SAND2024-14301C

What is my quantum computer good for? Quantum capability learning with physics-aware neural networks

Daniel Hothem
Quantum Performance Laboratory
Sandia National Laboratories
Livermore, CA 94550
`dhothem@sandia.gov`

Ashe Miller
Quantum Performance Laboratory
Sandia National Laboratories
Albuquerque, NM 87185
`anmille@sandia.gov`

Timothy Proctor
Quantum Performance Laboratory
Sandia National Laboratories
Livermore, CA 94550
`tjproct@sandia.gov`

Abstract

Quantum computers have the potential to revolutionize diverse fields, including quantum chemistry, materials science, and machine learning, but quantum computing hardware experiences errors that can cause quantum programs run on them to fail. Until quantum computers can reliably execute large quantum programs, stakeholders will need fast and reliable methods for assessing a quantum computer's capability—i.e., the programs it can run and how well it can run them. Past works have used off-the-shelf neural network architectures to model quantum computers' capabilities, but with limited success, due to these networks' inability to learn the complex quantum physics that determines real hardware's failures. We address this shortcoming with a new physics-aware architecture for building neural network capability models. We combine graph neural networks, for learning the rates and locations of different kinds of errors in quantum programs, with a network architecture that combines these error rates into a final prediction using a physics-aware approximation for success probability. Our approach achieves up to a $\sim 50\%$ and $\sim 75\%$ reduction in mean absolute error on experimental and simulated data, respectively, over state-of-the-art models based on convolutional neural networks.

1 Introduction

Quantum computers have the potential to efficiently solve classically intractable problems in quantum chemistry [Cao et al., 2019], material science [Rubin et al., 2023], machine learning [Harrow et al., 2009], and cryptography [Shor, 1997]. While contemporary quantum processors (QPUs) are quickly approaching the sizes and noise levels needed to solve interesting, practical problems, they are still far from being capable of reliably running most useful quantum algorithms. Until we build QPUs capable of executing *any and all* useful and interesting quantum algorithms, stakeholders will require fast, reliable, and scalable methods for predicting the algorithms that given QPU can reliably execute.

The task of learning which quantum algorithms or quantum circuits a QPU can reliably execute is known as quantum capability learning [Proctor et al., 2021a]. In general, quantum capability learning is a difficult task. The number of possible errors plaguing a QPU grows exponentially in the size of the device (e.g., the number of qubits) [Blume-Kohout et al., 2022], and errors in a quantum circuit can combine in difficult-to-predict ways. Most traditional approaches to capability learning restrict themselves to learning how well a QPU executes a limited set of circuits by running those circuits on the QPU, and using the data to estimate a success metric for each circuit [Hothem et al., 2023b]. While these methods are valuable for understanding a device, they provide no principled way to predict how well circuits outside of the chosen circuit set will run on the QPU. In other words, they offer little predictive power about a QPU's capability.

Recently, several groups have proposed building predictive models of a QPU's capability using neural networks [Elsayed Amer et al., 2022, Hothem et al., 2023c, Vadali et al., 2024, Wang et al., 2022]. In general, these neural network-based capability models achieve only a modest prediction accuracy on real hardware, in part due to their inability to learn the complex physics that determines real hardware's failures. We overcome this limitation by making two changes.

First, we restrict our focus to learning a QPU's capability on high-fidelity circuits. High-fidelity circuits are those circuits that a QPU correctly executes with a high success rate. High-fidelity circuits are arguably the most interesting circuits to study as we care far more about whether a QPU successfully executes a circuit 90% or 95% of the time rather than 10% or 40% of the time. Moreover, by restricting our focus to high-fidelity circuits, we are able to leverage known closed-form approximations to the success rate of high-fidelity circuits that rely on understanding the error rates of individual errors in the circuit. Therefore, instead of training a neural network to map a circuit direct to its success rate, we instead train neural networks to map a circuit to the error rates of physically relevant errors, using the signal provided by the success rate.

Second, we introduce a new physics-aware, graph neural network-based approach that is optimized for learning a QPU's capability on high-fidelity circuits. In our approach, we use a graph neural network to predict the error rates of physically relevant errors in each layer of a quantum circuit. These error rates are then efficiently combined using the internal logic of the circuit into an error vector for the entire circuit, and the resulting error vector is used as the input to a first-order approximation to the circuit's success rate. By leveraging the graph structure of each quantum circuit, restricting the neural network to focus exclusively on learning the error rates of physically relevant errors, and offloading the difficult-to-learn, yet classically tractable task of combining the error rates, our physics-aware neural networks vastly outperform the state-of-the-art convolutional neural network approach in [Hothem et al., 2023c] on both experimental and simulated data.

In a direct, head-to-head comparison study, our physics-aware neural networks achieve, on average a $\sim 50\%$, reduction in mean absolute error over the CNNs used in Hothem et al. [2023c] when predicting the probability of successful trials (PST) of the circuits in the experimental datasets used in Hothem et al. [2023c]. Our physics-aware neural networks achieve an average $\sim 32\%$ improvement over those CNNs even after fine-tuning the CNNs on a subset of the high-PST circuits in each data set.

We hypothesize that our physics-aware neural networks' improved performance is, in part, due to their improved ability to model the impact of coherent errors on a circuit's success rate.

Our physics-aware neural networks' improved performance is, in part, due to their improved ability to correctly model the impact of coherent errors on a circuit's success rate. Off-the shelf networks struggle with coherent errors Hothem et al. [2023c]. However, because we explicitly add in how coherent error rates combine into our new approach, our physics-aware networks are much better at predicting PST in the presence of coherent errors. To verify this claim, we demonstrate our networks' performance at predicting the entanglement fidelity of random circuits run on a simulated 4-qubit QPU experiencing only coherent gate errors. In these simulations, a physics-aware neural network obtain a $\sim 76\%$ reduction in MAE when compared to a CNN, and exhibit moderate performance when predicting the fidelity of out-of-distribution random mirror circuits [Proctor et al., 2021a] simulated on the same 4-qubit QPU.

2 Background

In this section, we review the necessary background in quantum computing to understand this paper. See Nielsen and Chuang [2010] for a more in-depth introduction to quantum computing and Blume-Kohout et al. [2022] for a more thorough description of the errors that occur in quantum processors.

2.1 Quantum computing

A quantum computer is a physical system that performs computations by exploiting the laws of quantum mechanics. The fundamental computational unit of a quantum computer is the qubit. A qubit is a two-level physical system whose pure states are normalized vectors in a complex two-dimensional Hilbert space, $\mathcal{H}$. The pure states of an n-qubit processor are normalized vectors in $\mathcal{H}^{\otimes n}$. The two orthonormal vectors $|0\rangle$ and $|1\rangle$ that are eigenvectors of the Z Pauli operator are identified as the "computational basis states" of $\mathcal{H}$. The computational basis states of $\mathcal{H}^{\otimes n}$ are all the tensor-product combinations of $|0\rangle$ and $|1\rangle$. Noise processes in real quantum computers mean that they are typically in mixed states ρ that are a probabilistic mixture of pure states.

A computation is performed by running a quantum circuit. Typically, an n-qubit, depth-d quantum circuit consists of the following instructions: (i) an instruction to prepare the all-zero state, $|0\rangle^{\otimes n}$; (ii) a sequence of d layers of logical instructions $\{L_i\}$; and (iii) an end-of-circuit measurement. The end-of-circuit measurement outputs a bitstring according to a probability distribution defined by the end-of-circuit state of the n-qubits. The initial state preparation and final measurement layers are fixed, so we define a quantum circuit by its set of logical instructions.

