Supervised learning of random quantum circuits via scalable neural networks

Quantum Science and Technology

CrossMark

OPEN ACCESS

RECEIVED
12 August 2022

REVISED
21 January 2023

ACCEPTED FOR PUBLICATION
16 March 2023

PUBLISHED
27 March 2023

PAPER

Supervised learning of random quantum circuits via scalable neural networks

Simone Cantori[1], **David Vitali**[1,2,3] **and Sebastiano Pilati**[1,2,*]

[1] School of Science and Technology, Physics Division, University of Camerino, I-62032 Camerino (MC), Italy
[2] INFN-Sezione di Perugia, 06123 Perugia, Italy
[3] CNR-INO, 50125 Firenze, Italy
[*] Author to whom any correspondence should be addressed.

E-mail: sebastiano.pilati@unicam.it

Keywords: random quantum circuits, quantum computing, deep learning

Abstract
Predicting the output of quantum circuits is a hard computational task that plays a pivotal role in the development of universal quantum computers. Here we investigate the supervised learning of output expectation values of random quantum circuits. Deep convolutional neural networks (CNNs) are trained to predict single-qubit and two-qubit expectation values using databases of classically simulated circuits. These circuits are built using either a universal gate set or a continuous set of rotations plus an entangling gate, and they are represented via properly designed encodings of these gates. The prediction accuracy for previously unseen circuits is analyzed, also making comparisons with small-scale quantum computers available from the free IBM Quantum program. The CNNs often outperform these quantum devices, depending on the circuit depth, on the network depth, and on the training set size. Notably, our CNNs are designed to be scalable. This allows us exploiting transfer learning and performing extrapolations to circuits larger than those included in the training set. These CNNs also demonstrate remarkable resilience against noise, namely, they remain accurate even when trained on (simulated) expectation values averaged over very few measurements.

1. Introduction

Universal quantum computers promise to solve some relevant computational problems which are intractable for classical computers [1, 2]. In fact, the claim of quantum speed-up is justified only when the targeted computational task cannot be completed by any classical algorithm in a comparable computation time [3]. On the other hand, precisely the lack of efficient classical simulation methods hinders the further engineering of quantum devices with more and more qubits, as their development has to proceed without benchmark data. In this context, machine-learning techniques represent an attractive alternative to direct classical simulations, since they might feature a lower computational cost. In fact, supervised machine learning from classically simulated datasets has already emerged as a promising and computationally feasible strategy to predict the ground-state properties of complex quantum systems, including, e.g. small molecules [4, 5], solid-state systems [6, 7], atomic gases [8, 9], and protein-ligand complexes [10, 11]. Moreover, it has recently been proven that data-based algorithms can solve otherwise classically intractable computational tasks [12], including predicting ground-state properties of quantum systems, and rigorous guarantees on the accuracy and on the scaling of the required training set size have been demonstrated [13]. Still, producing training sets for supervised learning via classical computers quickly becomes unfeasible as the system size increases. In the context of ground-state simulations, this problem has been addressed via scalable neural networks [14, 15]. These allow performing transfer learning from small to large systems [9, 16], and even to extrapolate to sizes larger than those included in the training set. So, it is natural to wonder whether neural networks might also be trained to emulate quantum circuits, and whether they might extrapolate to large qubit numbers where exact simulation methods become problematic.

The above considerations led us to investigate the supervised learning of gate-based quantum computers. Our goal is to demonstrate that deep convolutional neural networks (CNNs) can be trained to predict relevant output properties of quantum circuits, both from exact classical simulations of expectation values, as well as from noisy (simulated) measurements. Remarkably, we show that the CNNs trained on random circuits are able to emulate a broad category of quantum circuits, including, e.g. the Bernstein–Vazirani (BV) algorithm. Furthermore, thanks to a properly designed scalable structure, they provide accurate extrapolations for circuits larger than those included in the training set. Our findings also support the long-term perspective of using quantum computers to produce training data for supervised learning, possibly allowing them to emulate classically intractable algorithms. Clearly, if distributed to many users, such trained networks would allow these users to benefit from the quantum device even without having direct access to it.

In detail, in this article we consider large ensembles on quantum circuits built with gates randomly selected (mostly) from a discrete approximate universal set. It is worth mentioning that the sampling from random circuits is the computational task considered in the recent demonstrations of quantum supremacy [17, 18]. A second gate set, including continuous rotations plus an entangling layer, is addressed in the appendix. The target of the supervised learning is a partial description of the circuit output, namely, single-qubit and two-qubit expectation values. As we demonstrate, this limited information is still sufficient to emulate the category of quantum circuits with only one or two possible output bit strings. Interestingly, this category includes relevant circuits such as the BV algorithm [19]. An appropriate one-hot encoding of the constituent gates is designed to unequivocally describe the discrete circuits (for the continuous set, see the appendix). Their output is classically computed using the Qiskit library [20], considering, for the training data, numbers of qubits up to $N \simeq 10$ and circuit depths (number of gates per qubit) up to $P \simeq 10$. Significantly larger circuits are also considered in the testing processes performed via extrapolation procedures. Deep CNNs are trained to map the circuit descriptors to the output expectation values. The CNNs are tested on previously unseen circuits, and we analyze how the predictions accuracy varies as a function of the circuit size, of the network depth, and of the training-set size. We also compare the accuracy of the trained CNNs against the one of various small quantum computers available via IBM Quantum Experience [21]. Generally, the CNNs outperform the freely available noisy intermediate-scale quantum (NISQ) processors, unless the circuit's depth is increased at fixed CNN parameters. Notably, our CNNs are designed to be scalable. This allows us investigating transfer-learning and extrapolation protocols. Specifically, we show that the learning of large circuits can be accelerated via a pretraining performed on smaller circuits. Furthermore, we employ CNNs trained on small circuits to predict the output of circuits with more qubits, up to twice as much (or even more for continuous circuits). Interestingly, CNNs also learn (from random circuits) to emulate the BV algorithm, even when the number of qubits is increased by several orders of magnitude. Finally, we consider the training on noisy expectation values, obtained as averages over a variable number of (simulated) measurements. We find that the CNNs are able to filter this noise, providing remarkably accurate estimates of the exact expectation values even when very few measurements are considered in the training data. This validates the idea of using data produced by NISQ computers to train deep neural networks.

The rest of the article is organized as follows: in section 2 we describe the random circuits built using the discrete approximate universal sets, the appropriate one-hot encoding we design for supervised learning, the CNNs we adopt, and the target expectation values. Furthermore, we discuss the class of random circuits that can be emulated from the target values we address. The predictions of the trained CNNs are analyzed in section 3. In the same section these predictions are also compared with those obtained with small quantum computers available from IBM Quantum Experience. Then, transfer learning and extrapolation protocols are analyzed; notably, in the same section we also discuss the training on noisy expectation values. Section 4 reports our conclusions and an outlook on future perspectives. In the appendix, the extrapolation technique is tested on circuits with continuous outputs, built using single-qubit random rotations plus an entangling gate.

2. Methods

2.1. Representation of random circuits

Our goal is to train deep CNNs to map univocal representations of random circuits to their output expectation values. Specifically, we mostly consider circuits built with two single-qubit gates, namely, the T-gate (T) and the Hadamard gate (H), together with one two-qubit gate, namely, the controlled-not gate (CX). Notably, the set $\mathcal{S} = \{T, H, CX\}$ constitutes an approximately universal set [22, 23], meaning that any unitary operator can be implemented using these three gates. It is worth noticing that the identity $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ can be represented as $I = H^2$. Below, an extended set explicitly including the gate I will be

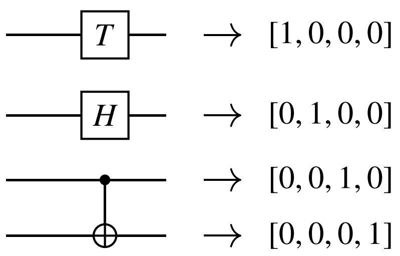


Figure 1. Visualization of the one-hot encoding that represents the quantum gates in the set $\mathcal{S}$. The full circle and the empty circle with plus sign represent the control and the target qubit of the CX gate, respectively.

discussed. We adopt the standard computational basis corresponding to the eigenstates of the Pauli matrix $Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$. In this basis, the three considered gates are represented by the following matrices:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix} \qquad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{1}$$

In the following, we consider circuits with N qubits and P layers of gates. Therefore, the integer P corresponds to the number of gates per qubit; this parameter will be referred to also as the circuit depth. In every layer, each qubit is processed by a gate randomly selected from the set $\mathcal{S}$. Notice that the two-qubit gate CX acts on a control and on a target qubit. To simplify the circuit representation, we allow only one CX gate at every layer. Circuits with more CX gates per layer can be emulated by deeper circuits satisfying the constraint. This constraint allows us adopting a relatively simple univocal circuit representation. It is based on the following one-hot encoding of the gate acting on each qubit: the T-gate corresponds to the vector $(1,0,0,0)$, the H-gate to $(0,1,0,0)$, the control qubit of the CX-gate corresponds to $(0,0,1,0)$, while the target qubit corresponds to $(0,0,0,1)$. This map is also represented in figure 1. Therefore, the feature vector representing a random circuit is a four-channel two-dimensional binary matrix with dimensions $N \times P \times 4$. Analogous gate-based descriptions of quantum circuits have been adopted in [24, 25]. Despite the constraint on the number of CX gates per layer, the number Q of possible circuits rapidly grows with N and P. This number can be computed as:

$$Q = \sum_{m=0}^{P} 2^{N \cdot P - 2m} \binom{P}{m} 2^m \binom{N}{2}^m, \tag{2}$$

where m is the number of CX-gates in the circuit. The first term, namely, $2^{N \cdot P - 2m}$, represents the possible combinations considering only the T-gates and the H-gates. The second term, namely, $\binom{P}{m}$, corresponds to the possible combinations of the CX-gates in P layers. The third term, namely, 2^m, corresponds to the choice of the control and of the target qubit for each CX gate. Finally, the term $\binom{N}{2}^m$ corresponds to the available pairs for each CX-gate. For example, for the smallest circuit size considered in this article, namely, $N = 3$ and $P = 5$, one has $Q = 3.2 \times 10^6$ possible random circuits. For the largest size, namely, $N = 20$ and $P = 6$, one has the astronomic number $Q \simeq 8^{48}$. This means that it is virtually impossible to create a dataset exhausting the whole ensemble of possible circuits. We instead resort to the generalization capability of deep CNNs. These are expected to provide accurate predictions for previously unseen instances, even when trained on (largely non-exhaustive) training sets including a number $N_{\text{train}} \ll Q$ of training instances.

While the set $\mathcal{S}$ is, in principle, universal, the choice of operating on all qubits in every layer implies that some unitary operators cannot be represented. Therefore, we also consider the extended set $\mathcal{S}^* = \mathcal{S} \bigcup \{I\}$, where the identity is explicitly included. For this set, a five channel one-hot encoding is needed. We use the map represented in figure 2. Most of the results reported in this article are based on the set $\mathcal{S}$, unless stated otherwise. Notably, this set is flexible enough to represent the BV algorithm, which we use as a relevant test bed. For a few representative test cases, we also consider the extended gate set $\mathcal{S}^*$, finding very similar performances. Furthermore, an additional gate set is addressed in the appendix, focusing on the extrapolation technique. This set includes single-qubit rotations with continuous random angles, plus CX-gates. It displays continuous outputs, as opposed to the approximate universal sets $\mathcal{S}$ and $\mathcal{S}^*$, which lead to discrete values of the outputs. Also, in this continuous set the number of entangling gates scales with N, thus possibly representing a more stringent test for the extrapolation procedure.

