Supplementary Material for: Can neural quantum states learn volume-law ground states?

Giacomo Passetti,[1] Damian Hofmann,[2] Pit Neitemeier,[1]
Lukas Grunwald,[1, 2] Michael A. Sentef,[3, 2] and Dante M. Kennes[1, 2]

[1] *Institut für Theorie der Statistischen Physik, RWTH Aachen University and JARA-Fundamentals of Future Information Technology, 52056 Aachen, Germany*
[2] *Max Planck Institute for the Structure and Dynamics of Matter, Center for Free-Electron Laser Science (CFEL), Luruper Chaussee 149, 22761 Hamburg, Germany*
[3] *H H Wills Physics Laboratory, University of Bristol, Bristol BS8 1TL, United Kingdom*

I. NETWORK ARCHITECTURE

A fully-connected feed-forward network (FFNN) is a composition of $\mu \in \mathbb{N}_{>0}$ layers where the lth network layer, $1 \le l \le \mu$, applies the transformation (writing $[n] := \{1, \dots, n\}$ for any $n \in \mathbb{N}_{>0}$)

$$z_i^{(l)} = \sum_j W_{ij}^{(l)} y_j^{(l)} + b_i^{(l)}, \qquad y_i^{(l+1)} = \phi(z_i^{(l)})$$
$$\begin{cases} i \in [L],\, j \in [\alpha L], & l = 1, \\ i, j \in [\alpha L], & l > 1, \end{cases} \tag{S1}$$

from input $y^{(l)}$ to output signal $y^{(l+1)}$. This is an affine transformation with weight matrix (or kernel) $W^{(l)}$, bias $b^{(l)}$, and a nonlinear activation function ϕ. The full set of network parameters, consisting of all weights and biases, is denoted by $\theta \in \mathbb{C}^{N_{\text{par}}}$. For the results presented in the main text, we have used the SELU activation function

$$\text{SELU}(x) = \lambda \begin{cases} x, & x > 0, \\ (a-1)e^x, & x \le 0 \end{cases} \tag{S2}$$

(where $\lambda \approx 1.0507$ and $a \approx 1.6733$), as implemented in JAX [1], which helps circumvent the vanishing gradient problem of training deep networks [2]. Since in our case the network acts on complex-valued signals, we apply SELU separately to real and imaginary part of the input (`reim_selu` in NETKET [3]), giving the complex activation

$$\phi(z) = \text{SELU}(\operatorname{Re} z) + i\,\text{SELU}(\operatorname{Im} z). \tag{S3}$$

Using the hyperbolic tangent as activation provides equivalent or (for deep networks) worse results, as we discuss below (Section III D). The log-probability amplitudes of the variational state are then obtained by performing the logsumexp operation on the output of the final layer, i.e.,

$$\log \langle x|\psi_\theta\rangle = \log\left(\sum_{i=1}^{\alpha L} \exp(y_i^{(\mu+1)})\right), \qquad y_i^{(1)} = x_i, \tag{S4}$$

with $\{|x\rangle \mid x \in \{0,1\}^L\}$ denoting the computational basis.

II. TRAINING PROTOCOL

A. Supervised learning

In this work we considered only system sizes up to $L = 18$, which are solvable via exact diagonalization (ED). We solve the ED problem for a given realization of the model $\hat{H}_{\text{SYK}}(J)$, finding the corresponding ground state $|\psi_{\text{GS}}(J)\rangle$. The key result of this work, the exponential scaling of the required network size to reach a specific error bound, is independent from the specific J realization for the couplings of the SYK model. This has been checked by performing the full analysis as described in the following paragraphs for two independent realizations of the random coupling matrix J.

