Connectivity determines the capability of sparse neural network quantum states

Brandon Barton[1,2], **Juan Carrasquilla**[1], **Christopher Roth**[2*], **Agnes Valenti**[2*]
[1]Institute for Theoretical Physics, ETH Zürich, Zürich 8093, Switzerland
[2]Center for Computational Quantum Physics, Flatiron Institute, New York, NY 10010, USA
{bbarton,jcarrasquilla}@ethz.ch, {croth,avalenti}@flatironinstitue.org

Abstract

The Lottery Ticket Hypothesis (LTH) posits that within overparametrized neural networks, there exist sparse subnetworks that are capable of matching the performance of the original model when trained in isolation from the original initialization. We extend this hypothesis to the unsupervised task of approximating the ground state of quantum many-body Hamiltonians, a problem equivalent to finding a neural-network compression of the lowest-lying eigenvector of an exponentially large matrix. Focusing on two representative quantum Hamiltonians, the transverse field Ising model (TFIM) and the toric code (TC), we demonstrate that sparse neural networks can reach accuracies comparable to their dense counterparts, even when pruned by more than an order of magnitude in parameter count. Crucially, and unlike the original LTH, we find that performance depends only on the structure of the sparse subnetwork, not on the specific initialization, when trained in isolation. Moreover, we identify universal scaling behavior that persists across network sizes and physical models, where the boundaries of scaling regions are determined by the underlying Hamiltonian. At the onset of high-error scaling, we observe signatures of a sparsity-induced quantum phase transition that is first-order in shallow networks. Finally, we demonstrate that pruning enhances interpretability by linking the structure of sparse subnetworks to the underlying physics of the Hamiltonian.

1 Introduction

Over the past decades, pruning has emerged as popular approach to reduce the number of parameters of neural networks [39, 33, 51, 32]. The goal of pruning is to speed up model inference and improve portability without compromising on accuracy. In practice, contemporary models can be pruned to 50% or even 90% sparsity while mostly retaining their original performance [30, 28, 64]. Pruning has also led to crucial insight on the inner workings of neural networks with many parameters: The lottery ticket hypothesis (LTH) states that sparse sub-networks ("winning tickets") are contained within densely parametrized models that can match or exceed the performance of the original model when trained in isolation with proper initialization [25]. The LTH has been verified in subsequent experimental and theoretical studies [26, 22, 55, 17] as well as extended to different types of neural networks [65, 15, 16, 66].

We here examine the LTH in a non-traditional use-setting: The task of approximating the ground state of a quantum many-body system with a neural network [10, 11]. Our task can be abstracted to the problem of learning a mapping from discrete data (spin-configurations $(\sigma_1, ...\sigma_N)$, where $\sigma_i \in \{+1, -1\}$ and N is the number of spin degrees of freedom) to a set of real numbers, called *amplitudes*, which specifies the ground state eigenvector of a many-body Hamiltonian.

*These authors contribitued equally to this work.

Preprint. Under review.

The object we represent is profoundly different from that of typical machine learning tasks such as image recognition [38], or language modeling [44]. There, a neural network is asked to represent extremely sparse objects, such as the subset of all RGB images that contain a cat, or the subset of random vocabulary strings that form coherent sentences. In contrast, a quantum many-body wavefunction is in general a fully dense representation, with probability amplitudes for different configurations spanning many orders of magnitudes. In order to capture the correct physics, these amplitudes must be encoded with high precision which typically necessitates the use of a more performant training algorithm than stochastic gradient descent. A common choice is natural gradient descent, which accounts for the curvature of the neural-network function.

A hidden benefit of considering physical systems is that we can readily apply physics-based analysis tools to better study the pruning dynamics. Since our neural network directly encodes a physical phase, we can use the language and mathematics of quantum phase transitions as a tool to understand the different sparse representations that we find with pruning.

In order to look for robust principles, we vary both the neural network architecture and the physical Hamiltonian. Throughout this work, we consider two distinct physical models: the transverse field Ising model for different values of transverse field and the topological toric code model [36]. We test the LTH on these cases and analyze the behavior of the pruned networks in terms of the physics of the ground state wavefunction.

Surprisingly, we find that an even stronger version of the LTH holds for the ground state search problem on the Hamiltonians we consider in this work. By using a learned sparse structure with $5 - 20\%$ weights of the dense model, we are able to match the performance of a fully dense model using a random initialization of weights. We find that the optimal sparsity depends on the physics of the considered quantum many-body model. Additionally, we see that starting from the weights from the "lottery ticket initialization" offers negligible benefit over random weights. Crucially, the *connectivity* of the sparse masks by itself constitutes a valid lottery ticket independent of its initialization (see Section (4.1)).

In Section (4.2), we proceed to examine the scaling behavior of shallow FFNNs as a function of pruning and identify scaling regions that are universal with respect to varying initial network size, in accordance with [54]: At both low and high sparsities, we find plateaus where the accuracy is fairly independent of the number of remaining parameters. Given that these plateaus occur at very different variational energies, they are likely representing different phases of matter. We gain further insights into the nature of these plateaus, and what happens in between, using tools to characterize phases of quantum matter. In particular, we show in Section (4.3) that the low-and high-error plateau correspond to physical phases of matter, connected by at least one (first-order) phase transition and possibly a third intermediate phase.

In Section (4.4), we then proceed to relate the structure and weights of the pruned sub-networks to the physics of the underlying model. We show that our pruning procedure discovers an efficient solution to the toric code problem with an error that can be made asymptotically small.

Contributions Our contributions are as follows:

- We show that neural network quantum many-body wavefunctions have a different form of the LTH. Here, the lottery tickets are sparse subnetworks, and the initial weights are not relevant to the final performance. This contrasts with the LTH in other problem domains [25, 15], where a lottery ticket consists of a sparse subnetwork initialized with particular parameter values.
- We demonstrate that neural network wavefunctions have distinct scaling regions across several quantum many-problems, consisting of high-and low error plateaus that bookend an approximately power law region. Changing the number of parameters only affects the performance of the models in the low-error plateau.
- We introduce quantum physics-based tools for interpreting pruning behavior in terms of phase transitions. We use this to identify a first-order phase transition as a function of sparsity for shallow FFNNs trained on the TFIM.
- We find an asympotically exact solution to the toric code through pruning, which we elucidate in detail. When the toric code is pruned past the minimum sparsity for this solution, it undergoes a phase transition.

2 Related work

Pruning neural-network quantum states In the context of neural-network quantum states, the effect of pruning has been studied for the task of reconstructing a known quantum many-body wave-function from a 1D Hamiltonian [29, 57], which is a supervised learning task. In contrast, we study an unsupervised learning task of finding the minimum energy variational wavefunction of 2D Hamiltonians. Our work also interprets the structure of the learned sparse representations in light of the physical Hamiltonian.

Scaling laws This work is inspired by previous studies on predictable scaling laws [2] of sparse neural networks during iterative pruning [54]. We test these methods on a non-traditional task of learning a quantum ground state. For neural quantum states, it is important to understand how the number of parameters necessary to achieve a desired relative error scales with system size. While this relation has been studied for dense models [12, 48, 21], here we examine scaling behavior through the lens of *sparse* networks by pruning initially dense models and analyzing the resulting error scaling.

Structure of sparse subnetworks Following the LTH, a stronger hypothesis [50] was proven [41, 49, 46, 8, 19, 23], showing that high-performing sparse subnetworks already exist at initialization ("strong lottery tickets") for a variety of neural network architectures. Traditionally, strong lottery tickets are therefore characterized by both a sparse subnetwork and a set of initial parameter values. In support of this, it has also been shown in the original LTH that changing the initialization of lottery tickets results in a performance decrease [25]. Our evidence points to the contrary for the case of neural-network quantum states, as the lottery ticket performance only depends on its structure, not its parameter initialization. Other related studies have shown that parameter sign configurations play an important role in obtaining winning tickets [27, 67, 45]. In this work, we are able to find sparse subnetwork representations that correspond to asymptotically exact solutions of the problem at hand in both absolute values of the weights as well as their sign structure.

Phase transitions in neural networks Several works have used tools from statistical mechanics in order to analyze neural network representations. One approach understands phase transitions as a function of training time, by considering the neural network as a classical Hamiltonian with individual neurons as spin variables, and the weights as time-dependent coupling strengths as a function of training under stochastic gradient descent [4, 63]. Another perspective characterizes phase transitions in a neural network's generalization capabilities as a function of training samples by employing replica methods from statistical mechanics [59, 31, 3, 56, 40, 5, 18]. Our work makes a direct connection to phase transitions in quantum many-body physics – we characterize phases directly by computing order parameters on a many-body wavefunction parameterized by a neural network. We find that these order parameters can change as a function of sparsity.

3 Experimental set-up

3.1 Neural network representations of quantum states

Throughout this work, we consider systems with N spin-$1/2$ degrees of freedom. We aim to solve the ground-state problem. In particular, we are given a Hamiltonian $\hat{H}$ for which we want to find the lowest-lying eigenstate, i.e.

$$\hat{H}|\Psi\rangle_G = E_G|\Psi\rangle_G, \tag{1}$$

where E_G is the system's ground state energy. In general, the dimensionality of this problem renders an exact solution unfeasible as the Hilbert space size scales as 2^N.

Neural network quantum states have emerged as a method to circumvent this exponential scaling. Concretely, we parametrize the many-body quantum state by expanding it in a basis of spin-configurations

$$|\Psi_\theta^{NN}\rangle = \sum_{\boldsymbol{\sigma}} \Psi_\theta^{NN}(\boldsymbol{\sigma})|\boldsymbol{\sigma}\rangle. \tag{2}$$

Here, the wave-function amplitudes $\Psi_\theta^{NN}(\boldsymbol{\sigma})$ are parametrized with the help of a neural network: The output value of the network is chosen to represent $\log(\Psi_\theta^{NN}(\boldsymbol{\sigma}))$ when given as input a spin-configuration $\boldsymbol{\sigma} = (\sigma_1, ...\sigma_N)$, $\sigma_i \in \{\pm 1\}$. The weights and biases of the network form the set

of parameters θ. Throughout this work, we only consider wave functions that are known to be real and positive. Our experiments are performed on three different network architectures: shallow feed-forward neural networks (FFNNs), shallow convolutional neural networks (CNNs) and few-layer residual convolutional neural networks (ResCNNs). More details on the employed neural networks and hyperparameters can be found in the Appendix.

