rho_A = \text{Tr}_{\bar{A}} \rho$ is the reduced density matrix obtained by tracing out the complement of the subsystem A . For this study, A is chosen as the first $L/2$ qubits. Since we focus on small system sizes in this work, S_A is simply computed using the exact diagonalization of the density matrix. The time evolution of the half-chain entanglement entropy for $L = 8, 10, 12, 14, 16$ is shown in figure 7. Here we consider two measurement rates: $p = 0.1$ and $p = 0.2$. For each case, we simulate $M_C = 100$ circuits, and their half-chain entanglement entropies S_A are averaged over these independent circuits. Moreover, note that the circuits follow a brick-wall layout, distinguishing between even and odd time steps. To further smooth the results, we average S_A over adjacent odd and even time steps.

As shown in figure 7, for all systems considered, the half-chain von Neumann entropy S_A first increases, then decreases, and eventually converges to a constant value. This behavior is consistent with the encoding-bulk circuit architecture as follows. In the encoding period, entangling unitaries continuously scramble the information, causing S_A to increase. After turning on the measurements in the bulk period, occasional measurements lower the S_A . Moreover, figure 7 demonstrates that a circuit depth of $T_{\text{encoding}} = T_{\text{bulk}} = 2L$ is sufficient to generate steady states; therefore, we adopt $T_{\text{encoding}} = T_{\text{bulk}} = 2L$ in the simulations presented in section 3.

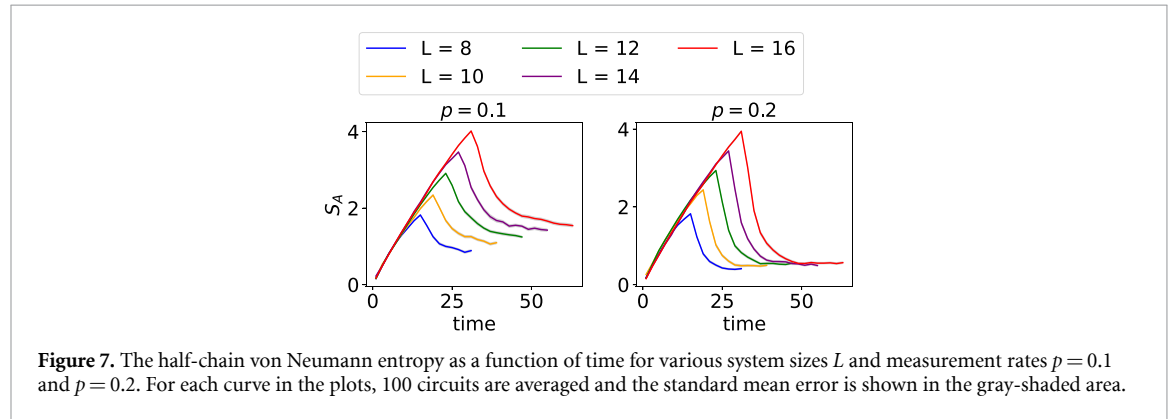


Figure 7. The half-chain von Neumann entropy as a function of time for various system sizes L and measurement rates $p = 0.1$ and $p = 0.2$. For each curve in the plots, 100 circuits are averaged and the standard mean error is shown in the gray-shaded area.

Appendix B. RNN models

B.1. Model parameters

This appendix provides the model parameters for the RNNs used in section 4.3. For the $p = 0.1$ circuit, the parameters are listed in table 1, and those for the $p = 0.2$ circuit are listed in table 2. For simplicity, at each M , the same set of model parameters is used for both ρ - and σ -RNNs, with Pytorch's default initialization applied [75]. The parameters listed in the tables are explained as follows.