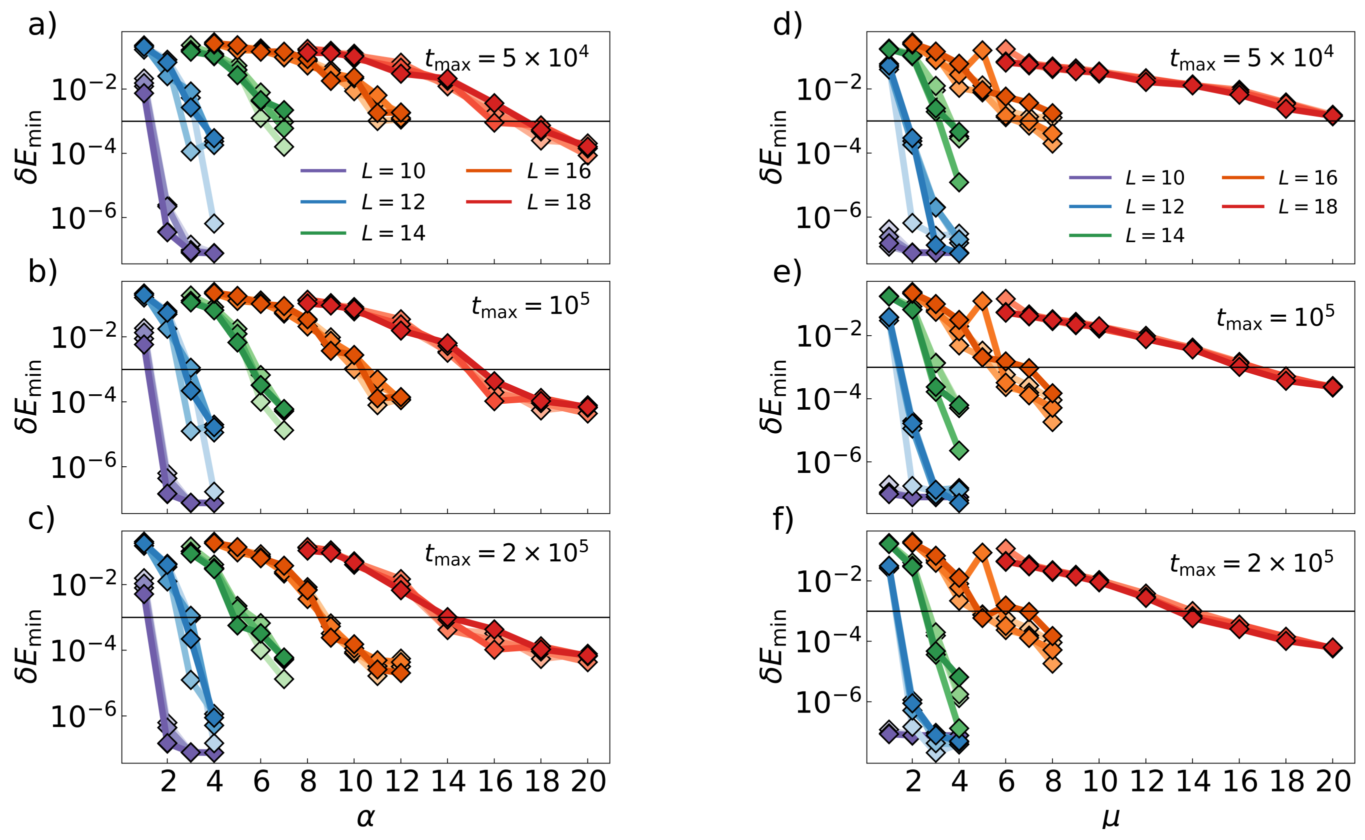

FIG. S1. Relative energy error δE as function of [(a), (b), (c)] the network width α and of [(d), (e), (f)] the number of layer μ for several system sizes and random initializations after [(a), (d)] 5×10^4, [(b), (e)] 10^5, and [(c), (f)] 2×10^5 simulation steps, respectively. Here we report the full dataset from which the average value plotted in the main text has been derived. The color of each set of data points corresponds to the system size L as indicated in the legend while different shades of the same color belong to independent realizations of the initial network weights.

Given the ED solution $|\psi_{\mathrm{GS}}\rangle$, it is possible to train the network to reproduce the correct probability amplitudes (up to norm and phase gauge freedoms) via supervised learning [4]. In practice, we have done so by optimizing the loss function

$$\delta O(\theta, J) = 1 - \left| \frac{\langle \psi_\theta | \psi_{\mathrm{GS}}(J) \rangle}{\langle \psi_\theta | \psi_\theta \rangle} \right| \tag{S5}$$

using the Adam optimizer [5]. Another training scheme can be implemented by directly minimizing the expected energy $E(\theta, J) = \langle \hat{H}_{\mathrm{SYK}}(J) \rangle_{|\psi_\theta\rangle}$ of the system as the loss function. For this loss function, it is possible to extend the training to system sizes beyond the reach of ED studies through variational Monte Carlo (VMC) sampling [6], which is not possible in the SL scheme employed here due to its reliance on the knowledge of the full solution vector. Since all our system sizes could be treated via full summation, we have not relied on VMC sampling in this work.

As our goal is to find the network size required to represent the SYK ground states at a given system size, we have compared training runs for both SL and energy minimization routines, finding the SL to be the best performing choice in the regime under consideration (compare Section III B). Therefore, we have selected the SL results for presentation in the main text. Our observations regarding the exponential scaling of required network size hold also for the variational energy optimization runs.