The "meat" of an n-qubit, depth-d circuit c is the sequence of d logical circuit layers, $\{L_i\}$. Together, these circuit layers specify the application of an n-qubit unitary $\mathcal{U}(c)$. Individually, a circuit layer L specifies the application of an n-qubit unitary map $\mathcal{U}(L)$ to the qubits. Usually, the n-qubit unitary is given as the parallel application of single-qubit and two-qubit gates.

2.2 Noise and errors in quantum computers

Unlike their idealized or classical counterparts, real quantum processors are highly error-prone. While the errors come in many forms, they all have the same effect. The errors accumulate over the execution of a circuit and lead to the measurement sampling from the incorrect distribution over bit strings.

This process can be modelled as follows. Model the noisy implementation $\tilde{\mathcal{U}}(L_i)$ of each circuit layer L_i as the ideal implementation $\mathcal{U}(L_i)$ proceeded by an error channel $\mathcal{E}_i$:

$$\tilde{\mathcal{U}}(L_i) = \mathcal{E}_i \circ \mathcal{U}(L_i). \tag{1}$$

Then the noisy implementation $\tilde{\mathcal{U}}(c)$ of a circuit c is modelled as

$$\tilde{\mathcal{U}}(c) = \prod_{i=1}^{d} \mathcal{E}_i \circ \mathcal{U}(L_i). \tag{2}$$

We may compute an end-of-circuit error channel by "propagating" each layer's error channel to the end of the circuit by commuting the error channels past each $\mathcal{U}(G_i)$,

$$\tilde{\mathcal{U}}(c) = \mathcal{E}_c \circ \mathcal{U}(c). \tag{3}$$

A convenient way to parameterize an error channel $\mathcal{E}$ is as $\mathcal{E} = \exp(\sum_j \epsilon_j G_j)$ where G_j are the set of $16^n - 1$ different *elementary error generators* introduced by Blume-Kohout et al. [2022]. The most common kinds of errors are the subset of $\{G_j\}$ known as Hamiltonian (H) and Pauli stochastic (S) errors, of which is indexed by an element of the n-qubit Pauli group, $\mathbb{P}_n$. There are $4^n - 1$ of H and $4^n - 1$ S errors and for the rest of this work we consider only the set of all such errors $\{G_j\}_{j=1}^{2(4^n-1)}$ and denote their rates by $\{\epsilon_P\}$ and $\{\theta_P\}$ respectively. The specific action of each error is unimportant Instead, what is important is that a Pauli stochastic error P occurs randomly at some rate s_P, while coherent errors occur deterministically with some strength θ_P. As a result, stochastic errors, to first order, accumulate linearly, while coherent errors, due to error cancellation, accumulate quadratically [Mądzik et al., 2022].

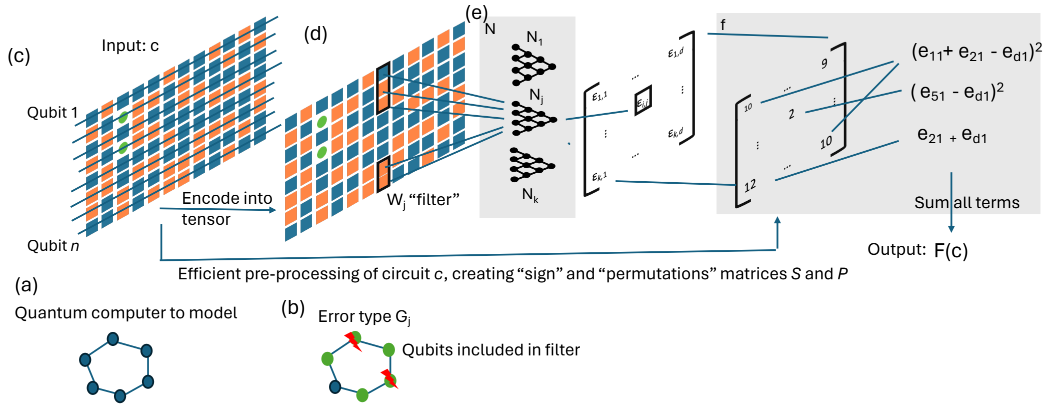

Figure 1: Our physics-aware neural network architecture for modelling the capabilities of real-world imperfect quantum computers.

2.3 Quantum capability learning

Because quantum processors are error-prone, knowing which quantum circuits a processor can successfully execute is important. Known as *quantum capability learning*, this task formally involves learning the mapping between a set of quantum circuits $c \in \mathcal{C}$ and some success metric $s(c) \in \mathbb{R}$ quantifying how well c runs on a processor $\mathcal{Q}$. While there are many interesting circuit sets and useful success metrics, we chose to focus on the probability of successful trial (PST) of definite-outcome Clifford circuits, and the *entanglement fidelity* of generic quantum circuits.

A circuit c is a definite-outcome circuit if its end-of-circuit measurement distribution, ideally, has support on a single bit string, $b(c)$. The probability of successful trial is defined as

$$\mathrm{PST}(c) = \Pr(\text{measuring } b(c) \text{ when executing } c \text{ on } \mathcal{Q}). \tag{4}$$

In practice, $\mathrm{PST}(c)$ is estimated by running c many times on $\mathcal{Q}$ and calculating

$$\widehat{\mathrm{PST}(c)} = \frac{\#\text{ observations of } b(c)}{N_{\text{shots}}}, \tag{5}$$

where N_{shots} is the number of shots or times c was run for.

In addition to being easily interpretable, $\mathrm{PST}(c)$ is a nice metric because it has a straightforward first-order approximation in terms of the Pauli stochastic $\{s_P\}$ and coherent error rates $\{\theta_p\}$ of the end-of-circuit error channel. If $\mathbb{P}_n^{X,Y}$ is the set of n-qubit Paulis containing at least one Pauli-X or Pauli-Y entry, then

$$\mathrm{PST}(c) \approx 1 - \sum_{P \in \mathbb{P}_n^{X,Y}} \left(s_P + \theta_p^2\right). \tag{6}$$

Equation 6 is a good approximation for high-PST circuits.

While $\mathrm{PST}(c)$ is a nice metric, it is not defined for non-definite-outcome circuits. Instead, we use entanglement fidelity, $F(c)$. Protocols exist for efficiently estimating fidelity on real hardware [Proctor et al., 2022], and, in simulations of small systems ($n <\sim 10$) it can be computed directly. Moreover, like $\mathrm{PST}(c)$, entanglement fidelity has a simple first-order approximation [Mądzik et al., 2022],

$$F(c) \approx 1 - \sum_{P \in \mathbb{P}_n} \left(s_P + \theta_p^2\right). \tag{7}$$

3 Neural network architecture

Our neural network architecture (see Fig. 1) for quantum capability learning combines neural network layers that have GNN-like structures with efficient approximations to the physics of errors in quantum computers. The overall action of our neural networks is to map an encoding of a circuit c to a prediction for $\mathrm{PST}(c)$ or $F(c)$. The same network can predict either $\mathrm{PST}(c)$ or $F(c)$ by simply toggling between two different output layers, that have no trainable parameters. Our architecture is divided into two sequential parts. The first part of our architecture is a neural network $\mathcal{N}$ that has the task of learning about the kinds and rates of errors that occur in quantum circuits. We use GNN-like structures within $\mathcal{N}$ to embed physics knowledge for how those errors depend on the quantum circuit being run. The second part of our architecture is a function f with no learnable parameters, that turns $\mathcal{N}$'s output into a prediction for $\mathrm{PST}(c)$ or $F(c)$.

