Adaptive Neural Quantum States: A Recurrent Neural Network Perspective

Jake McNaughton
Perimeter Institute
Waterloo, ON N2L 2Y5
jmcnaughton@perimeterinstitute.ca

Mohamed Hibat-Allah
Department of Applied Mathematics
University of Waterloo
Waterloo, ON N2L 3G1
mhibatallah@uwaterloo.ca

Abstract

Neural-network quantum states (NQS) are powerful neural-network ansätzes that have emerged as promising tools for studying quantum many-body physics through the lens of the variational principle. These architectures are known to be systematically improvable by increasing the number of parameters. Here we demonstrate an Adaptive scheme to optimize NQSs, through the example of recurrent neural networks (RNN), using a fraction of the computation cost while reducing training fluctuations and improving the quality of variational calculations targeting ground states of prototypical models. This Adaptive technique reduces computational cost through training small RNNs and reusing them to initialize larger RNNs. This work opens up the possibility for optimizing graphical processing unit (GPU) resources deployed in large-scale NQS simulations.

1 Introduction

Neural Quantum States (NQS) are the representation of a wave function as a neural network, whose parameters are optimized through the variational principle [31, 33], enabling a wide range of applications in quantum many-body physics [2]. In particular, finding ground states [25, 34] and simulating time-evolution of quantum many-body systems [8, 46]. In the literature, a variety of neural network architectures have been used as NQSs, including Restricted Boltzmann Machines (RBM) [8, 41], feedforward neural networks [6, 11], Convolutional Neural Networks (CNN) [12], Recurrent Neural Networks (RNN) [18, 44, 9, 30, 22, 35, 37, 38], and Transformers [59, 51, 43, 50, 26].

NQSs have the ability to be systematically improved by increasing the number of parameters, in a similar manner to the bond dimension parameter in DMRG [47, 17]. However, using larger models significantly increases the computational cost, requiring larger GPUs, more GPU units, and more time for training. As a result, despite the improved accuracy resulting from growing model complexities, most large-scale simulations in the literature still rely on a small subset of NQS architectures [40, 37, 38, 51].

To address this challenge, we propose an Adaptive training scheme where the dimension of the hidden state of an RNN wave function [18, 44] is iteratively increased during training. By implementing this training scheme, higher-dimensional models are trained for a fraction of the time required to train them from scratch, thereby reducing the computational resources used. We share the promising results obtained from our framework applied to the 1D transverse-field Ising Model (TFIM) and 2D Heisenberg Model, and demonstrate that these results indicate a superiority of the Adaptive RNN not only in terms of speed, but also in terms of accuracy and stability. Our implementation of the presented methods and all scripts needed to reproduce our results in this manuscript are openly available at `https://github.com/jakemcnaughton/AdaptiveRNNWaveFunctions`.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Machine Learning and the Physical Sciences.

2 Methods

To reduce the computational load of the model and the time taken in a variational calculation, we propose the *Adaptive* RNN where the size of the hidden state is gradually increased throughout training. As a result, the dimensions of the model parameters change with the hidden state size. We develop a method to increase the size of the parameters during training and transfer them to an RNN with a larger hidden state. The goal of this Adaptive scheme is to reduce training time and improve the accuracy of variational calculations, as we demonstrate in the results section.

When shifting from a model with hidden-state dimension $d_{\mathrm{h}}^{(i)}$ to one with dimension $d_{\mathrm{h}}^{(i+1)}$, where $d_{\mathrm{h}}^{(i)} < d_{\mathrm{h}}^{(i+1)}$, the weights and biases sizes increase from $d_{\mathrm{h}}^{(i)} \times d_{\mathrm{h}}^{(i)}$ and $d_{\mathrm{h}}^{(i)}$ to $d_{\mathrm{h}}^{(i+1)} \times d_{\mathrm{h}}^{(i+1)}$ and $d_{\mathrm{h}}^{(i+1)}$ respectively. To transfer the parameters from the smaller model (with parameters of size $m \times n$) to the larger model (with parameters of size $M \times N$), the larger model is initialized and then the trained parameters from the smaller model are embedded into the top $m \times n$ positions, replacing the initialized values in these positions. Xavier initialization was used for all RNNs.