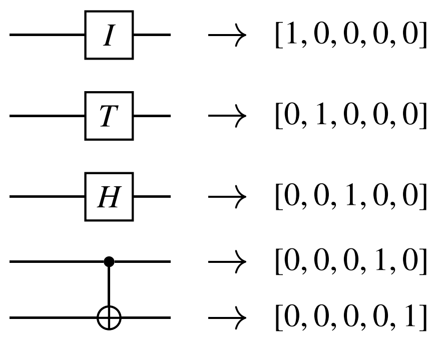


Figure 2. Visualization of the one-hot encoding that represents the quantum gates in the set $\mathcal{S}^*$.

2.2. Target values

The output state of a quantum circuit can be written as

$$|\psi\rangle = U|0\rangle^{\otimes N}, \tag{3}$$

where the tensor product $|0\rangle^{\otimes N} \equiv |0\rangle_1 \otimes \ldots |0\rangle_N$ is the input state and the unitary operator U represents the sequence of quantum gates that constitute the circuit. Here and in the following, we indicate with $|0\rangle_i$ and $|1\rangle_i$ the eigenvectors of the Pauli operator Z_i acting on qubit $i = 1, \ldots, N$, such that $Z_i |0\rangle_i = |0\rangle_i$ and $Z_i |1\rangle_i = -|1\rangle_i$. With this notation, each state $|\mathbf{x}\rangle$ of the many-qubit computational basis corresponds to a bit string $\mathbf{x} = x_1 \ldots x_N$, where $x_i = 0, 1$ for $i = 1, \ldots, N$. Our goal is to perform supervised learning of output expectation values. First, we focus on the single-qubit expectation values

$$\langle Z_i \rangle \equiv \langle \psi | Z_i | \psi \rangle. \tag{4}$$

These expectation values can be computed as

$$\langle Z_i \rangle = \left|\langle \psi | 0 \rangle_i\right|^2 - \left|\langle \psi | 1 \rangle_i\right|^2. \tag{5}$$

It is convenient to perform the following rescaling:

$$z_i = 1 - \frac{\langle Z_i \rangle + 1}{2}, \tag{6}$$

so that $z_i \in [0, 1]$, and $z_i = 0$ corresponds to $|0\rangle_i$, while $z_i = 1$ corresponds to $|1\rangle_i$. The rescaled variable z_i is the first target value we address for supervised learning. It is worth anticipating that we will consider both CNNs designed to predict only one expectation value, say, z_1, and CNNs that simultaneously predict all single-qubit expectation values z_i, for $i = 1, \ldots, N$. This is discussed with more details in section 2.4.

For illustrative purposes, we show in figure 3 the distribution of the target value z_1 over an ensemble of random circuits built with $\mathcal{S}$. Four representative circuit sizes are considered. Clearly, here the possible expectation values are discrete. In particular, one notices a relatively large probability of the output value $z_1 = 1/2$. Circuits with continuous outputs are considered in the appendix.

The second target values we consider are the two-qubit expectation values $\langle Z_i Z_j \rangle$, where, in general, $i, j = 1, 2, \ldots, N$. Specifically, we focus on the case $i = 1$ and $j = 2$, and the target value is the rescaled variable:

$$z_{12} = 1 - \frac{\langle Z_1 Z_2 \rangle + 1}{2}. \tag{7}$$

Clearly, single-qubit and two-qubit Pauli-Z expectation values represent a limited description of the circuit output. However, this information is sufficient to unambiguously identify the output of a significant category of quantum circuits. This category is described in the following subsection, where we also discuss some relevant examples belonging to the category.

2.3. Emulable quantum algorithms

For certain quantum algorithms, only a small subset of the 2^N output bit strings have non-zero measurement probability. In fact, some relevant circuits have only one possible outcome (in the absence of noise and errors). These are discussed below. First, we consider the more generic category for which only two output bit strings, which we indicate as $\mathbf{a}$ and $\mathbf{b}$, have non-zero measurement probabilities $p(\mathbf{a}) = |\langle \mathbf{a} | \psi \rangle|^2$ and

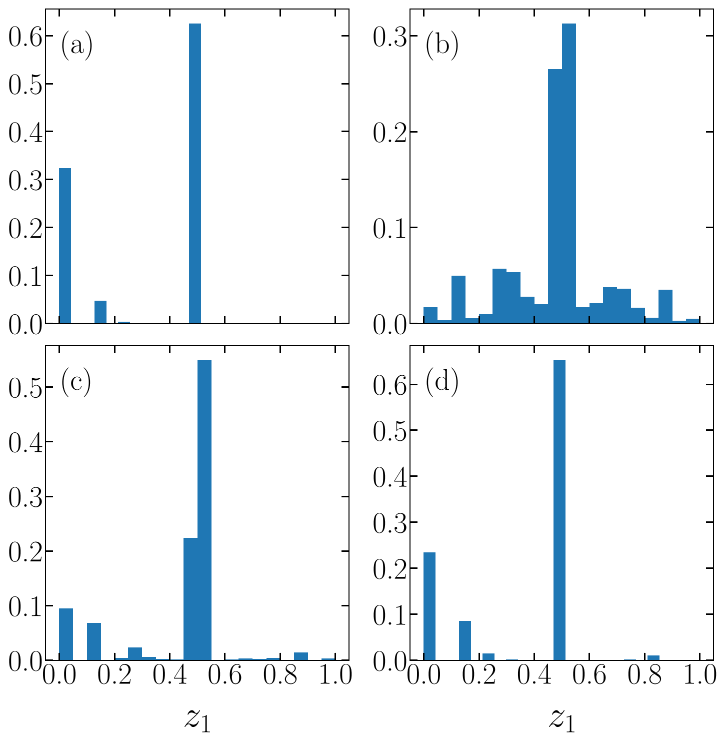


Figure 3. Normalized histograms of the rescaled single-qubit expectation values $z_1 = 1 - \frac{\langle Z_1\rangle + 1}{2}$ over ensembles of random circuits. Different number of qubits N and circuit depths P are considered in the four panels: $N = 3$ and $P = 5$ (a); $N = 3$ and $P = 20$ (b); $N = 5$ and $P = 10$ (c); $N = 20$ and $P = 6$ (d). The ensembles in (a)–(c), and the one in (d) include 5000 and 2000 circuits, respectively.

$p(\mathbf{b}) = |\langle \mathbf{b}|\psi\rangle|^2$. With this notation, one has $p(\mathbf{a}) + p(\mathbf{b}) = 1$. For this category of circuits, the expectation values $\langle Z_i\rangle$ and $\langle Z_iZ_j\rangle$ provide all the information required to unambiguously identify the bit strings **a** and **b**. This statement is proven here by providing an explicit algorithm. It is assumed that $p(\mathbf{x}) = 0$ for $\mathbf{x} \neq \mathbf{a}, \mathbf{b}$ and that the expectation values mentioned above are known. It is worth emphasizing that, in fact, if $p(\mathbf{a}) \neq p(\mathbf{b})$, knowledge of the single-qubit expectation values suffices. Indeed, the values $\langle Z_iZ_j\rangle$ are only used when $p(\mathbf{a}) = p(\mathbf{b})$—see case iv) below—and such case is easily identified since one must have $\langle Z_i\rangle = 0$ for at least one qubit i.

Proof. The qubit are analyzed in the order $i = 1, \ldots, N$. Four possible cases have to be separately treated:

(i) If $\langle Z_i\rangle = 1$, the corresponding bits are set to $a_i = b_i = 0$.
(ii) If $\langle Z_i\rangle = -1$, one sets $a_i = b_i = 1$.
(iii) If $\langle Z_i\rangle \in (-1, 1)$ and either $i = 1$ or $i > 1$ with $a_j = b_j$ for $j = 1, \ldots, i-1$ we arbitrarily set, without loss of generality, $a_i = 1$ and $b_i = 0$. Notice that we can also infer the two probabilities: $p(\mathbf{a}) = \frac{1-\langle Z_i\rangle}{2}$ and $p(\mathbf{b}) = \frac{1+\langle Z_i\rangle}{2}$.
(iv) Otherwise, when $p(\mathbf{a}) \neq p(\mathbf{b})$ (this is known from case iii)), two expectation values are possible. One is $\langle Z_i\rangle = \langle Z_j\rangle$, where the integer $j \in [1, i-1]$ is the first index such that $\langle Z_i\rangle \in (-1, 1)$ (at least one exists); in this case one sets $a_i = a_j$ and $b_i = b_j$. If, instead, $\langle Z_i\rangle \neq \langle Z_j\rangle$, one sets $a_i = 1 - a_j$ and $b_i = 1 - b_j$. When $p(\mathbf{a}) = p(\mathbf{b}) = 1/2$, one must have $\langle Z_i\rangle = 0$, and we also know that an integer $j \in [1, i-1]$ such that $\langle Z_j\rangle = 0$ exists. In this situation, $\langle Z_iZ_j\rangle = \pm 1$. If $\langle Z_iZ_j\rangle = 1$, one sets $a_i = a_j$ and $b_i = b_j$. If, instead, $\langle Z_iZ_j\rangle = -1$, one sets $a_i = 1 - a_j$ and $b_i = 1 - b_j$. □

As discussed in the previous paragraph, the single-qubit expectation values $\langle Z_i\rangle$, eventually combined with $\langle Z_iZ_j\rangle$, are sufficient to identify the output bit strings when only two of them have non-zero probability. Clearly, the values $\langle Z_i\rangle$ are sufficient when only one bit string is possible. Interestingly, some paradigmatic quantum algorithm belong to this group. A relevant example is the Deutsch–Jozsa algorithm [26]. This allows assessing whether a Boolean function $f\colon \{0,1\}^{(N-1)} \to \{0,1\}$ is either constant or balanced. The algorithm involves measuring the first $N-1$ qubits. If the outcome corresponds to the bit string $00\ldots0$, the function is constant, otherwise the function is balanced. Predicting $N-1$ single-qubit expectation values allows reaching the same result. Practically, if $\langle Z_i\rangle = 1$ for $i = 1, 2, \ldots, N-1$, the only possible bit string is the bit string $00\ldots0$, corresponding to a constant function. Otherwise, the function is balanced. Also in the BV algorithm [19] there is only one possible outcome. This algorithm is designed to identify an unknown bit string $\mathbf{w} \in \{0,1\}^{N-1}$, assuming we are given an oracle that implements the Boolean function $f\colon \{0,1\}^{N-1} \to \{0,1\}$ defined as $f(\mathbf{x}) = \mathbf{x} \circledcirc \mathbf{w}$, where the symbol $\circledcirc$ represents the dot product modulo 2. Notably, the BV algorithm provides the answer with one function query, outperforming classical computers which require $N-1$ queries. Single-qubit expectation values allow identifying $\mathbf{w}$: $\langle Z_i \rangle = 1$ corresponds to $w_i = 0$, while $\langle Z_i \rangle = -1$ corresponds to $w_i = 1$. The Grover algorithm with two searched items [27] and the quantum counting algorithm [28] have two output bit strings with much higher probabilities than the other strings. Therefore one could use the predictions for $\langle Z_i \rangle$ and $\langle Z_i Z_j \rangle$ to emulate also these latter algorithms.