In both cases, the learning rate is fixed at 10^{-3} throughout the training process. In the training, the dataset with size M is divided into a training set and a validation set. During each training epoch, the dataset is further split into batches to calculate the gradient of batch loss with respect to all parameters. The batch size and the validation set size are listed in the tables. As defined in section 4.1, N_h denotes the dimension of the hidden state, and the dropout rate for the dropout layer is also provided in the tables. As introduced in section 4.2, N_{sample} is the number of configurations we sample from the trained RNNs to compute the cross entropy. Recall that for the $p = 0.1$ circuit, we compute the χ_C by summing

Table 1. Model parameters of the RNNs for the $L = 8, p = 0.1$ circuit with various dataset sizes M . For detailed explanations of these parameters, refer to the main text.

M	Batch size	Validation size	N_h	Dropout rate
15 000	1000	3000	20	0.2
12 000	1000	2000	18	0.2
10 000	1000	2000	14	0.1
9000	1000	2000	13	0.1
8000	1000	1000	12	0.2
7000	1000	1000	12	0.2
6000	1000	1000	12	0.2
5000	1000	1000	12	0.35
4000	1000	1000	12	0.6
3000	600	600	10	0.5
2000	400	400	9	0.5
1000	200	200	7	0.6
500	100	100	5	0.6
100	20	20	3	0.6
p	Learning rate		N_{sample}	N_{epochs}
0.1	10^{-3} (constant)		∞	60 000

Table 2. Model parameters of the RNNs for the $L = 8, p = 0.2$ circuit with various dataset sizes M . For detailed explanations of these parameters, refer to the main text.

M	Batch size	val. size	N_h	p_{dropout}	N_{epochs}	N_{sample}
12 000	1000	2000	19	0.1	40 000	20 000
10 000	1000	2000	18	0.1	40 000	20 000
7000	1000	1000	18	0.2	40 000	50 000
5000	1000	1000	17	0.2	40 000	20 000
3000	600	600	14	0.2	40 000	50 000
1000	200	200	12	0.4	60 000	50 000
200	40	40	6	0.5	60 000	50 000

over all possible configurations as in equation (6). Namely, $N_{\text{sample}} = \infty$. The training time, represented as the total number of training epochs, is denoted as N_{epochs} in the tables.

B.2. Probability distributions

In the main text, we focus on the cross entropy estimated by the raw data and associated RNN models. As a sanity check, in this appendix, we compare the probability distributions formed by measurement outcomes and their corresponding trained RNNs.¹⁶ To do so, we consider the same circuit studied in section 4 with $N = 12$. Then the total number of possible output configurations is 4096 and we are able to graphically compare the two probability distributions obtained by the frequency count from the measurement data versus the trained RNN.

As a guiding example, we set the number of measurement runs to $M = 10^4$, and the RNNs trained in section 4. Blue lines in figure 8 show the frequency plots, representing the probability distributions formed by measurement data. Here, no renormalization is employed, namely the total count sums to $M = 10^4$, and the x -axis displays 2^{12} possible configurations. For the RNN models, after the training, one is able to sample as many configurations as desired and then generate a frequency plot of the results. However, recall that the output of the forward step in RNNs is the conditional probability, as explained in equation (9). We can directly compute the corresponding probability by inputting the configuration, without any sampling procedure. To compare with the training data, we further renormalize the result by a factor of M . The plots are shown as the orange curves in figure 8, demonstrating that the trained RNNs indeed capture the probability distribution formed by their training data.

Note that the frequency distributions (blue lines) are very fuzzy and the trained RNNs are anticipated to defuzzify them. Besides direct comparison through plots, we can also compute the cross entropy between the blue and orange distributions to get a quantitative sense, although we do not expect a very high cross-entropy score. The cross entropy for P_m^ρ 's is 0.836, and the cross entropy for P_m^σ is 0.851.

¹⁶ We thank one of the referees for suggesting this analysis.

