Neural Tensor Contractions and the Expressive Power of Deep Neural Quantum States

Or Sharir
The Hebrew University of Jerusalem
Jerusalem, 9190401, Israel
`or.sharir@cs.huji.ac.il`

Amnon Shashua
The Hebrew University of Jerusalem
Jerusalem, 9190401, Israel
`shashua@cs.huji.ac.il`

Giuseppe Carleo
Institute of Physics
École Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
`giuseppe.carleo@epfl.ch`

Abstract

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 non-linear activation functions. The resulting deep networks feature a number of edges that closely matches 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 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 PEPS, but that are instead efficiently expressible with neural network states.

1 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 [19] 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 [4].

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 [42, 34, 29, 41, 10]. Recently, neural-network-based representation of quantum states, dubbed NQS, have been introduced [6] 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.

Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).

Several theoretical properties of NQS have been established to date. General representation theorems for neural networks [11] 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 [5]. 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 [16]. 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 [13, 24, 17, 26]. 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 [18, 9, 37, 33, 23, 39] and chemistry [32, 22, 8].

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, however 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 efficiently contracted most notably encompass matrix product states (MPS) [42], 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) [40], is non polynomial problem that is known to belong to the #P complexity class [36, 20].

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 [14, 7, 25]. Also, mappings between the two classes of states have been realized, including between general fully-connected NQS and MPS with exponentially large bond-dimension [7]. An approach mapping MPS onto non-standard neural-networks has also been introduced [31]. Despite the important theoretical progress, however 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 [3, 30], 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 dimension and NQS. 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.

2 Preliminaries

We consider in the following a pure quantum system, constituted by N discrete degrees of freedom $\mathbf{s}{\equiv}(s_1,\ldots,s_N)$ (*e.g.* spins, occupation numbers, *etc.*) such that the wave-function (WF) amplitudes $\langle s|\Psi\rangle = \Psi(\mathbf{s})$ fully specify its state. Following the approach introduced in [6], 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[1] 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 $v(\mathbf{s}) = \sigma\left(b_v + \sum_{(u,v)\in E} W_{u,v} u(\mathbf{s})\right)$, 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 $\sigma : \mathbb{R} \to \mathbb{R}$ is some non-linear function known as the *activation function*, e.g., $\mathrm{ReLU}(x) = \max(x, 0)$ or $\mathrm{softplus}(x) = \log(\exp(x) + 1)$ [28, 15]. The root nodes

[1]This is the classical definition of a neural network. However, some of the models used today slightly deviate from it, e.g., self-attention modules use bilinear operations in addition to affine ones. While our proof for section 3 consider only the classical definition, extending it to support many of these variants is trivial.

of the network can optionally use the identity instead of a non-linear 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 also be viewed as a complex tensor $\mathcal{A}_{s_1,\ldots,s_N}$ that is in turn represented in terms of tensor factorization schemes. 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 App. A for a brief introduction to TN. In the next section we will present our main results on 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 (see App. A for a precise description). Second, we must precisely describe the computational circuit of a given TN 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*, 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 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 $v_{i,k} = \mathbb{1}\left[s_i = k\right]$. The depth of the circuit is defined the same as for neural networks. See Fig. 3 for an illustration of a simple TN to AC conversion.

3 Main Results

Here we present our main results. First, that NN can represent any quantum state that is modeled by a TN with the same efficiency. Second, that there exist states that NN can model efficiently, but require exponential time for common forms of TN. The main outcome of our work is the representability diagram in Fig. 1, 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.

While the relationship between NN and AC has not been well studied, several works did study the relationship between NN and other polynomial functions [27, 43, 38]. However, the prior methods do not yield sufficiently good bounds when applied to the problem at hand, resulting in 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 App. B for a longer discussion.

As opposed to 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 then $\|g - f\|_\infty < \epsilon$. 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. We use the infinity norm to measure the error of two states because it gives a precise estimate over all inputs. Another common measure for the closeness of two quantum states is their fidelity, i.e., $|\langle\psi|\phi\rangle|$. However, notice that closeness of the log-value under the infinity norm also implies closeness under the fidelity.