The optimization problem for a neural network is challenging on its own and there is unfortunately no general way to systematically identify the best choice of hyperparameters. Here, we have chosen to study the scaling with regard to width α and depth μ of the network by performing two sets of runs: Varying the width for a two-layer network and varying the depth of a network of fixed width $\alpha = 4$. Considering the trajectory of weights $\{\theta(t)\}_{t \in [t_{\max}]}$ (where $[t_{\max}] := \{1, \ldots, t_{\max}\} \subseteq \mathbb{N}$) over the sequence of optimization steps, we can the define the optimal energy relative error as $\delta E_{\min}(\theta, J) = \min_{t \in [t_{\max}]} E(\theta(t), J)$. In Fig. S1 we report $\delta E_{\min}$ for SL protocols for three different number of total iterations $t_{\max}$ as function of α. All the data reported in this plot have been obtained with $\mu = 2$, keeping the learning rate of the Adam optimizer constant throughout all the simulation. From this analysis we can make

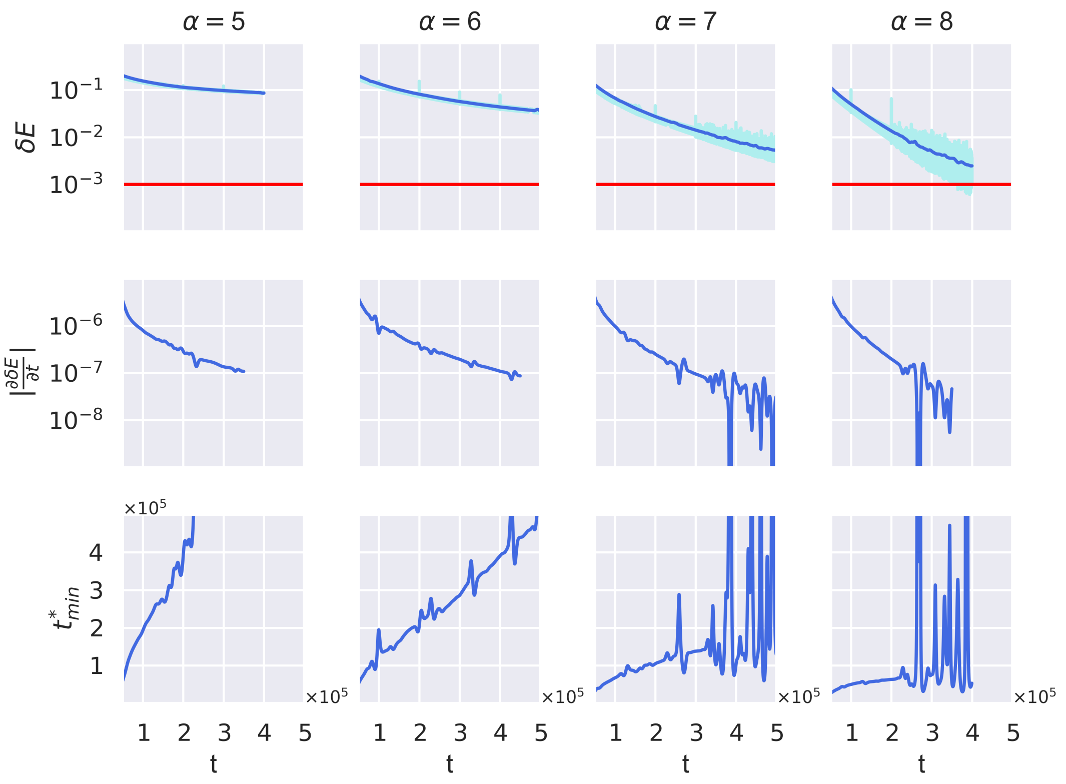

FIG. S2. (a) Relative energy error δE as function of the iteration step for shallow networks with $\mu = 2$ layers and width α as reported in the respective panels. The system size is $L = 16$. In light blue we report the raw data, in dark blue the result of a window smoothing filter applied to the raw data. (b) Absolute value of the numerical derivative $|\frac{\partial(\delta E)}{\partial t}|$. (c) Estimated minimum number of steps required to convergence $t^*_{\rm min}$ as defined by Eq. (S6).

two relevant observations. The first is that up to the network sizes simulated there is a monotonic improvement of the error with increasing width α. The second observation is that at each fixed number of simulation steps there is an exponential scaling of the $\alpha_{\rm min}$ required to bring $\delta E_{\rm min}$ below an arbitrary threshold (10^{-3} in our example) as a function of the system size. With the data reported in this paragraph, it cannot be ruled out that this exponential cost in network size is due to an exponential scaling of the number of steps required to bring the small α values to convergence. However, through an analysis of the training curves it is possible to make a strong argument to identify the $\alpha_{\rm min}$ discussed in the main text as a quantity independent from the number of training steps. This analysis is explained in the following section.