The neural network is trained to approximate the ground state by making use of the variational principle and minimizing the cost function

$$C(\theta) = \langle \hat{H} \rangle_\theta \geq E_{\mathrm{G}}. \tag{3}$$

Here, $\langle \cdot \rangle_\theta$ denotes the expectation value taken with respect to the wave-function $|\Psi_\theta^{NN}\rangle$. All expectation values are computed via Monte Carlo sampling. We minimize this cost function via natural gradient descent, in the context of the ground-state search also termed "stochastic reconfiguration" (SR) [60]. For more details on sampling and optimization, see Appendix. The simulations are carried out with the help of the package *netket* [9, 62], based on JAX [7] and FLAX [35].

Given a so-obtained ground-state approximation $|\Psi_\theta^{NN}\rangle$, we probe its accuracy using the relative error

$$\epsilon_{\mathrm{rel}}(E) = \left| \frac{E_{\mathrm{exact}} - \langle \hat{H} \rangle_\theta}{E_{\mathrm{exact}}} \right|, \tag{4}$$

where the reference exact ground-state energy E_{exact} is computed either via exact diagonalization on a small system, the density-matrix-renormalization group (DMRG) or the lowest energy that was obtained with our neural network quantum state. In some cases throughout this work, we choose to report the absolute error per spin $\Delta E = |E_{\mathrm{exact}} - \langle \hat{H} \rangle_\theta|/N$, when comparing errors in different quantum phases or across multiple system sizes. All DMRG computations are carried out using the package ITensor [24]. The implementation details can be found in the Appendix.

3.2 Physical models

We study the effect of pruning on the problem of the ground-state search for two different physical models which we introduce below: The transverse-field Ising model and the toric code model. We mainly focus on the transverse-field Ising model due to its versatility, and utilize the toric code model because of its exact solubility to interpret obtained sparse subnetworks.

Transverse-field Ising model We consider the transverse-field Ising model in two dimensions, with $N = L^2$ spin $1/2$-degrees of freedom on a $L \times L$ lattice. The system is described by the Hamiltonian

$$\hat{H}_{\mathrm{TFIM}} = -\sum_{\langle i,j \rangle} \hat{\sigma}_i^z \hat{\sigma}_j^z - \kappa \sum_i \hat{\sigma}_i^x, \tag{5}$$

with the Pauli matrices $\hat{\sigma}^{x/z}$. The interplay between spin coupling of nearest neighboring spins (denoted by $\langle i, j \rangle$) and the transverse magnetic field $\kappa > 0$ determines the quantum phase of the model. In particular, the Ising model exhibits a quantum phase transition as a function of magnetic field κ: For small magnetic field, the spin coupling term dominates and the ground state is ferromagnetic. In the case of a strong transverse field κ, spins are aligned in form of a paramagnet in the $+x$ direction. The quantum phase transition is estimated to occur at $\kappa_c \approx 3.04438(2)$ [6]. At the position of the phase transition, the system is at criticality and spin-spin correlation functions decay with power-law.

Toric code model As a use-case of a model with an exact solution and topological order, we consider the toric code model [36]. The toric code model is defined on a $L \times L$ lattice with periodic boundary conditions and $N = 2L^2$ spin-$1/2$ degrees of freedom on the edges. The Hamiltonian is given by

$$\hat{H}_{\mathrm{TC}} = -\sum_v \hat{A}_v - \sum_p \hat{B}_p, \tag{6}$$

where $\hat{A}_v = \prod_{i \in v} \hat{\sigma}_i^x$ and $\hat{B}_p = \prod_{i \in p} \hat{\sigma}_i^z$ act on the plaquettes and vertices on the given lattice. The ground-state manifold is four-fold degenerate and consists of eigenstates of the mutually commuting $\hat{A}_v$ and $\hat{B}_p$ with eigenvalue $+1$. The toric code serves as instructive example to test interpretability of neural-network quantum states, as several exact constructions of the toric code ground states with the help of neural networks have been brought forward in previous work [20, 11, 61, 14].

3.3 Sparse training strategies

The employed pruning strategy throughout this work is iterative magnitude pruning with weight rewinding (IMP-WR). We comment on this choice and compare it to other pruning strategies in the Appendix. We detail the pruning algorithm and describe isolated training strategies that are carried out using the sparse masks obtained via IMP-WR below.

Iterative magnitude pruning with weight rewinding Concretely, the IMP-WR procedure begins with training a randomly initialized dense neural network (pre-training phase). The optimized parameters θ_{wr} from this pre-training phase serve as *rewinding point*: Once this rewinding point is obtained, the IP-WR procedure proceeds in an iterative fashion. In each pruning iteration, the network's remaining non-zero weights are initialized in the rewinding point and then trained. After training, a fraction p_r of the weights with smallest magnitude removed in an unstructured fashion. We use a pruning ratio of $p_r = 0.12$ for FFNNs and ResCNNs, and a smaller pruning ratio of $p_r = 0.05$ for CNNs due to their initially lower parameter count. This process is repeated for a total of I pruning iterations. The number of remaining parameters n of the neural network at a particular iteration i is given by

$$n := (1 - p_{\mathrm{r}})^i n_{\mathrm{init}}, \tag{7}$$

where n_{init} is the number of parameters in the initial network. Throughout this work, we report the number of remaining parameters n, and investigate the ability of the NQS to approximate the ground state of physical models as the neural network becomes increasingly sparse. Further details on the pruning procedure are provided in the Appendix.

Isolated training variants To test the lottery ticket hypothesis, we perform different variants of isolated training from sparse initialized networks ("tickets" T). In particular, we denote the initialization of such a sparse subnetwork as a combination of paramaters θ and a pruning mask $m \in \{0, 1\}^{n_{\mathrm{init}}}$. Then, the parameters of the ticket $T(\theta, m)$ are given by element-wise product between θ and the pruning mask m. We then proceed to train θ in isolation, while keeping the pruning mask m fixed. In particular, we define three distinct such tickets:

- $T(\theta_{\mathrm{init}}, m_{\mathrm{imp}})$: Initialized to initial IMP-WR parameters θ_{init} with masks m_{imp}, obtained from IMP-WR at a selected pruning iteration.
- $T(\theta_{\mathrm{rand}}, m_{\mathrm{imp}})$: Initialized to random parameters θ_{rand} with masks m_{imp}, obtained from IMP-WR at a selected pruning iteration.
- $T(\theta_{\mathrm{init}}, m_{\mathrm{rand}})$: Initialized to initial IMP-WR parameters θ_{init} with random sparse masks m_{rand}.

The $T(\theta_{\mathrm{init}}, m_{\mathrm{imp}})$ variant tests the original LTH, while $T(\theta_{\mathrm{rand}}, m_{\mathrm{imp}})$ tests the viability of using the same mask on a new parameter initialization. Finally, $T(\theta_{\mathrm{init}}, m_{\mathrm{rand}})$ is a random sparse initialization of the pruning mask, which serves as a benchmark to the other two isolated training variants. The isolated training variants are tested with masks derived from single trajectories of iterative magnitude pruning, and we report the statistics over several random initializations in the Appendix.

4 Results

4.1 Testing the lottery ticket hypothesis

We test the validity of the LTH for the case of neural-network quantum states. For this purpose, we follow the sparse training strategies outlined in Section 3.3. In particular, we obtain the sparse masks m_{imp} used for our lottery ticket tests by applying IMP-WR on a neural network initialized with parameters θ_{init}. We then proceed with the isolated training of $T(\theta_{\mathrm{init}}, m_{\mathrm{imp}})$, $T(\theta_{\mathrm{rand}}, m_{\mathrm{imp}})$ and $T(\theta_{\mathrm{init}}, m_{\mathrm{rand}})$. The performances using various different model architectures are shown in Fig. 1 for the case of the TFIM at criticality ($\kappa = \kappa_c$) on a $N = 10 \times 10$ lattice.

The original LTH [25] asserts that it is the combination of masks *and* initial parameter values that forms a winning ticket. Thus we would expect training $T(\theta_{\mathrm{init}}, m_{\mathrm{imp}})$ in isolation to greatly outperform the other ticket initialization choices. While we find that winning ticket networks can replicate the performance of the dense network (up to a critical sparsity), we also observe that the

same sparse masks perform equally well when parameters are randomly initialized. On the other hand, we find that sparse networks based on masks with randomly chosen connectivity perform significantly more poorly; the performance begins to degrade with a small amount of pruning, and they reach high error much faster (see Fig. 1).

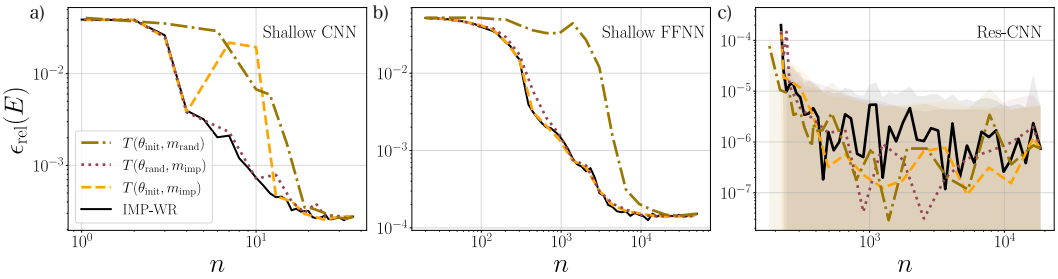

Figure 1: Test of the LTH at the critical point $\kappa = \kappa_c$ of the TFIM on the $N = 10 \times 10$ lattice, for various neural network architectures: (a) shallow CNN, (b) shallow FFNN, and (c) Res-CNN (shaded regions are statistical sampling errors). The relative error is computed with respect to the average energy of three best-performing dense ResCNNs by smallest variance (see Appendix).

In the case of the ResCNN architecture, the performance is fairly independent of both the initial parameters and the learned masks. Most likely this behavior has it roots in the starting point: the initial kernels are of size 3×3. This very localized structure already provides an extremely strong starting point, as we can encode the ground state nearly exactly with O(10^3) parameters.

From these experiments we conclude that a lottery ticket encoding of a quantum wavefunction is simply a sparse connectivity mapping. In Section 4.4 we elucidate this observation by showing that the connective structure of our pruned models can be understood in terms of the Hamiltonian that we simulate.