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-define. $f_{\min}$ can be extremely small, on the order of $10^{-10^{10}}$, without having a meaningful impact on our results and so it bares little effect in practice. 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 this settings, we proved that NN can simulate AC to almost arbitrary precision and with little overhead:

Theorem 1 *Let $f : \mathcal{X} \to \mathbb{C}$ be a complex-valued function given by an arithmetic circuit comprising n nodes and m edges, of depth l, and using complex parameters. Assume $0 < f_{\min} \equiv \inf_{x\in\mathcal{X}} \min\{|\mathrm{re}(f(x))|, |\mathrm{im}(f(x))|\}$, and define $W_{\max} \equiv \max\{1, \max_{e\in E} W_e\}$. Then, there exist a function $g : \mathcal{X} \to \mathbb{R}^2$ 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 $c(\epsilon, m, W_{\max}, f_{\min}) \equiv O\left(\ln^2\left(\frac{m}{\epsilon}\ln\left(\frac{W_{\max}}{f_{\min}}\right)\right) + \ln\left(\frac{1}{\epsilon}\right)\sqrt{\frac{1}{\epsilon}}\right)$.*

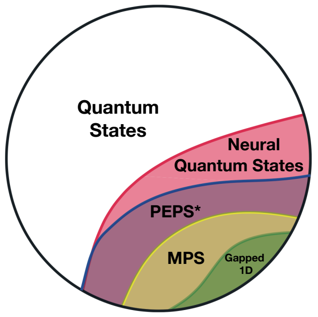

Figure 1: Expressive power of classically tractable variational quantum states. Different classes of quantum states describing a qudit system with N degrees of freedom and comprising $\mathrm{poly}(N)$ variational parameters are compared. Matrix Product States (MPS) can efficiently represent gapped ground-states of one-dimensional systems. PEPS* denotes here Projected Entangled Pair States of bond dimension χ that are exactly or approximately contracted in $\mathrm{poly}(N, \chi)$ time on a classical computer. Neural Quantum States (NQS) comprise all polynomially tractable TN, thus include MPS, and $\mathrm{PEPS}*$, while also representing additional states with volume law entanglement that are not efficiently described by such planar TN.

The proof of Theorem 1, which is given in full in app. C, 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:

$$\log(x_1{+}x_2) = \log(e^{o_1}{+}e^{o_2}) = o_1{+}\log\left(1{+}e^{o_2-o_1}\right) = o_1{+}\,\mathrm{softplus}(o_2{-}o_1). \tag{1}$$

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

Corollary 1 *For any tensor network quantum state with a 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) $O\left(k + \ln^2\left(\frac{kb}{\epsilon}\right) + \ln\left(\frac{1}{\epsilon}\right)\sqrt{\frac{1}{\epsilon}}\right)$.*

For the specific case of MPS, Cor. 1 translates to the following:

Corollary 2 *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:*

1. *Sequential: $l = \tilde{O}\left(N + \sqrt{1/\epsilon}\right)$ and $m = \tilde{O}\left(Nd\chi^2 + \sqrt{1/\epsilon}\right)$.*
2. *Parallel: $\tilde{O}(\ln N + \sqrt{1/\epsilon})$ and $m = \tilde{O}\left(N(d+\chi)\chi^2 + \sqrt{1/\epsilon}\right)$.*

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

In turn, this result also allows to use previously established rigorous results on MPS to directly quantify the expressive power of NQS on special classes of quantum systems. For example, Hastings famously established an area-law entanglement for the gapped ground state of one-dimensional systems [21] that directly translates into an efficient approximation by MPS [21, 2, 35, 12]. Our result in 2, in connection with the bound established in [21] implies:

Corollary 3 *Consider a 1D Hamiltonian H defined on N qudits of finite local dimension d, and with a non-vanishing 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 $m{=}O\left(\mathrm{poly}(N, 1/\epsilon)\right)$ edges.*

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

Corollary 4 *There exist quantum states that can be represented by NN with parameters and runtime polynomial in the number of sites, that MPS, MERA, and PEPS tensor networks cannot represent efficiently unless they use exponential number of parameters.*

The proof is based a prior work that used convolutional AC as indirect analogs to convolutional NN, and showed that convolutional AC can represent some volume-law states, which MPS, PEPS and MERA cannot represent efficiently. Using Theorem 1 we can translate their result to real-world NN. See Fig. 4 in App. D for an illustration. Cor. 4 leaves open the possibility of novel geometries for TNS that could be efficient. Nevertheless, using Theorem 1, even these hypothetical TNS could be represented by NQS.