3.1 Physics-aware neural networks for predicting errors in quantum circuits

The neural network $\mathcal{N}$'s input is a circuit c represented by (i) a tensor $I(c) \in \{0,1\}^{n \times d(c) \times n_{ch}}$ describing the gates in c, and (ii) a matrix $M(c) \in \{0,1\}^{2 \times n}$ describing the measurement of the qubits at the end of c. $\mathcal{N}$ maps $I(c)$ to a matrix $\mathcal{E} \in \mathbb{R}^{k \times d(c)}$ and $M(c)$ to a vector $\vec{m} \in \mathbb{R}^{k}$. $\mathcal{E}_{ij}$ is a prediction for the rate with which error type j occurs during circuit layer i, and m_j is a prediction for the rate with which error type j occurs when measuring the qubits at the end of a circuit. There are $2(4^n - 1)$ different possible error types that can occur in principle (see Section 2) so it is infeasible to predict all their rates beyond very small n. However, the overwhelming majority of these errors are implausible, i.e., they are not expected to occur in real quantum computers [Blume-Kohout et al., 2022]. Our networks therefore predict the rates of every error from a relatively small set of error types $\mathbb{G} = \{G_1, \ldots, G_k\}$ containing the k most plausible kinds of error. $\mathbb{G}$ is a hyperparameter of our networks. It can be chosen to reflect the known physics of a particular quantum computer and/or optimized using hyperparameter tuning. In our demonstrations, we choose $\mathbb{G}$ to contain all one-body H and S errors as well as all two-body H and S errors that interact pairs of qubits within h steps on the modelled quantum computer's connectivity graph for some constant h (see Fig. 1b). This choice for $\mathbb{G}$ encodes the physical principle that unwanted interactions between qubits are primarily two-body and local [Blume-Kohout et al., 2022]. The size of $\mathbb{G}$ grows with n, and for planar connectivity graphs (as in, e.g., contemporary superconducting qubit systems) it grows linearly in n. This results in $k = \mathcal{O}(n)$ errors whose rates $\mathcal{N}$ must learn to predict.

The internal structures of $\mathcal{N}$ are chosen to reflect general physical principles for how $\mathcal{E}$ and $\vec{m}$ depend on c. $\mathcal{E}_{ij}$ is a prediction for the rate that G_j occurs in circuit layer i, and this error corresponds to a space/time location within c—because it occurs at layer index or time i and G_j acts on a subset of the qubits $Q(G_j)$. This error's rate will therefore primarily depend only on the gates in a time- and space-local region around its location in c. Furthermore this dependence will be invariant under time translations (except for some exotic kinds of errors, that we discuss in Section 7.1). We can encode these structures into $\mathcal{N}$ by predicting $\mathcal{E}_{ij}$ from a space-time "window" of c around the associated error's location using a filter W_j that "slides" across the circuit to predict the rate of G_j versus time i. Stated more formally, we predict $\mathcal{E}_{ij}$ using a multilayer perceptron $\mathcal{N}_j$ whereby $\mathcal{N}_j(W_j[I(c), i]) = \mathcal{E}_{ij}$ and $W_j(I(c), i)$ is a snippet of $I(c)$ whose temporal origin is i (see Fig. 1e). The shape of each filter W_j is a hyperparameter of our networks and it can be designed to reflect general physical principles, the known physics of a particular quantum computing system, and/or optimized with hyperparameter tuning. The particular neural networks we present later herein use filters $W_j(I(c), i)$ that snip out only layer i and discard the parts of the layer that act on qubits more than l steps away from $Q(G_j)$ in the quantum computer's connectivity graph (see Fig. **??**). This structure has close connections to graph convolution layers, as well as CNNs. We choose this particular structure as it enables modelling the effects of spatially localized crosstalk errors, which are a ubiquitous but hard-to-model class of errors in quantum computers.

The network $\mathcal{N}$ must also predict the rates of errors that occur during measurements, but these are not typically closely related to the rates of gate errors (which are predicted by the $\mathcal{N}_j$). So we do not use the $\mathcal{N}_j$ and their convolutional filters W_j to make predictions for $\vec{m}$. Instead we use separate but structurally equivalent networks $\mathcal{N}_j'$ with corresponding filters W_j' that take $M(c)$ as input and implement only spatial filtering (i.e., W_j' simply discard rows from $M(c)$, as, unlike $I(c)$, $M(c)$ has no temporal dimension). The W_j' are hyperparameters of our networks and we can separately adjust the shape of each W_j' to reflect the known physics of errors induced by measuring qubits. We use the same kind of filters as the W_j, with l' steps on the connectivity graph.

3.2 Processing predicting error rates to predict capabilities

We process $\mathcal{N}$'s output to predict $\mathrm{PST}(c)$ or $F(c)$, using a function f with no learnable parameters. We do so for two reasons. First, the error matrix $\mathcal{E}$ predicted by $\mathcal{N}$ is not a directly observable quantity, and so we cannot easily train these networks in isolation. Generating the data needed to train $\mathcal{N}$ directly would require extraordinarily expensive quantum process tomography **?** and it is utterly infeasible except for very small n. In constract, both $\mathrm{PST}(c)$ and $F(c)$ can be efficiently estimated (see Section 2) for a given circuit c. So concatenating $\mathcal{N}$ with f makes training feasible. Secondly, this means that the network is being directly trained to predict the quantities of interest.

The function f computes an approximation to the $\mathrm{PST}(c)$ or $F(c)$ predicted by $\mathcal{E}$ and $\vec{m}$. The matrix $\mathcal{E}$ encodes the prediction that c's imperfect action is

$$\tilde{\mathcal{U}}(c) = \Lambda_d(\mathcal{E})\mathcal{U}(L_d) \cdots \Lambda_1(\mathcal{E})\mathcal{U}(L_1), \tag{8}$$

where the L_i are the d layers of c (see Section 2) and $\Lambda_i(\mathcal{E}) = \exp(\sum_{j=1}^{k} \mathcal{E}_{ij} G_j)$, i.e., $\Lambda_i(\mathcal{E})$ is an error channel parameterized by the ith column of $\mathcal{E}$. Equation (8) implies an exact prediction for $\mathrm{PST}(c)$ or $F(c)$ [e.g., $F(c) = \mathrm{Tr}(\tilde{U}(c)U^{-1}(c))/(4^n - 1)$], but exactly computing that prediction involves explicitly creating and multiplying together each of the $4^n \times 4^n$ matrices in Eq. (8). This is infeasible, except for very small n. Instead our f computes an efficient approximation to this prediction. f's action is most easily described by embedding $\mathcal{E}$ into the space of all possible errors $\{G_j\}_{j=1}^{2(4^n-1)}$, resulting in a $d \times 2(4^n - 1)$ matrix $\mathcal{E}_e$ whose columns are k-sparse, although note that we never construct these exponentially large matrices. Then we pull each error channel to the end of the circuit $\tilde{\mathcal{U}}(c) = \Lambda'_d(\mathcal{E}'_e) \cdots \Lambda'_1(\mathcal{E}'_e)\mathcal{U}(c)$ where $\Lambda'_d(\mathcal{E}'_e) = \exp(\sum_{j=1}^{2(4^n-1)} [\mathcal{E}'_e]_{ij} G_j)$ where $\mathcal{E}'_e$ has columns that are just c-dependent signed permutations of $\mathcal{E}_e$'s columns. The signed permutations required can be efficiently computed in advance (i.e., as an input encoding step) using an efficient representation of Clifford unitaries, utilizing a high-performance implementation of those methods in Stim, and can be efficiently represented in two $d \times k$ matrices: a sign matrix S containing ± 1 signs to be element-wise multiplied with $\mathcal{E}$ and a permutation indices matrix P containing values in $[1, 2(4^n - 1)]$ where P_{ij} specifies what error G_j because when pulled through the remaining circuit layers. Next, we combine the $\Lambda'_i(\mathcal{E})$ into a single error map $\Lambda(c)$ using a first-order Baker-Campbell-Hausdorff (BCH) expansion. Using our embedded representation, this means simply approximating $\Lambda(c)$ as $\Lambda(c) \approx \exp(\sum_j v_j G'_j)$ where $v_j = \sum_{i=1}^{d} [\mathcal{E}'_e]_{ij}$, i.e., we sum over the rows of $\mathcal{E}'_e$. To predict $F(c)$ we then simply apply Eq. (6) (meaning summing up v_j with those elements that correspond to Hamiltonian errors squared), and to predict $\mathrm{PST}(c)$ wealso combined in the measurement error map $\exp(\sum_{j=1}^{l} m_j G_j)$ and apply Eq. (7). The efficient representation of the overall action of f is illustrated in Fig. 1.