4.2 Sparse scaling behavior

We analyze the behavior of the neural network and the corresponding quantum wave-function as a function of pruning more in detail. To this end, we focus on the simplest use-case, the shallow FFNN of depth $d = 1$. For more details on the CNN and ResCNN scaling, see Appendix.

Universal scaling behavior In accordance with [54], the scaling behavior of the pruned FFNN error shown in Fig. 1 suggests three scaling regimes: (i) a low-error plateau (LEP) at high densities n (ii) a high-error plateau (HEP) at low densities and (iii) an approximate power-law region (APL) connecting the two plateaus. We find the error scaling in the (HEP) and the (APL) to be *universal* with respect to the width of the initial network; Fig. 2 demonstrates evidence that the error of all shallow FFNNs of different width coalesce onto one another. However, the LEP at large number of remaining parameters shows a slight dependence on the initial number of network parameters: These denser networks are qualitatively different than the very sparse networks in the APL and HEP, involving a more complex structure that is broken within the APL. We will further elaborate on this question in Section 4.3.

Universal scaling behavior within different phases of the TFIM Up to here, we have considered pruning of neural-network quantum states that approximate the ground state of the TFIM at criticality. This corresponds to setting the field strength κ in the Hamiltonian to the critical value $\kappa = \kappa_c$. We now examine the behavior of the pruned FFNN for the ferromagnet ($0 \leq \kappa < \kappa_c$) and the paramagnet ($\kappa > \kappa_c$). We find that each phase exhibits the same qualitative scaling behavior, i.e. showcases the three regions LEP, APL and HEP (see Fig. 2). However, the boundaries of these regions are unique to each phase. In particular, the onset to the HEP occurs at lower sparsity in the case of the ferromagnet in comparison to the paramagnet and the critical state. A possible explanation lies in the sparsity of the true ground state eigenvector: while the paramagnet and critical phases have non-zero amplitudes for many different σ_z magnetizations, the ferromagnet only has significant amplitudes when almost

all of the spins are aligned. Representing this very sparse state with a neural network only requires very few nonzero weights (see Appendix).

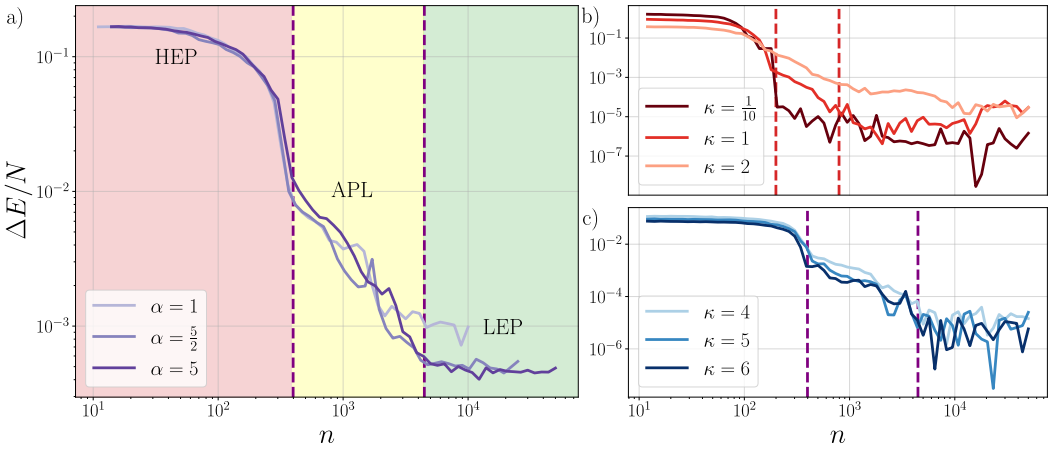

Figure 2: Plotted are regions of error scaling within IMP-WR for the TFIM on the $N = 10 \times 10$ lattice using FFNNs with depth $d = 1$. The subplots correspond to simulations at (a) fixed κ (at the critical point $\kappa = \kappa_c$) with varying network widths $w = \alpha N$ (b) varying $\kappa < \kappa_c$ within the ferromagnetic phase and fixed width $w = 5N$ (b) varying $\kappa > \kappa_c$ within the paramagnetic phase and fixed width $w = 5N$. We qualitatively highlight the scaling regions via colors in (a) and dashed lines in (b) and (c).

4.3 Sparsity-induced phase transition

So far, we have examined the generic neural-network scaling regions under pruning based on the behavior of the relative error. Now, we aim to get a better understanding of the pruning *dynamics*: For this purpose, we fix $\kappa = \kappa_c$ and study the evolution of the neural-network *quantum wave-function* as we vary the network sparsity. We are thereby able to link pruning dynamics of the neural network representation to physical quantum phases.

Fundamental properties in quantum many-body physics such as phases of matter are only well-defined in the thermodynamic limit, i.e. in the limit of an infinite number of (here spin 1/2)-degrees of freedom. In a machine-learning language, this translates into an input layer of infinite size. We here define the thermodynamic limit by sending $n, N \to \infty$ (where N is the number of spins) while at the same time keeping the ratio $\rho = n/N$ fixed.

We approach the thermodynamic limit via a finite-size scaling analysis: We repeat the pruning experiment for varying lattice (input layer) sizes. The results are depicted in Fig. 3. Crucially, when plotted as a function of the number of parameters per spin ρ, the error scaling curves at different lattice sizes coalesce. Thus, the observed features (i.e. the 3 scaling regimes) are well-defined in the thermodynamic limit and not an artifact of finite-size effects. Additionally, the transitions in between the regions occur at a scale-invariant number of parameters per spin. It is thus a natural question to ask, to what extent the scaling regions directly correspond to quantum phases of matter.

For this purpose, we can think of ρ as a phase-transition driving parameter, in analogy with the transverse field κ of the Ising model. Just as we can study the physics of the Ising model as a function of κ by computing correlation functions and order parameters, we can use the exact same toolkit to study $\Psi^{NN}(\rho)$.

What do we know about the quantum phase of $\Psi^{NN}(\rho)$ a priori? When ρ is very large and the neural-network approximation is very good the quantum phase of $\Psi^{NN}(\rho)$ will correspond to the same phase as the targeted state $\Psi^{exact}_{\kappa=\kappa_c}$, i.e. the exact ground state of the TFIM at $\kappa = \kappa_c$. We expect this to be true in the LEP. However, $\Psi^{NN}(\rho)$ may undergo a phase transition to a phase that is easier to represent as ρ is decreased. We find evidence of this behavior in a kink of the network error at

the transition between the APL and the HEP. This signifies a discontinuity in $\frac{dE}{d\rho}$, which is a typical signature of a (first-order) phase transition.

We sustain this claim by considering a more sensitive measure of quantum phase transitions, the *fidelity* $F(\rho) = \langle \Psi^{NN}(\rho)\Psi^{NN}(\rho+\epsilon)\rangle$, where ϵ is small (approaching zero). It measures the overlap between two quantum states states at adjacent points along the pruning trajectory. At the position of a phase transition, the two adjacent states are in two different phases and thus one shall expect the fidelity to approach zero. We find that the fidelity drops sharply at the transition between the APL to HEP scaling regions, providing further evidence for a phase transition. The transition we observe for the TFIM ($\kappa = \kappa_c$) occurs at a critical parameter count around $\rho_c \approx 3 - 5$.

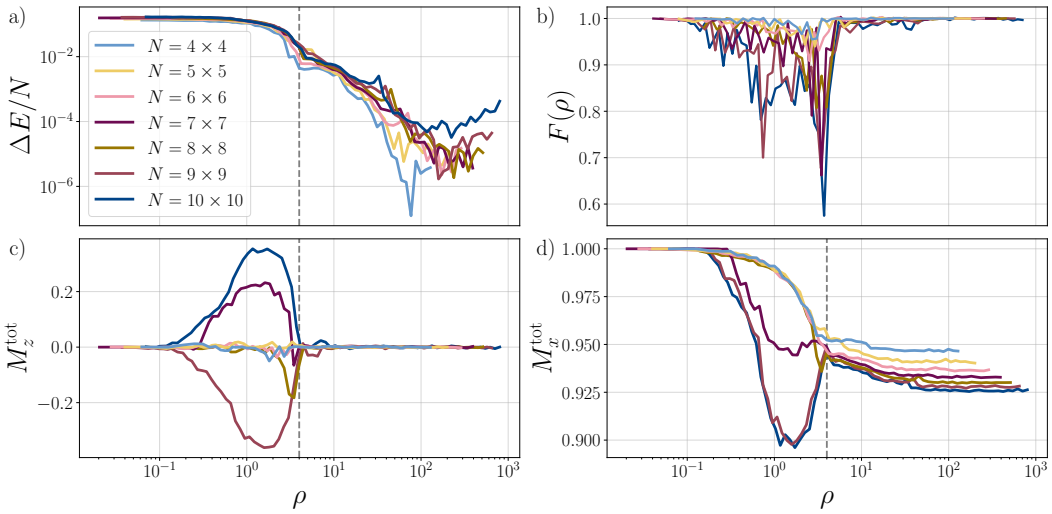

Figure 3: Evidence of a sparsity-induced quantum phase transition in shallow depth $d = 1$ FFNNs. This data corresponds to finite size scaling at the quantum critical point $\kappa = \kappa_c$ of the TFIM for $N = 4 \times 4$ through $N = 10 \times 10$ spins. We plot the (a) relative error per spin versus the number of parameters per spin (b) fidelity between neighboring pruned wave functions, (c,d) the total magnetization in the x and z directions. The phase transition is marked via a dashed line in (a,c,d).

At the same critical number of parameters ρ_c where the fidelity drops, there is a sudden change in the nature of the state, observed in the total x and z magnetization shown in 3. Here, the magnetization is given by $M_z^{\text{tot}} = \langle \sum_i \sigma_i^z \rangle_\theta / N$ (and similarly for M_x^{tot}). This sharp feature further points to the nature of the transition being first-order. We can understand this transition as a transition between a $\mathbb{Z}_2$-symmetric state (which corresponds to the symmetry of the exact ground state in a finite system) to a symmetry-broken state. No apparent phase transition occurs between the LEP and ALP regions, but the behavior within HEP itself changes drastically: The z-magnetization drops to zero as sparsity is increased. It is unclear if this is an adiabatic transition between phases or evidence for a second, possibly continuous, phase transition.