4 Discussion

In this work we have introduced a general mapping between tensor networks and deep artificial neural networks. This mapping allows to directly connect two of the most important classes of parametric representations of high dimensional functions, and allows to establish a representation diagram of modern variational many-body variational quantum states. Moreover, our results could be extended to support approximated contraction schemes as well (See App. E). We expect that our mapping will be especially useful to establish further rigorous representation results on neural-network based quantum states, using the well-developed theory of tensor-network representations. On the other hand, the kind of neural-network architectures and connectivity patterns resulting from our mapping might also inspire new practical applications inspired by successful tensor-network ideas. Along the same lines, our mapping can also help clarify in what circumstances gradient-based optimization strategies, ubiquitous in machine learning, are to be preferred over successful alternated optimization strategies instead commonly adopted for tensor networks.

Acknowledgments and Disclosure of Funding

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

References

[1] M. Andrecut. Parallel gpu implementation of iterative pca algorithms. *Journal of Computational Biology*, 16(11):1593–1599, 2009. PMID: 19772385.

[2] Itai Arad, Alexei Kitaev, Zeph Landau, and Umesh Vazirani. An area law and sub-exponential algorithm for 1D systems. *arXiv:1301.1162 [cond-mat, physics:quant-ph]*, January 2013. arXiv: 1301.1162.

[3] Artem Borin and Dmitry A. Abanin. Approximating power of machine-learning ansatz for quantum many-body states. *Physical Review B*, 101(19):195141, May 2020. Publisher: American Physical Society.

[4] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Rev. Mod. Phys.*, 91:045002, Dec 2019.

[5] Giuseppe Carleo, Yusuke Nomura, and Masatoshi Imada. Constructing exact representations of quantum many-body systems with deep neural networks. *Nature Communications*, 9(1):5322, 2018.

[6] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[7] Jing Chen, Song Cheng, Haidong Xie, Lei Wang, and Tao Xiang. Equivalence of restricted boltzmann machines and tensor network states. *Phys. Rev. B*, 97:085104, Feb 2018.

[8] Kenny Choo, Antonio Mezzacapo, and Giuseppe Carleo. Fermionic neural-network states for ab-initio electronic structure. *Nature Communications*, 11(1):2368, May 2020. Number: 1 Publisher: Nature Publishing Group.

[9] Kenny Choo, Titus Neupert, and Giuseppe Carleo. Two-dimensional frustrated ${J}_{1}\text{\ensuremath{-}}{J}_{2}$ model studied with neural network quantum states. *Physical Review B*, 100(12):125124, 2019.

[10] Ignacio Cirac, David Perez-Garcia, Norbert Schuch, and Frank Verstraete. Matrix Product States and Projected Entangled Pair States: Concepts, Symmetries, and Theorems. *arXiv:2011.12127 [cond-mat, physics:hep-th, physics:quant-ph]*, November 2020. arXiv: 2011.12127.

[11] G Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[12] Alexander M. Dalzell and Fernando G. S. L. Brandão. Locally accurate MPS approximations for ground states of one-dimensional gapped local Hamiltonians. *Quantum*, 3:187, September 2019. Publisher: Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften.

[13] Dong-Ling Deng, Xiaopeng Li, and S. Das Sarma. Machine learning topological states. *Physical Review B*, 96(19):195145, November 2017.

[14] Dong-Ling Deng, Xiaopeng Li, and S. Das Sarma. Quantum entanglement in neural network states. *Phys. Rev. X*, 7:021021, May 2017.

[15] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating second-order functional knowledge for better option pricing. In *NIPS*, pages 472–478, 2000.

[16] Xun Gao and Lu-Ming Duan. Efficient representation of quantum many-body states with deep neural networks. *Nature communications*, 8(1):662, 2017.

[17] Ivan Glasser, Nicola Pancotti, Moritz August, Ivan D. Rodriguez, and J. Ignacio Cirac. Neural-Network Quantum States, String-Bond States, and Chiral Topological States. *Physical Review X*, 8(1):011006, January 2018.

[18] Ivan Glasser, Nicola Pancotti, Moritz August, Ivan D. Rodriguez, and J. Ignacio Cirac. Neural-network quantum states, string-bond states, and chiral topological states. *Phys. Rev. X*, 8:011006, Jan 2018.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, Cambridge, Massachusetts, November 2016.

[20] Jonas Haferkamp, Dominik Hangleiter, Jens Eisert, and Marek Gluza. Contracting projected entangled pair states is average-case hard. *Physical Review Research*, 2(1):013010, January 2020. Publisher: American Physical Society.