4 Datasets

4.1 Experimental data

We used the 5-qubit data sets in Hothem et al. [2023c] for our experimental demonstrations. Each of these data sets $D = \{(c, \widehat{\mathrm{PST}(c)})\}$ was gathered by running random and periodic mirror circuits (two types of definite-outcome circuits) on the first five qubits of an IBMQ processor, and using the results to estimate the PST of each circuit. Each circuit ran between 1024 and 4096 times on the processor, with the exact shot count depending upon how often the circuit-generating process generated the circuit (some short, 1-qubit circuits were generated multiple times). The random and periodic mirror circuits ranged in width from 1 to 5 qubits, and ranged in depth from 3 to 515 layers (alt. 259 for the `ibmq_yorktown` dataset).

Because we are focused on high-PST circuits, we removed all circuits with a PST less than .85 from each data set, leaving between 864 (`ibmq_burlington`) to 1369 (`ibmq_yorktown` circuits in each data set. The remaining circuits were partitioned into training, validation, and test sets based upon their original assignment in Hothem et al. [2023c]. This choice allows us to directly compare our newly trained physics-aware neural networks to those used in the original paper. Training set sizes ranged from 682 circuits on `ibmq_burlington` to 1097 circuits on `ibmq_yorktown`, with an approximate training, validation, testing split of 80%, 10%, and 10%, respectively. See Figure 2(a) for a representative histogram of the PSTs from the `ibmq_vigo` processor.

4.2 Simulated data

For our simulations, we generated DGH: ONE datasets of 5000 high-fidelity ($> 90\%$) random circuits, for a 4-qubit processor with a "ring" geometry (see Figure **??**). The circuits ranged in width w from 1 to 4 qubits, and in depth from 1 to 180 circuit layers. Each circuit was designed to be applied to a randomly chosen (possibly disconnected) subset w of qubits. Each circuit layer was created by *i.i.d.* sampling from all possible circuit layers on the w active qubits. We used a gate set of $\{X(\pi/2), Y(\pi/2), X(3\pi/2), Y(3\pi/2), X(\pi), Y(\pi), Z(\pi)\}$ single-qubit rotation gates and two-qubit CNOT gates. See Appendix **??** for additional details.

All circuits were simulated under the same error model, consisting of unbiased local coherent errors. The exact error model was randomly selected (see Appendix **??** for the process). At a high-level, each gate was assigned a small overall error strength, which was distributed randomly across all possible single-qubit or two-qubit coherent errors. We chose a coherent-only error model as coherent errors are prevalent errors that CNNs are struggle to accurately model. See Figure 3 for a histogram of fidelities.

Each sampled circuit c was simulated under the chosen error model to compute its entanglement fidelity $s_F(c)$ exactly. After removing duplicate circuits, the resulting dataset(s) $D = \{(c, s_F(c))\}$ were partitioned into training, validation, and testing subsets, with a partition of 56.25%, 18.75%, and 25%, respectively.

We also generated DGH: ONE datasets of 750 random mirror circuits on the same processor. Again, random mirror circuits varied in width from 1 to 4 qubits, and were designed to be run on a randomly selected subset w of qubits. However, instead of *i.i.d.* sampling each circuit layer, each circuit was randomly sampled from the class of random mirror circuits on the w qubits. The depth of the mirror circuits ranged from 8 to 174 layers. We generated the mirror circuit datasets to evaluate how well the physics-aware and convolutional networks generalize to out-of-distribution circuits, as such they were used exclusively as testing sets. In order to ensure that no training was performed on the mirror circuits, we removed any mirror circuits that appeared in the random circuit sets (in actuality, there were no duplicates).

4.3 Encoding schemes

We used two different encoding schemes for converting each circuit c into a tensor: a shared scheme for the physics-aware networks and the CNNs on simulated data and a separate scheme for the CNNs on experimental data For the CNNs on experimental data, we used the same encoding scheme as Hothem et al. [2023c], as we used their data and networks. We now describe the encoding scheme used with the physics-aware networks and the CNNs on simulated data.

As outlined in Section 3, each width-w circuit c is represented by a three-dimensional tensor $I(c) \in \{0,1\}^{w \times d(c) \times n_{ch}}$ describing the gates in c and a matrix $M(C) \in \{0,1\}^{2 \times w}$ describing the measurement of the qubits. The ij-th entry of $I(c)$,

$$I_{ij}(c) = (I_{ij1}(c), \ldots, I_{ijn_{ch}}(c)), \tag{9}$$

is a one-hot encoded vector of what happens to qubit i in layer j. For the ring processor, $n_{ch} = 11$: one channel for each single-qubit gate and four channels for the CNOT gates. There are four CNOT channels to specify if the qubit i is the target or control qubit and if the interacting qubit is to the left or right of qubit i. We used an additional 4 or 8 CNOT channels for the experimental data. The first row in $M(c)$ is the bitstring specifying which qubits are measured at the end of c. When c is a definite-outcome circuit, the second row is its target bitstring, i.e., the sole bitstring supported by the end-of-circuit measurement distribution when c is executed without error. Both $I(c)$ and $M(c)$ are zero-padded to ensure a consistent tensor shape across a dataset.

In practice, we make a few modifications to the encoding scheme described above. First, we reshape $I(c)$ to have shape $nd(c) \times n_{ch}$ when working with the physics-aware network. This allows each row to store all the gate information of a circuit layer. Second, we discard $M(c)$ when predicting the entanglement fidelity, as entanglement fidelity is independent of measurement error.

Additionally, each circuit c is accompanied by a permutation matrix $P(c) \in \mathbb{N}^{w \times n_{\text{errors}}}$ and sign matrix $S(c) \in \{\pm 1\}^{w \times n_{\text{errors}}}$. The ij-entry of $P(c)$ specifies which error the j-th tracked error occurring after the i-th layer is transformed into at the end of the circuit. The ij-th entry of $S(c)$ specifies the sign of that error.

