


Neural tensor contractions and the expressive power of deep neural quantum states

Or Sharir^{*} and Amnon Shashua[†]

The Hebrew University of Jerusalem, Jerusalem 9190401, Israel

Giuseppe Carleo[‡]

Institute of Physics, École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland

 (Received 1 July 2021; revised 25 October 2022; accepted 2 November 2022; published 22 November 2022)

We establish a direct connection between general tensor networks and deep feed-forward artificial neural networks. The core of our results is the construction of neural-network layers that efficiently perform tensor contractions and that use commonly adopted nonlinear activation functions. The resulting deep networks feature a number of edges that closely match the contraction complexity of the tensor networks to be approximated. In the context of many-body quantum states, this result establishes that neural-network states have strictly the same or higher expressive power than practically usable variational tensor networks. As an example, we show that all matrix product states can be efficiently written as neural-network states with a number of edges polynomial in the bond dimension and depth that is logarithmic in the system size. The opposite instead does not hold true, and our results imply that there exist quantum states that are not efficiently expressible in terms of matrix product states or projected entangled pair states but that are instead efficiently expressible with neural network states.

DOI: [10.1103/PhysRevB.106.205136](https://doi.org/10.1103/PhysRevB.106.205136)

I. INTRODUCTION

Many fundamental problems in science can be formulated in terms of finding an explicit representation of complex high-dimensional functions, ranging from time-dependent vector fields to normalized probability densities. In recent years, machine learning (ML) techniques based on deep learning [1] have become the leading numerical approach for approximating high-dimensional functions found in industrial applications. Due to this success, ML methods have also been recognized as a prime computational tool to attack functional approximation problems in physics [2].

In quantum physics, one of the main theoretical challenges in describing interacting, many-body systems stems from the complexity of finding explicit representations of many-particle quantum wave functions. Tensor networks states (TNS) are a well-established general-purpose ansatz for representing such functions. TNS are intrinsically rooted in the notion of locality in quantum systems and constitute both a key theoretical language to analyze many-body phenomena as well as a powerful numerical tool for simulations [3–7]. Recently, neural-network-based representation of quantum

states, dubbed neural quantum states (NQS), have been introduced [8] and subsequently used in a variety of variational applications. A key theoretical question is how these two alternatives relate to each other and whether some families of quantum states are better described in terms of one of them.

Several theoretical properties of NQS have been established to date. General representation theorems for neural networks [9] guarantee that sufficiently large NQS can describe arbitrary quantum states. Moreover, exact representations of many-body ground states of local Hamiltonians can be analytically found in terms of deep Boltzmann machines [10]. Both representation results, however, do not bound the size of the corresponding NQS networks that, in the worst case, can be exponentially large in the number of physical degrees of freedom [11]. Despite the worst-case exponential bound on NQS, examples of physically relevant quantum states that can be efficiently represented are numerous. These encompass both analytical and numerical results. On the analytical side, for example exact and compact NQS representations of several correlated topological phases of matter are known [12–15]. On the numerical side, suitable learning algorithms have shown competitive results to find *ab initio* approximate description of many physical systems of interest in physics [14, 16–20] and chemistry [21–23].

As mentioned, a well-established paradigm for describing many-body quantum states are TNS. While generic TNS are widely believed to be general enough to compactly describe most physical quantum states, only a restricted subset of them are amenable for numerical calculations. A determining factor in the applicability of TNS as variational quantum states is played by how complex it is to use these representations to compute physical quantities, and it is in turn related to the complexity of contracting TNS. TNS that can be effi-

^{*}Current address: California Institute of Technology, Pasadena, CA, 91125; or.sharir@cs.huji.ac.il

[†]shashua@cs.huji.ac.il

[‡]giuseppe.carleo@epfl.ch

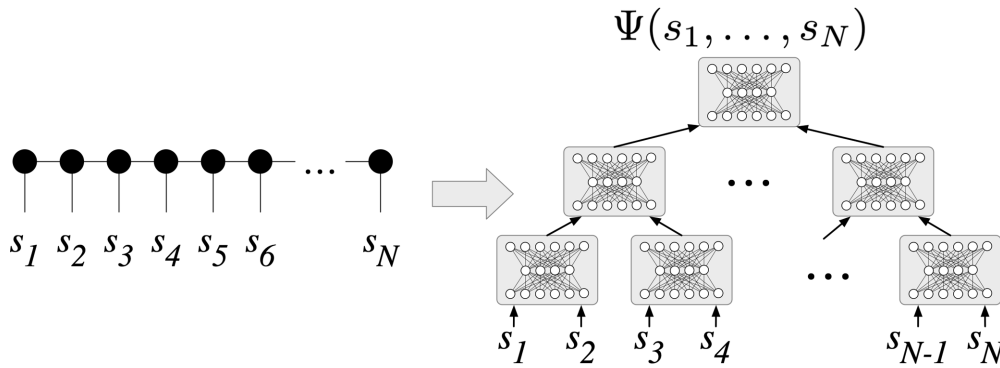


FIG. 1. We demonstrate a mapping from any tensor network with an efficient contraction algorithm to a compact neural network. Here we illustrate our coarse-grained construction for an ϵ approximation of a matrix product state over N sites, each of d degrees of freedom, and bond-dimension χ . The resulting network is of depth $\tilde{O}(\ln N + \sqrt{1/\epsilon})$ and has $\tilde{O}(N(d + \chi)\chi^2 + \sqrt{1/\epsilon})$ edges

ciently contracted most notably encompass matrix product states (MPS) [3], a very powerful representation of low-entangled states in one-dimension. Higher-dimensional TNS are in general to be contracted only approximately, and rigorous complexity results have been established. For example, computing expectation values of physical quantities over planar tensor networks in two dimension, the projected entangled pair states (PEPS) [24], is non polynomial problem that is known to belong to the $\#\text{P}$ complexity class [25,26].

Given the distinctive features of NQS and TNS, several works have studied possible connections between the two representations. For example, the volume-law entanglement capacity of neural networks has been established in several works [27–29]. Also, mappings between the two classes of states have been realized, including between general fully connected NQS and MPS with exponentially large bond dimension [28]. An approach mapping MPS onto nonstandard neural networks has also been introduced [30]. Despite the important theoretical progress, a direct mapping between generic, efficiently contractible TNS and standard NQS has not been established to date. This situation for example leaves open the possibility that TNS can offer a general representational advantage over NQS representations [31,32], and that there might exist compact, contractible TNS that cannot be expressed by means of compact NQS.

In this work, we establish a direct mapping between TNS in arbitrary dimensions and NQS, as illustrated in Fig. 1. By directly constructing neural-network layers that perform tensor contractions, we show that efficiently contractible TNS can be constructed in terms of polynomially sized neural networks. Our result, in conjunction with previously established results on the entanglement capacity of NQS, then demonstrates that NQS constitute a very flexible classical representation of quantum states and that TNS commonly used in variational applications are strictly a subset of NQS.

II. PRELIMINARIES

We consider in the following a pure quantum system, constituted by N discrete degrees of freedom $\mathbf{s} \equiv (s_1, \dots, s_N)$ (e.g., spins, occupation numbers, etc.) such that the wavefunction (WF) amplitudes $\langle s | \Psi \rangle = \Psi(\mathbf{s})$ fully specify its state.

Following the approach introduced in Ref. [8], we can represent $\log[\Psi(\mathbf{s})]$ as $g_1(\mathbf{s}) + i \cdot g_2(\mathbf{s})$, where g_1 and g_2 are two outputs of a feed-forward neural network, parametrized by a possibly large number of network connections. Given an arbitrary set of quantum numbers, \mathbf{s} , the output-value computation of the corresponding NQS can generally [33] be described as two roots of a directed acyclic graph (V, E) , where the value of each node $v \in V$ is recursively defined by $v(\mathbf{s}) = \sigma(b_v + \sum_{(u,v) \in E} W_{u,v} u(\mathbf{s}))$, where $\{W_{u,v} \in \mathbb{R}\}_{(u,v) \in E}$ and $\{b_v \in \mathbb{R}\}_{v \in V}$ are the parameters of the network, and $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ is some nonlinear function known as the *activation function*, e.g., $\text{ReLU}(x) = \max(x, 0)$ or $\text{softplus}(x) = \log[\exp(x) + 1]$ [34,35]. The root nodes of the network can optionally use the identity instead of a nonlinear activation function. The depth of a neural network is defined as the maximal distance between an input node and the roots.