B. Truncation scheme

In the first row of Fig. S2, we report $\delta E(\theta(t), J)$ for four different alpha values distributed around what we are going to define as the critical $\alpha_{\rm min}$. The light blue line corresponds to the raw data obtained from the training process. In order to remove the fluctuations around the moving average, we apply a flat window filter over the raw error, obtaining the profile corresponding to the dark blue line. This running average error is smooth and monotonically decreasing. Furthermore, the slope of these curves is also decreasing (in absolute value), implying a slowing rate of convergence of the optimization. In order to conserve computational resources, we truncate the runs that are expected to converge to a $\delta E(\theta, J)$ above the energy threshold of 10^{-3} based on the following criterion: Assuming that the absolute value of the slope is monotonically decreasing throughout all the training iterations, it is possible to estimate, at any step

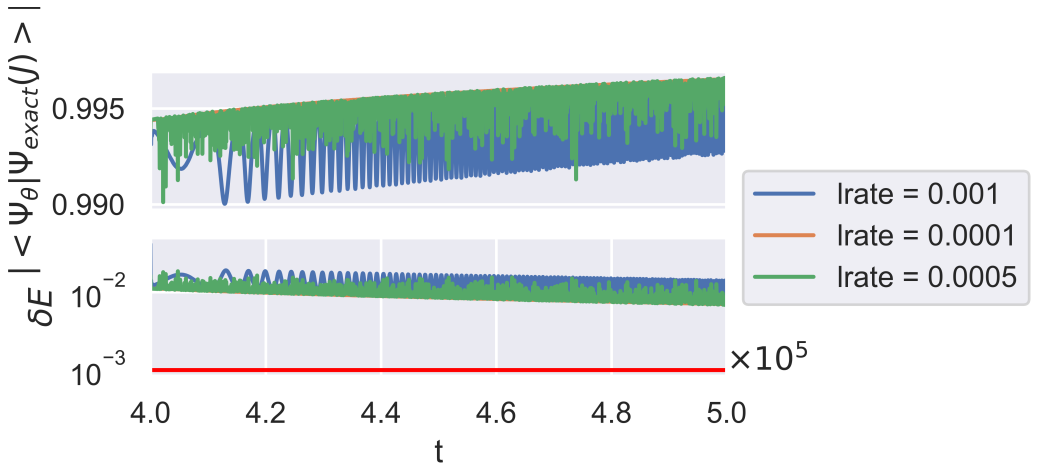

FIG. S3. (a) Overlap $\langle \Psi_\theta | \Psi_{\text{exact}}(J) \rangle$ and (b) relative energy error δE for a training protocol in the interval $t \in [4 \times 10^5, 5 \times 10^5]$ steps. The same pre-trained network here it has been updated with the three different learning ratios reported in the legend.

t, a lower bound for the number of steps required to reach the error threshold as

$$t^*_{\text{min}}(t) = \frac{\delta E(\theta(t), J) - \delta E^{\text{threshold}}}{|\partial_t \delta E(\theta(t), J)|}. \tag{S6}$$

Within the assumption of constant slope, the equality $\delta E(\theta(t + t^*_{\text{min}}(t)), J) = \delta E^{\text{threshold}}$ holds. If now we define a control interval Δt big enough to average out the noise around training curve, one can check if a candidate run is trending towards convergence below the error threshold based on the slope of $t^*_{\text{min}}(t)$,

$$\delta t^*_{\text{min}}(t) = \frac{t^*_{\text{min}}(t + \Delta t) - t^*_{\text{min}}(t)}{\Delta t}. \tag{S7}$$

The runs that are expected to reach convergence are those that satisfy

$$\delta t^*_{\text{min}}(t + \Delta t) < \delta t^*_{\text{min}}(t). \tag{S8}$$

We simulated all runs for at least $t_{\text{max}} = 2 \times 10^5$, for those runs that where still above the threshold for that step we used the criterion (S8) (with $\Delta t = 10^5$) in order to assess whether they can still be expected to reach the error threshold, based on their current rate of convergence. The runs for which converge below the threshold was ruled out by our criterion were stopped in order to conserve computational resources, while we did perform additional blocks of 10^5 steps for the remaining simulations, until the truncation criterion (S8) did apply or the error threshold was reached. The resulting α_{min} values reported in Fig. 4 of the main text are the smallest simulated value of α for which convergence below the threshold was reached. The μ scaling analysis was performed in an equivalent way.

III. OVERVIEW OF ALTERNATIVE TRAINING METHODS INVESTIGATED

In the previous paragraphs we have described the protocol and data analysis necessary to reproduce the results that we discussed in the main text. Here we are showing alternatives to the training scheme and hyperparameters. The alternatives which we have tested achieved either worse accuracy or no significant improvement compared to the results we selected for the main text.