An Adaptive RNN is a sequence of RNNs, each with a larger hidden-state dimension than the previous one. These RNNs are trained sequentially, the key idea being that the final RNN model can be trained for fewer training steps as it is pre-trained by the models earlier in the sequence, which are computationally cheaper to train. In our study, we double the hidden state size after each interval $d_{\mathrm{h}}^{(i+1)} = 2d_{\mathrm{h}}^{(i)}$. Note that we ensure that all parameters of the RNN $i+1$ are trainable, and we do not freeze the transferred set of parameters during training. In App. A we provide details of our RNN implementations in one- and two- dimensions Here, we use GRU cells as they provided the best stability for both the static and adaptive methods and therefore were an ideal testbed, however our method was also successfully implemented for both Vanilla and LSTM cells.

We refer to the traditional method of training RNNs with a single model of fixed hidden dimension as *Static*. We refer to our proposed method of training, where the hidden dimension is increased throughout training, as *Adaptive*.

3 Results

To compare Static and Adaptive RNN wave functions, we focus on finding the ground state of several prototypical Hamiltonians using the Variational Monte Carlo (VMC) framework [2]. This involves minimizing the variational energy $E_\theta = \langle \Psi_\theta | \hat{H} | \Psi_\theta \rangle$ of a variational ansatz $|\Psi_\theta\rangle$, such as an RNN wave function, which is normalized by construction [18]. To find an approximation of the ground state using VMC, the parameters are learned by training the RNN parameters through a gradient descent algorithm. In this study, we use Adam optimizer [23] and follow the same training scheme as in Ref. [18]. The hyperparameters used for all benchmarks can be found in App. B. Further results obtained from studying long-range Hamiltonians can be found in App. C

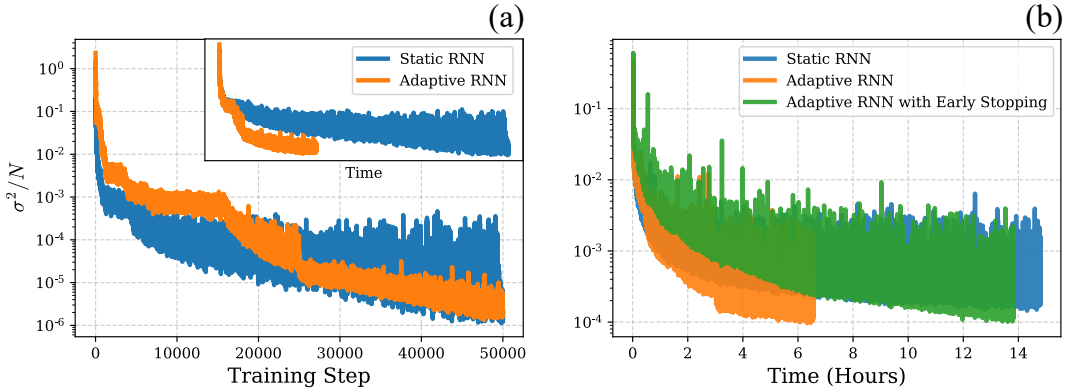

Figure 1: (a) Energy variance per spin throughout training for $N = 100$ spins in the one-dimensional transverse field Ising model. Inset: energy variance per spin against runtime. (b) Energy variance per spin vs Time (in hours) for the Heisenberg model on a lattice of 6×6 spins.

3.1 One-dimensional Transverse-field Ferromagnetic Ising Model

To demonstrate the effectiveness of the proposed Adaptive method, a one-dimensional RNN is used to study the 1D TFIM, within open boundary conditions (OBC), described by the following Hamiltonian

$$\hat{H}_{\mathrm{TFIM}} = -\sum_{i=1}^{N-1} \hat{\sigma}_i^z \hat{\sigma}_{i+1}^z - \Gamma \sum_{i=1}^{N} \hat{\sigma}_i^x. \tag{1}$$

Here $\sigma_i^{x,z}$ represents Pauli Matrices of the ith spin and Γ is the strength of the external transverse magnetic field [13].