Alternatively, a state $\Psi(\mathbf{s})$ can be viewed as a complex tensor $\mathcal{A}_{s_1, \dots, s_N}$ that is represented following some tensor factorization scheme. Most forms of tensor factorizations are conveniently described graphically via tensor networks (TN), undirected graphs whose nodes are tensors and edges specify contractions between connected tensors. See Appendix A for a brief introduction. In the next section we will present our main results on the efficiency of approximating TN by NN. To properly discuss the complexity of computing a TN, we have to be specific on how a given TN is computed. First, a contraction order must be selected, i.e., the order by which intermediate tensors are computed. The complexity of contracting a TN exactly is dependent on that order. While finding the optimal contraction order for an arbitrary TN is known to be NP-complete, for many common TN forms, e.g., matrix product states, efficient algorithms exist. Two such contraction schemes are the sequential and parallel contractions that are depicted in Fig. 2. Second, we must precisely describe the computational circuit of contraction to be able to characterize some structural properties, e.g., depth and number of neurons, of the NN approximating it. Given a contraction order, the value of $\Psi(\mathbf{s})$ can alternatively be described in the form of an *arithmetic circuit* (AC) [36], i.e., a computational graph comprising product and weighted-sum nodes. Specifically, the value for a product node $v \in P$ is given by $v(\mathbf{s}) = \prod_{(u,v) \in E} u(\mathbf{s})$, and for a weighted-sum node $v \in S$ is

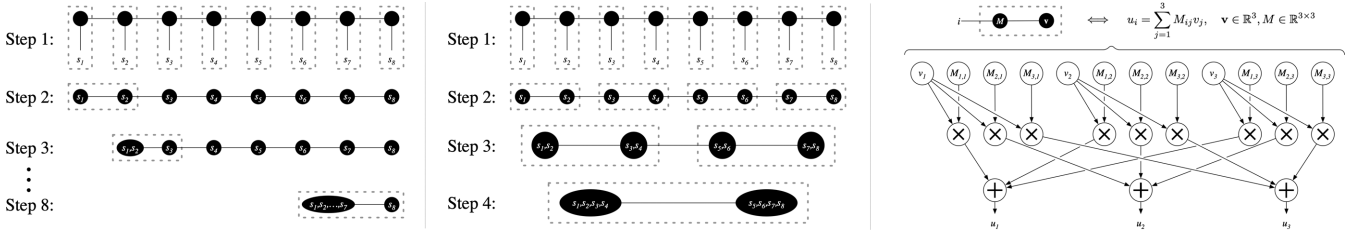


FIG. 2. Left: Sequential contraction scheme for matrix product states: At step 1, we map indices d_1, \dots, d_8 to their corresponding matrices (or vectors at boundaries), a $O(d\chi^2)$ -time operation. In each of the following steps, we contract a boundary vector with its neighboring matrix node, a $O(\chi^2)$ -time operation, amounting to a total of $O(Nd\chi^2)$ for the entire contraction, performed in N steps. Middle: Parallel [37] contraction scheme for matrix product states: Following step 1 as in the sequential contraction, we contract pairs of neighboring nodes in parallel, each an $O(\chi^3)$ -time operation, amounting to a total of $O(N(d + \chi)\chi^2)$ for the entire contraction, performed in $\log_2(N)$ steps. Right: Illustration of a simple contraction scheme, in this case matrix-vector multiplication, as an arithmetic circuit.

given by $v(\mathbf{s}) = \sum_{(u,v) \in E} W_{u,v} \cdot u(\mathbf{s})$, where $\{W_{u,v} \in \mathbb{C}\}_{(u,v) \in E}$ are the parameters of the circuit, corresponding to the tensor nodes in the tensor network. Input to the arithmetic circuit is represented by leaf input nodes, where for every s_i and possible value k there is an indicator node $v_{i,k} = \mathbb{1}[s_i = k]$. The depth of the circuit is defined the same as for neural networks. See Fig. 2 for an illustration of a simple TN to AC conversion.

III. RESULTS

Here we present our main results. First, NN can represent any quantum state that is described by a TN with the same efficiency. Second, there exist states that NN can describe efficiently but require exponential time for common forms of TN. The main outcome of our work is the representability diagram in Fig. 3, summarizing the expressive power of NN and TN as variational quantum states. As discussed, the expressive efficiency of TN is defined with respect to a given contraction scheme that gives rise to an explicit computation in the form of an AC, composed of product and weighted sum operations. Hence, the fundamental question is whether AC can be efficiently simulated by NN.

This section is organized as follows. We present our main theorem for the efficient approximation of AC by NN in Sec. III A. Using that, we prove our corollaries for approximating general tensor network contractions in Sec. III B and then examine more closely the implications for the special case of MPS in Sec. III C. In Sec. III D we prove that NQS can represent volume-law states very efficiently, improving on the bounds of prior works, which most of the commonly used TN geometries cannot represent efficiently.

A. Main theorem: Neural networks can efficiently simulate arithmetic circuits

While the exact relationship between NN and AC has not been well studied [38], several works did study the relationship between NN and other polynomial functions [39–41]. However, these prior methods do not yield sufficiently good bounds when applied to the problem at hand, leading to impractical results. This inefficiency is inherently related to focusing on linear metrics between functions rather than multiplicative. Because WF amplitudes are normalized, their absolute values are very small while their relative values are

often orders-of-magnitude apart. See Appendix B for a longer discussion.

Unlike prior approaches, we consider the approximation of the log-value of AC, i.e., finding g such that $\|g - \ln f\|_\infty < \epsilon$ —which translate to multiplicative bound in linear space—rather than $\|g - f\|_\infty < \epsilon$. Working in log-space has the advantage that more reasonable values (not dependent on N) for ϵ are sufficient for a meaningful approximation of WF amplitudes. Using the infinity norm to measure the distance of two states gives a precise estimate over all inputs.

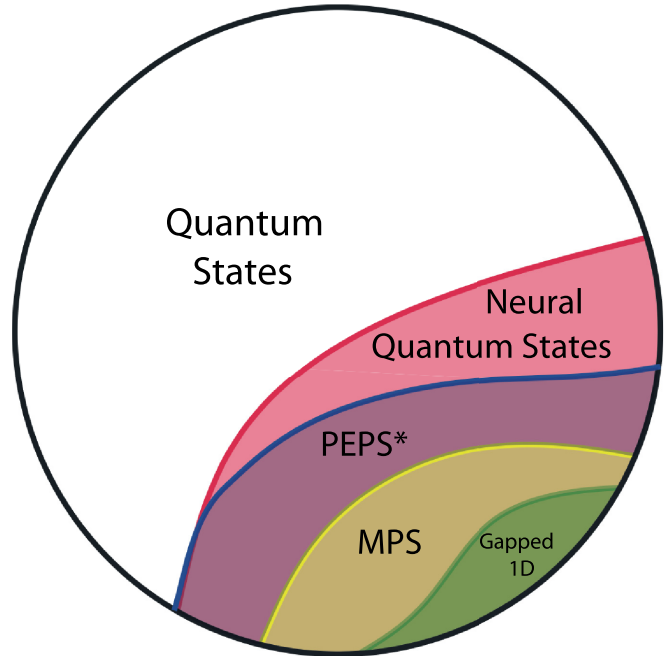


FIG. 3. Expressive power of classically tractable variational quantum states. Different classes of quantum states describing a qudit system with N degrees of freedom and comprising $\text{poly}(N)$ variational parameters are compared. MPS can efficiently represent gapped ground states of one-dimensional systems. PEPS* denotes projected entangled pair states of bond dimension χ that are exactly or approximately contracted in $\text{poly}(N, \chi)$ time on a classical computer. NQS comprise all polynomially tractable TN, thus include MPS, and PEPS*, while also representing additional states with volume-law entanglement that are not efficiently described by such planar TN.

Typically the fidelity is used to measure closeness of states, i.e., $F(\psi, \phi) = \frac{|\langle \psi | \phi \rangle|^2}{\langle \psi | \psi \rangle \langle \phi | \phi \rangle}$. However, notice that closeness of the log-value under the infinity norm also implies closeness under the fidelity, while infinity norm in the linear domain does not entail such relationship unless ϵ is very small (see Appendix C for proofs):

Claim 1. If $\|\ln \psi - \ln \phi\|_\infty < \epsilon < 1/4$, then $F(\psi, \phi) \geq 1 - 4\epsilon$.

Claim 2. There exist functions $\psi, \phi : 2^N \rightarrow \mathbb{C}$, where $\|\psi - \phi\|_\infty < \epsilon$, but $F(\psi, \phi) < 2^{-N}$ when $\epsilon^{-2} \ll 2^N$.

We assume the magnitude of the AC's output is strictly positive for all inputs and greater than some fix value, f_{\min} such that the log-value is well defined. f_{\min} can be extremely small, on the order of $10^{-10^{10}}$, without having a meaningful impact on our results and so it bares little practical significance. Furthermore, to simplify the presentation of our proofs, we assume the absolute value of both real and imaginary parts to be strictly positive, though this last assumption could be relaxed. Under these settings, we proved that NN can simulate AC to almost arbitrary precision and with little overhead:

Theorem 1. Let $f : \mathcal{X} \rightarrow \mathbb{C}$ be a complex-valued function described by an AC comprising n nodes and m edges, of depth l , and using complex parameters. Assume $0 < f_{\min} \equiv \inf_{x \in \mathcal{X}} \min\{|\operatorname{re}[f(x)]|, |\operatorname{im}[f(x)]|\}$, and define $W_{\max} \equiv \max\{1, \max_{e \in E} W_e\}$. Then there exist a function $g : \mathcal{X} \rightarrow \mathbb{R}^2$ described by a neural network comprising $O(n + m + c)$ nodes, $O(m + c)$ edges, of depth $O(l \log(m) + c)$, and using softplus activation functions and real parameters such that $\max_{x \in \mathcal{X}} |g_1(x) + i \cdot g_2(x) - \log[f(x)]| < \epsilon$, where $c(\epsilon, m, W_{\max}, f_{\min}) \equiv O(\ln^2(\frac{m}{\epsilon} \ln(\frac{W_{\max}}{f_{\min}})) + \ln(\frac{1}{\epsilon})\sqrt{\frac{1}{\epsilon}})$.