5 Experiments

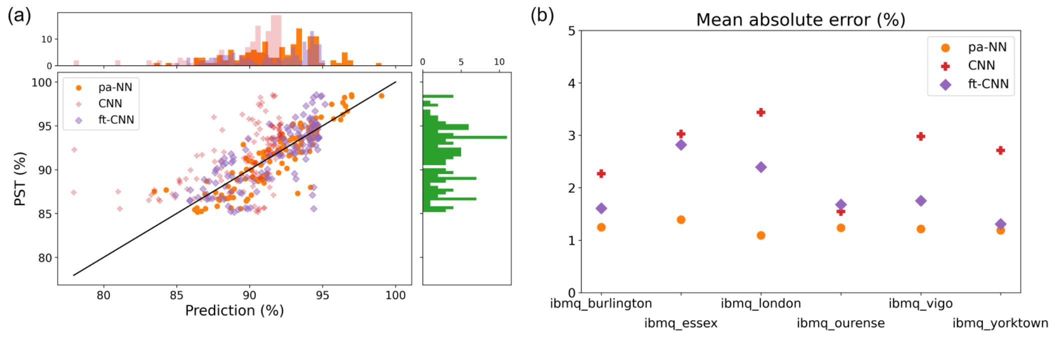

Figure 2: **Predictions and prediction accuracy on real hardware (a)** A scatter plot of the PST predictions on the `ibmq_vigo` test data by the physics-aware NN (pa-NN, ●), CNNs (o-CNN, +), and fine-tuned CNNs (ft-CNN, ♦). The top subplot depicts histograms of each model's predictions, while the right subplot is a histogram of the PST. **(b)** The mean absolute error on test data of the physics-aware NNs (pa-NN), CNNs (o-CNN), and fine-tuned CNNs (ft-CNN) trained on six different IBMQ processors.

We now present the results from our head-to-head comparison between the physics-aware neural networks and the CNNs on the 5-qubit datasets used in Hothem et al. [2023c]. Figure 2 summarizes the mean absolute error (MAE) achieved by the CNNs (red +) and the physics-aware neural networks (orange ● on each of the 5-qubit data sets. There is an across-the-board reduction in the MAE between the CNNs and the physics-aware networks, with an average 50.4% reduction in MAE ($\sigma = 15.25\%$). The ratio of the likehood of the physics-aware network to the likehood of the CNN (K, the Bayes factor) given the test data is between 10^{32} and 10^{380}, which is overwhelming evidence that the physics-aware network is a better model ($K \geq 10^2$ is typically considered decisive). These results strongly suggest that the extra infrastructure in the physics-aware models is making a difference.

Of course, it is possible that the improved performance is driven by a difference in model size; however, this is unlikely. For context, the `ibmq_london` CNN contains $6,649,531$ trainable parameters compared to the $1,218,348$ trainable parameters in the physics-aware network. Moreover, models of equivalent sizes to the physics-aware networks were included in the hyperparameter optimization space of the CNNs [Hothem et al., 2023c]. Therefore, we can conclude that simply changing the CNN's parameter count will not lead to a noticeable improvement in performance.

Nonetheless, comparing the physics-aware networks to the CNNs is still somewhat unfair as the CNNs were trained on out-of-distribution circuits (i.e., low-PST circuits). For a fairer comparison, we fine-tuned each CNN (Figure 2, purple ♦) on the high-PST circuits in each training set. Fine-tuning generally increased the CNNs' performances (mean 25% improvement, $\sigma = 20\%$); however, the physics-aware networks still achieved an average reduction of MAE of 32.2% ($\sigma = 15.8\%$) and outperformed the fine-tuned CNNs on all six datasets. K is between 10^{26} and 10^{237}, which is overwhelming evidence that the physics-aware network is a better model than the fine-tuned CNN.

6 Simulations

One reason why the extra infrastructure of our physics-aware networks may be necessary is that off-the-shelf networks struggle with modeling coherent errors [Hothem et al., 2023c]. To test our hypothesis, we compared the performance of a physics-aware neural network trained to predict the entanglement infidelity of random circuits executed on a simulated 4-qubit QPU experiencing solely coherent errors to a hyperparameter-tuned CNN trained on the same task. Figure 3 summarizes the results.

As expected, the physics-aware networks quantitatively and qualitatively outperform the CNNs. The CNNs struggle to extract any meaningful features from the training set, and learn to predict $\sim 100\%$ for almost all circuits. Whereas, the physics-aware networks learn to disambiguate between circuits.

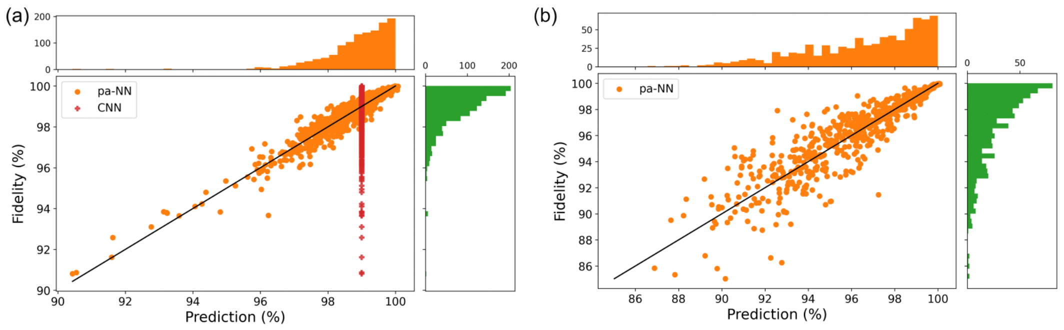

Figure 3: **CNN and physics-aware network predictions on simulated data.** **(a)** A scatter plot of the fidelity predictions on the simulated 4-qubit "ring" processor test data by the physics-aware NN (pa-NN, ●) and CNNs (o-CNN, +). The top subplot depicts histogram of the physics-aware network's predictions, while the right subplot is a histogram of the true fidelities. **(b)** A scatter plot of the fidelity predictions on the simulated 4-qubit "ring" processor mirror circuit data by the physics-aware NN (pa-NN, ●). Interestingly, the physics-aware network achieves modest prediction accuracy on this out-of-distribution task, suggesting that the physics-aware network is accurately learning error rates.

Overall, the physics-aware networks achieves an average 76.6% reduction in MAE, with a standard deviation of [FILL IN].

We also found that the physics-aware networks trained on random circuits are modest predictors of the infidelity of random mirror circuits. This phenomenon is an example of out-of-distribution generalization, as random mirror circuits contain (noiseless) idling gates on qubits, in addition to structure not present in random circuits that interacts more severely with coherent errors. On average, the physics-aware networks achieve an average MAE of $.72\%$ on the random mirror circuits. Although this result does represent a 3.2x increase in MAE, the strong linear relation between the network's predictions and the ground truth ($r = .91$) strongly suggests that the physics-aware network is learning information relevant to random mirror circuits.

7 Discussion

7.1 Limitations

While our results are a significant improvement over the state of the art, our approach does have several limitations:

1. As presently conceived, our approach assumes that a processor's physics are Markovian (i.e., they do not change over time). However, non-Markovian noise exists in QPUs [**?**]. In the future, we plan to address this issue by adding temporal information into our approach, perhaps with a temporal encoding.
2. Our approach only considers two error classes. Other Markovian error classes, like amplitude damping, exist, but their error rates contribute to PST and fidelity at order $\mathcal{O}(\varepsilon^3)$ [**?**]. If necessary, we can easily track these additional errors, and update f to account for their presence.
3. Propagating errors through a generic circuit, at scale, is computationally challenging, and so, for many-qubit QPUs, our current approach only works with Clifford circuits. While our approach works for generic, few-qubit ($\lesssim 10$ qubits) quantum circuits, we will need to develop approximate methods for propagating errors through generic circuits if we want our approach to scale for generic circuits.