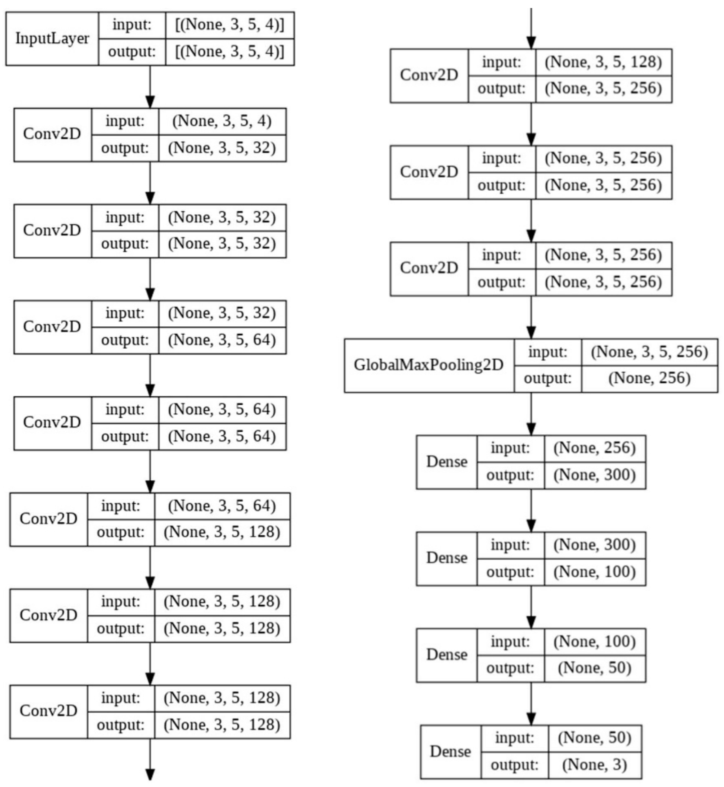
2.4. CNNs and training protocol

The CNNs considered in this article have the overall architecture described in figure 4. Relevant variations occur mostly in the last layer. Therein, the number of neurons corresponds to the desired number of outputs N_o. Specifically, we consider CNNs designed to predict only one expectation value at a time ($N_o = 1$), as well as N expectation values simultaneously ($N_o = N$). Clearly, in the first case, the last layer includes only one neuron. In the second case, it includes N neurons. The overall network structure we adopt is standard in fields such as, e.g. image classification or object detection. The first part includes N_c multi-channel convolutional layers, which create filtered maps of their input. The maps are created by scanning the input using (typically small) filters, featuring a fixed number of parameters. It is worth pointing out that the size of a convolutional layer's output scales with the corresponding input, even though the number of parameters in the filter is fixed. In turn, this means that the same filters could be applied to different input sizes. As in standard CNNs, the output of the convolutional part of the network is connected to a few dense layers (four in our case) with all-to-all interlayer connectivity. These dense layers constitute the venue where high-level operations on the effective features extracted by the convolutional layers occur. In standard CNNs, the connection between the last convolutional layer and the first dense layer is commonly performed through so-called flatten layers. This choice forces to scale the width (i.e. the number of neurons and of the corresponding parameters) of the first dense layer with the size of the network's input. Therefore, the whole network would be applicable only to one circuit size. Instead, we perform the connection using a global pooling layer. This extracts the maximum values of each (whole) map in the last convolutional layer. Thus, the output size of the convolutional part gets fixed: it corresponds to the number of filters in the last convolutional layer. This feature was adopted in [15] for the supervised learning of ground-state energies of quantum systems. It allowed implementing scalable CNNs, i.e. networks that can be trained on heterogeneous datasets including different system sizes and that can predict properties for sizes larger than those included in the training set. Scalable networks for physical and chemical systems have been implemented also in [9, 14, 16, 29, 30], using different strategies. Here, we exploit the global pooling layer to allow a single CNN addressing different circuit sizes. Notice, however, that full scalability is obtained only when the CNN predicts only one expectation value (with a single neuron in the output layer) at a time. More expectation values corresponding to different qubits can, in fact, be predicted even by the single output network. However, these predictions have to be performed in a sequential manner, by feeding the network with an appropriate swapping of the features. Specifically, when the goal is to predict, say, z_j, for any $j \in [2, N]$, one can employ a CNN trained to predict z_1, swapping the rows 1 and j of the network's input. If, instead, the goal is to simultaneously predict the expectation values corresponding to all qubits, obviously the number of neurons in the last layer has to be adapted to the targeted qubit number. In this case, full scalability is lost, since more parameters have to be trained if the qubit number increases.

The training of the CNN is performed by minimizing the loss function. For the discrete circuits, we adopt the binary cross-entropy:

$$\mathcal{L} = -\frac{1}{N_{\text{train}}} \sum_{k=1}^{N_{\text{train}}} \sum_{i=1}^{N_o} \left[y_i^{(k)} \log(\hat{y}_i^{(k)}) + (1 - y_i^{(k)}) \log(1 - \hat{y}_i^{(k)}) \right], \tag{8}$$

where N_{train} is the number of instances in the training set, N_o is the number of outputs (corresponding to the number of nodes in the last dense layer), and $\hat{y}_i^{(k)}$ is the network prediction corresponding to the ground-truth target value $y_i^{(k)}$. As discussed above, we consider the cases $N_o = N$ and $N_o = 1$. In the first case, the N target values correspond to all rescaled single-qubit expectation values: $y_i = z_i$, for $i = 1, \ldots, N$. In the latter case, we consider only one (rescaled) single-qubit expectation value, namely, $y_1 = z_1$, or one (rescaled) two-qubit expectation value, namely, $y_1 = z_{12}$. The optimization method we adopt is a successful variant of the stochastic gradient-descent algorithm, named Adam [31]. No benefit is found by introducing a regularization term in the loss function. Instead, batch normalization layers are included after every layer, before the application of the activation function. The chosen mini-batch size is in the range $N_b \in [128, 512]$,



InputLayer input: [(None, 3, 5, 4)] output: [(None, 3, 5, 4)]
Conv2D input: (None, 3, 5, 4) output: (None, 3, 5, 32)
Conv2D input: (None, 3, 5, 32) output: (None, 3, 5, 32)
Conv2D input: (None, 3, 5, 32) output: (None, 3, 5, 64)
Conv2D input: (None, 3, 5, 64) output: (None, 3, 5, 64)
Conv2D input: (None, 3, 5, 64) output: (None, 3, 5, 128)
Conv2D input: (None, 3, 5, 128) output: (None, 3, 5, 128)
Conv2D input: (None, 3, 5, 128) output: (None, 3, 5, 128)
Conv2D input: (None, 3, 5, 128) output: (None, 3, 5, 256)
Conv2D input: (None, 3, 5, 256) output: (None, 3, 5, 256)
Conv2D input: (None, 3, 5, 256) output: (None, 3, 5, 256)
GlobalMaxPooling2D input: (None, 3, 5, 256) output: (None, 256)
Dense input: (None, 256) output: (None, 300)
Dense input: (None, 300) output: (None, 100)
Dense input: (None, 100) output: (None, 50)
Dense input: (None, 50) output: (None, 3)

Figure 4. Representation of the CNN for the illustrative case of $N = 3$ qubits and circuit depth $P = 5$. Boxes report the input and the output shapes of each layer. This example includes $N_c = 10$ convolutional layers (Conv2D). Their input and output shapes have dimensions: (B, L_1, L_2, F), where $B =$ None is the mini-batch size (not specified), $L_1 = N$ and $L_2 = P$ denote the size of the two-dimensional feature maps, while F is the number of filters. For the dense layers (Dense), we only have the mini-batch size and the number of nodes. The figure omits the batch normalization layers, included in every layer before the application of the activation function. The latter corresponds to the *Mish* function [32], except for the last node where it corresponds to the *sigmoid* function. It is worth highlighting the global maximum pooling layer (GlobalMaxPooling2D) connecting the last convolutional layer to the first dense layer.

depending on the training set size. The CNNs are trained on large ensembles of random quantum circuits. These are implemented and simulated using the Qiskit library. These simulations provide numerically exact predictions for the considered expectation values. We also consider noisy estimates of the expectation values obtained from finite samples of simulated measurements. These estimates are employed in section 3.3 to inspect the impact of errors in the training set on the prediction accuracy. After training, the CNNs are tested on previously unseen random circuits. It is also worth mentioning that we remove possible circuit replicas, both in the training and in the test sets. In fact, only for the smallest circuits we consider, corresponding to $N = 3$ and $P = 5$, one can find within $N_{\mathrm{train}} \simeq 10^6$ training circuits a non-negligible number of identical replicas.

3. Results

3.1. Single-qubit expectation values

The CNNs described in section 2.4 are trained to map the circuit descriptors (see section 2.1) to various outputs. We first focus on a CNN designed to simultaneously predict the rescaled single-qubit expectation values z_i, with $i = 1, \ldots, N$. Here, the discrete gate sets are considered. Figure 5 displays the scatter plots of predicted versus ground-truth expectation values, for four representative circuit sizes. The tests are performed on random circuits distinct from those included in the training set. One notices a close correspondence in all four examples. In figure 6, we analyze how the prediction accuracy varies with the circuit depth P. Three datasets correspond to random circuits generated with the gates from the discrete set $\mathcal{S}$, but with different qubit numbers N. For the fourth dataset the gates are sampled from the extended set $\mathcal{S}^*$ (see section 2.1). To quantify the accuracy, we consider the coefficient of determination:

$$R^2 = 1 - \frac{\sum_{k=1}^{N_{\mathrm{test}}} \sum_{i=1}^{N_o} (y_i^{(k)} - \hat{y}_i^{(k)})^2}{\sum_{k=1}^{N_{\mathrm{test}}} \sum_{i=1}^{N_o} (y_i^{(k)} - \bar{y})^2} \tag{9}$$

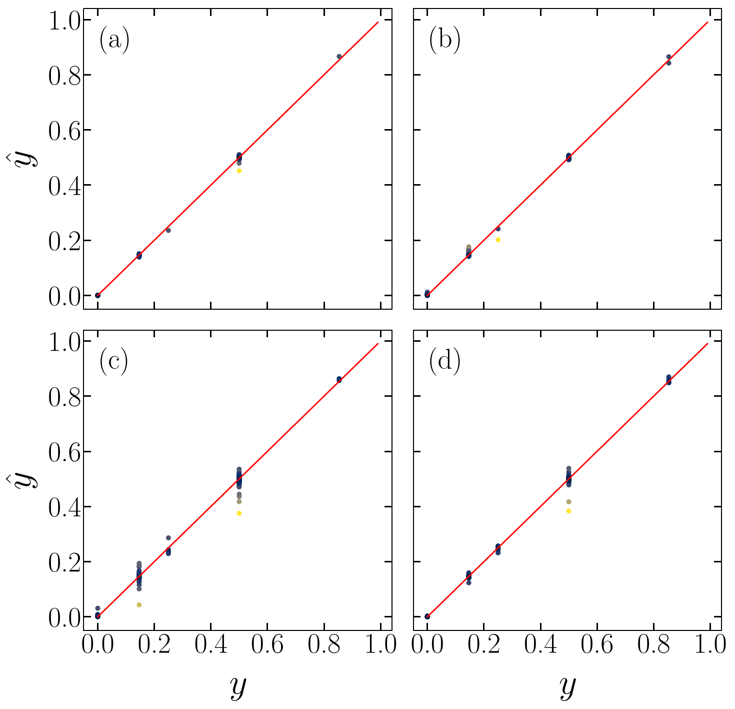


Figure 5. Rescaled single-qubit expectation values $\hat{y}$ predicted by the CNN versus the ground-truth results $y \equiv z_i$, with $i = 1, \ldots, N$. The latter results are simulated via Qiskit. Different number of qubits N and circuit depths P are considered in the four panels: $N = 3$ and $P = 5$ (a); $N = 3$ and $P = 7$ (b); $N = 5$ and $P = 5$ (c); $N = 5$ and $P = 7$ (d). These test sets include 100 random circuits. The color scale (blue to yellow) represents the absolute discrepancy $|\hat{y} - y|$. The (red) line represents the bisector $\hat{y} = y$.