A. Different learning rate

We have used the Adam optimizer using a constant learning rate throughout the simulations reported in the main text. One advantage of Adam with respect to SGD is that it automatically adapts the step size based on the history of loss gradients. Still, Adam has to be initialized with a specific learning rate and this can potentially influence the effectiveness of the training. While determining the optimal learning rate for our simulations, we have also tried

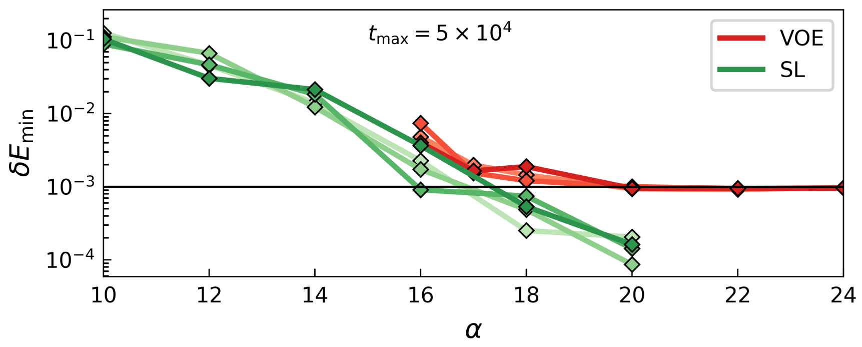

FIG. S4. Minimum relative energy error $\delta E_{\min}$ obtained using the SL (green) and a VOE (red) loss functions using $t_{\max} = 5 \times 10^4$ steps as function of the number of network width α. For the data reported in this plot, we considered system size $L = 18$ and a fixed $\mu = 2$

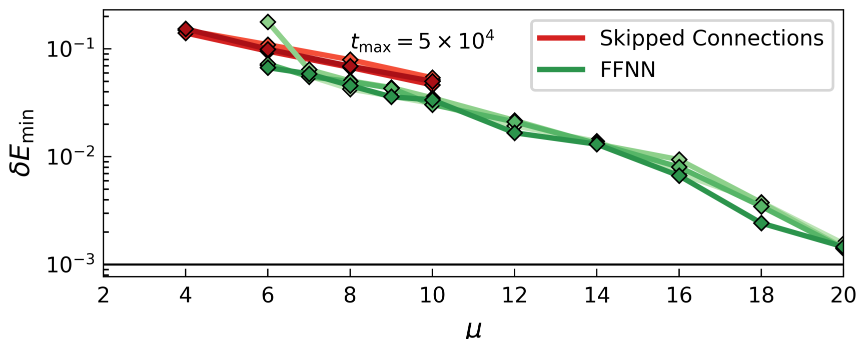

FIG. S5. Minimum relative energy error $\delta E_{\min}$ obtained during a SL training protocol of a deep FFNN (green) and of a Network with Skipped connections (red) with a number of steps set to $t_{\max} = 5 \times 10^4$ as function of the number of network laters μ. For the data reported in this plot we considered system size $L = 18$ and a fixed $\alpha = 4$

schemes involving an update of the learning rate during the simulation. We report an example of an unconverged training curve in Fig. S3 for which, after $4 \cdot 10^5$ steps at a fixed learning rate, we performed further 10^5 steps with three different adjusted learning rates. These changes of learning rate did not result in an improvement of training performance.

B. Variational energy optimization

Beside the SL protocol described above, we have also considered directly optimizing the variational energy

$$E(\theta, J) = \frac{\langle \psi_\theta | \hat{H}_{\text{SYK}}(J) | \psi_\theta \rangle}{\langle \psi_\theta | \psi_\theta \rangle}. \tag{S9}$$

For disambiguation, we refer to this loss function as the variationally optimized energy (VOE). Note that this loss does not involve the ED solution of the model and can be applied to system sizes beyond those limits, if it is used together with MC sampling of the quantum states [6]. However, as we are interested in comparing the performance of the SL and VOE protocols in the ED regime, we only use the loss (S9) obtained by full summation in the following. Within the parameter range tested, the SL protocol achieved systematically lower errors than the VOE loss. As an example of this, we show in Fig. S4 the resulting $\delta E_{\min}$ for a given system size ($L = 18$) and a fixed number of iteration steps. Although the trend is similar between the two curves, the SL reaches the convergence threshold at lower α values.