4.4 Interpretable sparse neural network solution to the toric code

We find that we can naturally obtain an efficient solution to the toric code by pruning a single-layer FFNN. In Fig. 4, the $T(\theta_{\text{init}}, m_{\text{imp}})$ and $T(\theta_{\text{rand}}, m_{\text{imp}})$ isolated training tests outperform the iteratively pruned network by around two orders of magnitude in accuracy, while the $T(\theta_{\text{init}}, m_{\text{rand}})$ benchmark reaches a high-error plateau at a much higher relative parameter count. Aligning closely to our conclusions for the TFIM, these results demonstrate that the structure of these sparse subnetworks contain a highly efficient and accurate representation of the ground state of the toric code.

Similarly to the case of the TFIM, we find evidence for a phase transition in the pruning dynamics of the toric code: the fidelity F between neural networks of slightly different sparsity drops rapidly at a critical parameter density of around $\rho_c = 16$, as shown in Fig. 4. This critical density corresponds to

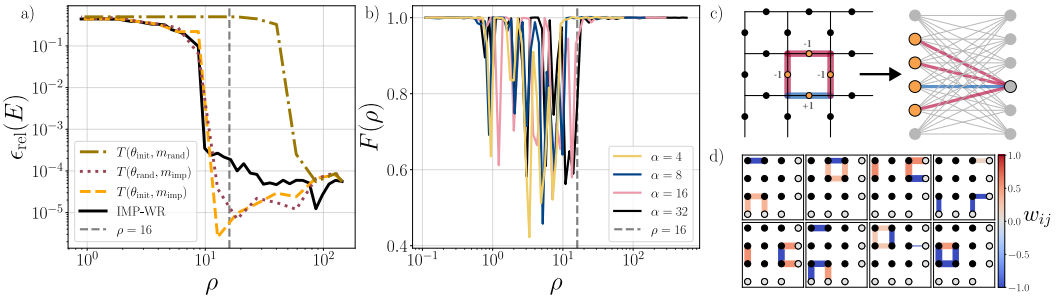

Figure 4: (a) LTH test for the toric code with $N = 18$ spins using a depth $d = 1$ FFNN of width $w = 8N$. (b) Fidelity between neighboring pruned NQS with dashed line at critical number of parameters $\rho_c = 16$. (c) Schematic of a single hidden neuron connected to a single plaquette. (d) Normalized weights and their sign configuration from eight single hidden neurons in a pruned FFNN.

Table 1: Explanation of our neural network solution to the toric code. The two rows consider configurations of spins that obey/violate the plaquette condition. The third column shows the average post-activation value of the filter convolved with the spin configuration. The last column shows the total contribution to Ψ^{NN} after all 8 orientations of the filter have been summed over.

Spin config. of $\boldsymbol{\sigma}$ of single plaquette p	In ground state	$\langle \text{ReLu}(f \cdot \boldsymbol{\sigma}) \rangle$	$\exp(\sum_f \text{ReLu}(f \cdot \boldsymbol{\sigma}))$
$(4\uparrow), (2\uparrow, 2\downarrow), (4\downarrow)$	✓	W	$\exp(8W)$
$(3\uparrow, 1\downarrow), (1\uparrow, 3\downarrow)$	✗	$0.5W$	$\exp(4W)$

16 weights per spin, or 32 weights per plaquette. We can understand this density as an asymptotically exact solution to the toric code, where the error decreases exponentially with the size of the weights.

This asymptotically exact solution consists of plaquette-shaped filters with three positive weights and one negative weight (each of magnitude W) or vice-versa as shown in Fig. 4, which we call "odd parity filters". We can interpret this structure by considering an exact toric code ground state solution, which consists of an equal superposition of all spin-configurations $|\boldsymbol{\sigma}\rangle$ that fulfill the "plaquette conditions" $\lambda_{B_p}(\boldsymbol{\sigma}) = \prod_{i \in p} \sigma_i^z = 1 \; \forall p$ [36]. In other words, to represent this ground state a neural-network solution has to fulfill $\Psi^{NN}(\boldsymbol{\sigma}) = 1$ if $\lambda_{B_p}(\boldsymbol{\sigma}) = 1 \; \forall p$, and $\Psi^{NN}(\boldsymbol{\sigma}) = 0$ else. Then, together with its activation function, an odd parity filter connected to plaquette p has the effect of exponentially amplifying the amplitude $\Psi^{NN}(\boldsymbol{\sigma})$ for spin-configurations $\boldsymbol{\sigma}$ that fulfill $\lambda_{B_p}(\boldsymbol{\sigma}) = 1$ relative to configurations with $\lambda_{B_p}(\boldsymbol{\sigma}) = -1$ (see Table 4.4). In total, the excited state contributions are suppressed relative to the ground state by a factor of $\exp(4W)$ for every broken plaquette condition. In the limit of $W \to \infty$, we thus arrive at the exact solution. Further details on the convergence of this solution over increasing magnitude weights can be found in the Appendix.

5 Conclusion

We study the dynamics of neural network pruning in a novel context, the problem of finding the ground state eigenvector of a quantum many-body system. We find that the problem and the method are symbiotic; by pruning we can discover how hard certain quantum systems are to represent, and through studying the physics of the sparse subnetwork representations we can understand the mechanisms by which neural networks make approximations.

In agreement with other pruning studies, we find that we are able to remove a large portion of neural network weights without any degradation in accuracy. Surprisingly, we find that the accuracy of models trained in isolation depends strongly on the connective structure of the weights, while the dependence on the parameter initialization is marginal. One possible explanation lies in a potentially strong connection between the sparse subnetwork connectivity and the physics of the model. Another explanation is the utilization of natural gradient descent, which can take larger steps across barren plateaus and may be able to overcome unfavorable initial parameter values.

We dig deeper into the dynamics of pruning on the transverse field Ising model, finding several fascinating features. Most saliently, at the critical point we find a first order phase transition *as a function of parameters per spin* from the critical phase to a state that breaks $\mathbb{Z}_2$ spin polarization symmetry.

Finally, we show that through pruning we are able to discover a new (asymptotically exact) solution to the toric code which is simpler than human-conceptualized solutions [11, 14, 20, 61], and does not require a priori architecture design principles. This demonstrates that pruning neural networks can elucidate how they are able to solve problems. We expect that pruning may uncover further exact solutions to other Hamiltonians, including those with topologically ordered ground states, those representing quantum error correction codes.

Understanding how to obtain a neural-network quantum state representation with minimal number of parameters may be relevant when it is beneficial to have a compressed model for further processing. For example, time evolution [58], quantum circuit simulation [42], and fine-tuning neural network quantum states [53] are all applications where an accurate compressed model could be advantageous. Furthermore, sparse network representations of quantum states may unlock the capability to use desired optimization strategies. Equipped with a pruned neural network that represents the initial state accurately, the fact that the architecture utilizes a small amount of parameters may allow for the use of SR in the $n < s$ regime where n the number of parameters and s is the number of samples. The ability to use SR may provide benefits over minSR [12], a variant of SR which is more efficient in the $n > s$ regime, but limited in terms of the number of samples one can use in practice. We leave the exploration of this idea to future studies.

Code and data availability

The code, all pretrained models, and data are available on Zenodo [1].

Acknowledgements

We would like to thank Bartholomew Andrews, Ao Chen, Anna Dawid, Matija Medvidović, Roger Melko and Schuyler Moss for useful discussions. The Center for Computational Quantum Physics at the Flatiron Institute is supported by the Simons Foundation. The computations in this work were, in part, run at facilities supported by the Scientific Computing Core at the Flatiron Institute. The Flatiron institute is a division of the Simons Foundation.

References

[1] Anonymous. Supplementary code and data: Interpretable scaling behavior in sparse subnetwork representations of quantum states: https://doi.org/10.5281/zenodo.15492729, 2025.

[2] Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27):e2311878121, 2024.

[3] Jean Barbier, Florent Krzakala, Nicolas Macris, Léo Miolane, and Lenka Zdeborová. Optimal errors and phase transitions in high-dimensional generalized linear models. *Proceedings of the National Academy of Sciences*, 116(12):5451–5460, 2019.

[4] Richard Barney, Michael Winer, and Victor Galitski. Neural networks as spin models: From glass to hidden order through training. *arXiv preprint arXiv:2408.06421*, 2024.

[5] Adriano Barra, Giuseppe Genovese, Peter Sollich, and Daniele Tantari. Phase transitions in restricted boltzmann machines with generic priors. *Physical Review E*, 96(4):042156, 2017.

[6] Henk W. J. Blöte and Youjin Deng. Cluster monte carlo simulation of the transverse ising model. *Phys. Rev. E*, 66:066110, Dec 2002.

[7] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. Jax: composable transformations of python+ numpy programs, 2018.

[8] Rebekka Burkholz, Nilanjana Laha, Rajarshi Mukherjee, and Alkis Gotovos. On the existence of universal lottery tickets. *arXiv preprint arXiv:2111.11146*, 2021.

[9] Giuseppe Carleo, Kenny Choo, Damian Hofmann, James ET Smith, Tom Westerhout, Fabien Alet, Emily J Davis, Stavros Efthymiou, Ivan Glasser, Sheng-Hsuan Lin, et al. Netket: A machine learning toolkit for many-body quantum systems. *SoftwareX*, 10:100311, 2019.

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

[11] Juan Carrasquilla and Roger G Melko. Machine learning phases of matter. *Nature Physics*, 13(5):431–434, 2017.

[12] Ao Chen and Markus Heyl. Empowering deep neural quantum states through efficient optimization. *Nature Physics*, 20(9):1476–1481, 2024.

[13] Ao Chen, Vighnesh Dattatraya Naik, and Markus Heyl. Convolutional transformer wave functions. *arXiv preprint arXiv:2503.10462*, 2025.

[14] Penghua Chen, Bowen Yan, and Shawn X Cui. Representing arbitrary ground states of the toric code by a restricted boltzmann machine. *Physical Review B*, 111(4):045101, 2025.

[15] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. The lottery ticket hypothesis for pre-trained bert networks. *Advances in neural information processing systems*, 33:15834–15846, 2020.

[16] Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A unified lottery ticket hypothesis for graph neural networks. In *International conference on machine learning*, pages 1695–1706. PMLR, 2021.

[17] Tianlong Chen, Zhenyu Zhang, Sijia Liu, Shiyu Chang, and Zhangyang Wang. Long live the lottery: The existence of winning tickets in lifelong learning. In *International Conference on Learning Representations*, 2020.