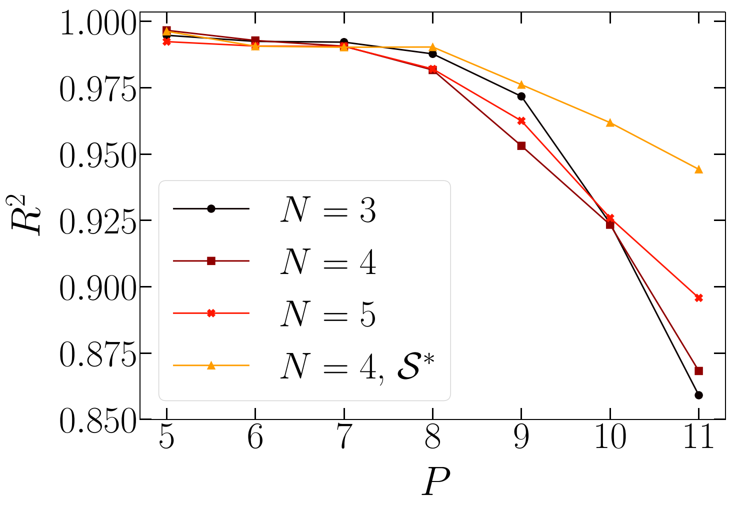


Figure 6. Coefficient of determination R^2 for rescaled single-qubit expectation values z_i as a function of the circuits depth P. Three datasets correspond to random circuits built with gates from the set $\mathcal{S}$, but having different qubit numbers N (see legend). For the fourth dataset, the extended gate set $\mathcal{S}^*$ is used. For $P = 5$ the neural network is trained from scratch on $N_{\text{train}} \simeq 10^6$ instances, while for $P > 5$ the training starts with the optimized weights and biases for $P - 1$.

where $\hat{y}_i^{(k)}$ is the prediction associated to the ground-truth target value $y_i^{(k)}$, $\bar{y}$ is the average of the target values, N_{test} is the number of random circuits in the test set, and N_o is the number of outputs. In this case, we have $N_o = N$ outputs. It is worth stressing that R^2 quantifies the accuracy in relation to the intrinsic variance of the test data. Indeed, it accounts for the (mean squared) deviations from the ground-truth values, compared to the typical fluctuations from the mean. This metric is therefore suitable for fair comparisons among different circuits sizes, which might display a more or less pronounced tendency of the output values to cluster at or close a mean value. For small P one observes remarkably high scores $R^2 \simeq 1$, corresponding to essentially exact predictions. However, the accuracy significantly decreases for deeper circuits. It is worth

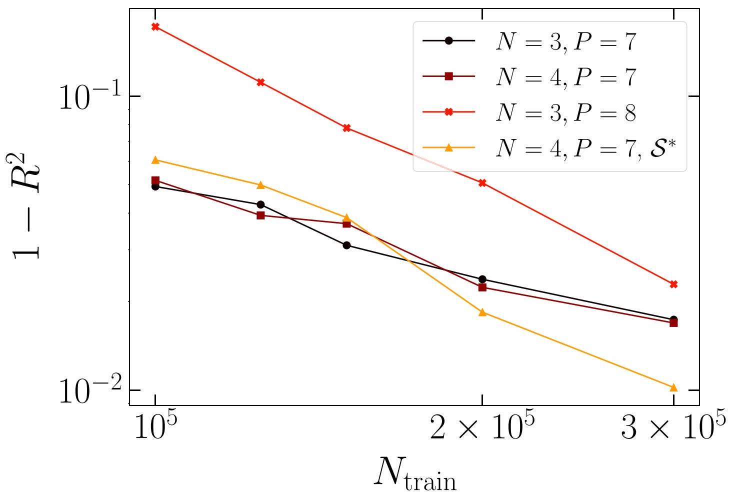


Figure 7. Normalized prediction error $1-R^2$ for (rescaled) single-qubit expectation values z_i as a function of the training set size N_{train}. Three datasets correspond to random circuits built with gates from the set $\mathcal{S}$, but having different qubit numbers N and circuit depths P (see legend). For the fourth dataset, the extended gate set $\mathcal{S}^*$ is used. The adopted CNN is described in figure 4.

pointing out that, in this analysis, the depth of the CNN is not varied, and the training set size is also fixed at $N_{\mathrm{train}} \simeq 10^6$.

The prediction accuracy can be improved by enlarging the training set, or by increasing the depth of the CNN. The first approach is analyzed in figure 7, for three representative circuit sizes. One notices the typical power-law suppression (see, e.g. [4, 33]) of the prediction error $1-R^2 \propto N_{\mathrm{train}}^{-\alpha}$, where $\alpha > 0$ is a non-universal exponent. The second approach is analyzed in figure 8. We find that the R^2 score systematically increases with the number of convolutional layers N_c. It is worth pointing out that the deepest CNNs adopted in this article include around 2×10^6 parameters. This number does not represent a noteworthy computational burden for modern high-performance computers, in particular if equipped with state-of-the-art graphic processing units. In fact, significantly deeper neural networks are routinely trained in the field of computer vision. Relevant examples are VGG-16, VGG-19 [34] and ConvNeXt [35]. Instead, creating copious training sets for circuits with $N > 10$ qubits becomes computationally expensive. In fact, simulating circuits with $N \gg 10$ qubits is virtually impossible, unless the analysis is restricted to specific circuits types for which tailored algorithms exist. Relevant examples are slightly entangling circuits and those dominated by Clifford gates; these two circuit types can be efficiently simulated via tensor-network methods and near-Clifford algorithms [36], respectively. Two strategies to circumvent the computational-cost problem are discussed in the section 3.2. With the second, predictions for large qubit numbers can be provided, without additional computational burdens in the training process, even in the regime where exact circuit simulations via general-purpose algorithms become impractical.

3.2. Transfer learning and extrapolation

We exploit the scalability of the CNNs featuring the global pooling layer to implement two strategies, namely, transfer learning and extrapolation. The first strategy is rather common in fields such as, e.g. computer vision [37]. It involves performing an extensive training of a deep CNN on a large generic database. Then, the pre-trained network is used as starting point in a second training performed on a more specific, typically smaller, database. At this stage the CNN learns to solve the targeted task. This approach has already been adopted for the supervised learning of ground-state properties of quantum systems [9, 14–16, 38]. Here, we use it to accelerate the learning of deep quantum circuits, exploiting a pre-training performed on computationally cheaper circuits with fewer gates per qubit. Specifically, we compare the learning speed of a CNN trained from scratch on circuits of depth $P=8$, with the one of a CNN pre-trained on circuits of depth $P=7$. The results are analyzed in figure 9. We find that the pre-trained CNN needs a significantly smaller training set to reach high R^2 scores.

The extrapolation strategy aims at predicting properties of circuits including more qubits than those included in the training set. As discussed in section 2.4, to allow flexibility in the number of qubits N, we adopt the CNN with one output neuron. This, in combination with the global pooling layer, provides the network full scalability, allowing the same network parameters to be applied to different circuit sizes. The results are analyzed in figure 10. Remarkably, we find that a CNN trained on (computationally affordable) circuits with $\tilde{N}=10$ qubits accurately predicts the (single qubit) expectation values of significantly larger

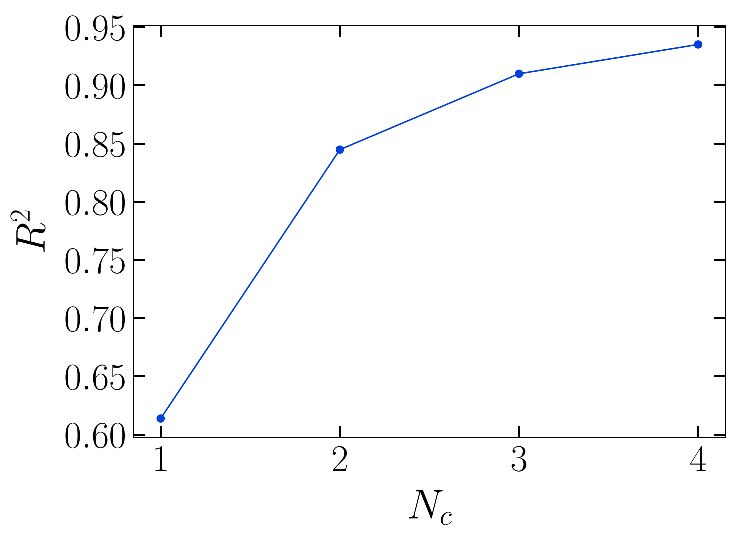


Figure 8. Coefficient of determination R^2 for rescaled single-qubit expectation values z_i as a function of the number of convolutional layers N_c. The qubit number is $N = 3$ and the circuit depth is $P = 7$. The training set includes $N_{\mathrm{train}} \simeq 10^6$ random circuits. The employed CNN is similar to the one depicted in figure 4, but with two fully-connected layers including 100 and 50 neurons. The N_c convolutional layers include $F = 32$ filters. The batch normalization layers and the output layer act as described in figure 4.

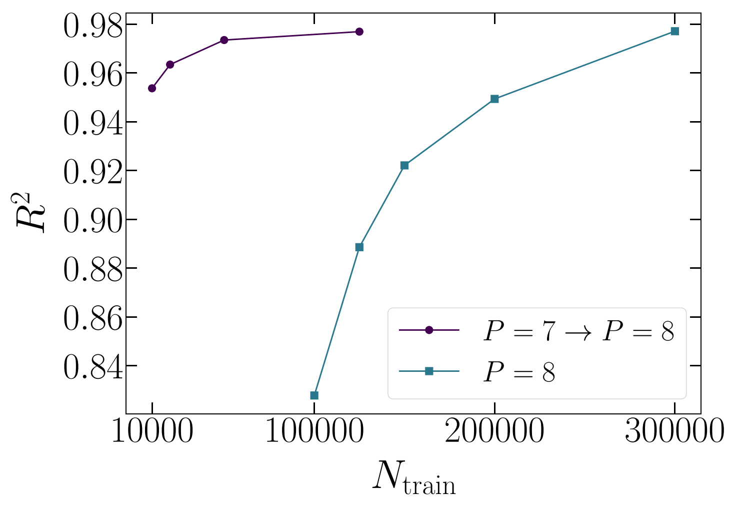


Figure 9. Coefficient of determination R^2 for the rescaled single-qubit expectation values z_1 as a function of the training set size N_{train}. The size of the test circuits is: $N = 3$ and $P = 8$. (Blue) squares correspond to training from scratch on the same circuit size. The (violet) circles correspond to transfer learning from $P = 7$ to $P = 8$ (same qubit number). The pretraining on $P = 7$ is performed with $N_{\mathrm{train}} \sim 10^6$ circuits. The CNN is as described in figure 4.

circuits, i.e. featuring $N = 20$ qubits. Instead, when the training is performed on smaller circuits ($\tilde{N} \simeq 5$), the R^2 score rapidly drops as the test-circuit size increases. This suggests that a minimum training circuit-size is needed to allow the CNN learning how to perform accurate extrapolations. It is worth stressing that the trained CNN can easily estimate the circuit output for qubit numbers larger than those considered here, but we do not have any benchmark data to quantify the prediction accuracy. Still, the almost constant accuracy shown in figure 10 suggests that such predictions would be reliable also for much larger qubit numbers. An almost constant extrapolation accuracy (up to $N = 24$) is obtained also for circuits with continuous output; see the appendix.