The proof of Theorem 1, which is given in full in Appendix D, is based on two steps. First, we show that AC with non-negative parameters and inputs can be exactly reconstructed with NN with real parameters and softplus activation functions. In this simple case, for any intermediate values $x_1, x_2 \geq 0$, we can set $o_i = \log(x_i)$ (where 0 is mapped to the right-side limit of $-\infty$) and then multiplication becomes summation, i.e., $\log(x_1 \cdot x_2) = o_1 + o_2$. For summation, softplus activations arise naturally:

$$\ln(e^{o_1} + e^{o_2}) = o_1 + \ln(1 + e^{o_2 - o_1}) = o_1 + \operatorname{softplus}(o_2 - o_1). \quad (1)$$

For log-space summation of n inputs, we can decompose it as a binary tree, which gives the $\log(m)$ correction to the depth of the network. Second, we reduce the complex case to the non-negative case plus a finite number of smooth operations, which can be approximated efficiently by employing various techniques. This is an extension of a known result that any real-valued AC can be reduced to the subtraction of two non-negative AC with similar number of gates as the original [36]. Since only a finite number of operations requires approximation, it results in the additive term $c(\epsilon, m, W_{\max}, f_{\min})$, which is merely logarithmic in the number of edges of the AC, and double logarithmic with respect to the magnitudes of the weights and the WF amplitudes. These weak dependencies of the target AC result in practically arbitrary precision.

B. Approximating general tensor networks

The immediate implication of Theorem 1 is that NQS can simulate TNS at least as efficiently as their TN representation:

Corollary 1. For any TNS with an exact contraction scheme of run-time k , and at most b bits of precision in computations and parameters, there exists a neural network that approximate it with a maximal error of ϵ and of run-time (number of edges) $O(k + \ln^2(\frac{kb}{\epsilon}) + \ln(\frac{1}{\epsilon})\sqrt{\frac{1}{\epsilon}})$.

While MPS can be contracted exactly, many TN that cannot be efficiently contracted exactly can still be used with approximated contraction algorithms. One of the most notable example for such a case are PEPS. The most common general form of approximated contraction scheme are compressed contractions [42]. This approach is based on iteratively contracting local parts of the TN graph exactly, followed by compressing the bond dimension of intermediate local tensors with singular-value decomposition (SVD), thus preventing the blow-out of exact contraction on generic graphs. While the theoretical SVD cannot be exactly represented by an AC, we can show that augmenting it with two other gates, namely division and square-root gates, allows us to represent some of the iterative SVD algorithms used in practice. These new gates can be exactly represented by a single neuron in our network, and thus we can *trace* the compressed contraction scheme and approximate it with a NN, resulting in the following corollary:

Corollary 2. For any TNS with a compressed contraction scheme of run-time k , approximation error $\epsilon/2$, and at most b bits of precision in computations and parameters, there exists a neural network that approximate it with a maximal error of ϵ and of run-time (number of edges) $O(k + \ln^2(\frac{kb}{\epsilon/2}) + \ln(\frac{1}{\epsilon/2})\sqrt{\frac{1}{\epsilon/2}})$.

To simulate SVD, we can rely on one of the iterative approximation algorithms [43] used to compute it in practice. Such algorithms involves iteratively employing matrix multiplications, computing the L_2 norm of a vector, and some divisions, hence the only missing operations that are not native to AC are divisions and square-root operations (for the L_2 -norm). Since our construction already represent the log-value of intermediate computations, then performing both divisions and computing the square root is just as easy as multiplications, where subtractions of log-values correspond to divisions, and multiplying a log value with $1/2$ corresponds to the logarithm of its square root. Combining these methods, an SVD can be simulated with NN, and hence some of the most common approximated contraction schemes as well.

C. Special case: Approximating matrix-product states

For the specific case of MPS, Corollary 2 translates to the following:

Corollary 3. For any MPS over N sites, each of local dimension d , with bond dimension χ , and fixed b bits of precision, there exists a neural network of depth l consisting of m edges that approximates its contraction algorithm up to ϵ , where l and m depend on the contraction scheme [44]:

$$(1) \text{ Sequential: } l = \tilde{O}(N + \sqrt{\frac{1}{\epsilon}}) \text{ and } m = \tilde{O}(Nd\chi^2 + \sqrt{\frac{1}{\epsilon}}).$$

(2) Parallel: $l = \tilde{O}(\ln N + \sqrt{\frac{1}{\epsilon}})$ and $m = \tilde{O}(N(d + \chi)\chi^2 + \sqrt{\frac{1}{\epsilon}})$,

where \tilde{O} denotes big-O while ignoring logarithmic factors.

To illustrate the above results, consider the simple case of $\chi = 2$, $d = 2$, and non-negative translationally symmetric parameters $\{A^{(0)}, A^{(1)} \in \mathbb{R}_{\geq 0}^{2 \times 2}\}$, where $\Psi(\mathbf{s}) = \text{Tr}[A^{(s_1)} \dots A^{(s_N)}]$. Some well-known quantum states can be represented in this form, e.g., the Greenberger-Horne-Zeilinger (GHZ) state [45] (i.e., $|\text{GHZ}\rangle = \frac{|0\rangle^{\otimes N} + |1\rangle^{\otimes N}}{\sqrt{2}}$) uses $A^{(0)} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, $A^{(1)} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$. With slight modifications to support negative values and $d = 3$, the same construction could also represent the AKLT model [46].

The corresponding NQS according to Corollary 3 has the following recursive form (see full derivation in Appendix E):

$$\ln \Psi(\mathbf{s}) = W^1 \tilde{v}^{L,1} + \text{softplus}(W^2 \tilde{v}^{L,1}), \quad (2)$$

$$\tilde{v}^{l,k_l} = W^3 \left[\begin{array}{c} \tilde{v}^{l-1,2k_l-1} \\ \tilde{v}^{l-1,2k_l} \end{array} \right] + \text{softplus} \left(W^4 \left[\begin{array}{c} \tilde{v}^{l-1,2k_l-1} \\ \tilde{v}^{l-1,2k_l} \end{array} \right] \right), \quad (3)$$

$$\tilde{v}^{0,k_0} = \tilde{A} \hat{\mathbf{s}}^{k_0}, \quad (4)$$

where $\tilde{v}^{l,k_l} \in \mathbb{R}^4$ for $0 \leq l \leq \log_2 N$ and $1 \leq k_l \leq 2^{\log_2(N)-l}$, \mathbf{s} is represented with one-hot encoded vectors $\hat{\mathbf{s}}^1, \dots, \hat{\mathbf{s}}^N \in \{0, 1\}^d$ such that $\hat{\mathbf{s}}_j^i = 1 \iff s_i = j$, and define $\tilde{A} \in \mathbb{R}^{4 \times 2}$ where $\tilde{A}_{ij} = \ln A_{\lfloor i/2 \rfloor, \text{mod}(i,2)}^{(j)}$ (using zero-indexing). Any zeros in A could be replaced with some arbitrarily small ϵ . The weights $W^1, W^2 \in \mathbb{R}^{1 \times 4}$ and $W^3, W^4 \in \mathbb{R}^{4,8}$ are constant matrices that do not depend on the values of $A^{(0)}$ and $A^{(1)}$.

The above could be described as convolutional layers with window length and stride equal to 1 and 2 for Eq. (E7) and Eq. (E8), respectively. Note that this construction is actually suboptimal for representing GHZ, as more compact NQS exist: $\ln \Psi(\mathbf{s}) = c \mathbf{1}_{1 \times 2} \text{ReLU}[\mathbf{1}_{1 \times N} \mathbf{s} - (N-1)\mathbf{1}_{2 \times 1}] - c - \ln \sqrt{2}$, where $c > 0$ and $\mathbf{1}_{n \times m}$ is the n -by- m matrix of ones—as $c \rightarrow \infty$ then $\Psi \rightarrow \text{GHZ}$ with exponential convergence. This illustrates that our results merely show worst-case complexity bounds, but NQS could be even more efficient for specific states.

Furthermore, Corollary 3 enables us to directly quantify NQS's expressive power on special classes of quantum systems using previous, rigorous results on MPS. For example, Hastings famously established an area-law entanglement for the gapped ground state of one-dimensional systems [47] that directly translates into an efficient approximation by MPS [47–50], which together with Corollary 3 implies:

Corollary 4. Consider a one-dimensional (1D) Hamiltonian H defined on N qudits of finite local dimension d , and with a nonvanishing spectral gap Δ . The ground state of a H can be written as a deep neural network of depth $l = O(\ln N + \sqrt{1/\epsilon})$ and $m = O(\text{poly}(N, 1/\epsilon))$ edges.

D. Efficient volume-law neural quantum state

Though we have established a strictly inclusive relationship, we can also show that the reverse is not true, i.e., there are NQS that cannot be efficiently reproduced by most commonly used variational TNS:

Corollary 5. There exist quantum states that can be represented by NN with linear (1D) or sublinear ($\geq 2D$) parameters in the number of sites and that MPS and PEPS cannot represent efficiently unless they use exponentially many parameters.

The proof is based on prior results [29,52] that used convolutional AC (ConvAC) as *indirect analogues* to convolutional NN and showed that ConvAC can represent some volume-law states, which MPS and PEPS cannot represent efficiently. Theorem 1 closes this theoretical gap, transferring these results to conventional NN. It is especially noteworthy that our constructive approximation of ConvAC coincidentally results in the ResNet [51] architecture—arguably the most prominent architectures in computer vision. See Fig. 4 for an illustration. Corollary 5 leaves open the possibility of other novel TNS geometries that might be efficiently evaluated while supporting volume-law states. Nevertheless, Theorem 1 shows even such hypothetical TNS could be represented by NQS. Overall, we have established the representability diagram of Fig. 3.

IV. DISCUSSION

We introduced a general mapping between tensor networks and deep neural networks. This mapping allows us to directly connect two of the most important classes of parametric representations of high-dimensional functions and to establish a representation diagram of modern many-body variational quantum states.