[21] M. B. Hastings. An area law for one-dimensional quantum systems. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(08):P08024, August 2007. Publisher: IOP Publishing.

[22] Jan Hermann, Zeno Schätzle, and Frank Noé. Deep-neural-network solution of the electronic Schrödinger equation. *Nature Chemistry*, 12(10):891–897, October 2020. Number: 10 Publisher: Nature Publishing Group.

[23] Mohamed Hibat-Allah, Martin Ganahl, Lauren E. Hayward, Roger G. Melko, and Juan Carrasquilla. Recurrent neural network wave functions. *Physical Review Research*, 2(2):023358, June 2020. Publisher: American Physical Society.

[24] Raphael Kaubruegger, Lorenzo Pastori, and Jan Carl Budich. Chiral topological phases from artificial neural networks. *Physical Review B*, 97(19):195136, May 2018.

[25] Yoav Levine, Or Sharir, Nadav Cohen, and Amnon Shashua. Quantum entanglement in deep learning architectures. *Phys. Rev. Lett.*, 122:065301, Feb 2019.

[26] Sirui Lu, Xun Gao, and L.-M. Duan. Efficient representation of topologically ordered states with restricted Boltzmann machines. *Physical Review B*, 99(15):155136, April 2019.

[27] Hrushikesh Mhaskar, Qianli Liao, and Tomaso Poggio. When and why are deep networks better than shallow ones? *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017.

[28] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[29] Román Orús. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1(9):538–550, September 2019. Number: 9 Publisher: Nature Publishing Group.

[30] Chae-Yeun Park and Michael J. Kastoryano. Are neural quantum states good at solving non-stoquastic spin Hamiltonians? *arXiv:2012.08889 [cond-mat, physics:quant-ph]*, December 2020. arXiv: 2012.08889.

[31] Lorenzo Pastori, Raphael Kaubruegger, and Jan Carl Budich. Generalized transfer matrix states from artificial neural networks. *Physical Review B*, 99(16):165123, April 2019. Publisher: American Physical Society.

[32] David Pfau, James S. Spencer, Alexander G. D. G. Matthews, and W. M. C. Foulkes. Ab initio solution of the many-electron Schr\"odinger equation with deep neural networks. *Physical Review Research*, 2(3):033429, September 2020. Publisher: American Physical Society.

[33] Markus Schmitt and Markus Heyl. Quantum Many-Body Dynamics in Two Dimensions with Artificial Neural Networks. *Physical Review Letters*, 125(10):100503, September 2020. Publisher: American Physical Society.

[34] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011.

[35] Norbert Schuch and Frank Verstraete. Matrix product state approximations for infinite systems. *arXiv:1711.06559 [cond-mat, physics:quant-ph]*, November 2017. arXiv: 1711.06559.

[36] Norbert Schuch, Michael M. Wolf, Frank Verstraete, and J. Ignacio Cirac. Computational Complexity of Projected Entangled Pair States. *Physical Review Letters*, 98(14):140506, April 2007. Publisher: American Physical Society.

[37] Or Sharir, Yoav Levine, Noam Wies, Giuseppe Carleo, and Amnon Shashua. Deep Autoregressive Models for the Efficient Variational Simulation of Many-Body Quantum Systems. *Physical Review Letters*, 124(2):020503, 2020. Publisher: American Physical Society.

[38] Matus Telgarsky. Neural networks and rational functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3387–3393, International Convention Centre, Sydney, Australia, Aug 2017. PMLR.

[39] Giacomo Torlai, Guglielmo Mazzola, Juan Carrasquilla, Matthias Troyer, Roger Melko, and Giuseppe Carleo. Neural-network quantum state tomography. *Nature Physics*, 14(5):447, May 2018.

[40] Frank Verstraete and J Ignacio Cirac. Renormalization algorithms for quantum-many body systems in two and higher dimensions. *arXiv preprint cond-mat/0407066*, 2004.

[41] Frank Verstraete, Valentin Murg, and J Ignacio Cirac. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics*, 57(2):143–224, 2008.

[42] Steven R White. Density matrix formulation for quantum renormalization groups. *Physical review letters*, 69(19):2863, 1992.