3.3. Real quantum computers and noisy simulators

Supervised learning with scalable CNNs is being discussed as a potentially useful benchmark for quantum computers. Therefore, it is interesting to compare the predictions provided by trained CNNs with those of actual physical devices. For this purpose, we execute random circuits on five devices freely available through IBM Quantum Experience [21]. In figure 11, the prediction accuracy of a CNN trained on $N_{\mathrm{train}} \simeq 8^7$

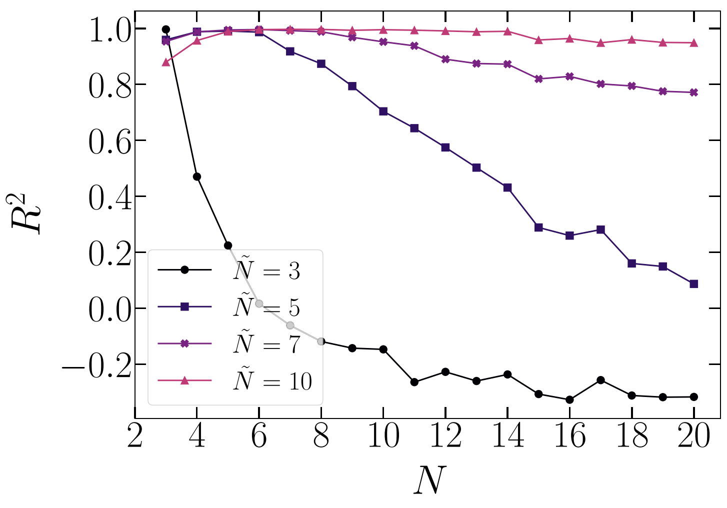


Figure 10. Coefficient of determination R^2 for rescaled single-qubit expectation values z_1 as a function of the number of qubits N in the test circuits. Both training and test circuits have depth $P = 6$. Different datasets correspond to different number of qubits $\tilde{N}$ in the training circuits (see legend). The employed CNN is as shown in figure 4 except for the last layer, which has only one neuron.

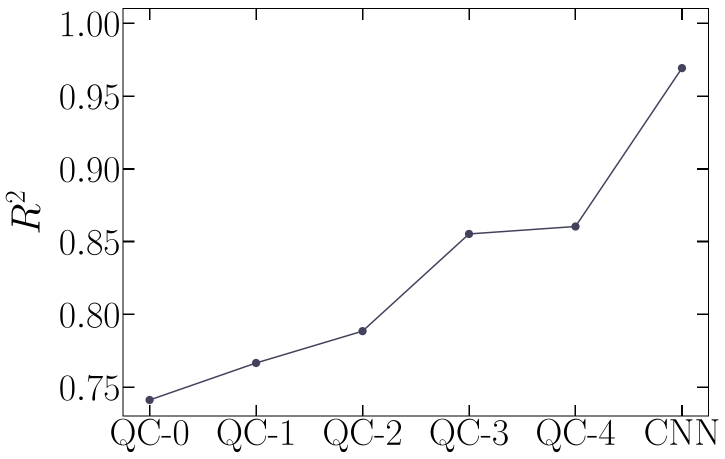


Figure 11. Coefficient of determination R^2 for single-qubit expectation values measured on five IBM quantum computers and predicted by a trained CNN. The circuit size is: $N = 5$ and $P = 10$. The five quantum computers, namely, QC-0 $\equiv$ ibmq_lima, QC-1 $\equiv$ ibmq_bogota, QC-2 $\equiv$ ibmq_belem, QC-3 $\equiv$ ibmq_quito, and QC-4 $\equiv$ ibmq_manila, are ordered for increasing R^2 score. For each quantum circuit, the expectation values are estimated using $N_{\mathrm{measure}} = 2048$ measurements. The training set size is $N_{\mathrm{train}} \simeq 8^7$. The CNN is as described in figure 4.

classically simulated circuits is compared with the corresponding scores reached by the IBM devices. The quantum circuits include $N = 5$ qubits and $P = 10$ gates per qubit. In this case, the neural network outperforms the chosen physical quantum devices. Another comparison between CNNs and quantum computers is shown in figure 12. Here, only three IBM devices are considered, and the accuracy scores R^2 are plotted as a function of the circuit depth P, for a fixed number of qubits $N = 3$. Notably, for $P > 9$, two out of the three quantum computers outperform the CNN. Notice that the latter is trained on a (fixed) training set with $N_{\mathrm{train}} \simeq 10^6$ instances. In fact, it is quite feasible to improve the CNN's accuracy, even for larger qubit numbers. As a term of comparison, we consider in figure 12 also a CNN trained on $N_{\mathrm{train}} \simeq 10^7$ circuits with $N = 10$ qubits, and used to extrapolate predictions for $N = 11$. One observes that this CNN outperforms all of the considered physical devices. We recall that, in figure 10 (see also figure 18 below), accurate extrapolations to even more challenging qubit numbers $N = 20$ are demonstrated. These findings indicate that scalable CNNs trained via supervised learning on classically simulated quantum circuits represent a potentially useful benchmark for the development of quantum devices.

One can envision the use of data produced by physical quantum devices to train CNNs. This could allow them learning how to emulate classically intractable quantum circuits. However, physical devices only allow estimating output expectation values via finite number of measurements. In the era of NISQ computers [39],

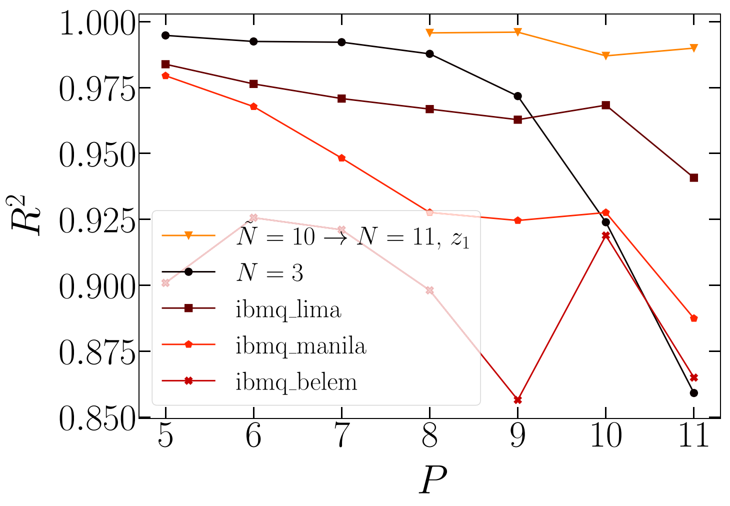


Figure 12. Coefficient of determination R^2 for (rescaled) single-qubit expectation values as a function of the circuit depth P. The five datasets correspond to three IBM quantum computers with $N = 3$ qubits, and to two CNNs, with $N = 3$ and with $N = 11$ qubits, respectively. The quantum computers are tested on 100 test circuits, estimating the three expectation values via $N_{\text{measure}} = 2048$ measurements. The CNN for $N = 3$ is trained on $N_{\text{train}} \simeq 10^6$ random circuits with the same number of qubits, and it simultaneously predicts the three (rescaled) single-qubit expectation values z_1, z_2, and z_3. For $N = 11$, the CNN has one output neuron. It is trained on $N_{\text{train}} \simeq 10^7$ circuits with $\tilde{N} = 10$ qubits, and it extrapolates to $N = 11$ performing a single prediction for z_1. The training of both CNNs starts with the optimized weights and biases for $P - 1$, except for the smallest P, where the training starts from scratch.

one should expect this number to be quite limited, leading to noisy estimates affected by significant statistical fluctuations. Therefore, it is important to analyze the impact of this noise on the supervised training of CNNs. For this, we consider as training target values the noisy estimates obtained by simulating via Qiskit finite numbers of measurements N_{measure}. In the testing phase, the CNN's predictions are compared against exact expectation values. This comparison is shown in the scatter plot of figure 13, for the case of $N_{\text{measure}} = 32$. One notices that, while the noisy estimates display large random fluctuations, the CNN's predictions accurately approximate the exact expectation value. This effect is quantitatively analyzed via the R^2 score in figure 14. Notably, the CNN reaches remarkably accuracies $R^2 \gtrsim 0.99$ for numbers of measurements as small as $N_{\text{measure}} \sim 32$, despite the fact that the estimated expectation values are instead significantly inaccurate, corresponding to $R^2 \simeq 0.9$. An analogous resilience to noise in training data was first observed in applications of CNNs to image classification tasks [40]. It was also demonstrated in the supervised learning of ground-state energies of disordered atomic quantum gases [8]. It is quite relevant to recover this property in the case of quantum computing, where noise represents a major obstacle to be overcome. Chiefly, this resilience paves the way to the use of physical quantum devices for the production of training datasets. Deep CNNs could be trained to solve classically intractable circuits, and then distributed to practitioners more easily than a physical device. This would allowing these practitioners to exploit the benefit of the quantum device even without having direct access to it.

3.4. Emulation of the BV algorithm

As discussed in section 2.3, the BV algorithm can be emulated by predicting the single-qubit expectation values $\langle Z_i \rangle$, with $i = 1, \ldots, N-1$. This means that these expectation values allow one unequivocally identifying the sought-for bit string $\mathbf{w} \in \{0,1\}^{N-1}$. Here we analyze how accurately our scalable CNNs emulate this algorithm. Notably, we challenge the CNN in the extrapolation task, i.e. we use it to emulate BV circuits with (many) more qubits than those included in the training set. Conventionally, the BV algorithm is implemented using the following gates: I, Z, H, and CX. However, it can also be realized using only gates from the set $\mathcal{S}$. An example of this alternative implementation is visualized in figure 15. Notice that a dangling T gate, acting on the Nth qubit in the last layer, needs to be inserted. However, this does not affect the relevant output expectation values. The tests we perform are limited to sought-for bit strings $\mathbf{w}$ where all bits except one, two, or three, have zero value; that is, only one, two, or three indices $i_\alpha \in [1, N-1]$ exist such that $w_{i_\alpha} = 1$, where $\alpha \in \{1,2,3\}$ spans the group of (up to three) non-zero bits. These indices are randomly selected. Specifically, a CNN is trained on circuits with $\tilde{N} = 10$ qubits and depth $P = 7, 8,$ or 9, for one, two, or three non-zero bits, respectively. It is then invoked to predict the single qubit expectation values of BV circuits with larger N. To visualize the prediction accuracy, we show in figure 16 the expectation values z_i, for $i \in [1, N]$, for a BV circuit with as many as $N = 5 \times 10^5$ qubits. Notice that also the Nth expectation value,

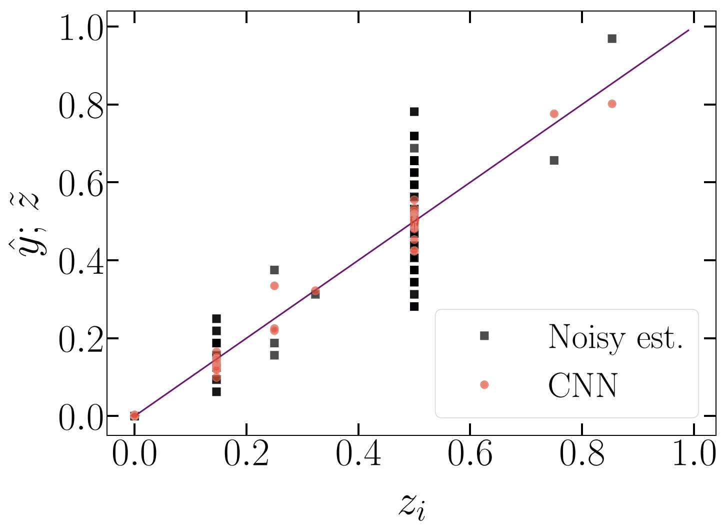


Figure 13. Predictions of (rescaled) single-qubit expectation values versus ground-truth results $y \equiv z_i$ (see equation (6)). The CNN predictions $\hat{y}$ (red circles) are compared against the noisy estimates $\tilde{z}_i$ (black squares) obtained by averaging $N_{\text{measure}} = 32$ (simulated) measurements. The circuits size is $N = 3$ and $P = 7$. The CNN is trained on the noisy estimates corresponding to $N_{\text{train}} \simeq 10^6$ random circuits.