[18] Hugo Cui, Freya Behrens, Florent Krzakala, and Lenka Zdeborová. A phase transition between positional and semantic learning in a solvable model of dot-product attention. *Advances in Neural Information Processing Systems*, 37:36342–36389, 2024.

[19] Arthur da Cunha, Emanuele Natale, and Laurent Viennot. Proving the lottery ticket hypothesis for convolutional neural networks. In *International Conference on Learning Representations*, 2022.

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

[21] Zakari Denis, Alessandro Sinibaldi, and Giuseppe Carleo. Comment on" can neural quantum states learn volume-law ground states?". *arXiv preprint arXiv:2309.11534*, 2023.

[22] James Diffenderfer and Bhavya Kailkhura. Multi-prize lottery ticket hypothesis: Finding accurate binary neural networks by pruning a randomly weighted network. *arXiv preprint arXiv:2103.09377*, 2021.

[23] Damien Ferbach, Christos Tsirigotis, Gauthier Gidel, et al. A general framework for proving the equivariant strong lottery ticket hypothesis. *arXiv preprint arXiv:2206.04270*, 2022.

[24] Matthew Fishman, Steven R. White, and E. Miles Stoudenmire. The ITensor Software Library for Tensor Network Calculations. *SciPost Phys. Codebases*, page 4, 2022.

[25] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[26] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis. *arXiv preprint arXiv:1903.01611*, 2019.

[27] Advait Gadhikar and Rebekka Burkholz. Masks, signs, and learning rate rewinding. *arXiv preprint arXiv:2402.19262*, 2024.

[28] Zhe Gan, Yen-Chun Chen, Linjie Li, Tianlong Chen, Yu Cheng, Shuohang Wang, Jingjing Liu, Lijuan Wang, and Zicheng Liu. Playing lottery tickets with vision and language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 652–660, 2022.

[29] Anna Golubeva and Roger G Melko. Pruning a restricted boltzmann machine for quantum state reconstruction. *Physical Review B*, 105(12):125124, 2022.

[30] Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*, 2020.

[31] Géza Györgyi. First-order transition to perfect generalization in a neural network with binary synapses. *Physical Review A*, 41(12):7097, 1990.

[32] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[33] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

[34] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 1970.

[35] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2024.

[36] A Yu Kitaev. Fault-tolerant quantum computation by anyons. *Annals of physics*, 303(1):2–30, 2003.

[37] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[39] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[40] Antoine Maillard, Bruno Loureiro, Florent Krzakala, and Lenka Zdeborová. Phase retrieval in high dimensions: Statistical and computational phase transitions. *Advances in Neural Information Processing Systems*, 33:11071–11082, 2020.

[41] Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. Proving the lottery ticket hypothesis: Pruning is all you need. In *International Conference on Machine Learning*, pages 6682–6691. PMLR, 2020.

[42] Matija Medvidović and Giuseppe Carleo. Classical variational simulation of the quantum approximate optimization algorithm. *npj Quantum Information*, 7(1):101, 2021.

[43] Matija Medvidović and Javier Robledo Moreno. Neural-network quantum states for many-body physics. *arXiv preprint arXiv:2402.11014*, 2024.

[44] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*, 2023.

[45] Junghun Oh, Sungyong Baik, and Kyoung Mu Lee. Find a winning sign: Sign is all we need to win the lottery. *arXiv preprint arXiv:2504.05357*, 2025.

[46] Laurent Orseau, Marcus Hutter, and Omar Rivasplata. Logarithmic pruning is all you need. *Advances in Neural Information Processing Systems*, 33:2925–2934, 2020.

[47] Chae-Yeun Park and Michael J Kastoryano. Geometry of learning neural quantum states. *Physical Review Research*, 2(2):023232, 2020.

[48] Giacomo Passetti, Damian Hofmann, Pit Neitemeier, Lukas Grunwald, Michael A Sentef, and Dante M Kennes. Can neural quantum states learn volume-law ground states? *Physical Review Letters*, 131(3):036502, 2023.

[49] Ankit Pensia, Shashank Rajput, Alliot Nagle, Harit Vishwakarma, and Dimitris Papailiopoulos. Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. *Advances in neural information processing systems*, 33:2599–2610, 2020.

[50] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11893–11902, 2020.

[51] Russell Reed. Pruning algorithms-a survey. *IEEE transactions on Neural Networks*, 4(5):740–747, 1993.

[52] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

[53] Riccardo Rende, Sebastian Goldt, Federico Becca, and Luciano Loris Viteritti. Fine-tuning neural network quantum states. *Physical Review Research*, 6(4):043280, 2024.

[54] Jonathan S Rosenfeld, Jonathan Frankle, Michael Carbin, and Nir Shavit. On the predictability of pruning across scales. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9075–9083. PMLR, 18–24 Jul 2021.

[55] Pedro Savarese, Hugo Silva, and Michael Maire. Winning the lottery with continuous sparsification. *Advances in neural information processing systems*, 33:11380–11390, 2020.

[56] Henry Schwarze. Learning a rule in a multilayer neural network. *Journal of Physics A: Mathematical and General*, 26(21):5781, 1993.

[57] Dan Sehayek, Anna Golubeva, Michael S Albergo, Bohdan Kulchytskyy, Giacomo Torlai, and Roger G Melko. Learnability scaling of quantum states: Restricted boltzmann machines. *Physical Review B*, 100(19):195125, 2019.

[58] Alessandro Sinibaldi, Clemens Giuliani, Giuseppe Carleo, and Filippo Vicentini. Unbiasing time-dependent variational monte carlo by projected quantum evolution. *Quantum*, 7:1131, 2023.

[59] Haim Sompolinsky, Naftali Tishby, and H Sebastian Seung. Learning from examples in large neural networks. *Physical Review Letters*, 65(13):1683, 1990.

[60] Sandro Sorella. Green function monte carlo with stochastic reconfiguration. *Physical review letters*, 80(20):4558, 1998.

[61] Agnes Valenti, Eliska Greplova, Netanel H Lindner, and Sebastian D Huber. Correlation-enhanced neural networks as interpretable variational quantum states. *Physical Review Research*, 4(1):L012010, 2022.

[62] Filippo Vicentini, Damian Hofmann, Attila Szabó, Dian Wu, Christopher Roth, Clemens Giuliani, Gabriel Pescia, Jannes Nys, Vladimir Vargas-Calderón, Nikita Astrakhantsev, et al. Netket 3: Machine learning toolbox for many-body quantum systems. *SciPost Physics Codebases*, page 007, 2022.

[63] Mike Winer and Boris Hanin. Deep nets as hamiltonians. *arXiv preprint arXiv:2503.23982*, 2025.

[64] Sangyeop Yeo, Yoojin Jang, Jy-yong Sohn, Dongyoon Han, and Jaejun Yoo. Can we find strong lottery tickets in generative models? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 3267–3275, 2023.

[65] Haonan Yu, Sergey Edunov, Yuandong Tian, and Ari S Morcos. Playing the lottery with rewards and multiple languages: lottery tickets in rl and nlp. *arXiv preprint arXiv:1906.02768*, 2019.

[66] Guibin Zhang, Kun Wang, Wei Huang, Yanwei Yue, Yang Wang, Roger Zimmermann, Aojun Zhou, Dawei Cheng, Jin Zeng, and Yuxuan Liang. Graph lottery ticket automated. In *The Twelfth International Conference on Learning Representations*, 2024.

[67] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *Advances in neural information processing systems*, 32, 2019.

A Pruning procedures

Algorithm 1 Iterative Magnitude Pruning with Weight Rewinding (IMP-WR) [25]

Input: A randomly initialized dense network $f(\boldsymbol{\sigma}; \theta_{\text{init}} \odot m)$ with binary mask $m = \{1\}^{|\theta_{\text{init}}|}$.
Output: A sparse network $f(\boldsymbol{\sigma}; \theta' \odot m')$.
Train the network for j steps. {Pre-training phase}
Save the parameters θ_{wr} for rewinding.
for each pruning iteration in I **do**
  Select $p_{\text{r}}\%$ of weights (among unmasked) with lowest magnitude.
  Set corresponding entries in the mask m to zero.
  Reset unmasked weights to their values in θ_{wr}. {Weight rewinding}
  Train the network for k steps.
end for

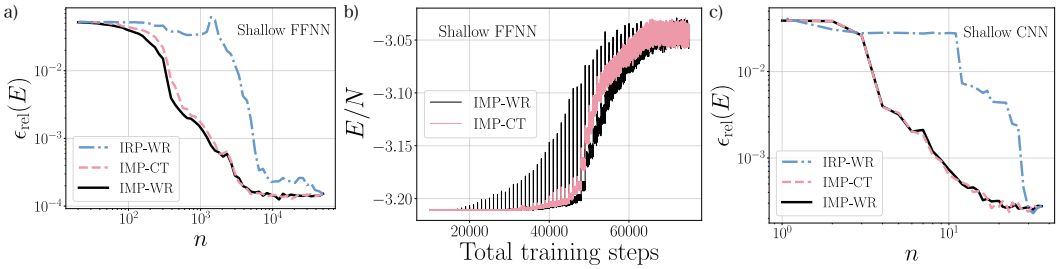

Figure 5: Comparison of iterative pruning variants: weight rewinding (IMP-WR), continued training (IMP-CT), and random pruning with weight rewinding (IRP-WR). This data corresponds to the TFIM on a $N = 10 \times 10$ lattice at the critical point ($\kappa = \kappa_c$). We plot the relative error of these three variants starting from the same dense (a) FFNN ($d = 1, w = 5N$), and (c) CNN ($d = 1, n_f = 4$). In (b), we show the energy per spin over the total training steps in 65 pruning iterations from the shallow FFNN in (a).

Comparison of IMP-WR to other pruning procedures In this work, we use Iterative Magnitude Pruning with Weight Rewinding (IMP-WR) as described in Algorithm 1 and Section 3.3 in the main text. We compare this baseline with two iterative pruning variants:

- **IMP-CT (Continued Training):** Follows the same procedure as IMP-WR, except that after each pruning step, the remaining weights are not rewound to their initial values. Instead, training continues from their current values. This approach is often referred to as *fine-tuning*.
- **IRP-WR (Iterative Random Pruning with Weight Rewinding):** Also mirrors the IMP-WR procedure, but replaces magnitude-based pruning with uniform random selection of weights to prune. This variant serves as a baseline to assess the efficacy of magnitude-based criteria.