[43] Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103 – 114, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We do our best to specify what can and cannot be deduced from our theoretical claims.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] The paper deals purely with the theoretical question of the representational power of two computational models used for approximating quantum states, and so have no societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Assumptions are included either as part of claims or in preceding paragraphs.
   (b) Did you include complete proofs of all theoretical results? [Yes] In appendices.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Introduction to Tensor Networks

Here we give a brief introduction to the basic concepts of tensor networks (TN). See Fig. 2 for the accompanying illustrations. TN is a graphical notation for describing common tensor operations and factorization schemes. Nodes in the graph represent 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 Matrix Product States (MPS), Tree Tensor Networks (TTN), Projected Entangled Pair States (PEPS), and Multi-scale Entanglement Renormalization Ansatz (MERA).

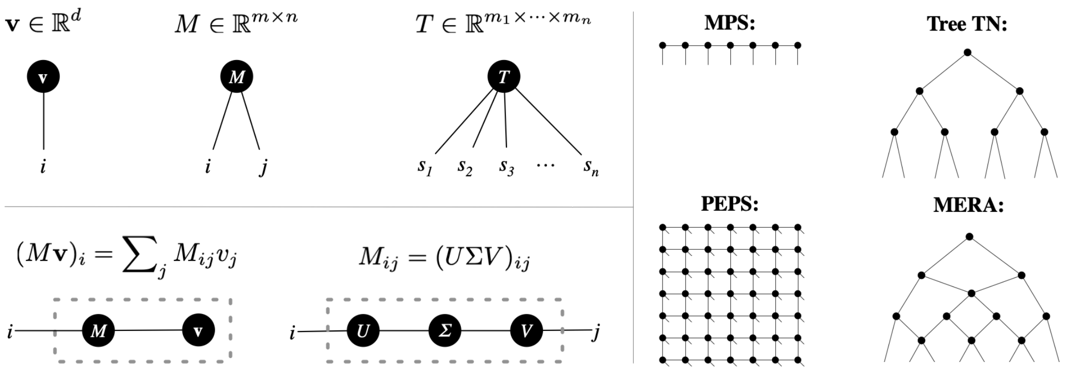
$\mathbf{v} \in \mathbb{R}^d$
$M \in \mathbb{R}^{m\times n}$
$T \in \mathbb{R}^{m_1\times\cdots\times m_n}$
MPS:
Tree TN:
PEPS:
MERA:
$(M\mathbf{v})_i = \sum_j M_{ij}v_j$
$M_{ij} = (U\Sigma V)_{ij}$

Figure 2: Illustrations of basic elements and operations of tensor networks, as well as common tensor networks types.

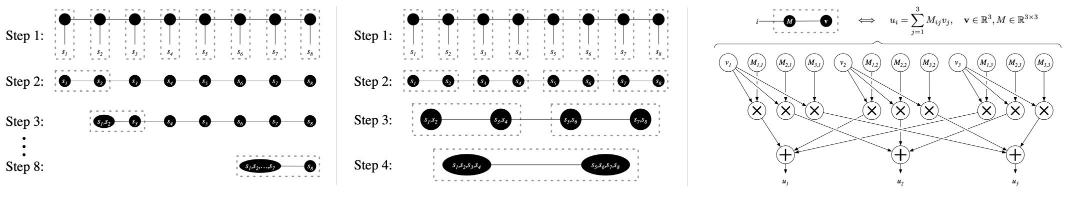
Step 1:
Step 2:
Step 3:
Step 4:
Step 8:
$u_i = \sum_{j=1}^{3} M_{ij}v_j, \quad \mathbf{v} \in \mathbb{R}^3, M \in \mathbb{R}^{3\times 3}$

Figure 3: **(left)** Sequential contraction scheme for Matrix Product States: At step 1, we map indices $d_1, \ldots, d_8$ 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[3]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.

The complexity of contracting a TN exactly is dependent on its contraction 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. 3.

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 straitforward for NN, the product operation is not trivially simulated by NN and has been the topic of several works [27, 43, 38] 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 [43] on approximating products with NN demonstrates a construction with a width and a depth at most $O(\log(M/\epsilon))$ such that $\max_{x,y\in[-M,M]} |\mathrm{NN}(x,y) - x\cdot y| < \epsilon$. While this 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 $\mathrm{poly}(N)$ 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-

[3]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 $O(\chi^2 \log N)$.

dimensional system would require at least hundreds of NN layers regardless of the complexity of the TNS.

C 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.