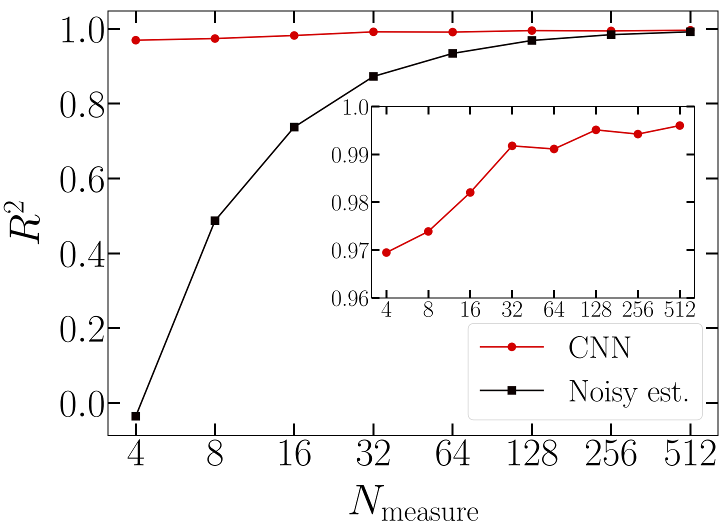


Figure 14. Coefficient of determination R^2 as a function of the number of simulated measurements N_{measure}. In the main panel, the R^2 score of the noisy estimates $\tilde{z}_i$ with respect to the ground-truth (rescaled) single-qubit expectation values z_i (black squares) is compared with the corresponding score of the CNN predictions (red circles). The circuits size is $N = 3$ and $P = 7$. The inset displays the CNN data on a narrower scale.

corresponding to the ancilla qubit, is shown. One notices that, beyond the Nth qubit, all but one, two, or three expectation values are small, meaning that the CNN is able to identify the sought-for indices i_α corresponding to $w_{i_\alpha} = 1$. It is remarkable that a CNN trained on random circuits learns to emulate a rather peculiar algorithm such as the BV circuit, even for larger qubit numbers.

3.5. Two-qubit expectation values

The scalable CNN can also be trained to predict two-qubit expectation values. We consider only the first two qubits, i.e. the CNN predicts the rescaled expectation value z_{12} defined in equation (7). Henceforth, only one neuron is included in the output layer (see discussion in section 2.4). Again, it is worth pointing out that the same CNN could predict also other two-qubit expectation values, corresponding to any pair (i, j). These predictions are obtained by performing the double exchange of row indices $(1,2) \longleftrightarrow (i,j)$ in the circuit descriptor matrix. In figure 17, the prediction accuracy is analyzed as a function of the circuit depth P. One

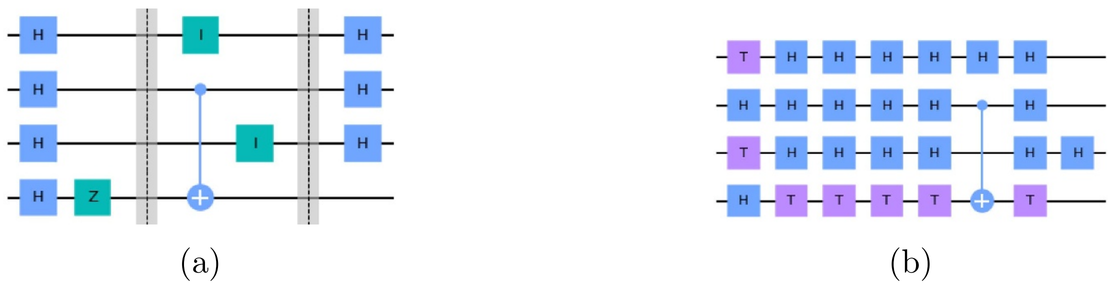


Figure 15. Representation of the Bernstein–Vazirani (BV) algorithm for a sought-for string $\mathbf{w} \in \{0,1\}^3$, corresponding to a circuit with $N = 4$ qubits. Panel (a) displays the conventional implementation using the gates I, Z, H, and CX, for the bit string $\mathbf{w} = 010$. Panel (b) displays an alternative implementation using only gates from the set $\mathcal{S}$. The dangling T-gate on the last qubit is required for shape consistency, and it does not affect the relevant output probabilities.

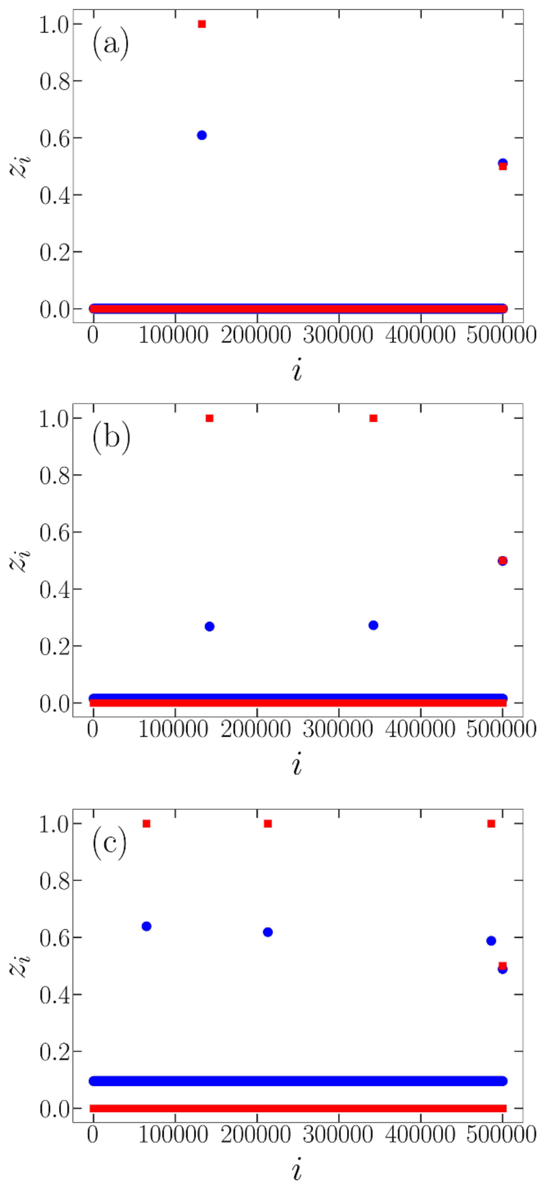


Figure 16. Rescaled output expectation values z_i (see equation (6)) as a function of the qubit index $i = 1, \ldots, N$, for a BV algorithm with $N = 5 \times 10^5$ qubits. The CNN predictions (blue circles) are compared to the expected values (red squares). The three panels correspond to sought-for bit strings $\mathbf{w}$ with one non-zero bit $i_1 = 132\,121$ (panel (a)), with two non-zero bits $(i_1, i_2) = (341\,924, 141\,725)$ (panel (b)), and with three non-zero bits $(i_1, i_2, i_3) = (64\,793, 212\,973, 485\,883)$ (panel (c)). The CNNs are trained on $N_{\text{train}} \simeq 10^6$ (panel (a)) and $N_{\text{train}} \simeq 10^7$ (panels (b) and (c)) random circuits with $N = 10$ qubits.

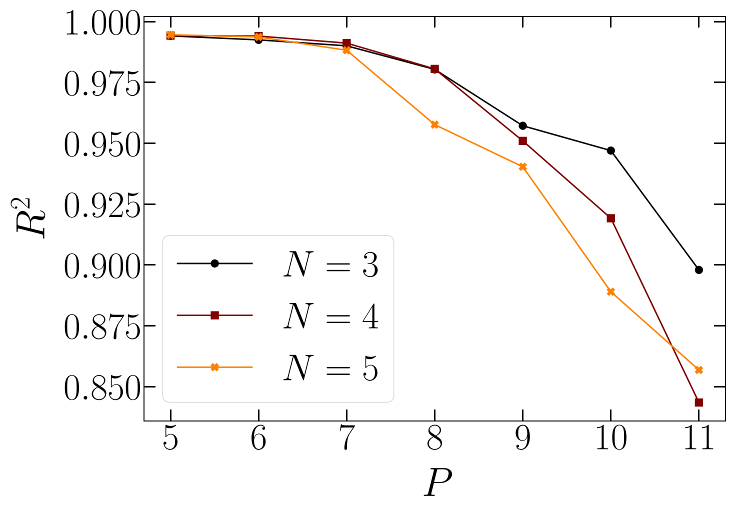


Figure 17. Coefficient of determination R^2 for the rescaled two-qubit expectation value z_{12} (see equation (7)) as a function of the circuit depth P. The three datasets correspond to different qubit numbers N. The training set size is $N_{\text{train}} \simeq 10^6$. For $P = 5$ the CNN is trained from scratch, while for $P > 5$ the weights and biases are initialized at the optimized values for $P - 1$. This allows a significant reduction in computation time.

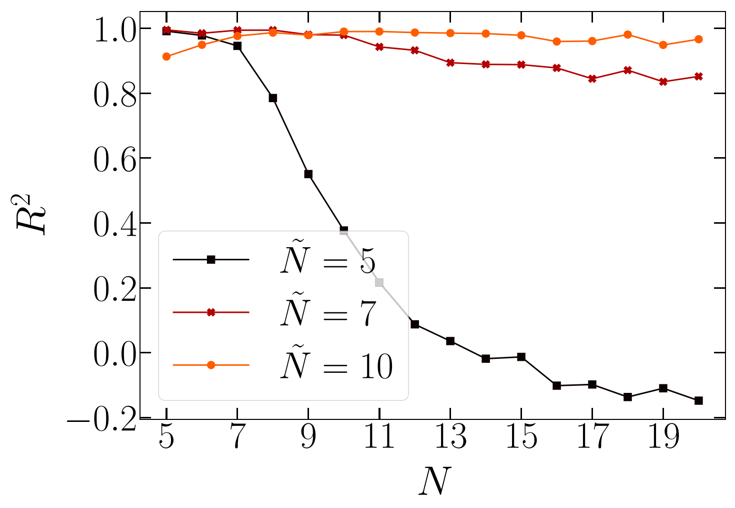


Figure 18. Coefficient of determination R^2 for the rescaled two-qubit expectation value z_{12} as a function of the number of qubits N of the test circuits. The three datasets correspond to different qubit numbers $\tilde{N}$ of the training circuits. The training set includes $N_{\text{train}} \simeq 10^6$ random circuits. Both training and test circuits have depth $P = 6$.

observes remarkably high scores $R^2 \simeq 1$ for small and intermediate circuits depths, and a moderate accuracy degradation for deeper circuits. As already shown for single-qubit expectation values (see section 3.1), we stress that also in this case the prediction accuracy can be further improved by increasing the training set size or deepening the CNN (data not shown).

The scalable CNN is also tested in the extrapolation task, i.e. in predicting the two-qubit expectation value for circuits larger than those included in the training set. The accuracy score R^2 is plotted in figure 18. The three datasets corresponds to different training qubit numbers $\tilde{N} = 5$, 7, and 10, and the extrapolation is extended up to $N = 20$. Notably, if the training qubit number is sufficiently large, the predictions remain remarkably accurate for significantly larger circuits.