We expect that our mapping will be especially useful for establishing further rigorous representation results on neural-network-based quantum states using the well-developed theory of tensor-network representations. Indeed, our analysis is limited in scope to TNS that support efficient amplitude evaluation, i.e., $\langle s | \psi \rangle$, which do cover all planar TNS that are most commonly adopted in practical computations. However, our work also lays the important foundations for studying the relation to TNS that support efficient computation of expectation values, for a restricted set of observables, even if they lack the ability to efficiently compute (or approximate) $\langle s | \psi \rangle$, e.g., multiscale entanglement renormalization ansatz (MERA) and its variants [53,54].

Furthermore, the kind of neural-network architectures and connectivity patterns resulting from our mapping might inspire practical applications leveraging tensor-network algorithms. Similarly, our mapping can help clarify when gradient-based optimization, ubiquitous in ML, should be considered over successful alternating-optimization strategies commonly employed with tensor networks.

ACKNOWLEDGMENT

This research was supported by the ERC (European Research Council) and the ISF (Israel Science Foundation).

APPENDIX A: INTRODUCTION TO TENSOR NETWORKS

Here we give a brief introduction to the basic concepts of TN. See Fig. 5 for the accompanying illustrations. TN is a graphical notation for describing common tensor operations and factorization schemes. Nodes in the graph represent

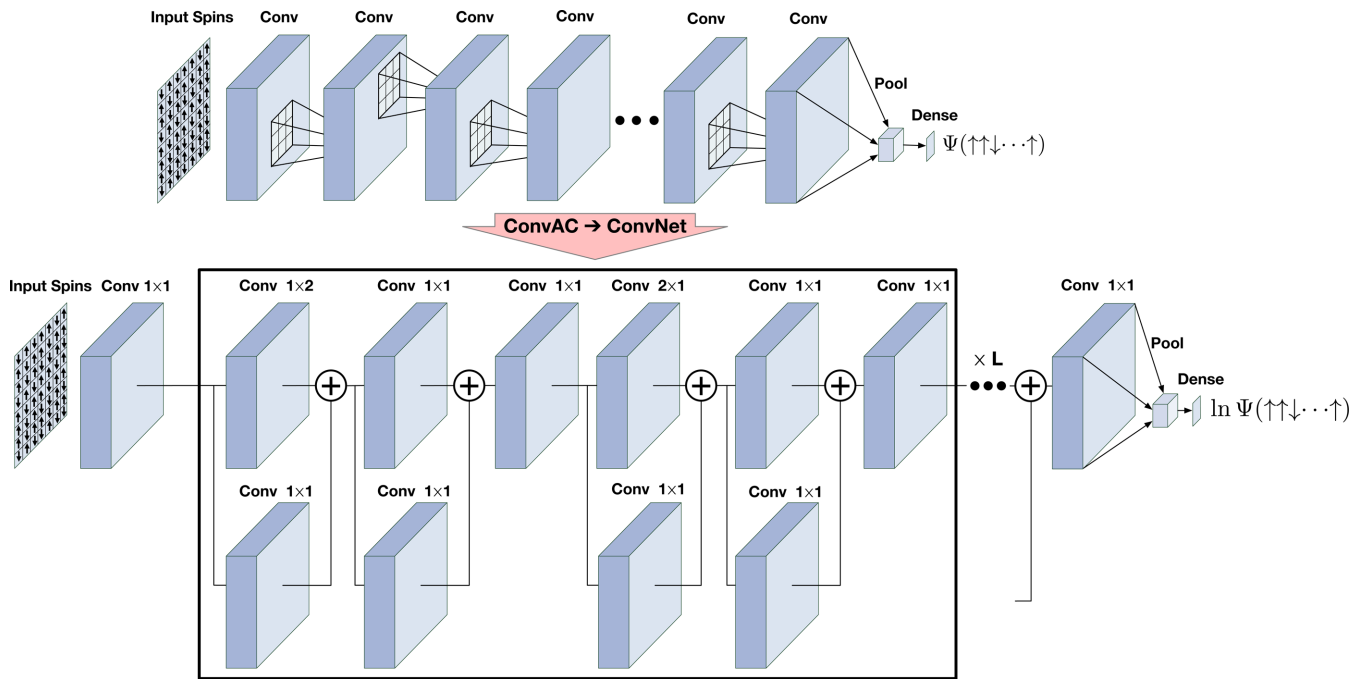


FIG. 4. Convolutional arithmetic circuit is a theoretical ansatz capable of representing volume-law states with polynomial time and memory complexity [29], unlike most planar TNS such as PEPS. Here we illustrate its transformation into a conventional convolutional neural network following Theorem 1 that give rise to a ResNet-like architecture [51]. The residual connections are a direct result of the $o_1 + \text{softplus}(o_2 - o_1)$ construction found in Eq. (D1).

tensors, where edges correspond to indices, ranging from vectors (top left) and matrices (top middle) to arbitrary high-dimensional tensors (top right). Connected nodes represent tensor contractions, i.e., a summation over matching indices of the products of all tensor nodes in the graph, e.g., matrix-vector multiplication (bottom left). Tensor networks are useful for describing tensor factorizations, e.g., SVD factorization of matrices (bottom right). The most commonly used forms of TN are MPS, tree tensor networks, PEPS, and MERA.

APPENDIX B: RELATED WORKS ON APPROXIMATING POLYNOMIAL FUNCTIONS WITH NEURAL NETWORKS

When examining the ability of NN to approximate polynomials, one can notice that while the weighted sum operation is straightforward for NN, the product operation is not trivially simulated by NN and has been the topic of several works [39–41] in the context of the approximation power of NN. Nevertheless, we could not base our approximation scheme on these claims without attaining worse bounds. The most recent result [40] on approximating products with NN demonstrates a construction with a width and a depth at most $O(\log(M/\epsilon))$ such that $\max_{x,y \in [-M,M]} |\text{NN}(x,y) - x \cdot y| < \epsilon$. While this

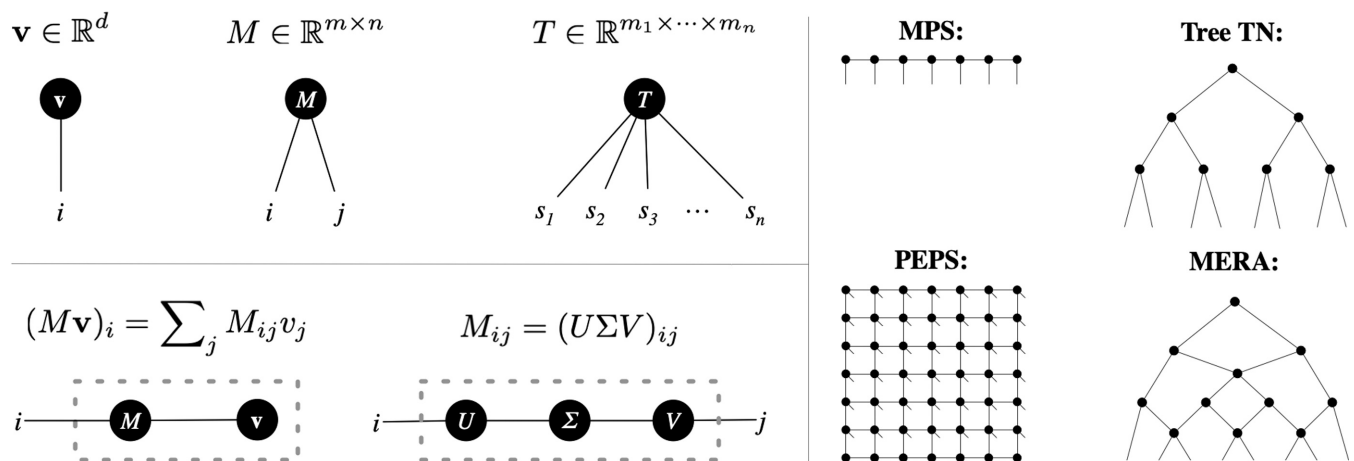


FIG. 5. Illustrations of basic elements and operations of tensor networks, as well as common tensor networks types.

impressive rate of approximation is sufficient for many purposes, it is less suitable for quantum states representation.

Consider for example an arbitrary N -qubit system, then due to normalization at least half of its wave-function amplitudes are, in modulus, less than $2^{-N/2}$, which entails $\epsilon < 2^{-N/2}$ for a meaningful approximation. Thus, using this construction would require at least poly(N) width and depth for every product operation, resulting in a multiplicative polynomial penalty to the runtime. In practice, this polynomial penalty would have major ramifications. To put this in perspective, a 10×10 two-dimensional system would require at least hundreds of NN layers regardless of the complexity of the TNS.

APPENDIX C: PROOFS OF CLAIMS ON THE RELATIONSHIP BETWEEN THE FIDELITY AND THE INFINITY NORM

1. Proof of Claim 1

Let ψ and ϕ be two WF with nonzero magnitudes everywhere, such that their logarithm is well defined, i.e., $\ln \psi(s) = a(s) + i \cdot b(s)$ and $\ln \phi(s) = c(s) + i \cdot d(s)$. Assume $\|\ln \psi - \ln \phi\|_\infty < \epsilon < \frac{1}{4}$.