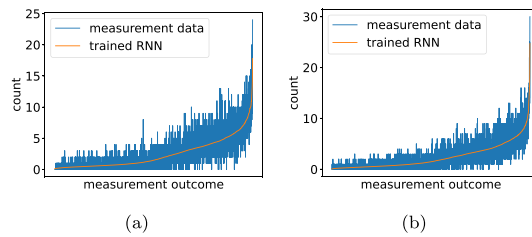


Figure 8. Probability distributions formed by frequency approach from the measurement outcomes (in blue) and the trained RNN models (in orange). For the same set of parameters, (a) shows the result for the ρ -circuit, and (b) shows the result for the σ -circuit.

B.3. Training costs

Although the primary focus of this paper is not on reducing the overall computational resources, this appendix provides supplementary details on the training costs. The computational resources required for the RNN scheme proposed in this paper depend on the model parameters and the cross entropy computation. Moreover, the hardware and additional code optimizations (such as parallel computation) play a significant role in determining the job's wall time.¹⁷

For reference, below we list the training cost for two training examples: $N = 12$ and $N = 24$ circuits with $M = 10^4$ and RNN models have parameters listed in appendix B.1. (1) For $N = 12$, we compute the cross entropy every 10 epochs. On a single A100 node with one GPU and one CPU, the CPU time is 11 h, 18 min, and 56 s, which is also the job's wall time.¹⁸ The memory usage is 520.81 MB. (2) For $N = 24$, we compute the cross entropy every 50 epochs by sampling $N_{\text{sample}} = 2 \times 10^4$ configurations. On a single A100 node with one GPU and one CPU, the CPU time is 18 h, 47 min, and 30 s; and the memory usage is 7.76 GB.

ORCID iDs

Yangrui Hu  0000-0002-3548-8574

Yi Hong Teoh  0000-0003-0692-3327

William Witzak-Krempa  0000-0003-2350-2583

Roger G Melko  0000-0002-5505-8176

References

- [1] Kaplan J, McCandlish S, Henighan T, Brown T B, Chess B, Child R, Gray S, Radford A, Wu J and Amodei D 2020 Scaling laws for neural language models (arXiv:2001.08361) [cs.LG]
- [2] Wei J et al 2022 Emergent abilities of large language models (arXiv:2206.07682)
- [3] Carrasquilla J 2020 Machine learning for quantum matter *Adv. Phys. X* **5** 1797528
- [4] Dawid A et al 2022 Modern applications of machine learning in quantum sciences (arXiv:2204.04198) [quant-ph]
- [5] Melko R G and Carrasquilla J 2024 Language models for quantum simulation *Nat. Comput. Sci.* **4** 11
- [6] Carrasquilla J, Torlai G, Melko R G and Aolita L 2019 Reconstructing quantum states with generative models *Nat. Mach. Intell.* **1** 155
- [7] Torlai G and Melko R G 2020 Machine-learning quantum states in the NISQ era *Annu. Rev. Condens. Matter Phys.* **11** 325
- [8] Sweke R, Kesselring M S, van Nieuwenburg E P and Eisert J 2020 Reinforcement learning decoders for fault-tolerant quantum computation *Mach. Learn.: Sci. Technol.* **2** 025005
- [9] Teoh Y H, Drygala M, Melko R G and Islam R 2020 Machine learning design of a trapped-ion quantum spin simulator *Quantum Sci. Technol.* **5** 024001
- [10] Torlai G, Wood C J, Acharya A, Carleo G, Carrasquilla J and Aolita L 2023 Quantum process tomography with unsupervised learning and tensor networks *Nat. Commun.* **14** 2858
- [11] Durrer R, Kratochwil B, Koski J V, Landig A J, Reichl C, Wegscheider W, Ihn T and Greplova E 2020 Automated tuning of double quantum dots into specific charge states using neural networks *Phys. Rev. Appl.* **13** 054019
- [12] Moon H et al 2020 Machine learning enables completely automatic tuning of a quantum device faster than human experts *Nat. Commun.* **11** 4161
- [13] Czischek S et al 2021 Miniaturizing neural networks for charge state autotuning in quantum dots *Mach. Learn.: Sci. Technol.* **3** 015001
- [14] Zwolak J P and Taylor J M 2023 Colloquium: advances in automation of quantum dot devices control *Rev. Mod. Phys.* **95** 011006
- [15] Li Y, Zou Y, Glorioso P, Altman E and Fisher M P A 2023 Cross entropy benchmark for measurement-induced phase transitions *Phys. Rev. Lett.* **130** 220404

¹⁷ We leave the consideration of these improvements to the future.