A GRU-based RNN is trained on the system sizes $N = 20$, 40, 60, 80, and 100 spins at the critical point $\Gamma = 1$. Both Static and Adaptive models are trained for 50,000 gradient descent steps. The Static RNN has a hidden dimension equal to 256 throughout training, whereas the Adaptive RNN starts with $d_{\mathrm{h}} = 2$, and increases at a fixed interval (every 6,250 training iterations).

Fig. 1(a) shows the variance per spin throughout training of the Adaptive and Static RNNs for $N = 100$ spins. The Static RNN demonstrates a quick convergence compared to the Adaptive RNN in the first half of training. Nevertheless, in the second half for $d_{\mathrm{h}} \geq 64$, the Adaptive RNN reaches a comparable variance till the end of training. Looking at the variance evolution with time in the inset of Fig. 1(a), we observe that the Adaptive model maintains a lower variance from the beginning, and finishes training in 34% of the time. This result demonstrates that the Adaptive RNN can achieve an accurate result faster compared to the Static RNN. We find that the time ratio of the time taken by the Adaptive RNN over that of the Static RNN evolves with the number of spins. The ratio in the asymptotic limit is estimated around 25.6% in the case of our Adaptive scheme with fixed intervals. Additionally, we report pronounced fluctuations in the Static RNN training trajectory relative to the Adaptive RNN, underscoring the improved stability achieved through the Adaptive training strategy as suggested in Fig. 1(a).

Tab. 1 presents the final results for all system sizes when computed with 1,000,000 samples after training is complete. In addition to showing the Static and Adaptive RNN results for $d_{\mathrm{h}} = 256$, we also provide data for the trained penultimate RNN model in the Adaptive sequence with $d_{\mathrm{h}} = 128$. The results clearly demonstrate that the Adaptive RNN yields compatible energies with those of the Static RNN within error bars, except for $N = 40$ where the Adaptive RNN provided the best relative error. We also note that the Adaptive RNN outperforms the Static RNN on system sizes $N = 40, 80$, and 100 in terms of energy variance. The latter is a good indicator of the quality of a variational calculation [2, 1, 54]. Furthermore, the penultimate model with $d_{\mathrm{h}} = 128$ achieved comparable energies to the Static RNN with $d_{\mathrm{h}} = 256$, with a much shorter runtime and requiring less GPU resources. This result highlights the possibility of getting comparable accuracy with a lower number of parameters by virtue of the enhanced trainability provided by the Adaptive scheme.

3.2 Two-dimensional Heisenberg Model

We now focus on the 2D Heisenberg model on a square lattice to assess the Adaptive RNN's performance in two spatial dimensions. Historically, this model has served as a very useful playground for the development of numerical methods in computational quantum matter [45, 29, 52, 8]. The following Hamiltonian describes this model within OBC:

$$\hat{H} = \frac{1}{4} \sum_{\langle i,j \rangle} \hat{\sigma}_i^x \hat{\sigma}_j^x + \hat{\sigma}_i^y \hat{\sigma}_j^y + \hat{\sigma}_i^z \hat{\sigma}_j^z. \tag{2}$$

Here $\langle i, j \rangle$ indicates that the indices being summed over are nearest neighbor pairs on the square lattice. To use a positive 2D RNN, we apply a Marshall sign rule, which is equivalent to finding the ground state of the XXZ Hamiltonian where all the off-diagonal elements are negative [32, 7].

A Static 2D RNN with $d_{\mathrm{h}} = 256$ is trained for 200,000 gradient steps. In the Adaptive setting, we start with a 2D RNN that has $d_{\mathrm{h}} = 32$, and double it every 50,000 steps, for a total of 200,000 steps. In addition to this setup, we also implement an Adaptive framework where d_{h} doubles after each early stopping criterion, given by the energy variance, is triggered until reaching a model with $d_{\mathrm{h}} = 256$ where the criterion triggers training to stop.