We begin by finding a lower bound for the inner product:

$$\begin{aligned} |\langle \psi | \phi \rangle| &\geq |\operatorname{Re}(\langle \psi | \phi \rangle)| = \left| \sum_s e^{a(s)+c(s)} \cos[d(s) - b(s)] \right| = \left| \sum_s |\psi(s)|^2 e^{c(s)-a(s)} \cos[d(s) - b(s)] \right| \\ &\times \geq \left| \sum_s |\psi(s)|^2 e^{-\epsilon} \cos(\epsilon) \right| = \|\psi\|_2^2 e^{-\epsilon} \cos(\epsilon) \end{aligned}$$

and similarly we can find an upper bound for the norm of ϕ :

$$\|\phi\|_2^2 = \sum_s |\phi(s)|^2 = \sum_s |\psi(s)|^2 e^{2[c(s)-a(s)]} \leq \|\psi\|_2^2 e^{2\epsilon}.$$

Using the above, we can find a lower bound for the fidelity:

$$F(\psi, \phi) = \frac{|\langle \psi | \phi \rangle|^2}{\|\psi\|_2^2 \|\phi\|_2^2} \geq e^{-4\epsilon} (\cos \epsilon)^2 \geq 1 - 4\epsilon.$$

The last step can be confirm by looking at the first and second derivative of the difference between the left-hand and right-hand sides, showing they are both strictly positives in the range (0,0.25):

$$\begin{aligned} f(x) &= e^{-4x} (\cos x)^2 - 1 + 4x \\ f'(x) &= 4 - 4e^{-4x} (\cos x)^2 - 2e^{-4x} \cos x \sin x \\ f''(x) &= 2e^{-4x} + 12e^{-4x} (\cos x)^2 + 16e^{-4x} \cos x \sin x, \end{aligned}$$

$f''(x)$ is composed of strictly positive terms in the given range, and so it is strictly positive. Because $f'(0) = 0$ and $f''(x) > 0$ for $x \in (0, 1/4)$ then $f'(x) > 0$ in that range as well, and similarly for $f(x)$, proving the lower bound.

2. Proof of Claim 2

Let $s' \in 2^N$ an arbitrary point and define:

$$\begin{aligned} \psi(s) &= \begin{cases} 1 & s = s' \\ \frac{1}{2^{N-1}} & \text{otherwise} \end{cases}, \\ \phi(s) &= \begin{cases} 1 - \epsilon & s = s' \\ \epsilon + \frac{1}{2^{N-1}} & \text{otherwise} \end{cases}. \end{aligned}$$

Clearly $\|\psi - \phi\|_\infty \leq \epsilon$, and yet:

$$\begin{aligned} |\langle \psi | \phi \rangle|^2 &= \left(1 + \frac{1}{2^{N-1}} \right)^2 \\ \|\psi\|_2^2 &= 1 + \frac{1}{2^{N-1}} \end{aligned}$$

$$\begin{aligned} \|\phi\|_2^2 &= 1 + \frac{1}{2^{N-1}} + 2^N \epsilon^2 \\ F(\psi, \phi) &= \frac{1}{1 + (2^N - 1)\epsilon^2} \\ &\approx \begin{cases} 2^{-N} & \epsilon^{-2} \ll 2^N \\ 1 - (2^N - 1)\epsilon^2 & \epsilon^{-2} \gg 2^N \end{cases}. \end{aligned}$$

APPENDIX D: PROOF OF THEOREM 1

In this section we describe the proof of Theorem 1. We begin by providing a sketch of the proof, followed by the full proof. As mentioned in the main text, we prove the theorem in two steps. First, prove the theorem for the case of non-negative AC. Second, reduce the general complex case to the non-negative case.

1. Proof sketch

The proof is based on two steps. First, we show that AC with non-negative parameters and inputs can be exactly reconstructed with NN with real parameters and softplus activation functions. Let $o_1 = \log(x_1)$, $o_2 = \log(x_2)$ for $x_1, x_2 \geq 0$. Then, working in log-space, multiplication becomes summation, i.e., $\log(x_1 \cdot x_2) = o_1 + o_2$, making input-input multiplication trivial for NN, unlike before. For every input-parameter multiplication, i.e., a sum-node edge in the AC graph, we add an auxiliary neuron with a single input. The AC's parameters are stored in the bias terms of these auxiliary neurons, adding m nodes to the NN but with negligible effect on runtime (number of edges). For summation, softplus activations arise naturally:

$$\begin{aligned} \log(x_1 + x_2) &= \log[\exp(o_1) + \exp(o_2)] \\ &= o_1 + \log[1 + \exp(o_2 - o_1)] \\ &= o_1 + \operatorname{softplus}(o_2 - o_1). \end{aligned} \quad (D1)$$

For log-space summation of n inputs, we can decompose it as a binary tree, which gives the $\log(m)$ correction to the depth of

the network. With both log-space NN analogs in place, a non-negative AC can be exactly reproduced with same asymptotic time complexity.

For the second step, we reduce the general complex case to the non-negative case. A real number $x \in \mathbb{R}$ can be represented with a redundant representation of two non-negative numbers $x_+, x_- \geq 0$ by $x = x_+ - x_-$. Addition and multiplication can be applied directly on this representation:

$$\begin{aligned} x + y &= (x_+ + y_+) - (x_- + y_-) \\ x \cdot y &= (x_+ \cdot y_+ + x_- \cdot y_-) - (x_- \cdot y_+ + x_+ \cdot y_-). \end{aligned}$$

Thus, a real AC can be expressed as the difference of two non-negative AC, and a complex AC by representing the real and imaginary parts in this fashion. Finally, to compute the logarithm of this redundant complex representation, i.e., the log-magnitude and phase, we employ various univariate approximation schemes. Since these two operations are smooth and used only at the end of the network, it results in the additive term $c(\epsilon, m, W_{\max}, f_{\min})$, which is merely logarithmic in the number of edges of the AC, and double logarithmic with respect to the magnitudes of the weights and the WF amplitudes. Due to these weak dependencies of the target AC, it allows for an approximation with a practically arbitrary precision.

2. Non-negative case

For the first step, we assume an AC with non-negative inputs and parameters. The inputs and AC parameters are transformed to their log-value, where we extend the real-line with $\pm\infty$ and represent $\log(0) = -\infty$. For most practical considerations, $-\infty$ could be substituted with a large but finite negative constant.

In our NN construction, we freely use the identity instead of a softplus activation function when it is more convenient. We can do so because the identity operation can be simulated with arbitrary precision using the weighted sum of just two neurons with softplus activations:

$$\begin{aligned} x &= \max(x, 0) - \max(-x, 0), \\ \max(x, 0) &= \lim_{\delta \rightarrow \infty} \frac{\text{softplus}(\delta x)}{\delta} = \lim_{\delta \rightarrow \infty} \frac{1}{\delta} \ln[1 + \exp(\delta x)], \\ &= \lim_{\delta \rightarrow \infty} \begin{cases} \frac{1}{\delta} \overbrace{\ln[1 + \exp(\delta x)]}^{\rightarrow 0} & x \leq 0, \\ x + \frac{1}{\delta} \overbrace{\ln[1 + \exp(-\delta x)]}^{\rightarrow 0} & x > 0 \end{cases}, \\ \Rightarrow x &= \lim_{\delta \rightarrow \infty} \frac{\text{softplus}(\delta x) - \text{softplus}(-\delta x)}{\delta}. \end{aligned}$$

The above workaround can at most double the number of neurons and edges in our construction, and thus does not affect our asymptotic bounds.

Every product node with k in-edges in the AC is replaced by a neuron with k in-edges, whose weights are set to 1 and bias to 0, representing multiplication in log-space, i.e., $\log(\prod_{i=1}^k x_i) = \sum_{i=1}^k o_i$, where $\{o_i = \exp(x_i)\}_{i=1}^k$ are the log-values of the connected nodes.

Every weighted-sum node with k in-edges and parameterized by $\mathbf{w} \in \mathbb{R}_{\geq 0}^k$ is replaced by the following NN subgraph

of $O(k)$ nodes and $O(k)$ edges. Every input-parameter multiplication term, i.e., $w_i \cdot x_i$, is represented by a single neuron with a single in-edge with weights set to 0 and bias set to w_i , resulting in $p_i \equiv \log(w_i \cdot x_i) = w_i + o_i$. Without losing our generality, assume $k = 2^t$ for some $t \in \mathbb{N}$, and so we can decompose $\sum_{i=1}^k p_i$ as a complete binary tree of depth t , $2k - 1$ nodes, and $2k - 1$ in-edges in total. Each node in the tree represent a binary addition, which can be realized with two neurons, one with softplus activation and one with identity:

$$\begin{aligned} \log(x_1 + x_2) &= \log[\exp(o_1) + \exp(o_2)] \\ &= o_1 + \log[1 + \exp(o_2 - o_1)] \\ &= o_1 + \text{softplus}(o_2 - o_1). \end{aligned}$$

Applying the above transformations to a non-negative AC with n nodes, m edges, and depth l results in a NN of depth $l \log(m)$ with $O(n + m)$ nodes and $O(m)$ edges, concluding the proof of the first step.

3. Complex case

For the second step, we begin initially by transforming a complex AC into four distinct non-negative AC graphs, representing the following four ‘‘parts’’ of a complex number: positive real, negative real, positive imaginary, and negative imaginary.

Every real number $x \in \mathbb{R}$ can be represented with the redundant form $x = x_+ - x_-$, where $x_+, x_- \in \mathbb{R}_{\geq 0}$. Multiplication and addition can be performed directly within that representation using the following identities:

$$\begin{aligned} x + y &= [x_+ + y_+] - [x_- - y_-], \\ x \cdot y &= [x_+ \cdot y_+ + x_- \cdot y_-] - [x_+ \cdot y_- + x_- \cdot y_+]. \end{aligned}$$

Similarly, a complex number $z \in \mathbb{C}$ can be represented with four components, $z = z_{\text{re},+} - z_{\text{re},-} + i \cdot (z_{\text{im},+} - z_{\text{im},-})$, where $z_{\text{re},+}, z_{\text{re},-}, z_{\text{im},+}, z_{\text{im},-} \in \mathbb{R}_{\geq 0}$.