For shallow FFNNs, we find that IMP-WR converges to a lower energy minima at each pruning iteration, compared to IMP-CT as shown in Fig. 5, in agreement with previous work comparing weight rewinding and continued training [52]. Despite the large perturbation of the network due to rewinding the weights shown in Fig. 5, the network is able to recover to a lower energy with few training steps at each pruning iteration. Conversely, continued training does not induce such a large perturbation, and the network retains an energy close to the ground state, yet these minima are higher in energy than those of IMP-WR. Compared to the baseline of IRP-WR, we find that networks pruned on magnitude based criteria in IMP-WR and IMP-CT drastically outperform those pruned randomly with IRP-WR. This result further demonstrates that magnitude-based pruning criteria finds a non-trivial structure in the sparse subnetworks, that achieves significantly better than a random structure. The differences between weight rewinding and continued training are not as distinct for shallow CNNs compared to shallow FFNNs, shown in Fig. 5, which is likely due to the very number of weights in the model from initialization.

B Scaling behavior in the variance of the energy

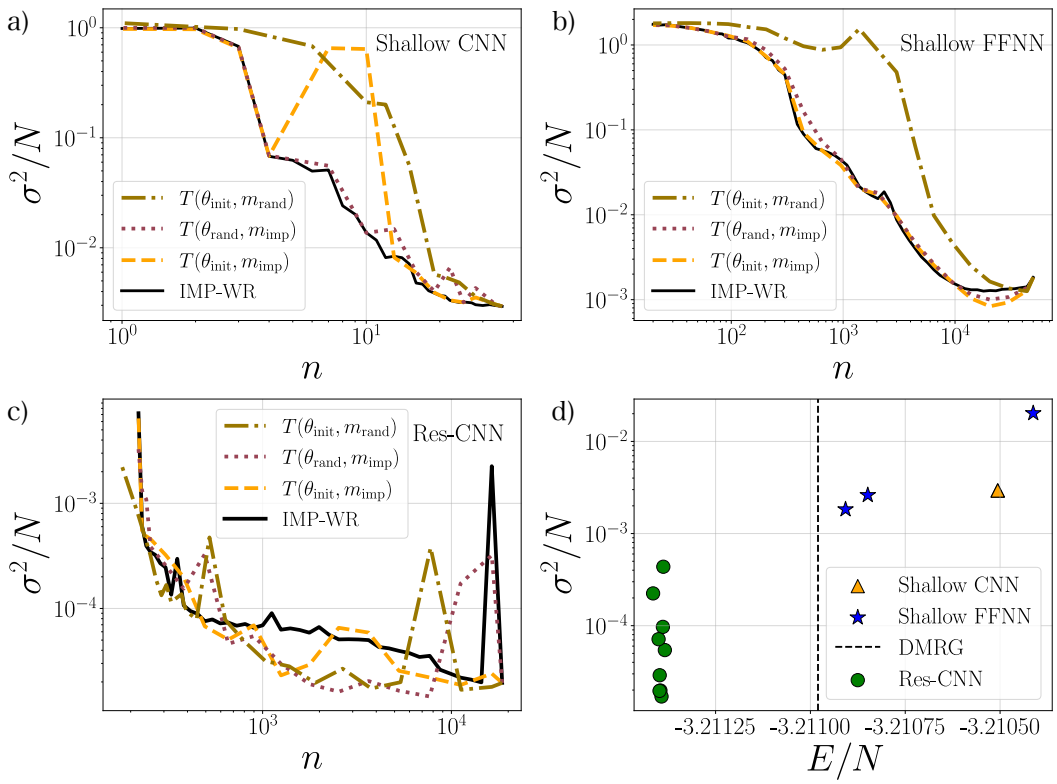

Figure 6: Variance of the energy per spin σ^2/N over remaining parameters n corresponding to the data in Fig. 1 in the main text, for the $N = 10 \times 10$ TFIM at the critical point ($\kappa = \kappa_c$). We include IMP-WR and isolated training variants from various networks: (a) shallow FFNN, (b) shallow CNN, and (c) ResCNN. In (d), we show extended data from the inset in Fig. 1 for theenergy variance per spin over the energy per spin E/N associated to various dense initializations of single layer FFNNs varying in width $w = \alpha N; \alpha = \{1, 5/2, 5\}$, single layer CNNs varying in width by the number of filter $n_f = \{4, 8, 16\}$, and ResCNNs varying in depth and width. In (d), we include the energy per spin obtained from DMRG with bond dimension $\chi = 3000$ as a vertical dashed line.

In this section, we report the energy variance per spin for the iteratively pruned and isolated trained networks presented in the main text. We define the energy variance as

$$\sigma^2 = \langle \hat{H}^2 \rangle - \langle \hat{H} \rangle^2 \tag{8}$$

which is estimated as the sample variance of the local energy defined in Eq. 18. This quantity controls the statistical uncertainty σ_E in the estimated energy:

$$\sigma_E = \sqrt{\frac{\sigma^2}{N_{\text{s}}}}, \tag{9}$$

where N_s is the number of samples. As a result, the energy variance provides a meaningful measure of the magnitude of fluctuations in $\langle \hat{H} \rangle$ and the quality of the ground-state approximation, as it vanishes when the variational wavefunction is an exact eigenstate and governs the statistical error in energy estimates obtained via Monte Carlo sampling.

We find that the scaling behavior of the energy variance follows the same scaling behavior of the relative error of the energy. This demonstrates that our conclusions drawn from the scaling of relative error are well supported, and not simply due to fluctuations in the sampled energy estimates.

Table 2: Robustness of m_{imp} mask structure at different iterations of pruning for the $N = 10 \times 10$ TFIM (corresponding to panel (b) of Fig. 1), and $N = 18$ toric code (corresponding to panel (a) of Fig. 4) Hamiltonians. We report the original relative energy error for θ_{init} and the mean relative energy error over 5 different initializations for θ_{rand}, in addition to their 99.9% confidence interval.

Hamiltonian	Pruning iteration	Initialization type θ_{init}	θ_{rand}	99.9% CI
TFIM	$i = 12$	1.54e-4	1.50e-4	[1.44e-4, 1.56e-4]
	$i = 15$	1.65e-4	1.67e-4	[1.58e-4, 1.77e-4]
	$i = 18$	1.96e-4	1.99e-4	[1.50e-4, 2.19e-4]
TC	$i = 9$	2.94e-5	1.83e-5	[0, 4.55e-5]
	$i = 12$	1.64e-5	1.52e-5	[0, 3.45e-5]
	$i = 15$	9.71e-6	1.60e-5	[0, 4.30e-5]

C Robustness over many random initializations

To substantiate the claim of the importance of the structure in the masks, and the relative unimportance of parameter initialization, we test the parameter sensitivity in the reduced model dimension for the $T(\theta_{\text{rand}}, m_{\text{imp}})$ isolated training tests, starting from many random initializations. We confirm the robustness of the subnetwork structure by reporting the mean of five additional random instances of the isolated training variant at several pruning iterations and their 99.9% confidence interval shown in Table 2. All random initializations remain close to the θ_{init} initialization and are in agreement at the 99.9% level. This confirms the robustness of the mask structure, and demonstrates that under many random initializations, a good mask is sufficient to obtain a matching error to the lottery ticket initialization.

D Sparse neural network solution to the toric code

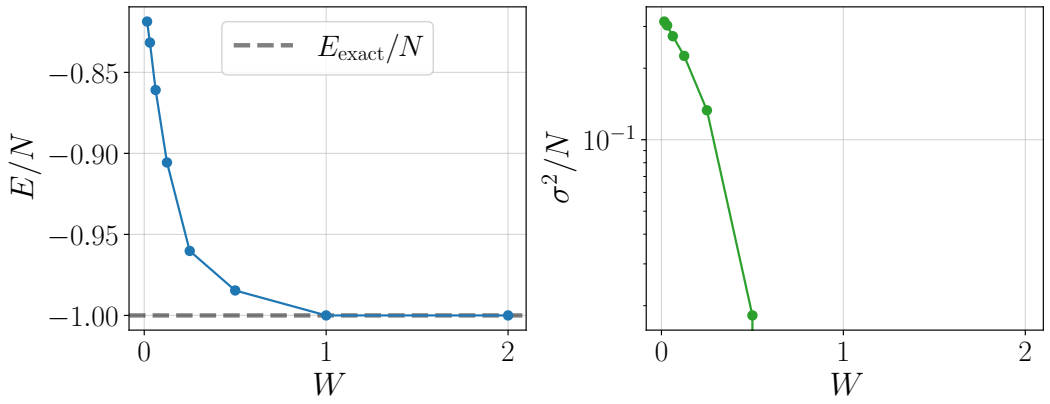

Figure 7: Exponential convergence to the ground state of the toric code, driven by the magnitude of the weights $W = |w_{ij}|$ in odd-parity filters within single hidden neurons of $d = 1, w = 4N$ FFNN. In (a), we plot E/N, the energy per spin from *untrained* sparse initialized networks with odd-parity single hidden neuron filters. In (b), we plot the energy variance, which abruptly goes to zero, signifying an exact solution to the toric code.

In this section, we further detail the sparse FFNN solution to the toric code. In Section 4.4 of the main text, we provide a single hidden neuron analysis for a single plaquette spin configuration. We show that odd-parity filters exponentially amplify the wavefunction coefficients of ground state configurations by the magnitude of positive and negative weights in odd-parity filters.

In Fig. 7, we demonstrate this exponential convergence by computing the energy from a sparse initialized networks with odd-parity filter structure. When the magnitude of the weights increase, we find that the energy converges to the exact ground state energy, with zero variance.