4. Conclusions

We explored the supervised learning of random quantum circuits. These were built using either discrete universal sets of gates, or using continuous random rotations plus entangling gates (see the appendix). Deep

CNNs have been trained to map properly designed encodings of the circuits' quantum gates to the corresponding single-qubit and two-qubit output expectation values. After training on sufficiently large datasets of classically simulated circuits, the CNN provided remarkably accurate predictions, even superior to those provided by small quantum computers available for free from the IBM quantum program. Notably, we implemented scalable CNNs. This allowed them predicting the properties of circuit sizes larger than those included in the training set, displaying consistent accuracy for all sizes for which we could produce testing results. Notice that these CNNs can easily provide predictions for even larger circuits, but, with the computational resources and algorithms currently available to us, it is not possible to obtain benchmark values to test their accuracy. The considered expectation values represent an admittedly limited description of the circuits' output. However, we demonstrated that circuits with only one or two possible output bit strings can be emulated using these targeted expectation values. Notably, the BV algorithm belongs to this category, and we use it as a relevant benchmark for the CNN's predictions. In fact, we verified that the scalable CNN can emulate it even for qubit numbers much larger than those included in the training set. The supervised learning turned out to be remarkably robust against random errors in the training set. Specifically, the CNN provided accurate predictions for expectation values even when trained on noisy averages performed over few simulated measurements. This finding supports the idea that CNNs could be trained on data produced by NISQ computers [39]. This way, the CNN could be trained to replicate the behavior of a quantum computer, perhaps tackling a classically intractable computational problem, and then distributed to many users, allowing them to take advantage of the quantum device even without having direct access to it. To produce the results reported in this article, several training and test datasets have been produced, and various CNNs have been trained. To facilitate future comparative studies, but avoiding cluttering the repository, we provide the datasets and the code used for one emblematic test, namely, the one analyzed in figure 10, through [41].

Classical simulations of quantum algorithms play a pivotal role in the development of quantum computing devices. On the one hand, they provide benchmark data for validation. On the other hand, they represent an indispensable term of comparison to justify claims of quantum speed-up in the solution of computational problems [3]. For adiabatic quantum computers, quantum Monte Carlo algorithms have emerged as the standard benchmark [42–45]. This stems from their ability of simulating the tunneling dynamics of quantum annealers based on sign-problem free Hamiltonians [46–50]. Simulating universal gate-based quantum computers is more challenging. Direct simulation methods, such as those based on tensor networks [51], are being continuously improved [52–56], but they anyway suffer from an exponentially-scaling computational cost for strongly entangling circuits. Supervised machine-learning algorithms were recently proven to be able of solving computational tasks that are intractable for algorithms that did not learn from data [13]. In this article, we investigated their efficiency in simulating a limited description of quantum circuits' output. At the current stage, it is not clear whether scalable supervised learning can emulate circuits that are otherwise absolutely intractable for any other classical algorithm. To support such a strong statement, approximate and/or special purpose methods should be proven unfeasible for the circuit types considered in this Article. Our findings raise various relevant questions related to the computational complexity of supervised learning of quantum circuits. In particular, the required amount of data, depending on the network structure, the training protocol, and the included gates, should be further analyzed to disclose or rule out a possible exponential scaling of the computational cost. While some relevant results have been reported here, an exhaustive analysis necessarily requires further investigations. Chiefly, these investigation should address more complete descriptions of the circuit output, considering, e.g. multi-qubit expectation values, and they should shed further light on what might make a quantum circuit intractable for scalable supervised learning. It is worth mentioning that the combination of classical machine learning and quantum computers has already been discussed in various contexts [25, 57–59]. For example, in [60], generative neural networks trained via unsupervised learning were used to accelerate the convergence of expectation-value estimation. One can envision the use of stochastic generative neural network to predict a more complete description of the circuits' outputs such as, e.g. the classical shadow [13, 61]. We leave this endeavor to future investigations.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://github.com/simonecantori/Supervised-learning-of-quantum-circuits.git.

Acknowledgments

This work was supported by the Italian Ministry of University and Research under the PRIN2017 Project CEnTraL 20172H2SC4, and by the European Union Horizon 2020 Programme for Research and Innovation through the Project No. 862644 (FET Open QUARTET). S P acknowledges PRACE for awarding access to the Fenix Infrastructure resources at Cineca, which are partially funded by the European Union's Horizon 2020 research and innovation program through the ICEI project under the Grant Agreement No. 800858. S Cantori acknowledges partial support from the B-GREEN project of the italian MiSE—Bando 2018 Industria Sostenibile. This work was also supported by the PNRR MUR Project PE0000023-NQSTI. The authors 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.

Appendix. Circuits of random rotations plus two-qubit gates

The circuits described in section 2 are built using the universal sets $\mathcal{S}$ or $\mathcal{S}^*$. Their single-qubit and two-qubit output expectation values are discrete. Furthermore, the number of entangling (two-qubit) gates does not scale with the qubit number N. To further support the results on prediction accuracy and on scalability reported in section 3, it is worth considering also an alternative circuit design featuring complementary properties. In this appendix, we consider circuits made of two layers including random rotations along the y-axis of the Bloch sphere for each qubit, separated by one layer featuring CX gates, connecting every pair of adjacent qubits. This circuit design is visualized in figure A1. The rotations are represented by the R_y gate, defined by the matrix:

$$R_y(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}. \tag{A.1}$$

For each qubit $i = 1, \ldots, N$ and rotation layer $r = 1, 2$, a rotation angle $\theta_{i,r} \in [0, 2\pi]$ is randomly sampled from a continuous uniform distribution. With this design, the set of random angles $\theta_{1,1}, \theta_{2,1}, \ldots, \theta_{N,1}$ plus $\theta_{1,2}, \theta_{2,2}, \ldots, \theta_{N,2}$ suffices to provide a univocal representation of each circuit. Henceforth, we use a two channel encoding, whereby the corresponding values of $\sin\theta_{i,r}$ and $\cos\theta_{i,r}$ are stored in the first and in the second channel, respectively. The resulting $N \times 2 \times 2$ tensor constitutes the CNN input. The chosen target value is $y \equiv \langle Z_1 \rangle$. This can take continuous values in the range $y \in [-1, 1]$. One might suspect that this represents a more challenging target for supervised learning, as compared to the finite discrete values mentioned above. It is also worth emphasizing that, with this design, the number of two-qubit gates scales with N. In fact, this arrangement of CX gates, but combined with a H gate, is also used to create the Greenberger–Horne–Zeilinger states [62], which feature maximal entanglement according to various measures [63]. A similar design has also been adopted to tackle combinatorial optimization problems via the variational quantum eigensolver [64]. The CNN we adopt here is similar to the one shown in figure 4, but with the removal of one of the convolutional layers with $F = 128$ and one with $F = 256$, resulting in a total of $N_c = 8$ convolutional layers. The four dense layers contain 512, 128, 64 and 1 neurons, respectively. The activation function in the last layer is the identity function. The loss function optimized during training is the mean squared error:

$$\mathcal{L} = \frac{1}{N_{\text{train}}} \sum_{k=1}^{N_{\text{train}}} (y^{(k)} - \hat{y}^{(k)})^2. \tag{A.2}$$

The prediction accuracy in the extrapolation task is analyzed in figure A2. We find that the CNN trained on circuits including only $\tilde{N} = 10$ qubits accurately extrapolates single-qubit expectation values of circuits with up to $N = 20$ qubits. Figure A3 displays the scatter plots of predicted versus ground-truth expectation values for even larger quantum circuits with $N = 24$ (beyond the $N = 20$ case), showing again remarkable accuracy. Only 100 test circuits are considered for $N = 24$, since with the Qiskit library their simulations require in total approximately 48 hours on a Intel(R) Xeon(R) Gold 6154 CPU. These findings corroborate the results shown in figures 10 and 18. They indicate that scalable neural networks allow one to accurately predict output expectation values of circuits significantly larger than those used for training.

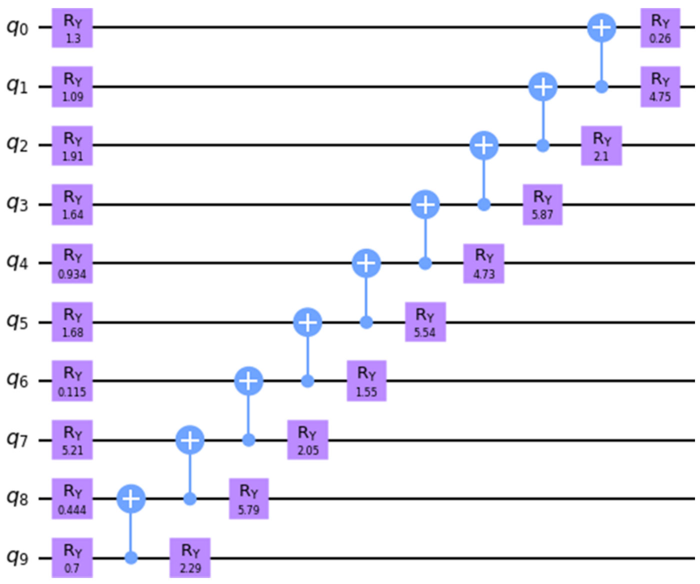


Figure A1. Representation of the continuous quantum circuits for the case $N = 10$. The rotation angles for the R_y gates are sampled from a continuous uniform distribution.

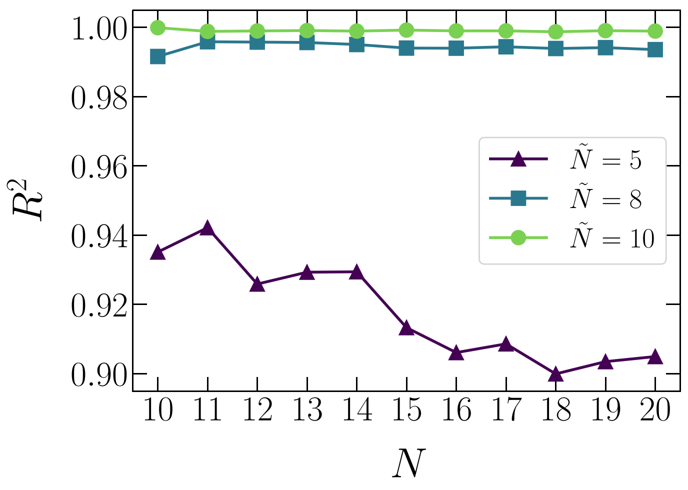


Figure A2. Coefficient of determination R^2 for the single-qubit expectation values $\langle Z_1 \rangle$ as a function of the number of qubits N in the test circuits. Different datasets correspond to different number of qubits $\tilde{N}$ in the training circuits (see legend). Here, the circuits are built with the continuous gate set.

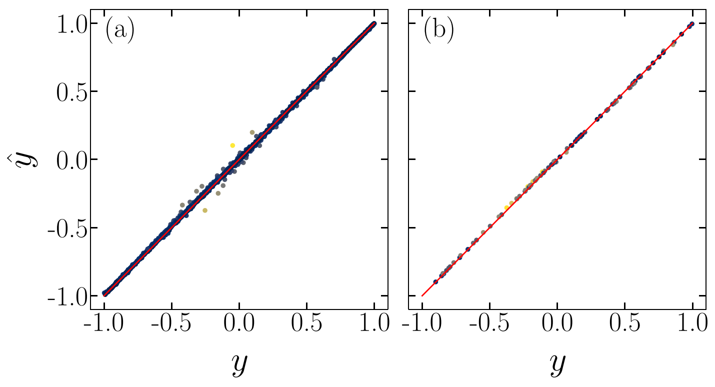


Figure A3. Single-qubit expectation values $\hat{y}$ predicted by the CNN versus the ground-truth results $y \equiv \langle Z_1 \rangle$. The CNN is trained on (continuous) quantum circuits with $\tilde{N} = 10$ qubits and tested on circuits with different number of qubits N: $N = 20$ with 2000 samples (a); $N = 24$ with 100 samples ($R^2 \simeq 0.9997$) (b). The color scale (blue to yellow) represents the absolute discrepancy $|\hat{y} - y|$. The (red) line represents the bisector $\hat{y} = y$.