Given a complex AC with m edges, n nodes, and of depth l , we can use the above redundant representation for its inputs, parameters, and intermediate computations. Propagating the operations with the above identities through the complex AC graph, results in four non-negative AC, each with $O(m)$ edges, $O(n)$ nodes, and of depth $O(l)$, denoting each component of the complex AC’s output, i.e., $AC(z) = AC(\hat{z})_{\text{re},+} - AC(\hat{z})_{\text{re},-} + i \cdot [AC(\hat{z})_{\text{im},+} - AC(\hat{z})_{\text{im},-}]$, where $\hat{z} = (z_{\text{re},+}, z_{\text{re},-}, z_{\text{im},+}, z_{\text{im},-})$. The logarithm of each of these non-negative AC can be represented with a NN according to the first step.

What remains is to convert the redundant representation to a log-polar form, i.e., $\log(z) = \log(|z|) + i \cdot \arg(z)$, per the desired output described in Theorem 1. We employ various approximation techniques to simulate this operation. In the following we denote the components of the redundant representation and its log-value by $o_{\text{re},+} = \ln z_{\text{re},+}$, $o_{\text{re},-} = \ln z_{\text{re},-}$, $o_{\text{im},+} = \ln z_{\text{im},+}$, and $o_{\text{im},-} = \ln z_{\text{im},-}$.

a. Estimating log |z|

In this subsection, we describe the estimation of $\log(|z|)$ by softplus networks.

$\log(|z|)$ can be expressed with respect to the redundant representation's components as:

$$\begin{aligned}\log |z| &= \ln \left(\sqrt{|z_{\text{re}}|^2 + |z_{\text{im}}|^2} \right), \\ &= \ln |z_{\text{re}}| + \frac{1}{2} \ln [1 + \exp(2 \ln |z_{\text{im}}| - 2 \ln |z_{\text{re}}|)], \\ &= \ln |z_{\text{re}}| + \frac{1}{2} \text{softplus}(2 \ln |z_{\text{im}}| - 2 \ln |z_{\text{re}}|), \quad (\text{D2})\end{aligned}$$

where $z_{\text{re}} = z_{\text{re},+} - z_{\text{re},-}$ and $z_{\text{im}} = z_{\text{im},+} - z_{\text{im},-}$.

In the rest of this subsection we focus on the approximation of $\ln |z_{\text{re}}|$, where the same methods can be applied for $\ln |z_{\text{im}}|$. We begin by defining $o_{\text{re},\text{max}} = \max(o_{\text{re},+}, o_{\text{re},-})$ and $o_{\text{re},\text{min}} = \min(o_{\text{re},+}, o_{\text{re},-})$, and similarly for the imaginary part. Recall that $\max(x, y) = y + \max(x - y, 0)$ and $\min(x, y) = y - \max(y - x, 0)$, and so both can be approximated to arbitrary precision with softplus networks. With that, we can write:

$$\begin{aligned}\ln |z_{\text{re}}| &= \ln [\max(z_{\text{re},+}, z_{\text{re},-}) - \min(z_{\text{re},+}, z_{\text{re},-})] \\ &= \ln [\exp(o_{\text{re},\text{max}}) - \exp(o_{\text{re},\text{min}})], \\ &= o_{\text{re},\text{min}} + \ln [\exp(o_{\text{re},\text{max}} - o_{\text{re},\text{min}}) - 1], \\ &= o_{\text{re},\text{min}} + \text{softplus}^{-1}(o_{\text{re},\text{max}} - o_{\text{re},\text{min}}),\end{aligned}$$

where softplus^{-1} is the inverse of the softplus function. To approximate the inverse, we employ two strategies: (i) For large values, $\text{softplus}^{-1}(x) \approx x$ to a high precision, and (ii) for smaller values, we estimate the inverse using root-finding algorithms, and specifically, the bisection method.

Let $\epsilon > 0$, and $x = o_{\text{re},\text{max}} - o_{\text{re},\text{min}}$. For $x > x_{\text{large}} \equiv -\ln[1 - \exp(-\epsilon)]$ it holds that $|x - \text{softplus}^{-1}(x)| < \epsilon$. For realizing the bisection method, we first set the initial search range for $y^* = \text{softplus}^{-1}(x)$. y_{max}^* can be set to x_{large} because $\text{softplus}^{-1}(x) \leq x$. For y_{min}^* we can bound the minimal value of x as follows:

$$\begin{aligned}x = o_{\text{re},\text{max}} - o_{\text{re},\text{min}} &= \ln \left(\frac{z_{\text{re},\text{max}}}{z_{\text{re},\text{min}}} \right) = \ln \left(\frac{|z_{\text{re}}| + z_{\text{re},\text{min}}}{z_{\text{re},\text{min}}} \right) \\ &= \ln \left(\frac{|z_{\text{re}}|}{z_{\text{re},\text{min}}} + 1 \right) \geq \ln \left(\frac{f_{\text{min}}}{z_{\text{re},\text{min}}} + 1 \right).\end{aligned}$$

Next, we upper bound the value of $z_{\text{re},\text{min}}$ by finding an upper bound on the value of a generic non-negative AC with m edges. First, we replace every nonzero weight with the maximal weight in the graph. Then we can replace every weighted sum with $v(\mathbf{s}) = \sum_{(u,v) \in E} W_{u,v} u(\mathbf{s}) \leq \{ \{(u,v) \in E\} (\max_{e \in E} W_e) [\max_{(u,v) \in E} u(\mathbf{s})] \}$. Finally, we can prove by induction along the topological order of the graph that the output of every subgraph of m' edges is upper bounded by $[m' \max_{(v,u) \in E} |W_{v,u}|]^{m'}$. Thereby, we can set $x_{\text{min}} \equiv \ln \left[\frac{f_{\text{min}}}{(mW_{\text{max}})^m} + 1 \right]$, and thus $y_{\text{min}}^* \equiv \ln \left[\frac{f_{\text{min}}}{(mW_{\text{max}})^m} \right]$.

To simulate the bisection algorithm, we define the approximate Heaviside function by $H_\delta(x) \equiv \max(\frac{x}{2\delta} + \frac{1}{2}, 0) - \max(\frac{x}{2\delta} - \frac{1}{2}, 0)$ that satisfies $H = \lim_{\delta \rightarrow 0} H_\delta$, and use the following recursive update rule for $T \equiv \lceil \log_2(y_{\text{max}}^* - y_{\text{min}}^*/\epsilon) \rceil$ steps:

$$\begin{aligned}m_i &\equiv \frac{y_{i-1,\text{min}} + y_{i-1,\text{max}}}{2}, \\ c_i &\equiv H_\delta[\text{softplus}(m) - x]\end{aligned}$$

$$y_{i,\text{min}} \equiv c_i y_{i-1,\text{min}} + (1 - c_i) m_i,$$

$$y_{i,\text{max}} \equiv c_i m_i + (1 - c_i) y_{i-1,\text{max}},$$

where the multiplications are approximated according to Yarotsky [40], which requires an additional $O(\ln(\max\{|y_{\text{max}}^*|, |y_{\text{min}}^*|\}/\tilde{\epsilon}))$ edges and depth per multiplication, where $\tilde{\epsilon} \equiv \epsilon/8T$. The usual bisection method relies on the exact Heaviside function; however, if δ is chosen to be small enough, then it, too, satisfies the range halving property, i.e., it holds that $y_{i,\text{max}} - y_{i,\text{min}} = \frac{y_{i-1,\text{max}} - y_{i-1,\text{min}}}{2}$ and $\text{softplus}^{-1}(x) \in [y_{i,\text{min}}, y_{i,\text{max}}]$. The latter holds because either $|\text{softplus}(m) - x| \geq \delta$, a regime at which $H_\epsilon = H$, or $|\text{softplus}(m) - x| < \delta$, which due to the Lipschitzness of softplus^{-1} it holds that $|m - \text{softplus}^{-1}(x)| \leq L|\text{softplus}(m) - x| \leq L\delta$. Thus, for $\delta < \epsilon/2L$, the claim holds. Similarly, we can use the approximated Heaviside function once more to combine both regimes of x , by outputting $H_\delta(x - x_{\text{large}})x + [1 - H_\delta(x - x_{\text{large}})]m_T$.

In total, to approximate $\log |z|$ up to ϵ , requires $O(\ln^2(\frac{m}{\epsilon} \ln(\frac{W_{\text{max}}}{f_{\text{min}}})))$ nodes, edges, and depth on top of the base NN used to approximate the four non-negative AC.

b. Estimating $\arg z$

In this subsection, we describe the estimation of $\arg z$ by softplus networks, building on the approximations of $\ln |z_{\text{re}}|$ and $\ln |z_{\text{im}}|$ described in the previous subsection.