Fig. 1(b) shows the energy variance per spin throughout training for the three models trained on this benchmark task. Similar to the 1D TFIM, the Adaptive RNN completes training in less than half the

Method	N	Energy	σ^2/N $[\times 10^{-6}]$	Time (hh:mm:ss)
Static RNN (256)	20	-25.107793(5)	**1.067(2)**	00:08:44
Adaptive RNN ($2 \to 128$)		-25.107794(5)	1.203(2)	**00:03:59**
Adaptive RNN ($2 \to 256$)		-25.107785(6)	1.877(3)	00:05:27
Static RNN (256)	40	-50.569396(9)	2.147(3)	00:23:47
Adaptive RNN ($2 \to 128$)		-50.56941(2)	2.144(3)	**00:07:48**
Adaptive RNN ($2 \to 256$)		-50.569426(8)	**1.749(3)**	00:11:12
Static RNN (256)	60	-76.033138(9)	**1.228(2)**	00:45:33
Adaptive RNN ($2 \to 128$)		-76.03312(3)	2.013(3)	**00:12:14**
Adaptive RNN ($2 \to 256$)		-76.03314(1)	2.042(3)	00:18:21
Static RNN (256)	80	-101.49738(2)	2.972(4)	01:13:22
Adaptive RNN ($2 \to 128$)		-101.49737(4)	2.190(3)	**00:17:25**
Adaptive RNN ($2 \to 256$)		-101.49739(1)	**1.433(2)**	00:27:21
Static RNN (256)	100	-126.96182(2)	2.313(3)	01:54:56
Adaptive RNN ($2 \to 128$)		-126.96184(3)	3.638(5)	**00:24:31**
Adaptive RNN ($2 \to 256$)		-126.96185(1)	**2.048(3)**	00:39:46

Table 1: Comparison between the Static and Adaptive RNNs in terms of the final energies, and variances per spin σ^2/N using 1,000,000 samples. Times taken for training are also reported. The best values, while taking error bars into account, for variance are shown in bold. The error bar on the variance is estimated by assuming a Gaussian distribution over the local energies [36]. All simulations were run using A100 GPUs.

time, reaching lower variances. The early stopping variant also reaches lower variances but does not provide a significant improvement in training time, as it is trained for about 540,000 steps. However, we believe that there is still room for exploring optimal stopping criteria.

Each model is sampled to output 1,000,000 configurations once training is complete. The Adaptive model achieves the best energy, error, and variance, completing training in 6 hours and 33 minutes. The Adaptive method with early stopping outperforms the Static model in terms of energy, error, and variance, taking 13 hours and 49 minutes, whereas the Static model takes 14 hours and 48 minutes.

In addition to the advantage of the Adaptive RNN highlighted in the two previous benchmarks, we would also like to note that in App. C, we provide additional empirical evidence of the superiority of our adaptive architecture on long-range quantum benchmarks, namely the long-range TFIM and the 1D Cluster Hamiltonians.

4 Conclusion

In this paper, we propose a framework for training RNN wave functions by gradually scaling up the hidden state dimension throughout training. This technique resulted in a significant reduction in the time taken for training when applied to prototypical spin models studied, while reaching similar or improved levels of accuracy. Our study also demonstrates that our Adaptive RNN can reach accurate energies using a lower hidden state dimension, highlighting the improved trainability using our Adaptive scheme.

An optimal early stopping mechanism is expected to further improve the performance of the Adaptive framework by ensuring each model in the sequence is trained for long enough to gain the time advantage, but not overtrained. Additionally, developing an adaptive learning rate scheme that depends on the stage of our Adaptive method could improve training and speed of convergence. Furthermore, combining our Adaptive RNN with the iterative retraining technique of RNNs [44, 20, 37, 38] will also allow targeting large lattice sizes using a fraction of the computational cost, leveraging the inherent weight sharing in RNNs. The latter provides a key advantage of Adaptive RNNs compared to Adaptive RBMs used in Refs. [57, 56]. We also highlight that variational energies obtained in this work could be further improved by applying tensorization [20, 53, 37] and leveraging symmetries [18, 20, 39].

Acknowledgments

Computer simulations were made possible thanks to the Digital Research Alliance of Canada and the Math Faculty Computing Facility at the University of Waterloo. M.H acknowledges support from Natural Sciences and Engineering Research Council of Canada (NSERC), and the Digital Research Alliance of Canada. Research at Perimeter Institute is supported in part by the Government of Canada through the Department of Innovation, Science and Economic Development and by the Province of Ontario through the Ministry of Colleges and Universities.