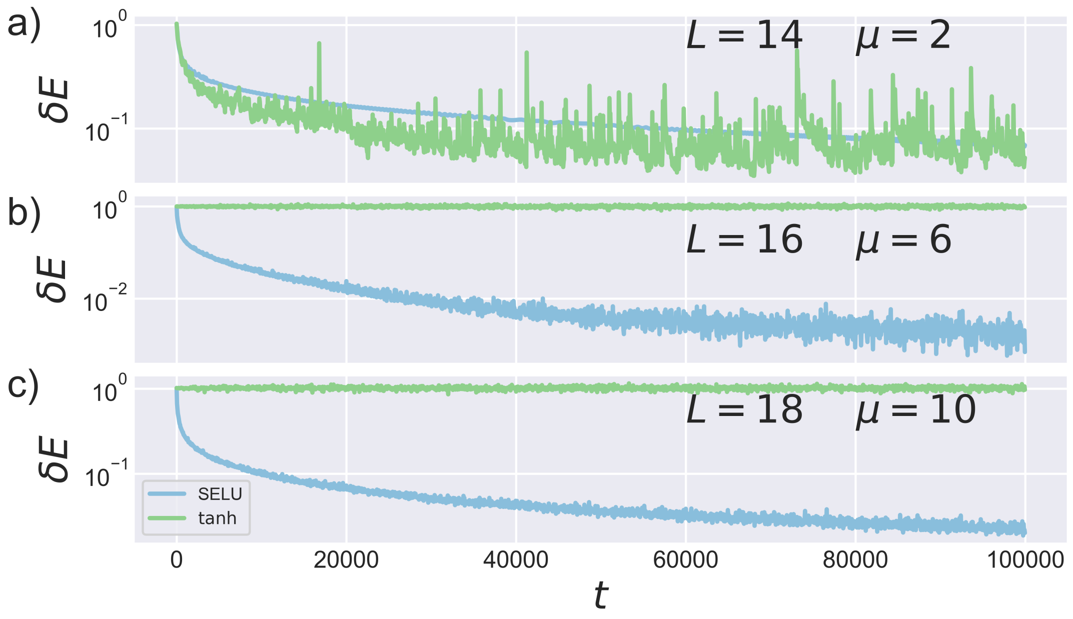

FIG. S6. Relative energy error δE as function of the iteration step t using the SL training protocol with SELU (light blue) and tanh (light green) activation functions. The figure shows the data for (a) $L = 14$, $\mu = 2$; (b) $L = 16$, $\mu = 6$; and (c) $L = 18$, $\mu = 10$.

C. Deep networks with skip connections

In the main part of this work, we presented an extensive discussion of the results obtained when training a FFNN with the architecture defined in Section I. Among the two different hyperparameter dependencies discussed there, the number of layers μ scaling requires some extra attention. It is known that while adding more layers enhances the networks expressive capability, at the same time it can make the training more difficult [7]. One common remedy to this is the addition of skip connections between the layers of the neural network [8]. We report in Fig. S5 a comparison between the training at a fixed number of iteration steps $t_{\max} = 5 \cdot 10^4$ between the FFNN discussed in the main text and a fully-connected network with skip connections. Specifically, our network is divided into n_B blocks, each containing ℓ layers. The total number of layers is thus $\mu = n_B \ell$. After the application of each block (i.e., after applying ℓ fully-connected layers), the content of the original input layer (with the first affine transformation applied) is added to the output before it is passed to the following block. For the SL simulation runs considered (Fig. S5), this architecture does not give an improvement in energy error when compared to a simple FFNN. We have found improved convergence (compared to the simple FFNN) of the skip connection networks in some runs using the VOE loss. However, in those cases the number of layers already exceeded the critical μ (around which the network with skip connections still performed worse), so no improvement with regard to compression was obtained this way.

D. Hyperbolic tangent activation function

As already stated, for all the data discussed in the main part, the FFNN implemented where following the definition presented in Eq. (S1) using SELU as activation function ϕ. While in principle there are many viable choices with regard to which specific activation function utilizing, we opted to restrict our studies to a particular activation in order to investigate extensively the dependence of the representability power as function of the network hyperparameters, keeping a manageable number of simulations to perform. Thus the need to choose an activation function able to achieve reliable performance when dealing with both deep and shallow networks. We present here a comparison of the performances achieved by the same training protocol and network architecture while using another common activation function, the hyperbolic tangent. In Fig. S6, we present training profiles for the relative energy error δE as function of the iteration step t. We show this for both a network with SELU and a network with tanh activation function for several numbers of layers μ. As one could expect from the vanishing gradient problem that effects *tanh*, we observe that also for the network sizes relevant to our problem, above a certain number of layers the tanh activated networks are not able to improve the energy error anymore. At the same time SELU displayed a monotonic improvement of

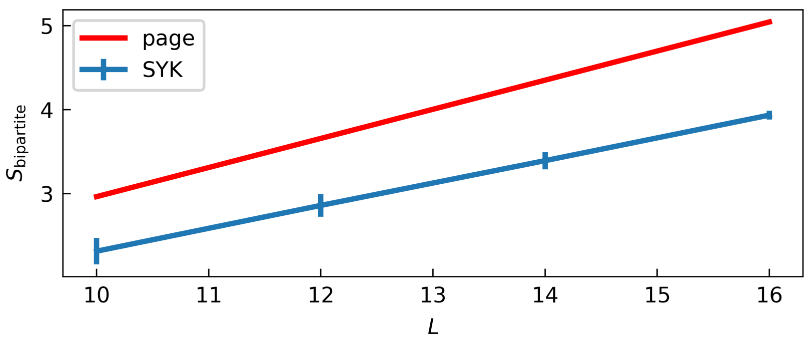

FIG. S7. Bipartite entanglement entropy averaged over 4 independent realizations of the SYK model (error bar shows the standard deviation) compared to bipartite entanglement of a random state given by the Page value (red).

the energy error with the number of layers (up to scales relevant for our specific problem) and thus is an effective choice (compare Fig. S1).