$\arg z$ can be computed according to the following formula:

$$\arg z = \text{atan2}(z_{\text{im}}, z_{\text{re}})$$

$$= \begin{cases} \arctan \left(\frac{z_{\text{im}}}{z_{\text{re}}} \right) & z_{\text{re}} > 0 \\ \arctan \left(\frac{z_{\text{im}}}{z_{\text{re}}} \right) + \pi & z_{\text{re}} < 0 \wedge z_{\text{im}} \geq 0 \\ \arctan \left(\frac{z_{\text{im}}}{z_{\text{re}}} \right) - \pi & z_{\text{re}} < 0 \wedge z_{\text{im}} < 0 \\ +\frac{\pi}{2} & z_{\text{re}} = 0 \wedge z_{\text{im}} > 0 \\ -\frac{\pi}{2} & z_{\text{re}} = 0 \wedge z_{\text{im}} < 0 \\ \text{undefined} & z_{\text{re}} = 0 \wedge z_{\text{im}} = 0 \end{cases}.$$

Since we assumed $|z_{\text{im}}|, |z_{\text{re}}| > 0$, then only the first three cases are relevant. Therefore, we can write the formula using the following compact form:

$$\arg z = \arctan \left(\frac{z_{\text{im}}}{z_{\text{re}}} \right) + H(-z_{\text{re}}) \text{sgn}(z_{\text{im}}) \pi,$$

where H is the Heaviside function. Furthermore, we can rewrite in terms of $\ln |z_{\text{im}}|$ and $\ln |z_{\text{re}}|$:

$$\begin{aligned}\arg z &= \text{sgn}(z_{\text{re}} z_{\text{im}}) \arctan \left(\left| \frac{z_{\text{im}}}{z_{\text{re}}} \right| \right) + H(-z_{\text{re}}) \text{sgn}(z_{\text{im}}) \pi, \\ &= \text{sgn}(z_{\text{re}} z_{\text{im}}) \arctan \exp(\ln |z_{\text{im}}| - \ln |z_{\text{re}}|) \\ &\quad + H(-z_{\text{re}}) \text{sgn}(z_{\text{im}}) \pi.\end{aligned}$$

The signs of z_{re} can be computed as $\text{sgn}(z_{\text{re}}) = H(o_{\text{re},+} - o_{\text{re},-}) - H(o_{\text{re},-} - o_{\text{re},+})$, which can be approximated with softplus networks using the approximated Heaviside function, H_δ (defined in previous subsection). Since we proved in the previous section that $|o_{\text{re},+} - o_{\text{re},-}| \geq \ln \left[\frac{f_{\text{min}}}{(mW_{\text{max}})^m} + 1 \right]$ then using $0 < \delta < \ln \left[\frac{f_{\text{min}}}{(mW_{\text{max}})^m} + 1 \right]$ the approximated Heaviside function will be equivalent to the exact Heaviside in the regime of our network. Similarly, $H(-z_{\text{re}}) = \frac{1 - \text{sgn}(z_{\text{re}})}{2}$, and so could be computed exactly as well. The

multiplications between these terms and $\arctan \exp(\cdot)$ can be approximated via Yarotsky [40], where in this case since the values are all bounded by ± 2 , then we only need $O(\ln(1/\epsilon))$ nodes, edges and depth for this subnetwork.

To approximate $t(x) \equiv \arctan \exp(x)$, we start with a piecewise-linear approximation, which can then be approximated to arbitrarily precision with softplus networks. Since $t(x) = \frac{\pi}{2} - t(-x)$, then it is enough to show an approximation for $x \leq 0$, and the $x > 0$ case can be constructed with the above identity. For $x \leq 0$, t is a $1/2$ -smooth convex function because its derivative, $\frac{1}{\exp(x) + \exp(-x)}$, is strictly increasing and bounded by $1/2$ in this range. Hence, for any $x, y < 0$ it holds that $|t(x) + t'(x)(y-x) - t(y)| \leq \frac{1}{4}|y-x|^2$.

For any $x_{\min} < 0$ and $n \in \mathbb{N}$, define the following piecewise linear function. Use $n+1$ uniformly spaced anchor points in the $[x_{\min}, 0]$ range, where the first and last anchors are the boundaries. For every anchor point x , denote the first-order linear approximation at this point by $l_x(y) = t(x) + t'(x)(y-x)$. Since t is convex in this range, then $l_x(y) \leq t(y)$, and so for every two neighboring anchor points $x_1 < x_2$ the intersection point of l_{x_1} and l_{x_2} must lie in the range (x_1, x_2) . Define the segments of the piecewise linear function according to the intersection points and the matching linear approximations l_x of the anchor point within each segment. For any two neighboring anchors $x_1 < x_2$ and $x_1 \leq y \leq x_2$, this function can be denoted by $\max\{l_{x_1}(y), l_{x_2}(y)\}$. Using the last inequality, we can bound the error in the $[x_{\min}, 0]$ range with $\frac{1}{4}(\frac{x_2-x_1}{2})^2 \leq \frac{x_{\min}^2}{16n^2}$. For $x < x_{\min}$, we extend the segment of the anchor x_{\min} until its intersection with the x -axis, followed by an open segment for the zero function. Since $\arctan x \leq x$ for any $x > 0$, then $t(x) \leq \exp(x)$, and so for $x_{\min} = -\ln(1/\epsilon)$ it holds that $\forall x \leq x_{\min}, |t(x)| < \epsilon$. Thus, a piecewise linear function with $O(\ln(1/\epsilon)\sqrt{\frac{1}{\epsilon}})$ segments can approximate $t(x)$ up to ϵ maximal difference. Finally, a piecewise linear function with k segments can be realized with a ReLU network of $O(k)$ nodes and edges, and of constant depth.

APPENDIX E: DERIVATION OF NQS FORM FOR MPS WITH BOND DIMENSION $\chi = 2$ AND LOCAL DIMENSION $d = 2$

Below is a derivation for the transformation of an MPS with bond dimension $\chi = 2$, local dimension $d = 2$, and non-negative translationally symmetric parameters, $\{A^{(0)}, A^{(1)} \in \mathbb{R}_{\geq 0}^{2 \times 2}\}$, with the resulting state following the formula $\Psi(\mathbf{s}) =$

$\text{Tr}[A^{(s_1)} \dots A^{(s_N)}]$, into an NQS following the construction of Corollary 3.

The evaluation of $\Psi(\mathbf{s})$ is given by the following recursive formula:

$$\Psi(\mathbf{s}) = V_{11}^{L,1} + V_{22}^{L,1}, \quad (\text{E1})$$

$$V_{ij}^{l,k_l} = V_{i1}^{l-1,2k_l-1} V_{1j}^{l-1,2k_l} + V_{i2}^{l-1,2k_l-1} V_{2j}^{l-1,2k_l}, \quad (\text{E2})$$

$$V_{ij}^{0,k_0} = A_{ij}^{(s_{k_0})}, \quad (\text{E3})$$

where $0 \leq l \leq L = \log_2(N)$ and $1 \leq k_l \leq 2^{\log_2(N)-l}$.

In the NQS form, we represent \mathbf{s} with the one-hot encoded vectors $\hat{s}^1, \dots, \hat{s}^N \in \{0, 1\}^d$ such that $\hat{s}_j^i = 1 \iff s_i = j$, and denote the logarithms of the A and V matrices with \tilde{A} and \tilde{V} , respectively. Any zeros in A could be replaced with some arbitrarily small ϵ . This results in

$$\ln \Psi(\mathbf{s}) = \tilde{V}_{11}^{L,1} + \text{softplus}(\tilde{V}_{22}^{L,1} - \tilde{V}_{11}^{L,1}), \quad (\text{E4})$$

$$\begin{aligned} \tilde{V}_{ij}^{l,k_l} &= \tilde{V}_{i1}^{l-1,2k_l-1} + \tilde{V}_{1j}^{l-1,2k_l} \\ &+ \text{softplus}(\tilde{V}_{i2}^{l-1,2k_l-1} - \tilde{V}_{i1}^{l-1,2k_l-1} + \tilde{V}_{2j}^{l-1,2k_l} - \tilde{V}_{1j}^{l-1,2k_l}), \end{aligned} \quad (\text{E5})$$

$$\tilde{V}_{ij}^{0,k_0} = \tilde{A}_{ij}^{(0)} \hat{s}_1^{k_0} + \tilde{A}_{ij}^{(1)} \hat{s}_2^{k_0}. \quad (\text{E6})$$

In vectorized form, where $\tilde{v}^{l,k_l} \in \mathbb{R}^4$, we could rewrite the above in more conventional terms as:

$$\ln \Psi(\mathbf{s}) = W^1 \tilde{v}^{L,1} + \text{softplus}(W^2 \tilde{v}^{L,1}), \quad (\text{E7})$$

$$\tilde{v}^{l,k_l} = W^3 \begin{bmatrix} \tilde{v}^{l-1,2k_l-1} \\ \tilde{v}^{l-1,2k_l} \end{bmatrix} + \text{softplus} \left(W^4 \begin{bmatrix} \tilde{v}^{l-1,2k_l-1} \\ \tilde{v}^{l-1,2k_l} \end{bmatrix} \right), \quad (\text{E8})$$

$$\tilde{v}^{0,k_0} = \tilde{A} \hat{\mathbf{s}}^{k_0}, \quad (\text{E9})$$

where the weights $W^1, W^2 \in \mathbb{R}^{1 \times 4}$ and $W^3, W^4 \in \mathbb{R}^{4 \times 8}$ are defined as $W^1 = (1 \ 0 \ 0 \ 0)$, $W^2 = (-1 \ 0 \ 0 \ 1)$,

$$W^3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \quad (\text{E10})$$

$$W^4 = \begin{pmatrix} -1 & 1 & 0 & 0 & -1 & 0 & 1 & 0 \\ -1 & 1 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 & -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 & -1 & 0 & 1 \end{pmatrix}. \quad (\text{E11})$$

- [1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016).
- [2] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Machine learning and the physical sciences, *Rev. Mod. Phys.* **91**, 045002 (2019).
- [3] S. R. White, Density Matrix Formulation for Quantum Renormalization Groups, *Phys. Rev. Lett.* **69**, 2863 (1992).