¹⁸ Note that one could reduce the job's wall time significantly by implementing a multi-threaded job with multiple CPUs on the same node.

- [16] Skinner B, Ruhman J and Nahum A 2019 Measurement-induced phase transitions in the dynamics of entanglement *Phys. Rev. X* **9** 031009
- [17] Li Y, Chen X and Fisher M P A 2018 Quantum zeno effect and the many-body entanglement transition *Phys. Rev. B* **98** 205136
- [18] Barratt F, Agrawal U, Gopalakrishnan S, Huse D A, Vasseur R and Potter A C 2022 Field theory of charge sharpening in symmetric monitored quantum circuits *Phys. Rev. Lett.* **129** 120604
- [19] Li Y, Vijay S and Fisher M P 2023 Entanglement domain walls in monitored quantum circuits and the directed polymer in a random environment *PRX Quantum* **4** 010331
- [20] Li Y, Chen X and Fisher M P A 2019 Measurement-driven entanglement transition in hybrid quantum circuits *Phys. Rev. B* **100** 134306
- [21] Zabalo A, Gullans M J, Wilson J H, Gopalakrishnan S, Huse D A and Pixley J H 2020 Critical properties of the measurement-induced transition in random quantum circuits *Phys. Rev. B* **101** 060301
- [22] Agrawal U, Zabalo A, Chen K, Wilson J H, Potter A C, Pixley J H, Gopalakrishnan S and Vasseur R 2022 Entanglement and charge-sharpening transitions in U(1) symmetric monitored quantum circuits *Phys. Rev. X* **12** 041002
- [23] Sang S, Li Y, Zhou T, Chen X, Hsieh T H and Fisher M P 2021 Entanglement negativity at measurement-induced criticality *PRX Quantum* **2** 030313
- [24] Majidy S, Agrawal U, Gopalakrishnan S, Potter A C, Vasseur R and Halpern N Y 2023 Critical phase and spin sharpening in SU(2)-symmetric monitored quantum circuits *Phys. Rev. B* **108** 054307
- [25] Avakian S J, Peregr-Barnea T and Witczak-Krempa W 2024 Long-range multipartite entanglement near measurement-induced transitions (arXiv:2404.16095) [quant-ph]
- [26] Hu Y and Zou Y 2024 Petz map recovery for long-range entangled quantum many-body states *Phys. Rev. B* **110** 195107
- [27] Choi S, Bao Y, Qi X-L and Altman E 2020 Quantum error correction in scrambling dynamics and measurement-induced phase transition *Phys. Rev. Lett.* **125** 030505
- [28] Gullans M J and Huse D A 2020 Dynamical purification phase transition induced by quantum measurements *Phys. Rev. X* **10** 041020
- [29] Weinstein Z, Bao Y and Altman E 2022 Measurement-induced power-law negativity in an open monitored quantum circuit *Phys. Rev. Lett.* **129** 080501
- [30] Skinner B 2023 Lecture Notes: introduction to random unitary circuits and the measurement-induced entanglement phase transition (arXiv:2307.02986) [cond-mat.stat-mech]
- [31] Chan A, Nandkishore R M, Pretko M and Smith G 2019 Unitary-projective entanglement dynamics *Phys. Rev. B* **99** 224307
- [32] Jian C-M, You Y-Z, Vasseur R and Ludwig A W W 2020 Measurement-induced criticality in random quantum circuits *Phys. Rev. B* **101** 104302
- [33] Bao Y, Choi S and Altman E 2020 Theory of the phase transition in random unitary circuits with measurements *Phys. Rev. B* **101** 104301
- [34] Li Y, Chen X, Ludwig A W W and Fisher M P A 2021 Conformal invariance and quantum nonlocality in critical hybrid circuits *Phys. Rev. B* **104** 104305
- [35] Fisher M P A, Khemani V, Nahum A and Vijay S 2023 Random quantum circuits *Annu. Rev. Condens. Matter Phys.* **14** 335
- [36] Agrawal U, Lopez-Piqueres J, Vasseur R, Gopalakrishnan S and Potter A C 2024 Observing quantum measurement collapse as a learnability phase transition *Phys. Rev. X* **14** 041012