7.2 Conclusion

In this paper, we presented a new physics-aware neural network architecture for modelling a quantum processor's capability that significantly improves upon the state of the art. The new architecture has two parts: a graph neural network that uses gate information and a QPU's connectivity graph to predicting the rates of errors in a circuit layer, and a non-trainable component that turns the per-layer error rates into a capability prediction. By imbuing the network with knowledge about how errors propagate through a circuit and restricting the graph neural network to predict error rates, our new networks outperform state-of-the-art convolutional neural network-based capability models by $\sim 50\%$ on experimental data and $\sim 75\%$ in simulated data. We also provided evidence that these physics-aware networks are learning the physical error rates, as they exhibit modest prediction accuracy when predicting the fidelity of out-of-distribution quantum circuits.

Understanding which quantum circuits a quantum processor can run, and how well it can run them, is an important, yet challenging component of understanding a quantum processor's power. Given the complexity of the problem, neural networks are likely to play a large role in its solution. As our results demonstrate, our new physics-aware network architecture should play a critical role in building fast and reliable neural network-based capability models.

Acknowledgments and Disclosure of Funding

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

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

A. Anonymous. Supplemental code and data. . To be released. See attached anonymized zip file.

R. Blume-Kohout, M. P. da Silva, E. Nielsen, T. Proctor, K. Rudinger, M. Sarovar, and K. Young. A taxonomy of small markovian errors. *PRX Quantum*, 3:020335, May 2022. doi: 10.1103/PRXQuantum.3.020335. URL `https://link.aps.org/doi/10.1103/PRXQuantum.3.020335`.

Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis, and A. Aspuru-Guzik. Quantum chemistry in the age of quantum computing. *Chemical Reviews*, 119(19):10856–10915, 2019. doi: 10.1021/acs.chemrev.8b00803. URL `https://doi.org/10.1021/acs.chemrev.8b00803`. PMID: 31469277.

F. Chollet et al. Keras. `https://keras.io`, 2015.

N. Elsayed Amer, W. Gomaa, K. Kimura, K. Ueda, and A. El-Mahdy. On the learnability of quantum state fidelity. *EPJ Quantum Technology*, 9:31, 2022. URL `https://epjquantumtechnology.springeropen.com/articles/10.1140/epjqt/s40507-022-00149-8#citeas`.

C. Gidney. Stim: a fast stabilizer circuit simulator. *Quantum*, 5:497, July 2021. ISSN 2521-327X. doi: 10.22331/q-2021-07-06-497. URL `https://doi.org/10.22331/q-2021-07-06-497`.

A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009. doi: 10.1103/PhysRevLett.103.150502. URL `https://link.aps.org/doi/10.1103/PhysRevLett.103.150502`.

D. Hothem, T. Catanach, K. Young, and T. Proctor. Learning a quantum computer's capability using convolutional neural networks [Data set]. `https://doi.org/10.5281/zenodo.7829489`, 2023a. Published: 2023-04-12.

D. Hothem, J. Hines, K. Nataraj, R. Blume-Kohout, and T. Proctor. Predictive models from quantum computer benchmarks. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 709–714, Los Alamitos, CA, USA, sep 2023b. IEEE Computer Society. doi: 10.1109/QCE57702.2023.00086. URL `https://doi.ieeecomputersociety.org/10.1109/QCE57702.2023.00086`.

D. Hothem, K. Young, T. Catanach, and T. Proctor. Learning a quantum computer's capability using convolutional neural networks, 2023c.

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *Proceedings of the 3rd International Conference for Learning Representations*. Microtome, 2015. URL `https://arxiv.org/abs/1412.6980`.

M. T. Mądzik, S. Asaad, A. Youssry, B. Joecker, K. M. Rudinger, E. Nielsen, K. C. Young, T. J. Proctor, A. D. Baczewski, A. Laucht, V. Schmitt, F. E. Hudson, K. M. Itoh, A. M. Jakob, B. C. Johnson, D. N. Jamieson, A. S. Dzurak, C. Ferrie, R. Blume-Kohout, and A. Morello. Precision tomography of a three-qubit donor quantum processor in silicon. *Nature*, 601(7893):348–353, Jan. 2022.

E. Nielsen, K. Rudinger, T. Proctor, A. Russo, K. Young, and R. Blume-Kohout. Probing quantum processor performance with pyGSTi. *Quantum Sci. Technol.*, 5(4):044002, July 2020. ISSN 2058-9565. doi: 10.1088/2058-9565/ab8aa4. URL `https://iopscience.iop.org/article/10.1088/2058-9565/ab8aa4`.

M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout. Measuring the capabilities of quantum computers. *Nature Phys*, 18(1):75, Dec. 2021a. ISSN 1745-2473. doi: 10.1038/s41567-021-01409-7. URL `https://www.nature.com/articles/s41567-021-01409-7`.

T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout. Scalable randomized benchmarking of quantum computers using mirror circuits [Data set]. `https://doi.org/10.5281/zenodo.5197499`, 2021b. Accessed: 2023-04-12.

T. Proctor, S. Seritan, E. Nielsen, K. Rudinger, K. Young, R. Blume-Kohout, and M. Sarovar. Establishing trust in quantum computations, 2022.

N. Rubin, D. Berry, A. Kononov, F. Malone, T. Khattar, A. White, J. Lee, H. Neven, R. Babbush, and A. Baczewski. Quantum computation of stopping power for inertial fusion target design. *arXiv preprint*, 2023. URL `https://arxiv.org/abs/2308.12352`.

P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. doi: 10.1137/S0097539795293172. URL `https://doi.org/10.1137/S0097539795293172`.

A. Vadali, R. Kshirsagar, P. Shyamsundar, and G. N. Perdue. Quantum circuit fidelity estimation using machine learning. *Quantum Mach. Intell.*, 6(1), June 2024.

H. Wang, P. Liu, J. Cheng, Z. Liang, J. Gu, Z. Li, Y. Ding, W. Jiang, Y. Shi, X. Qian, et al. Quest: Graph transformer for quantum circuit reliability estimation. *arXiv preprint arXiv:2210.16724*, 2022.

A Compute resources

All of the physics-aware neural networks were trained using a 6-Core Intel Core i9 processor on a MacBookPro 15.1 with 32GB of memory. Each model took roughly 15-20 wall clock minutes to train. Total training time, across the paper, totaled $\sim$ 160 wall clock minutes.

All of the simulations and data pre-processing was performed using a 6-core Intel Core i9 processor on a MacBookPro 15.1 with 32GB of memory. Each dataset took approximately 1 hour of wall clock time to create. This total includes the initial circuit creation, simulating the circuits, and encoding each circuit into a tensor.

B Code and data availability

The simulated 5-qubit data as well as records of all the physics-aware networks can be found in Anonymous. The CNNs and 5-qubit experimental datasets used in Hothem et al. [2023c] are available at Hothem et al. [2023a]. The data sets were originally located at Proctor et al. [2021b]. Each dataset was released under a CC-BY 4.0 International license.

All simulations were performed using a combination of `pygsti` version 0.9.11.2 [Nielsen et al., 2020] and `stim` version 1.13.0 [Gidney, 2021]. Models were trained and developed using `Keras` version 2.12.0 [Chollet et al., 2015] and `TensorFlow` version 2.12.0 [Abadi et al., 2015]. The physics-aware network model classes (`CircuitErrorVecScreenZErrorsWithMeasurementsBitstrings` for PST and `CircuitErrorVec` for entanglement fidelity) are available in the Supplementary Material.