E Neural network architectures

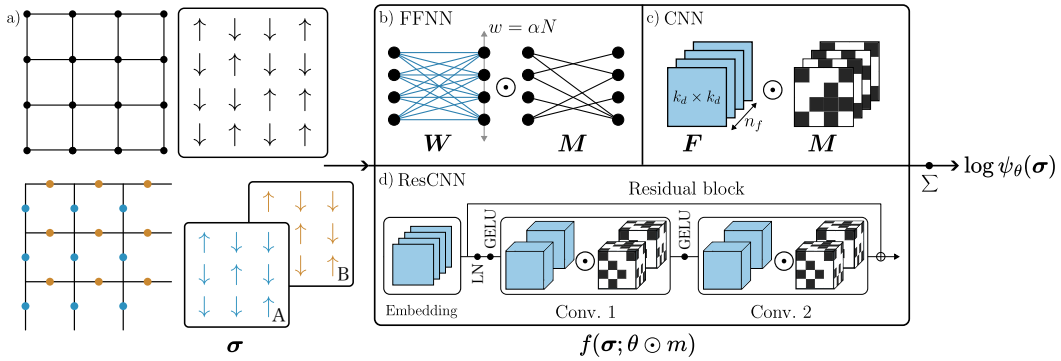

Figure 8: Schematic of neural network architectures from (a) input spin configurations $\boldsymbol{\sigma}$, to (b,c,d) neural network ansatz $f(\boldsymbol{\sigma}; \theta \odot m)$, which outputs wavefunction coefficients $\log \psi_\theta(\boldsymbol{\sigma})$. In (a), we show the input configurations of the (top) TFIM and (bottom) A and B sublattices of toric code, which are all flattened upon input to FFNNs. We detail the masking procedure in (a) single layer FFNN, (b) single layer CNN, and (c) ResCNN with a single residual block. Each residual block in (c) consists of a layer normalization (LN), GELU activation, convolutional layer (Conv. 1), GELU activation, convolutional layer (Conv. 2), and residual connection. The outputs of the last layer are summed, to output the logarithm of the wave function coefficient.

Feed-forward neural networks Throughout this work, we use a shallow feed-forward neural network (FFNN) architecture of constant depth $d = 1$, varying in width $w = \alpha N$, where α is a scaling factor of the input dimension. We choose to scale the size of the networks with the system-size, such that the input dimension N denotes the number of spins, and the latent dimension is scaled by $\alpha \in \mathbb{R}$. Therefore, the set of parameters θ are simply defined by a synaptic weight matrix $\boldsymbol{W} \in \mathbb{R}^{N \times \alpha N}$. We initialize $\boldsymbol{W}$ from a normal distribution with a standard deviation $\sigma = 0.1$. The network has bias terms $\boldsymbol{b}$, pre-activations $\boldsymbol{h}$, and post-activations $\tilde{\boldsymbol{\sigma}}$. The forward propagation dynamics of the neural network are given by

$$\tilde{\boldsymbol{\sigma}} = \phi(\boldsymbol{h}), \quad \boldsymbol{h} = \boldsymbol{W}\boldsymbol{\sigma} + \boldsymbol{b}, \tag{10}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function, and the input to the network is $\boldsymbol{\sigma} \in \mathbb{R}^N$. We use the Rectified Linear Unit (ReLU) activation function throughout this work. For the simplicity of interpreting weights without the effect of bias, we choose to remove the bias terms, effectively setting $\boldsymbol{b} = 0$ in Eq. 10. The removal of bias terms produced no noticeable differences in network performance. The post activations are then summed to produce the logarithm of the wave function coefficients, therefore giving the exponential ansatz defined by

$$\psi(\boldsymbol{\sigma}) = \exp\left[\sum_{i=1}^{\alpha N} \tilde{\boldsymbol{\sigma}}_i\right]. \tag{11}$$

The total number of parameters in the network at dense initialization is $n_{\text{init}} = \alpha N^2$. Thus, the number of parameters in this shallow FFNN architecture scales as $\mathcal{O}(N^2)$, which is quadratic in the number of spins.

Shallow convolutional neural networks We use a shallow convolutional neural-network (CNN) architecture of constant $d = 1$ depth, varying in width by n_f, the number of convolutional kernels. The forward propagation dynamics of the neural network are given by

$$\tilde{\boldsymbol{\sigma}} = \phi(\boldsymbol{h}), \quad \boldsymbol{h} = \boldsymbol{F} * \boldsymbol{\sigma} + \boldsymbol{b}, \tag{12}$$

where $\boldsymbol{F} * \boldsymbol{\sigma}$ denotes the convolution operation of n_f kernels $f \in \boldsymbol{F}$. The kernels are convolved with VALID padding [35] on the two-dimensional input $\boldsymbol{\sigma}$. The kernels $f \in \mathbb{R}^{k_d \times k_d}$ have dimension $k_d = 3$, which we initialize with a truncated normal distribution (LeCun normal) [37]. Similar to the FFNN architecture, we set the bias terms to zero. The activation ϕ we use is a Gaussian Error Linear Unit (GELU). The post activations are summed in a similar manner to Eq. 11 giving

$$\psi(\boldsymbol{\sigma}) = \exp\left[\sum_{i=1}^{N} \tilde{\boldsymbol{\sigma}}_i\right]. \tag{13}$$

The total number of parameters of a CNN at dense initialization is therefore $n_{\text{init}} = n_f k_d^2$, which scales as a constant with the system size N, due to the nature of weight sharing in the convolution operation.

Residual convolutional neural networks We use a residual layer convolutional neural network (ResCNN) introduced in [12, 13] that vary by (width) the number convolutional kernels n_f, and (depth) the number of residual blocks n_b. The ResCNN architecture consists of an initial convolutional embedding layer, followed by pre-activation residual blocks with two convolutional layers per block. We detail the structure of the ResCNN in Fig. 8. The embedding layer of the CNN produces the input to the residual blocks

$$\boldsymbol{h}^1 = \boldsymbol{F}_{\text{embed}} * \boldsymbol{\sigma}, \tag{14}$$

where $\boldsymbol{F}_{\text{embed}} \in \mathbb{R}^{n_f \times k_d \times k_d}$ is the embedding convolution. We use no biases for the ResCNN architecture, similar to the FFNN and CNN architectures. The forward propagation dynamics of the residual blocks are given by

$$\boldsymbol{h}^{i+1} = \left(\boldsymbol{F}_2 * \phi\left\{\boldsymbol{F}_1 * \text{LN}\left[\phi(\boldsymbol{h}^i)\right]\right\}\right) + \boldsymbol{h}^i, \quad i = 1, \ldots, n_b. \tag{15}$$

Here, LN denotes a layer normalization, ϕ is the GELU activation, and $\boldsymbol{F}_1, \boldsymbol{F}_2 \in \mathbb{R}^{n_f \times n_f \times k_d \times k_d}$ denote two convolutional layers (Conv. 1 and Conv.2) in each residual block. We use a custom layer normalization that computes the mean and variance only on non-zero inputs, due to the sparsity induced from pruning. We found that masking zero inputs in the layer normalization produced more stable training dynamics. All convolutional layers are initialized from a truncated normal distribution (LeCun normal) [37]. Upon obtaining $\boldsymbol{h}^{n_b}$ from the last residual block, we apply a final layer normalization to obtain $\tilde{\boldsymbol{\sigma}} = \text{LN}\left[\boldsymbol{h}^{n_b}\right]$, and sum the outputs to produce the wave function coefficients using Eq. 13.

The total number of parameters in a ResCNN is defined by $n_{\text{init}} = n_{\text{embed}} + n_b n_f^2 k_d^2$, where the $n_{\text{embed}} = n_f k_d^2$ denotes the number of parameters in the embedding layer. We do not prune the embedding layer, and prune the residual blocks in a global unstructured fashion.

F Sampling and optimization procedures

Monte Carlo sampling To estimate expectation values of observables such as the energy or its gradient, we sample configurations $\boldsymbol{\sigma}$ from the probability distribution $|\psi_\theta(\boldsymbol{\sigma})|^2$ using the Metropolis-Hastings algorithm [34]. New configurations $\boldsymbol{\sigma}'$ are proposed via local update rules and accepted with probability

$$P(\boldsymbol{\sigma} \to \boldsymbol{\sigma}') = \min\left(1, \frac{|\psi_\theta(\boldsymbol{\sigma}')|^2}{|\psi_\theta(\boldsymbol{\sigma})|^2}\right). \tag{16}$$

Provided the proposal distribution is symmetric, this procedure yields samples from the desired distribution after thermalization.

The choice of proposal distribution depends on the physical model. For the transverse field Ising model, we use single-spin flips. For the toric code, where single flips have vanishing amplitude in the ground state sector, we instead propose updates from two rules with equal probability: single-spin flips and *plaquette flips*, where all four spins on a plaquette are flipped simultaneously. This ensures efficient exploration of relevant ground state configurations.

Efficient computation of expectation values Expectation values of an operator $\hat{O}$ are then estimated by

$$\langle \hat{O} \rangle \approx \frac{1}{N_s} \sum_{i=1}^{N_s} O_{\text{loc}}(\boldsymbol{\sigma}_i), \tag{17}$$

as the sample mean over N_s configurations drawn from $|\Psi_\theta^{NN}(\boldsymbol{\sigma})|^2$. Here,

$$O_{\text{loc}}(\boldsymbol{\sigma}) = \frac{\langle \boldsymbol{\sigma} | \hat{O} | \Psi_\theta^{NN} \rangle}{\langle \boldsymbol{\sigma} | \Psi_\theta^{NN} \rangle}, \tag{18}$$

is a local expectation value, $|\Psi_\theta^{NN}\rangle$ is the neural network quantum state, and $|\boldsymbol{\sigma}\rangle$ denotes a basis configuration.

Stochastic reconfiguration We optimize neural network quantum states using stochastic reconfiguration (SR) [60], a technique equivalent to natural gradient descent in the variational parameter space. This method accounts for the geometry of the Hilbert space by introducing a metric tensor, ensuring more stable and efficient optimization trajectories compared to standard gradient descent.

Given a wavefunction $|\Psi_\theta\rangle$ parametrized by θ, the SR update rule is

$$\theta \to \theta - \eta \sum_{\theta'} S_{\theta\theta'}^{-1} \frac{\partial \langle \hat{H} \rangle_\theta}{\partial \theta'}, \tag{19}$$

where η is the learning rate (LR), $\langle \hat{H} \rangle_\theta = \langle \Psi_\theta | \hat{H} | \Psi_\theta \rangle / \langle \Psi_\theta | \Psi_\theta \rangle$ is the variational energy, and $S_{\theta\theta'}$ is the quantum Fisher information matrix (or S-matrix), defined as

$$S_{\theta\theta'} = \langle O_\theta^* O_{\theta'} \rangle - \langle O_\theta^* \rangle \langle O_{\theta'} \rangle, \tag{20}$$

with $O_\theta(\boldsymbol{\sigma}) = \partial \log \Psi_\theta(\boldsymbol{\sigma}) / \partial \theta$ denoting the log-derivatives of the wavefunction and expectations taken over the distribution of $|\Psi_\theta(\boldsymbol{\sigma})|^2$. The gradient of the energy with respect to the parameters is given by:

$$\frac{\partial \langle \hat{H} \rangle_\theta}{\partial \theta} = 2\,\text{Re}\left[\langle O_\theta^* E_{\text{loc}} \rangle - \langle O_\theta^* \rangle \langle E_{\text{loc}} \rangle \right], \tag{21}$$

where the local energy is defined by Eq. 18. To ensure that the inverse of the S-matrix in Eq. 20 is well defined, we use an explicit regularization $S = S + \lambda I$, where λ is called the diagonal shift (DS). We keep the learning rate and diagonal shift constant throughout this work, and report the values we use in Section G.