- [4] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, *Ann. Phys.* **326**, 96 (2011).
- [5] R. Orús, Tensor networks for complex quantum systems, *Nat. Rev. Phys.* **1**, 538 (2019).
- [6] F. Verstraete, V. Murg, and J. I. Cirac, Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems, *Adv. Phys.* **57**, 143 (2008).

- [7] I. Cirac, D. Perez-Garcia, N. Schuch, and F. Verstraete, Matrix product states and projected entangled pair states: Concepts, symmetries, and theorems, *Rev. Mod. Phys.* **93**, 045003 (2021).
- [8] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, *Science* **355**, 602 (2017).
- [9] G. Cybenko, Approximation by superpositions of a sigmoidal function, *Math. Contr. Sign. Syst.* **2**, 303 (1989).
- [10] G. Carleo, Y. Nomura, and M. Imada, Constructing exact representations of quantum many-body systems with deep neural networks, *Nat. Commun.* **9**, 5322 (2018).
- [11] X. Gao and L.-M. Duan, Efficient representation of quantum many-body states with deep neural networks, *Nat. Commun.* **8**, 662 (2017).
- [12] D.-L. Deng, X. Li, and S. Das Sarma, Machine learning topological states, *Phys. Rev. B* **96**, 195145 (2017).
- [13] R. Kaubruegger, L. Pastori, and J. C. Budich, Chiral topological phases from artificial neural networks, *Phys. Rev. B* **97**, 195136 (2018).
- [14] I. Glasser, N. Pancotti, M. August, I. D. Rodriguez, and J. I. Cirac, Neural-Network Quantum States, String-Bond States, and Chiral Topological States, *Phys. Rev. X* **8**, 011006 (2018).
- [15] S. Lu, X. Gao, and L.-M. Duan, Efficient representation of topologically ordered states with restricted Boltzmann machines, *Phys. Rev. B* **99**, 155136 (2019).
- [16] K. Choo, T. Neupert, and G. Carleo, Two-dimensional frustrated J_1 - J_2 model studied with neural network quantum states, *Phys. Rev. B* **100**, 125124 (2019).
- [17] O. Sharir, Y. Levine, N. Wies, G. Carleo, and A. Shashua, Deep Autoregressive Models for the Efficient Variational Simulation of Many-Body Quantum Systems, *Phys. Rev. Lett.* **124**, 020503 (2020).
- [18] M. Schmitt and M. Heyl, Quantum Many-Body Dynamics in Two Dimensions with Artificial Neural Networks, *Phys. Rev. Lett.* **125**, 100503 (2020).
- [19] M. Hibat-Allah, M. Ganahl, L. E. Hayward, R. G. Melko, and J. Carrasquilla, Recurrent neural network wave functions, *Phys. Rev. Res.* **2**, 023358 (2020).
- [20] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, Neural-network quantum state tomography, *Nat. Phys.* **14**, 447 (2018).
- [21] D. Pfau, J. S. Spencer, A. G. D. G. Matthews, and W. M. C. Foulkes, Ab initio solution of the many-electron Schrödinger equation with deep neural networks, *Phys. Rev. Res.* **2**, 033429 (2020).
- [22] J. Hermann, Z. Schätzle, and F. Noé, Deep-neural-network solution of the electronic Schrödinger equation, *Nat. Chem.* **12**, 891 (2020).
- [23] K. Choo, A. Mezzacapo, and G. Carleo, Fermionic neural-network states for *ab-initio* electronic structure, *Nat. Commun.* **11**, 2368 (2020).
- [24] F. Verstraete and J. I. Cirac, Renormalization algorithms for quantum-many body systems in two and higher dimensions, [arXiv:cond-mat/0407066](https://arxiv.org/abs/cond-mat/0407066) (2004).
- [25] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac, Computational Complexity of Projected Entangled Pair States, *Phys. Rev. Lett.* **98**, 140506 (2007).
- [26] J. Haferkamp, D. Hangleiter, J. Eisert, and M. Gluza, Contracting projected entangled pair states is average-case hard, *Phys. Rev. Res.* **2**, 013010 (2020).
- [27] D.-L. Deng, X. Li, and S. Das Sarma, Quantum entanglement in neural network states, *Phys. Rev. X* **7**, 021021 (2017).
- [28] J. Chen, S. Cheng, H. Xie, L. Wang, and T. Xiang, Equivalence of restricted Boltzmann machines and tensor network states, *Phys. Rev. B* **97**, 085104 (2018).
- [29] Y. Levine, O. Sharir, N. Cohen, and A. Shashua, Quantum Entanglement in Deep Learning Architectures, *Phys. Rev. Lett.* **122**, 065301 (2019).
- [30] L. Pastori, R. Kaubruegger, and J. C. Budich, Generalized transfer matrix states from artificial neural networks, *Phys. Rev. B* **99**, 165123 (2019).
- [31] A. Borin and D. A. Abanin, Approximating power of machine-learning ansatz for quantum many-body states, *Phys. Rev. B* **101**, 195141 (2020).
- [32] C.-Y. Park and M. J. Kastoryano, Are neural quantum states good at solving non-stoquastic spin Hamiltonians? *Phys. Rev. B* **106**, 134437 (2022).
- [33] We employ the classical definition of a neural network. Some modern architectures assume a broader definition extending its expressiveness.
- [34] V. Nair and G. E. Hinton, Rectified linear units improve restricted Boltzmann machines, in *Proceedings of the 27th International Conference on Machine Learning* (Omnipress, Madison, WI, 2010), pp. 807–814.
- [35] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, Incorporating second-order functional knowledge for better option pricing, in *Advances in Neural Information Processing Systems* (MIT Press, Cambridge, MA, 2000), pp. 472–478.
- [36] A. Shpilka and A. Yehudayoff, Arithmetic circuits: A survey of recent results and open questions, *Found. Trends Theor. Comput. Sci.* **5**, 207 (2010).
- [37] When parallelizing across sites, the *effective* run-time in practice depends mostly on the number of steps, i.e., $\log_2 N$, and χ^2 rather than χ^3 because each matrix multiplication itself can be parallelized across the coordinates of the output matrix, resulting in $O(\chi^2 \log N)$.
- [38] N. Cohen, O. Sharir, Y. Levine, R. Tamari, D. Yakira, and A. Shashua, Analysis and design of convolutional networks via hierarchical tensor decompositions, [arXiv:1705.02302](https://arxiv.org/abs/1705.02302) (2018).
- [39] H. Mhaskar, Q. Liao, and T. Poggio, When and why are deep networks better than shallow ones, in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31 (Association for the Advancement of Artificial Intelligence, Palo Alto, CA, 2017).
- [40] D. Yarotsky, Error bounds for approximations with deep ReLU networks, *Neural Netw.* **94**, 103 (2017).
- [41] M. Telgarsky, Neural networks and rational functions, in *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70, edited by D. Precup and Y. W. Teh (PMLR, International Convention Centre, Sydney, Australia, 2017), pp. 3387–3393.
- [42] J. Gray and G. K.-L. Chan, Hyper-optimized compressed contraction of tensor networks with arbitrary geometry, [arXiv:2206.07044](https://arxiv.org/abs/2206.07044).
- [43] M. Andrecut, Parallel gpu implementation of iterative pca algorithms, *J. Comput. Biol.* **16**, 1593 (2009).
- [44] The bounds on the two contraction schemes serve to highlight different characteristics. The sequential scheme demonstrates that NN can approximate MPS with the same runtime.

- The parallel scheme demonstrates that logarithmic depth is sufficient and is better suited for parallel execution as leveraged by GPUs.
- [45] D. M. Greenberger, M. A. Horne, and A. Zeilinger, Going beyond bell's theorem, in *Bell's Theorem, Quantum Theory and Conceptions of the Universe*, edited by M. Kafatos (Springer Netherlands, Dordrecht, 1989), pp. 69–72.
 - [46] I. Affleck, T. Kennedy, E. H. Lieb, and H. Tasaki, Rigorous Results on Valence-Bond Ground States in Antiferromagnets, *Phys. Rev. Lett.* **59**, 799 (1987).
 - [47] M. B. Hastings, An area law for one-dimensional quantum systems, *J. Stat. Mech.: Theory Exp.* (2007) P08024.
 - [48] I. Arad, A. Kitaev, Z. Landau, and U. Vazirani, An area law and sub-exponential algorithm for 1D systems, [arXiv:1301.1162](https://arxiv.org/abs/1301.1162) (2013).
 - [49] N. Schuch and F. Verstraete, Matrix product state approximations for infinite systems, [arXiv:1711.06559](https://arxiv.org/abs/1711.06559) (2017).
 - [50] A. M. Dalzell and F. G. S. L. Brandão, Locally accurate MPS approximations for ground states of one-dimensional gapped local Hamiltonians, *Quantum* **3**, 187 (2019).
 - [51] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for imagerecognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, New York, 2016), pp. 770–778.
 - [52] O. Sharir and A. Shashua, On the expressive power of overlapping architectures of deep learning, in *Proceedings of the 6th International Conference on Learning Representations (ICLR)* (The International Conference on Learning Representations, 2018).
 - [53] G. Evenbly and G. Vidal, Class of Highly Entangled Many-Body States that can be Efficiently Simulated, *Phys. Rev. Lett.* **112**, 240502 (2014).
 - [54] G. Evenbly and G. Vidal, Scaling of entanglement entropy in the (branching) multiscale entanglement renormalization ansatz, *Phys. Rev. B* **89**, 235113 (2014).