- [37] Noel C et al 2022 Measurement-induced quantum phases realized in a trapped-ion quantum computer *Nat. Phys.* **18** 760
- [38] Myerson A H, Szwer D J, Webster S C, Allcock D T C, Curtis M J, Imreh G, Sherman J A, Stacey D N, Steane A M and Lucas D M 2008 High-fidelity readout of trapped-ion qubits *Phys. Rev. Lett.* **100** 200502
- [39] Christensen J E, Hucul D, Campbell W C and Hudson E R 2020 High-fidelity manipulation of a qubit enabled by a manufactured nucleus *npj Quantum Inf.* **6** 35
- [40] Taylor J M and Calarco T 2008 Wigner crystals of ions as quantum hard drives *Phys. Rev. A* **78** 062331
- [41] Blatt R and Roos C F 2012 Quantum simulations with trapped ions *Nat. Phys.* **8** 277
- [42] Bruzewicz C D, Chiaverini J, McConnell R and Sage J M 2019 Trapped-ion quantum computing: progress and challenges *Appl. Phys. Rev.* **6** 021314
- [43] Pino J M et al 2021 Demonstration of the trapped-ion quantum CCD computer architecture *Nature* **592** 209
- [44] Foss-Feig M, Pagano G, Potter A C and Yao N Y 2024 Progress in trapped-ion quantum simulation (arXiv:2409.02990) [quant-ph]
- [45] Debnath S, Linke N M, Figgatt C, Landsman K A, Wright K and Monroe C 2016 Demonstration of a small programmable quantum computer with atomic qubits *Nature* **536** 63–66
- [46] Monroe C et al 2021 Programmable quantum simulations of spin systems with trapped ions *Rev. Mod. Phys.* **93** 025001
- [47] Crain S, Mount E, Baek S and Kim J 2014 Individual addressing of trapped $^{171}\text{Yb}^{+}$ ion qubits using a microelectromechanical systems-based beam steering system *Appl. Phys. Lett.* **105** 181115
- [48] Shih C-Y, Motlakunta S, Kotibhaskar N, Sajjan M, Hablützel R and Islam R 2021 Reprogrammable and high-precision holographic optical addressing of trapped ions for scalable quantum control *npj Quantum Inf.* **7** 57
- [49] Koh J M, Sun S-N, Motta M and Minnich A J 2023 Measurement-induced entanglement phase transition on a superconducting quantum processor with mid-circuit readout *Nat. Phys.* **19** 1314
- [50] Ippoliti M and Khemani V 2021 Postselection-free entanglement dynamics via spacetime duality *Phys. Rev. Lett.* **126** 060501
- [51] Ippoliti M, Rakovszky T and Khemani V 2022 Fractal, logarithmic and volume-law entangled nonthermal steady states via spacetime duality *Phys. Rev. X* **12** 011045
- [52] Lu T-C and Grover T 2021 Spacetime duality between localization transitions and measurement-induced transitions *PRX Quantum* **2** 040319
- [53] Potter A C and Vasseur R 2022 Entanglement dynamics in hybrid quantum circuits *Entanglement in Spin Chains: From Theory to Quantum Technology Applications* (Springer) pp 211–49
- [54] Dehghani H, Lavasani A, Hafezi M and Gullans M J 2023 Neural-network decoders for measurement induced phase transitions *Nat. Commun.* **14** 2918
- [55] Iouchtchenko D, Gonthier J F, Perdomo-Ortiz A and Melko R G 2023 Neural network enhanced measurement efficiency for molecular groundstates *Mach. Learn. Sci. Technol.* **4** 015016
- [56] Czischek S, Torlai G, Ray S, Islam R and Melko R G 2021 Simulating a measurement-induced phase transition for trapped ion circuits *Phys. Rev. A* **104** 062405
- [57] Pogorelov I et al 2021 Compact ion-trap quantum computing demonstrator *PRX Quantum* **2** 020343
- [58] Gaebler J P et al 2016 High-fidelity universal gate set for $^{9}\text{Be}^{+}$ ion qubits *Phys. Rev. Lett.* **117** 060505