C Datasets

Device	Geometry	Circuit types	Circuit widths	Circuit depths	Training set size	Validation set size	Test set size
ibmq_london	t-bar	mirror	1-5 qubits	3-515 layers	711 circuits	104 circuits	91 circuits
ibmq_ourense	t-bar	mirror	1-5 qubits	3-515 layers	930 circuits	124 circuits	114 circuits
ibmq_essex	t-bar	mirror	1-5 qubits	3-515 layers	713 circuits	93 circuits	86 circuits
ibmq_burlington	t-bar	mirror	1-5 qubits	3-515 layers	682 circuits	90 circuits	92 circuits
ibmq_vigo	t-bar	mirror	1-5 qubits	3-515 layers	1029 circuits	137 circuits	126 circuits
ibmq_yorktown	bowtie	mirror	1-5 qubits	3-515 layers	1097 circuits	132 circuits	140 circuits
Ring (x10)	ring	*i.i.d.* random	1-4 qubits	1-180 layers	2813 circuits	938 circuits	1250 circuits
Ring (x10)	ring	mirror	1-4 qubits	8-174 layers	-	-	750 circuits

Table 1: **Summary data of every dataset used in the paper.** The data for the ring processors is averaged over the 10 simulated datasets. See Figure 4 for images of each processor geometry (i.e., the qubit connectivity graph).

We provide additional details on the datasets used in the paper. Table 1 summarizes each dataset. We tracked all weight-2 errors with support on qubits connected by 2 hops in all the datasets. Below, we provide additional details on the circuit and error model generating processes.

C.1 Creating the circuits

In this subsection, we go over how the random *i.i.d.*-layer circuits and random mirror circuits were created for this paper. We will begin by explaining how we generated the random *i.i.d.*-layer circuits

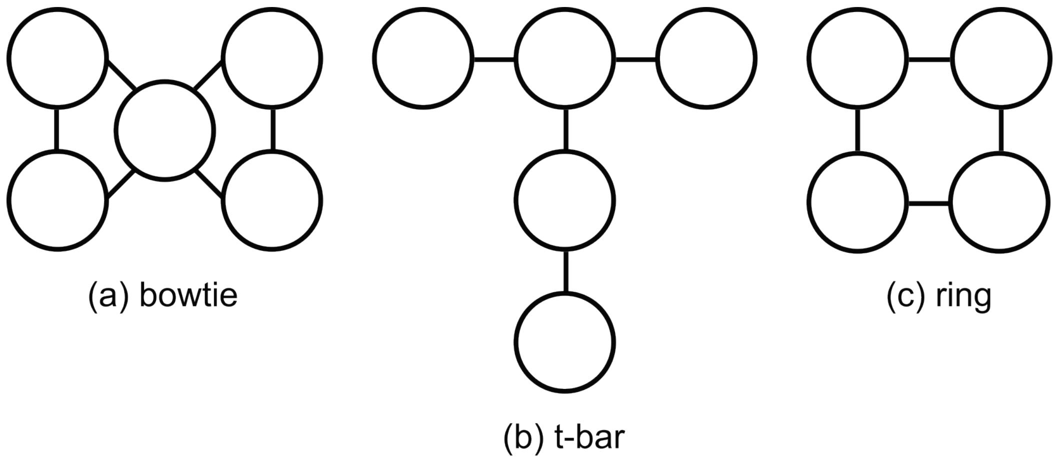

Figure 4: **Device geometries.** The connectivity graphs for the **(a)** 5-qubit `ibmq_yorktown` "bowtie" processor; **(b)** the remaining 5-qubit experimental "t-bar" processors; and **(c)** the 4-qubit simulated "ring" processor.

for a 4-qubit ring processor, and then explain the modifications needed to generate the random mirror circuits. This subsection's content is conceptual. The actual circuits were created in `pygsti` using the code in the Supplementary Material.

Each random *i.i.d.*-layer circuit c was created by a multi-step process. First, we randomly sampled a connected subset $\mathbb{Q}_c \subset \{\mathrm{Q0}, \mathrm{Q1}, \mathrm{Q2}, \mathrm{Q3}\}$ of qubits for which c is designed for. Then, we uniformly sampled c's depth from between 1 and a d_w, a pre-determined, circuit-width-dependent maximum depth. The depths d_w were selected to ensure that $s_F(c) > 90\%$ given the maximum error strengths used to create the simulated error model (Section C.2). Third, we randomly sampled a two-qubit gate density $\rho_{2\mathrm{Q}}$ between 0 and $2/3$. The density $\rho_{2\mathrm{Q}}$ determines the average number of two-qubit gates in each of c's layers. We then sampled each layer *i.i.d.* from all possible circuit layers on the qubits in $\mathbb{Q}_c$.

The random mirror circuits were generated using a similar multi-step process with two differences. The first difference is that we used a pre-determined maximum depth of $d_w/6$. We chose to reduce the pre-determined, circuit-width-dependent maximum depth so that the deepest random mirror circuits had roughly the same length as the deepest random *i.i.d.* circuits. The second difference is that we created a random mirror circuit on $\mathbb{Q}_c$. See Proctor et al. [2021a] for more details.

C.2 Creating an error model

In this subsection, we explain how we constructed the 4-qubit Markovian local coherent error model used in Section 6. Here we provide a conceptual explanation. The actual error model was created in `pygsti` using the code found in the Supplementary Material.

The 4-qubit Markovian local coherent error model was specified using the error generator framework explained in Section **??** and Blume-Kohout et al. [2022]. The error model consists of operation-dependent errors sampled according to a two-step process. The error strengths for each gate and qubit(s) pairs were independently sampled. First, we sampled an overall error strength ε_g for each one- and two-qubit gate g by randomly sampling from $[0, 1]$ and scaling by a pre-determined maximum error strength ($.025\%$). Then we sampled the relative error strengths $\vec{\varepsilon}_{g,\mathrm{rel}}$ of each of the $4^n - 1$ coherent errors, where $n = 1, 2$ for one- and two-qubit gates, respectively. We then normalized $\vec{\varepsilon}_{g,\mathrm{rel}}$ to obtain the actual error strengths according to the following equation:

$$\vec{\varepsilon}_g = \frac{\sqrt{\varepsilon_g} \cdot \vec{\varepsilon}_{g,\mathrm{rel}}}{\sqrt{\sum_i \varepsilon_{g,i}^2}}. \tag{10}$$

The re-scaling ensures that, to first-order, gate g contributes approximately ε_g to the circuit's entanglement infidelity (or PST, if appropriate).

D Networks

D.1 Physics-aware network details

Dataset	Metric	Model size	N_{hops}	N_{errors}	Dense units
5-qubit t-bar	$\mathrm{PST}(c)$	1218348	3	174	[30, 20, 10, 5, 5, 1]
5-qubit bowtie	$\mathrm{PST}(c)$	1596420	3	210	[30, 20, 10, 5, 5, 1]
4-qubit ring	$s_F(c)$	299772	2	132	[30, 20, 10, 5, 5, 1]

Table 2: **Summary data for the physics-aware networks used in the paper.**

Table 2 briefly outlines the hyperparameters and model sizes of the physics-aware neural networks used in this paper. All dense subunits used a ReLU activation function. All models were trained using `keras`'s Adam [Kingma and Ba, 2015] optimizer with a step size of $1e-3$ and with MSE as the loss function. Model training was cut short using early stopping. To help with training, we scaled $\mathrm{PST}(c)$ and $s_F(c)$ by 10000 when training the physics-aware networks. The notebooks in the Supplementary Material contain more details.

D.2 Convolutional neural network details

Details on the specific convolutional neural networks used in this paper are located in Hothem et al. [2023c]. We fine-tuned each network on high-PST experimental data using the Adam optimizer and early stopping.

The model architecture for the convolutional neural networks used in our simulations was selected via hyperparameter tuning. We tuned the number of convolutional layers, dense layers, etc. We performed XX search rounds using $\texttt{keras}-\texttt{tuner}$'s built-in Bayesian optimization class.

E Experimental results

In this section we provide prediction plots for each experiment.