The SR method can also be derived as an approximation to imaginary-time evolution, $|\Psi_{t+1}\rangle \approx \exp(-\eta \hat{H}) |\Psi_t\rangle$, where η plays the role of an imaginary-time step. For further details on this connection, see [60, 47, 43].

G Resources for reproducability

Hyperparameters and training settings All symbols, acronyms, and hyperparameters used in this work are reported in Table 3. We report the all hyperparameters and training settings used across all experiments in Tables 4, 5, 6, and 7 to ensure reproducibility and facilitate comparison in future work.

Implementation details for density matrix renormalization group To benchmark the quality of expected observables calculated with neural network wave functions, we used DMRG to find the ground state wave function for the TFIM. DMRG optimizes an MPS to find the lowest eigenvector of the Hamiltonian H_{TFIM}, and the corresponding energy eigenvalue. We implemented DMRG with the ITensor julia package [24]. The hyperparameters we used are reported in Table 8.

Computational resources for reproducibility All neural network training was performed on up to 30 NVIDIA A100 graphical processing units (GPUs), with 80 gigabytes (GB) of memory each, housed on an internal cluster. Iterative pruning and training as outlined in Algorithm 1 was performed in serial on a single GPU, while isolated sparse training variants were carried out on multiple GPUs

in parallel. The time of execution for each experiment is outline in Table 9 We used an approximate total of 3200 GPU hours in this work, including 2200 GPU hours during testing, and 1000 GPU hours for final experiments. The DMRG algorithm was run on a single CPU with 80GB of memory, for a total of 30 hours.

Table 3: Symbols, acronyms, and hyperparameters used throughout this work.

Symbol	Description	Applies to
$\boldsymbol{\sigma}$	Spin configuration	-
N	Number of spins	-
E	Variational energy	-
Δ	Absolute error	-
ϵ_{rel}	Relative error	-
σ^2	Energy variance	-
θ	Neural network parameters	-
w	Individual neural network weight	-
W	Magnitude of a neural network weight	-
Ψ_θ^{NN}	Neural network quantum state	-
F	Monte Carlo fidelity	-
$M_{x/z}^{\text{tot}}$	Magnetization in the x or z directions	-
T	A sparse neural network trained in isolation	-
θ_{init}	Initialized parameters from IMP-WR	-
θ_{rand}	Randomly initialized parameters	-
m_{imp}	Sparse mask obtained from from IMP-WR	-
m_{rand}	Random sparse mask	-
n	Number of remaining parameters	-
n_{init}	Number of initial parameters	-
ρ	Number of parameters per spin	-
$\sigma^{x/z}$	Pauli matrices	-
κ	Transverse field strength	-
TFIM	Transverse field Ising model	-
TC	Toric code	-
LTH	Lottery ticket hypothesis	-
DMRG	Density matrix renormalization group	-
FFNN	Feed forward neural-network	-
CNN	Convolutional neural-network	-
ResCNN	Residual convolutional neural-network	-
IMP-WR	Iterative magnitude pruning with weight rewinding	-
IMP-CT	Iterative magnitude pruning with continued training	-
IRP-WR	Iterative random pruning with weight rewinding	-
LEP	Low-error plateau	-
APL	Approximate power law	-
HEP	High-error plateau	-
$\boldsymbol{W}$	Weight matrix	FFNN
$\boldsymbol{b}$	Bias vector	FFNN
w	Network width	FFNN
α	Width scaling factor	FFNN
d	Network depth	FFNN, CNN
f	Individual convolutional kernel	CNN, ResCNN
$\boldsymbol{F}$	Convolutional kernel	CNN, ResCNN
n_f	Number of kernels (width)	CNN, ResCNN
k_d	Kernel dimension	CNN, ResCNN
n_b	Number of residual blocks (depth)	ResCNN
n_{embed}	Number of parameters in embedding layer	ResCNN
LN	Layer normalization	ResCNN
j	Number of pre-training steps	All
k	Number of training steps in each pruning iteration	All
p_r	Pruning ratio	All
I	Total number of pruning iterations	All
n_{init}	Number of parameters at dense initialization	All
N_s	Number of samples per optimization step	All
η	Learning rate (LR)	All
λ	Diagonal shift (DS) in stochastic reconfiguration	All

Table 4: Hyperparameters used in Fig. 1 in the main text, for each architecture in panels (a) CNN, (b) FFNN, and (c) ResCNN. Definitions of all symbols are provided in Table 3 in the appendix.

NETWORK	w/n_f	d/n_b	k_d	j	k	p_r	I	n_{init}	N_{s}	η	λ
A) CNN	$n_f = 4$	$d = 1$	3	1E4	1E3	0.05	31	36	1024	8E-3	1E-3
B) FFNN	$w = 5N$	$d = 1$	–	1E4	1E3	0.12	65	50000	1024	8E-3	1E-4
C) RESCNN	$n_f = 16$	$n_b = 4$	3	1E4	1E3	0.12	43	18576	1024	8E-3	1E-3

Table 5: Hyperparameters for Fig. 2 in the main text, panels (a,b,c). Definitions of all symbols are provided in Table 3 in the appendix.

PANEL	κ	w	d	j	k	p_r	I	n_{init}	N_{s}	η	λ
(A)	3.04438	N	1	1E4	1E3	0.12	54	10000	1024	8E-3	1E-4
(A)	3.04438	$2.5N$	1	1E4	1E3	0.12	60	25000	1024	8E-3	1E-4
(A)	3.04438	$5N$	1	1E4	1E3	0.12	65	50000	1024	8E-3	1E-4
(B)	$\{0.1, 1, 2\}$	$5N$	1	1E4	1E3	0.12	65	50000	1024	8E-3	1E-4
(C)	$\{4, 5, 6\}$	$5N$	1	1E4	1E3	0.12	65	50000	1024	8E-3	1E-4

Table 6: Hyperparameters for Fig. 3 in the main text, panel (a). Definitions of all symbols are provided in Table 3 in the appendix.

SYSTEM SIZE N	w	d	j	k	p_r	I	n_{init}	N_{s}	η	λ
4×4	$8N$	1	1E4	1E3	0.12	51	2048	1024	8E-3	1E-4
5×5	$8N$	1	1E4	1E3	0.12	58	5000	1024	8E-3	1E-4
6×6	$8N$	1	1E4	1E3	0.12	64	10368	1024	8E-3	1E-4
7×7	$8N$	1	1E4	1E3	0.12	69	19208	1024	8E-3	1E-4
8×8	$8N$	1	1E4	1E3	0.12	65	32768	1024	8E-3	1E-4
9×9	$8N$	1	1E4	1E3	0.12	74	52488	1024	8E-3	1E-4
10×10	$8N$	1	1E4	1E3	0.12	74	80000	1024	8E-3	1E-4

Table 7: Hyperparameters for Fig. 4 in the main text, panels (a,b). Definitions of all symbols are provided in Table 3 in the appendix.

PANEL(S)	w	d	j	k	p_r	I	n_{init}	N_{s}	η	λ
(A,B)	$4N$	1	1E4	1E3	0.12	47	1296	1024	8E-3	1E-3
(B)	$8N$	1	1E4	1E3	0.12	53	2592	1024	8E-3	1E-3
(B)	$16N$	1	1E4	1E3	0.12	58	5184	1024	8E-3	1E-3
(B)	$32N$	1	1E4	1E3	0.12	64	10368	1024	8E-3	1E-3

Table 8: DMRG parameters used for calculating reference observable quantities in the two-dimensional TFIM. Each sweep is bound to a maximum bond dimension $\chi_{\max}$, which increases at each sweep. The cutoff defines the truncation error in the MPS, which is constant for each sweep.

SWEEPS	MAXIMUM BOND DIMENSION $\chi_{\max}$	CUT OFF
12	$10 - 3000$	1E-15

Table 9: Execution times (in GPU hrs) rounded up to the nearest integer for iterative pruning (IP) variants, and isolated training (IT) variants for different network architectures varying in dimension, across different system sizes of the TFIM, and toric code Hamiltonians. The execution times for isolated training variants correspond to training a single network.

HAMILTONIAN	SYSTEM SIZE	NETWORK	NETWORK DIM.	IP (GPU HRS)	IT (GPU HRS)
TFIM	$N = 4 \times 4$	FFNN	$d = 1, w = 8N$	4	1
TFIM	$N = 5 \times 5$	FFNN	$d = 1, w = 8N$	5	1
TFIM	$N = 6 \times 6$	FFNN	$d = 1, w = 8N$	6	1
TFIM	$N = 7 \times 7$	FFNN	$d = 1, w = 8N$	10	2
TFIM	$N = 8 \times 8$	FFNN	$d = 1, w = 8N$	13	2
TFIM	$N = 9 \times 9$	FFNN	$d = 1, w = 8N$	16	2
TFIM	$N = 10 \times 10$	FFNN	$d = 1, w = N$	20	3
TFIM	$N = 10 \times 10$	FFNN	$d = 1, w = 2.5N$	21	3
TFIM	$N = 10 \times 10$	FFNN	$d = 1, w = 5N$	22	3
TFIM	$N = 10 \times 10$	FFNN	$d = 1, w = 8N$	23	3
TFIM	$N = 10 \times 10$	CNN	$d = 1, n_f = 4$	12	3
TFIM	$N = 10 \times 10$	RESCNN	$n_b = 4, n_f = 16$	28	5
TORIC CODE	$N = 18$	FFNN	$d = 1, w = 4N$	1	1
TORIC CODE	$N = 18$	FFNN	$d = 1, w = 8N$	1	1
TORIC CODE	$N = 18$	FFNN	$d = 1, w = 16N$	1	1
TORIC CODE	$N = 18$	FFNN	$d = 1, w = 32N$	1	1