C.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:

$$\begin{aligned}\log(x_1 + x_2) &= \log(\exp(o_1) + \exp(o_2)) \\ &= o_1 + \log\left(1 + \exp(o_2 - o_1)\right) \\ &= o_1 + \operatorname{softplus}(o_2 - o_1). \end{aligned} \tag{2}$$

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 reproduce 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.

C.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:

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

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 $\mathbf{w} \in \mathbb{R}^k_{\ge 0}$ is replaced by the following NN sub-graph 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 loosing 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 2 neurons, one with softplus activation and one with identity:

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

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.

C.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 $x_+, x_- \in \mathbb{R}_{\ge 0}$. 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 $z_{\text{re},+}, z_{\text{re},-}, z_{\text{im},+}, z_{\text{im},-} \in \mathbb{R}_{\ge 0}$.

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 $o_{\text{im},-} = \ln z_{\text{im},-}$.

C.3.1 Estimating $\log |z|$

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

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

$$\begin{aligned}\log|z| &= \ln\left(\sqrt{|z_{\text{re}}|^2 + |z_{\text{im}}|^2}\right),\\ &= \ln|z_{\text{re}}| + \frac{1}{2}\ln\left(1+\exp\left(2\ln|z_{\text{im}}| - 2\ln|z_{\text{re}}|\right)\right),\\ &= \ln|z_{\text{re}}| + \frac{1}{2}\operatorname{softplus}(2\ln|z_{\text{im}}| - 2\ln|z_{\text{re}}|), \qquad (3)\end{aligned}$$

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 sub-section 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 $o_{\text{re},\max} = \max(o_{\text{re},+}, o_{\text{re},-})$ and $o_{\text{re},\min} = \min(o_{\text{re},+}, o_{\text{re},-})$, 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:

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

where $\operatorname{softplus}^{-1}$ is the inverse of the softplus function. To approximate the inverse, we employ two strategies: (i) for large values, $\operatorname{softplus}^{-1}(x) \approx x$ 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 $x = o_{\text{re},\max} - o_{\text{re},\min}$. For $x > x_{\text{large}} \equiv -\ln(1-\exp(-\epsilon))$ it holds that $\left|x - \operatorname{softplus}^{-1}(x)\right| < \epsilon$. For realizing the bisection method, we first set the initial search range for $y^* = \operatorname{softplus}^{-1}(x)$. $y^*_{\max}$ can be set to x_{large} because $\operatorname{softplus}^{-1}(x) \leq x$. For $y^*_{\min}$ we can bound the minimal value of x as follows

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

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

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 $H = \lim_{\delta\to 0} H_\delta$, and use the following recursive update rule for $T \equiv \lceil \log_2(y^*_{\max} - y^*_{\min}/\epsilon)\rceil$ steps:

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

where the multiplications are approximated according to **(author?)** [43], which requires an additional $O(\ln(\max\{|y^*_{\max}|,|y^*_{\min}|\}/\tilde{\epsilon}))$ 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 $y_{i,max} - y_{i,min} = \frac{y_{i-1,max}-y_{i-1,min}}{2}$ and $\text{softplus}^{-1}(x) \in [y_{i,min}, y_{i,max}]$. 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 $\left|m - \text{softplus}^{-1}(x)\right| \leq L\,|\text{softplus}(m) - x| \leq L\delta$. 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 $O\left(\ln^2\left(\frac{m}{\epsilon}\ln\left(\frac{W_{\max}}{f_{\min}}\right)\right)\right)$ nodes, edges, and depth on top of the base NN used to approximate the four non-negative AC.

C.3.2 Estimating $\arg z$

In this sub-section, 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 sub-section.

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

$$\begin{aligned}\arg z &= \text{atan2}(z_{\text{im}}, z_{\text{re}}) \\ &= \begin{cases} \arctan\left(\frac{z_{\text{im}}}{z_{\text{re}}}\right) & z_{\text{re}} > 0 \\ \arctan\left(\frac{z_{\text{im}}}{z_{\text{re}}}\right) + \pi & z_{\text{re}} < 0 \wedge z_{\text{im}} \geq 0 \\ \arctan\left(\frac{z_{\text{im}}}{z_{\text{re}}}\right) - \pi & z_{\text{re}} < 0 \wedge z_{\text{im}} < 0 \\ +\frac{\pi}{2} & z_{\text{re}} = 0 \wedge z_{\text{im}} > 0 \\ -\frac{\pi}{2} & z_{\text{re}} = 0 \wedge z_{\text{im}} < 0 \\ \text{undefined} & z_{\text{re}} = 0 \wedge z_{\text{im}} = 0 \end{cases}.\end{aligned}$$