ORCID iDs

Simone Cantori https://orcid.org/0000-0002-6071-9987
David Vitali https://orcid.org/0000-0002-1409-7136
Sebastiano Pilati https://orcid.org/0000-0002-4845-6299

References

[1] Steane A 1998 Quantum computing *Rep. Prog. Phys.* **61** 117–73
[2] Aharonov D 1999 Quantum computation *Annual Reviews of Computational Physics VI* (Singapore: World Scientific) pp 259–346
[3] Rønnow T F, Wang Z, Job J, Boixo S, Isakov S V, Wecker D, Martinis J M, Lidar D A and Troyer M 2014 Defining and detecting quantum speedup *Science* **345** 420–4
[4] Faber F A, Hutchison L, Huang B, Gilmer J, Schoenholz S S, Dahl G E, Vinyals O, Kearnes S, Riley P F and Von Lilienfeld O A 2017 Prediction errors of molecular machine learning models lower than hybrid DFT error *J. Chem. Theory Comput.* **13** 5255–64
[5] Hansen K, Biegler F, Ramakrishnan R, Pronobis W, von Lilienfeld O A, Müller K-R and Tkatchenko A 2015 Machine learning predictions of molecular properties: accurate many-body potentials and nonlocality in chemical space *J. Phys. Chem. Lett.* **6** 2326–31 PMID: 26113956
[6] Schütt K T, Glawe H, Brockherde F, Sanna A, Müller K R and Gross E K U 2014 How to represent crystal structures for machine learning: towards fast prediction of electronic properties *Phys. Rev.* B **89** 205118
[7] Ryczko K, Mills K, Luchak I, Homenick C and Tamblyn I 2018 Convolutional neural networks for atomistic systems *Comput. Mater. Sci.* **149** 134–42
[8] Pilati S and Pieri P 2019 Supervised machine learning of ultracold atoms with speckle disorder *Sci. Rep.* **9** 5613
[9] Mujal P, Miguel À, Polls A, Juliá-Díaz B and Pilati S 2021 Supervised learning of few dirty bosons with variable particle number *SciPost Phys.* **10** 73
[10] Ballester P J and Mitchell J B 2010 A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking *Bioinformatics* **26** 1169–75
[11] Khamis M A, Gomaa W and Ahmed W F 2015 Machine learning in computational docking *Artif. Intell. Med.* **63** 135–52
[12] Huang H-Y, Broughton M, Mohseni M, Babbush R, Boixo S, Neven H and McClean J R 2021 Power of data in quantum machine learning *Nat. Commun.* **12** 1–9
[13] Huang H-Y, Kueng R, Torlai G, Albert V V and Preskill J 2021 Provably efficient machine learning for quantum many-body problems (arXiv:2106.12627)
[14] Mills K, Ryczko K, Luchak I, Domurad A, Beeler C and Tamblyn I 2019 Extensive deep neural networks for transferring small scale learning to large scale systems *Chem. Sci.* **10** 4129–40
[15] Saraceni N, Cantori S and Pilati S 2020 Scalable neural networks for the efficient learning of disordered quantum systems *Phys. Rev.* E **102** 033301
[16] Jung H, Stocker S, Kunkel C, Oberhofer H, Han B, Reuter K and Margraf J T 2020 Size-extensive molecular machine learning with global representations *ChemSystemsChem* **2** e1900052
[17] Arute F *et al* 2019 Quantum supremacy using a programmable superconducting processor *Nature* **574** 505–10
[18] Wu Y *et al* 2021 Strong quantum computational advantage using a superconducting quantum processor *Phys. Rev. Lett.* **127** 180501
[19] Bernstein E and Vazirani U 1997 Quantum complexity theory *SIAM J. Comput.* **26** 1411–73
[20] Treinish M *etal* 2022 Qiskit/qiskit: qiskit 0.36.2 (https://doi.org/10.5281/zenodo.6560959)
[21] IBM Quantum 2021 (available at: https://quantum-computing.ibm.com/)
[22] Boykin P, Mor T, Pulver M, Roychowdhury V and Vatan F 2000 A new universal and fault-tolerant quantum basis *Inf. Process. Lett.* **75** 101–7
[23] Toffalori C, Corradini F, Leonesi S and Mancini S 2005 *Teoria Della Computabilità e Della Complessità* (*Collana di Istruzione Scientifica)* (New York: McGraw-Hill)
[24] Zlokapa A and Gheorghiu A 2020 A deep learning model for noise prediction on near-term quantum devices (arXiv:2005.10811)
[25] Zhang S-X, Hsieh C-Y, Zhang S and Yao H 2021 Neural predictor based quantum architecture search *Mach. Learn.: Sci. Technol.* **2** 045027
[26] Deutsch D and Jozsa R 1992 Rapid solution of problems by quantum computation *Proc. R. Soc.* A **439** 553–8
[27] Grover L K 1996 A fast quantum mechanical algorithm for database search *Proc. Annual ACM Symp. Theory Computing (STOC '96)* (New York: Association for Computing Machinery) pp 212–9
[28] Nielsen M A and Chuang I L 2010 *Quantum Computation and Quantum Information* 10th Anniversary edn (Cambridge: Cambridge University Press)
[29] Costa E, Scriva G, Fazio R and Pilati S 2022 Deep-learning density functionals for gradient descent optimization *Phys. Rev.* E **106** 045309
[30] Blania A, Herbig S, Dechent F, van Nieuwenburg E and Marquardt F 2022 Deep learning of spatial densities in inhomogeneous correlated quantum systems (arXiv:2211.09050)
[31] Kingma D P and Ba J 2014 Adam: a method for stochastic optimization (arXiv:1412.6980)
[32] Misra D 2019 Mish: a self regularized non-monotonic neural activation function *CoRR* (arXiv:1908.08681)
[33] Hestness J, Narang S, Ardalani N, Diamos G, Jun H, Kianinejad H, Patwary M M A, Yang Y and Zhou Y 2017 Deep learning scaling is predictable, empirically (arXiv:1712.00409)
[34] Simonyan K and Zisserman A 2014 Very deep convolutional networks for large-scale image recognition (arXiv:1409.1556)
[35] Liu Z, Mao H, Wu C-Y, Feichtenhofer C, Darrell T and Xie S 2022 A convnet for the 2020s (arXiv:2201.03545)
[36] Bravyi S and Gosset D 2016 Improved classical simulation of quantum circuits dominated by clifford gates *Phys. Rev. Lett.* **116** 250501
[37] Caruana R 1997 Multitask learning *Mach. Learn.* **28** 41–75
[38] Scherbela M, Reisenhofer R, Gerard L, Marquetand P and Grohs P 2022 Solving the electronic Schrödinger equation for multiple nuclear geometries with weight-sharing deep neural networks *Nat. Comput. Sci.* **2** 331–41
[39] Preskill J 2018 Quantum computing in the NISQ era and beyond *Quantum* **2** 79
[40] Rolnick D, Veit A, Belongie S and Shavit N 2017 Deep learning is robust to massive label noise (arXiv:1705.10694)

[41] Cantori S, Vitali D and Pilati S 2022 Supervised learning of quantum circuits (available at: https://github.com/simonecantori/Supervised-learning-of-quantum-circuits)
[42] Santoro G E, Martoňák R, Tosatti E and Car R 2002 Theory of quantum annealing of an Ising spin glass *Science* **295** 2427–30
[43] Boixo S, Albash T, Spedalieri F M, Chancellor N and Lidar D A 2013 Experimental signature of programmable quantum annealing *Nat. Commun.* **4** 1–8
[44] Boixo S, Rønnow T F, Isakov S V, Wang Z, Wecker D, Lidar D A, Martinis J M and Troyer M 2014 Evidence for quantum annealing with more than one hundred qubits *Nat. Phys.* **10** 218–24
[45] Heim B, Rønnow T F, Isakov S V and Troyer M 2015 Quantum versus classical annealing of Ising spin glasses *Science* **348** 215–7
[46] Isakov S V, Mazzola G, Smelyanskiy V N, Jiang Z, Boixo S, Neven H and Troyer M 2016 Understanding quantum tunneling through quantum Monte Carlo simulations *Phys. Rev. Lett.* **117** 180402
[47] Mazzola G, Smelyanskiy V N and Troyer M 2017 Quantum Monte Carlo tunneling from quantum chemistry to quantum annealing *Phys. Rev.* B **96** 134305
[48] Brady L T and van Dam W 2016 Quantum Monte Carlo simulations of tunneling in quantum adiabatic optimization *Phys. Rev.* A **93** 032304
[49] Inack E M, Giudici G, Parolini T, Santoro G and Pilati S 2018 Understanding quantum tunneling using diffusion Monte Carlo simulations *Phys. Rev.* A **97** 032307
[50] Parolini T, Inack E M, Giudici G and Pilati S 2019 Tunneling in projective quantum Monte Carlo simulations with guiding wave functions *Phys. Rev.* B **100** 214303
[51] Felser T, Notarnicola S and Montangero S 2021 Efficient tensor network ansatz for high-dimensional quantum many-body problems *Phys. Rev. Lett.* **126** 170603
[52] Jones T, Brown A, Bush I and Benjamin S C 2019 QuEST and high performance simulation of quantum computers *Sci. Rep.* **9** 1–11
[53] Guerreschi G G, Hogaboam J, Baruffa F and Sawaya N P 2020 Intel quantum simulator: a cloud-ready high-performance simulator of quantum circuits *Quantum Sci. Technol.* **5** 034007
[54] Steiger D S, Häner T and Troyer M 2018 ProjectQ: an open source software framework for quantum computing *Quantum* **2** 49
[55] Villalonga B, Lyakh D, Boixo S, Neven H, Humble T S, Biswas R, Rieffel E G, Ho A and Mandrà S 2020 Establishing the quantum supremacy frontier with a 281 pflop/s simulation *Quantum Sci. Technol.* **5** 034003
[56] Pan F and Zhang P 2022 Simulation of quantum circuits using the big-batch tensor network method *Phys. Rev. Lett.* **128** 030501
[57] Schuld M, Sinayskiy I and Petruccione F 2015 An introduction to quantum machine learning *Contemp. Phys.* **56** 172–85
[58] Seif A, Landsman K A, Linke N M, Figgatt C, Monroe C and Hafezi M 2018 Machine learning assisted readout of trapped-ion qubits *J. Phys.* B **51** 174006
[59] Torlai G *et al* 2019 Integrating neural networks with a quantum simulator for state reconstruction *Phys. Rev. Lett.* **123** 230504
[60] Torlai G, Mazzola G, Carleo G and Mezzacapo A 2020 Precise measurement of quantum observables with neural-network estimators *Phys. Rev. Res.* **2** 022060
[61] Huang H-Y, Kueng R and Preskill J 2020 Predicting many properties of a quantum system from very few measurements *Nat. Phys.* **16** 1050–7
[62] Greenberger D M, Horne M A and Zeilinger A 2007 Going beyond bell's theorem (arXiv:0712.0921)
[63] Enríquez M, Wintrowicz I and Życzkowski K 2016 Maximally entangled multipartite states: a brief survey *J. Phys.: Conf. Ser.* **698** 012003
[64] Barkoutsos P K, Nannicini G, Robert A, Tavernelli I and Woerner S 2020 Improving variational quantum optimization using CVaR *Quantum* **4** 256