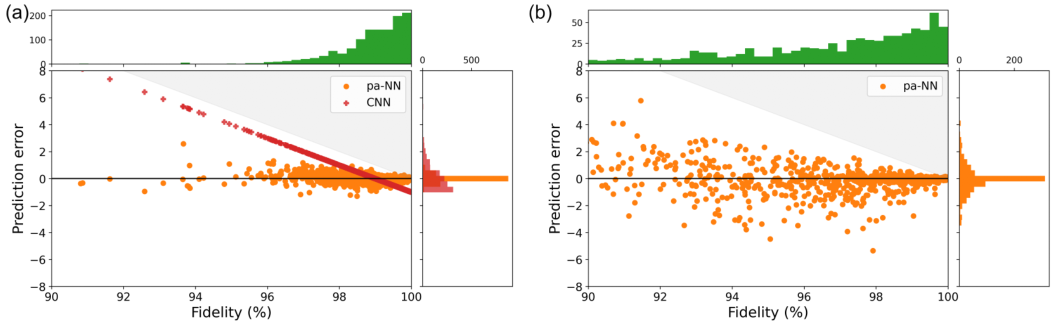

Figure 5: **Combined prediction error plot for the simulation**

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

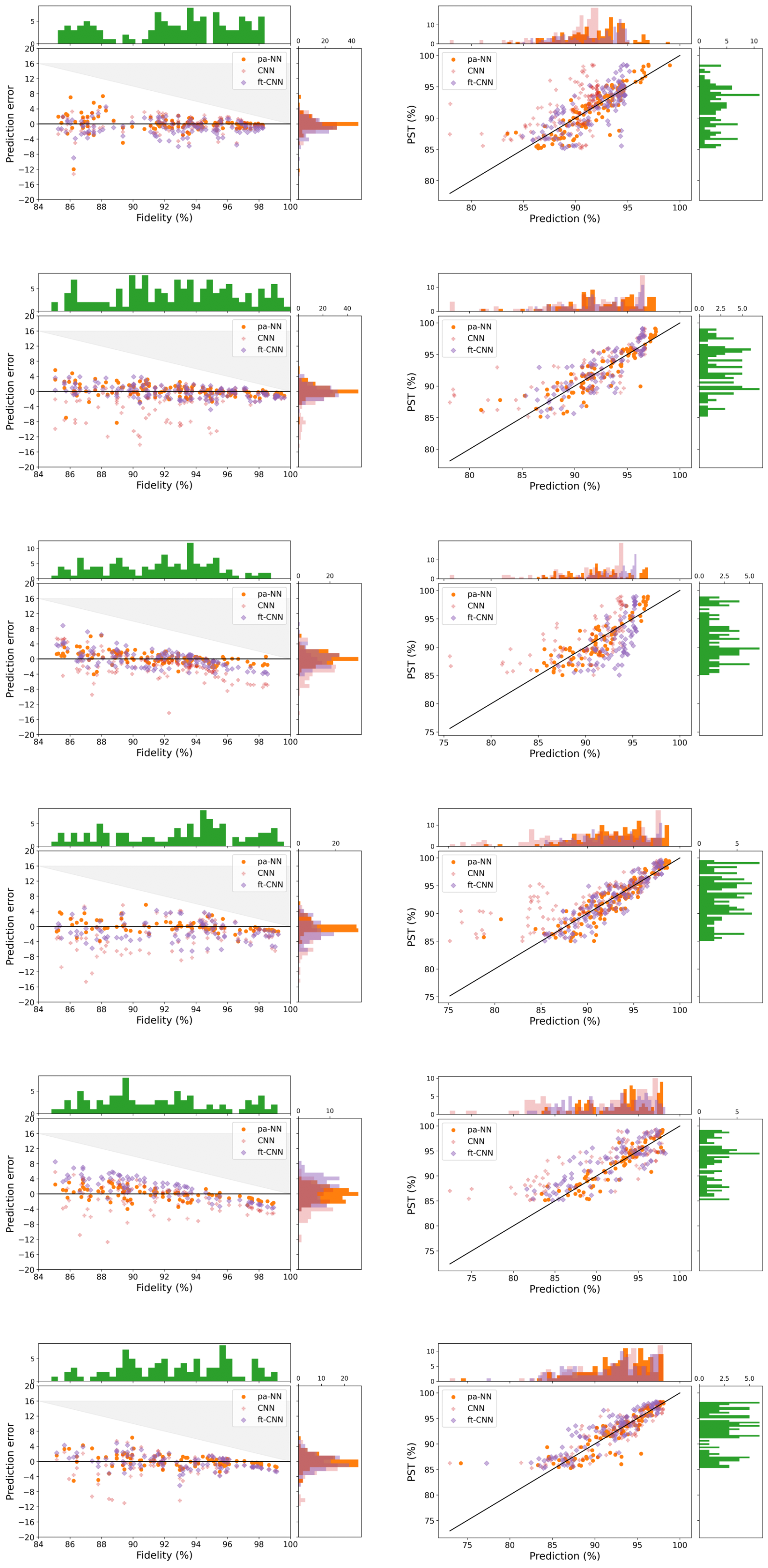

Figure 6: **All experiment results.**

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, we list the main claims and the section in which they are answered below.

   (a) We developed a new physics-aware neural network architecture for quantum capability learning: Section 3.

   (b) Our new approach achieves up to a $\sim 50\%$ and $\sim 76\%$ improvement over state-of-the-art convolutional neural networks on experimental and simulated data, respectively: Section 5 and 6.

   (c) Our new approach beats state-of-the-art convolutional networks, in part, due to their improved ability to model coherent errors: Section 6.

   (d) Our new approach achieves moderate prediction accuracy on an out-of-distribution prediction task: Section 6.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: We include a discussion of the limitations of the work in Section 7.1.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: We do not include any new theoretical results or proofs.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We believe that we provide sufficient details to reproduce the main experimental results of the paper. Readers should be able to recreate our results based on the details in the main body of the paper and the appendix, or by using the notebooks in the supplemental material.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We have provided an anonymized version of the code and novel data used in the paper with our submission. We provide explicit instructions on how to access the experimental data in Appendix B.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In Section 4 we explain how we processed the experimental and simulated datasets. Appendices C and **??** contain additional details on the datasets, specific network instantiations, and model training.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report error bars on the mean absolute error of the models' predictions in Section 6. We do not report error bars on the models' predictions in the experimental data as the original paper [Hothem et al., 2023c] reported their error bars as being trivial. However, we do provide the standard deviation of the percent change in the MAE across the experimental datasets, and we report log-likelihood ratios for each model on each experimental dataset, demonstrating substantial improvement by the physics-aware networks.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide details on the compute resources and compute time used in this work in Appendix A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We believe that we have conducted our research in a manner that conforms, in every respect, with the NeurIPS Code of Ethics. The only relevant areas of concern are the use of deprecated datasets and respect for copyright and fair use. We believe that we have not violated either of these requirements, although the cloud-accessed processors used in the experimental section are no longer available (but the datasets are not deprecated).

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We believe that there is little to no societal impact of our work. We believe that the following quote from the checklist guidelines is relevant: "it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster." While quantum computers might some day have a large societal impact, our work does not directly improve their ability to run programs with societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We explicitly reference each existing dataset in the References, and state each existing datasets license in Appendix B. All existing assets were licensed under a CC-BY 4.0 license, requiring proper attribution.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: At the time of submission, we have released code used in Section 6 and the new simulated dataset in the supplemental material. In the future we will release additional assets, such as the models trained on experimental data, publicly after receiving approval from our employer.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.