- [59] Kong L, Li Z and Liu Z-W 2024 Convergence efficiency of quantum gates and circuits (arXiv:2411.04898) [quant-ph]
- [60] Happer W 1972 Optical pumping *Rev. Mod. Phys.* **44** 169
- [61] Smith M C, Leu A D, Miyanishi K, Gely M F and Lucas D M 2024 Single-qubit gates with errors at the 10^{-7} level (arXiv:2412.04421) [quant-ph]
- [62] Hibat-Allah M, Ganahl M, Hayward L E, Melko R G and Carrasquilla J 2020 Recurrent neural network wave functions *Phys. Rev. Res.* **2** 023358
- [63] Hou W, Li M and You Y-Z 2023 Quantum generative modeling of sequential data with trainable token embedding (arXiv:2311.05050)
- [64] Fitzek D, Teoh Y H, Fung H P, Dagnew G A, Merali E, Moss M S, MacLellan B and Melko R G 2024 RydbergGPT (arXiv:2405.21052) [quant-ph]
- [65] Yang T-H, Soleimanifar M, Bergamaschi T and Preskill J 2024 When can classical neural networks represent quantum states? (arXiv:2410.23152) [quant-ph]
- [66] Cho K, van Merriënboer B, Bahdanau D and Bengio Y 2014 On the properties of neural machine translation: encoder–decoder approaches *Proc. of Sst-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* ed D Wu, M Carpuat, X Carreras and E M Vecchi (Association for Computational Linguistics) pp 103–11
- [67] Sehayek D, Golubeva A, Albergo M S, Kulchytskyy B, Torlai G and Melko R G 2019 Learnability scaling of quantum states: restricted Boltzmann machines *Phys. Rev. B* **100** 195125
- [68] Vaswani A 2017 Attention is all you need *Advances in Neural Information Processing Systems*
- [69] Zhang Y-H and Di Ventura M 2023 Transformer quantum state: a multipurpose model for quantum many-body problems *Phys. Rev. B* **107** 075147
- [70] Lange H, Bornet G, Emperauger G, Chen C, Lahaye T, Kienle S, Browaeys A and Bohrdt A 2024 Transformer neural networks and quantum simulators: a hybrid approach for simulating strongly correlated systems (arXiv:2406.00091)
- [71] Zhang Z and You Y-Z 2024 Observing schrödinger's cat with artificial intelligence: emergent classicality from information bottleneck *Mach. Learn.: Sci. Technol.* **5** 015051
- [72] Yao J and You Y-Z 2024 Shadowgpt: learning to solve quantum many-body problems from randomized measurements (arXiv:2411.03285)
- [73] Wang H, Li P, Chen M, Cheng J, Liu J and Chen T 2024 GroverGPT: a large language model with 8 billion parameters for quantum searching (arXiv:2501.00135) [quant-ph]
- [74] Yangrui H 2025 RNN-enhanced-XEB (available at: <https://github.com/yhu1996/RNN-enhanced-XEB>)
- [75] Paszke A et al 2019 Pytorch: an imperative style, high-performance deep learning library *Advances in Neural Information Processing Systems* vol 32