Since we assumed $|z_{\text{im}}|, |z_{\text{re}}| > 0$, then only the first 3 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 $\ln|z_{\text{re}}|$:

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

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 sub-section). Since we proved in the previous section that $|o_{\text{re},+} - o_{\text{re},-}| \geq \ln\left(\frac{f_{\min}}{(mW_{\max})^m} + 1\right)$ then using $0 < \delta < \ln\left(\frac{f_{\min}}{(mW_{\max})^m} + 1\right)$ 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 $\arctan\exp(.)$ can be approximated via **(author?)** [43], where in this case the since the values are all bounded by ± 2, then we only need $O(\ln(1/\epsilon))$ nodes, edges and depth for this sub-network.

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 $\frac{1}{4}\left(\frac{x_2 - x_1}{2}\right)^2 \leq \frac{x_{\min}^2}{16n^2}$. 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 $O\left(\ln(1/\epsilon)\sqrt{\frac{1}{\epsilon}}\right)$ 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.

D Illustration of a Volume-Law Neural-Network Quantum State

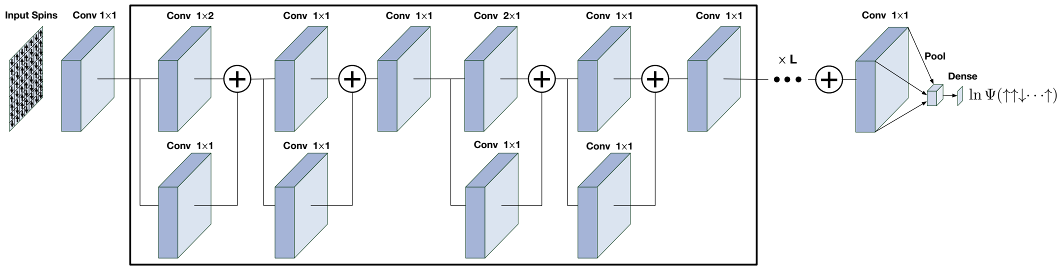

Figure 4: An illustration of a convolutional neural network (ConvNet), according to Cor. 4, that is capable of representing two-dimensional quantum spin ($d = 2$) states with volume-law entanglement entropy that a PEPS model cannot represent unless it employs exponential (in number of sites) parameters. The ConvNet is made up of a sequence of L blocks. Each block has two spatial (either $2{\times}1$ or $1{\times}2$ window size) convolutional layers, eight (in general, $2\log_2(2d)$) local convolutional layers ($1{\times}1$ windows) and residual connections between them. The residual connections are a direct result of the $o_1 + \mathrm{softplus}(o_2 - o_1)$ construction found in Eq. 2. There are no pooling layers except in the next to last layer, in which a global average pooling operation is applied. The network ends with a dense layer reducing the dimension to a scalar that represents the log-amplitude of the quantum state. If a system employs a $q{\times}q$ grid ($N = q^2$), then a ConvNet with $L = q/2$ blocks can be used to represent some volume-law states, which PEPS cannot represent efficiently. For one-dimensional systems, a network can be similarly constructed, demonstrating separation from both MPS and MERA as well. The same construction can also be extended to higher dimensions and for $d > 2$.

E Proof Sketch for Extending Results to Approximated Contraction Schemes

Some TN that cannot be efficiently contracted exactly can still be used with approximate. One of the most notable example for such a case are PEPS. When examining these approximated contraction algorithms, they typically mostly involve iterative application of linear operations, except for employing a Singular-Value Decomposition operation. Hence, if we could simulate the SVD operation with a NN, then we could *trace* the operations of the approximated contraction scheme, use Theorem 1 for simulating the linear operations and then approximate the remaining SVD operations.

To simulate SVD, we can rely on one of the iterative approximation algorithms [1] used to compute it in practice. This algorithm involves iteratively employing matrix multiplications, computing the L_2 norm of a vector, and some divisions, hence the only missing part is approximating divisions and square-root operations (for the L_2-norm). Since our construction already represent the log-value of intermediate computations, then performing divisions is just as easy as multiplications – simply a subtraction. As for the square-root, it can be approximated with Newton's method with a quadratic convergence, using just additions, multiplications and divisions. Combining these methods, an SVD can be simulated with NN, and hence some of the most common approximated contraction schemes as well.