IV. SYK BIPARTITE ENTANGLEMENT

Following the standard definition, the bipartite entanglement of a density matrix ρ corresponds to

$$S_{\text{bipartite}} = -\text{Tr}\left[\rho_A \log\left(\rho_A\right)\right] = -\text{Tr}\left[\rho_B \log\left(\rho_B\right)\right] \tag{S10}$$

where $\rho_{A(B)} = \text{Tr}_{B(A)}\left[\rho\right]$ is the reduced density matrix obtained from the density matrix ρ and tracing over the degrees of the subpartition B(A) of the total hilbert space, in the special case of $\dim(A) = \dim(B)$. We compare the $S_{\text{bipartite}}$ scaling for four independent finite size SYK model realizations together with the Page value [9], that quantifies the entanglement for a pure random state. In the special case case of bipartite entanglement, the page value is

$$S_{\text{page}} = N\log(2) - \frac{1}{2}. \tag{S11}$$

As one can see in Fig. S7, the entanglement scaling of the SYK model follows a volume law scaling that does not saturate the page value, showing that the SYK states exhibit structure beyond a pure random state.

V. ENERGY CONVERGENCE TOWARD THE THERMODYNAMIC LIMIT

Considering the complex SYK model defined by

$$\hat{H}_{\text{SYK}}\left(J\right) = \frac{1}{(2L)^{3/2}} \sum_{ijkl} J_{ij;kl}\, \hat{c}_i^\dagger \hat{c}_j^\dagger \hat{c}_k \hat{c}_l, \tag{S12}$$

it is possible to analytically evaluate the ground state energy density in the large L limit, this results to be $e_{\text{GS}}^{(\text{TD})} \approx -0.0826$. In Fig. S8 we show the ground state energy from the ED simulations for the system sizes considered in the main part of the paper. There one can see that while the energy is slowly converging toward the thermodynamic limit, at $L = 18$ the relative error in the energy between the thermodynamic limit and the finite size results to be $\delta E = \left(e_{\text{GS}}|_{L=18} - e_{\text{GS}}^{(\text{TD})}\right)/e_{\text{GS}}^{(\text{TD})} \approx 0.5$. Despite the absolute value of the energy not yet being converged to the thermodynamic limit, we can see that as for the entanglement entropy also this observable exhibits self-averaging properties in the sizes simulated.

VI. NETWORK COMPRESSION

Trained neural networks can be amenable to compression in order to reduce the total number of parameters. This is usually done for the purpose of reducing memory and computational requirements in order to run a network on

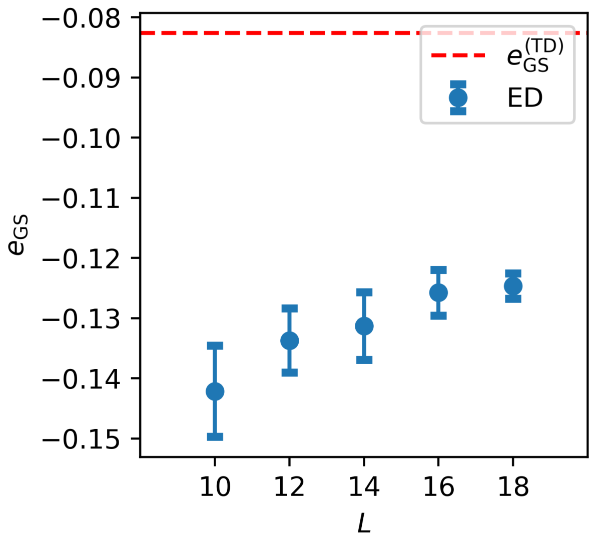

FIG. S8. Groundstate energy density e_{GS} evaluated from ED for different finite sizes. The errorbar corresponds to the standard deviation. The number of independent realizations for the data shown depends on the system size and corresponds to: ($L =$ 10, 40 realizations), ($L = 12$, 30 realizations), ($L = 14$, 20 realizations), ($L = 16$, 10 realizations), ($L = 18$, 4 realizations)

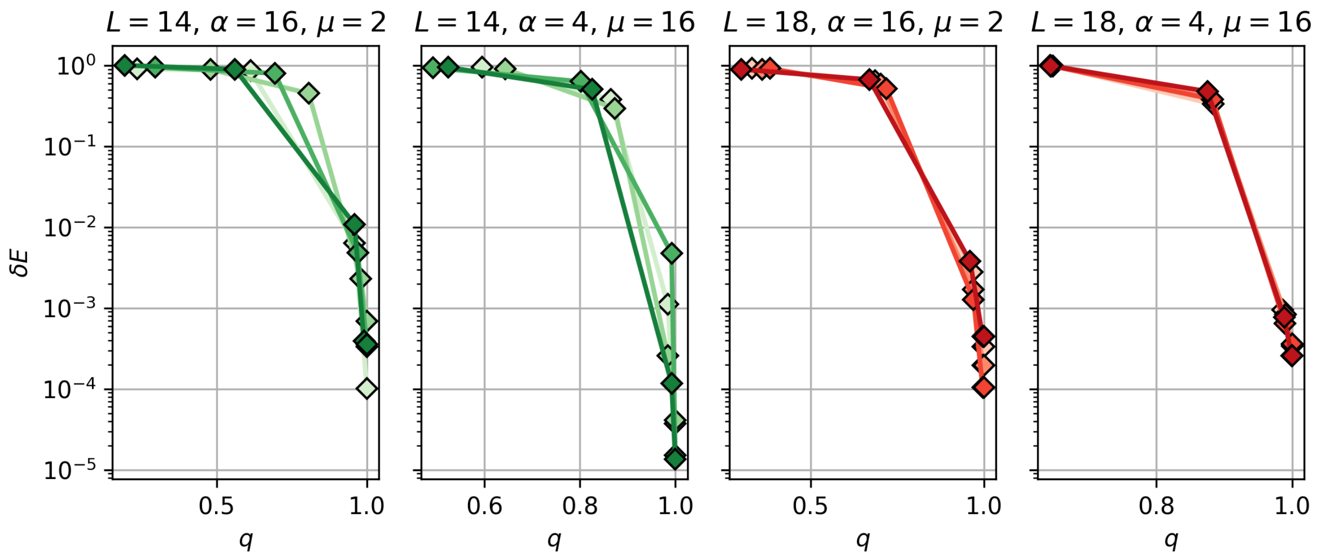

FIG. S9. Relative energy error as a function of the fraction of retained singular values q for four different choices of hyperparameters indicated in the title of each panel. The color corresponds to different network initializations as in Fig. S1.

a lower powered device. Here, we have explored a compression scheme based on a lower-rank approximation of the weight matrices in each layer [10] in order to gauge whether any potential redundancy in the learned variational parameters can be identified this way. This is done by performing a singular value decomposition (SVD)

$$W^{(l)} = U^{(l)} S^{(l)} V^{(l)} \tag{S13}$$

where $U^{(l)}$, $V^{(l)}$ are unitary matrices and $S^{(l)} = \mathrm{diag}(\sigma_1^{(l)}, \ldots, \sigma_M^{(l)})$ is the matrix of singular values $\sigma_1^{(l)} \geq \ldots \geq \sigma_M^{(l)} \geq$ 0. We truncate this spectrum by discarding all singular values below a threshold λ relative to the largest singular value, i.e., based on the criterion $\sigma_i^{(l)}/\sigma_1^{(l)} < \lambda$. For simplicity, we chose λ uniformly for each layer. As a measure of the approximation error, we report in Fig. S9 the relative energy error after truncation as a function of the fraction

$$q = \frac{\text{singular values retained}}{\text{total singular values}} \tag{S14}$$

of singular values retained over all weight matrices of the network for different system sizes and choices of hyper-parameters. This data shows that already a limited amount of truncation ($0.95 \leq q < 1.0$) results in a significant increase in energy error. We therefore conclude that at uniform SVD truncation of the weight matrices does not reveal redundancy in our learned SYK ground states.

[1] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, JAX: composable transformations of Python+NumPy programs (2018).
[2] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, Self-normalizing neural networks (2017), arXiv:1706.02515.
[3] F. Vicentini, D. Hofmann, A. Szabó, D. Wu, C. Roth, C. Giuliani, G. Pescia, J. Nys, V. Vargas-Calderón, N. Astrakhantsev, and G. Carleo, NetKet 3: Machine Learning Toolbox for Many-Body Quantum Systems, SciPost Phys. Codebases , 7 (2022).
[4] B. Jónsson, B. Bauer, and G. Carleo, Neural-network states for the classical simulation of quantum computing (2018).
[5] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization (2014), arXiv:1412.6980.
[6] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, 602 (2017).
[7] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016) pp. 770–778.
[8] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, Visualizing the loss landscape of neural nets, in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, edited by S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (2018) pp. 6391–6401.
[9] D. N. Page, Average entropy of a subsystem, Phys. Rev. Lett. **71**, 1291 (1993).
[10] J. Xue, J. Li, and Y. Gong, Restructuring of deep neural network acoustic models with singular value decomposition, in *Interspeech 2013* (ISCA, 2013).