A Recurrent Neural Networks (RNNs)

RNNs belong to the class of autoregressive models, which take advantage of the probability chain rule:

$$P(\sigma_1, \sigma_2, \ldots, \sigma_N) = P(\sigma_1)P(\sigma_2|\sigma_1)\ldots P(\sigma_N|\sigma_1, \ldots \sigma_{N-1}). \tag{3}$$

Hereafter $(\sigma_1, \sigma_2, \ldots, \sigma_N)$ stands for a configuration of spins of size N where $\sigma_n = 0, 1$. To illustrate how RNNs take advantage of the chain rule, let us take the example of the simplest RNN cell called the Vanilla RNN [28], where a spin configuration is generated sequentially through the following recursion relation:

$$\boldsymbol{h}_n = f(W\boldsymbol{h}_{n-1} + V\boldsymbol{\sigma}_{n-1} + \boldsymbol{b}), \tag{4}$$

where W, V and $\boldsymbol{b}$ are respectively the weights and the biases. $\boldsymbol{\sigma}_{n-1}$ is the one hot encoding of the spins σ_{n-1}. Furthermore, f is a non-linear activation function. The initialization of the recursion relation is given by $\boldsymbol{h}_0 = \boldsymbol{0}$, $\boldsymbol{\sigma}_0 = \boldsymbol{0}$. The hidden state $\boldsymbol{h}_n$ can be used to compute the parameterized conditional probability of getting σ_n as:

$$P_{\boldsymbol{\theta}}(\sigma_n|\sigma_{<n}) = \text{Softmax}\left(U\boldsymbol{h}_n + \boldsymbol{c}\right) \cdot \boldsymbol{\sigma}_n. \tag{5}$$

The product of the conditionals for each step n, allows us to obtain a parameterized joint probability distribution for the spin configurations. Note that the use of the vector $\boldsymbol{h}_n$ allows to model the conditional dependencies. For this reason, $\boldsymbol{h}_n$ is called the memory state (or the hidden state). The size of this state, called d_{h}, controls the expressiveness of the RNN. Additionally, the RNN construction is also key for enabling perfect (autoregressive) sampling from the joint probability P, where σ_n can be sampled sequentially from the conditional probabilities [18]. In this paper, we use a specific type of RNN cell, known as Gated Recurrent Units (GRU) [10] as described in App. A.

RNNs can model not only one-dimensional distributions, but can also be generalized to model two-dimensional quantum states [18]. Encoding two-dimensional correlations can be achieved using a two-dimensional RNN (2D RNN) through a two-dimensional recursion relation

$$\boldsymbol{h}_{i,j} = f(W[\boldsymbol{h}_{i-(-1)^j,j}; \boldsymbol{h}_{i,j-1}; \boldsymbol{\sigma}_{i-(-1)^j,j}; \boldsymbol{\sigma}_{i,j-1}] + \boldsymbol{b}),$$

where $[.\ ;\ .\ ;\ .\ ;\ .]$ is a concatenation operation. Note that the previous recursion relation can be adapted to take next-nearest neighbors or other geometries into account [18, 19, 22]. The 1D path for sampling and inference can be chosen as a zigzag path. We can use the Softmax layer to compute the conditional probabilities as in the case of the 1DRNNs. For the two-dimensional benchmarks, we use a 2D GRU variant.

A quantum state amplitude $\Psi(\boldsymbol{\sigma})$ could be modeled as follows:

$$\Psi(\boldsymbol{\sigma}) = \sqrt{P(\boldsymbol{\sigma})}\exp\left(\mathrm{i}\phi(\boldsymbol{\sigma})\right),$$

where P is a joint probability and ϕ is a phase. A large family of Hamiltonians, so-called stoquastic Hamiltonians [4], admits ground states with positive amplitudes. As a result, the ground state amplitudes can be modeled as the square root of a joint probability:

$$\Psi(\boldsymbol{\sigma}) = \sqrt{P(\boldsymbol{\sigma})},$$

where $P(\boldsymbol{\sigma})$ is a product of conditional probabilities computed using the RNN. This RNN wave function is denoted as a positive RNN (pRNN) wave function [18]. For non-stoquastic Hamiltonians, we can use a complex RNN (cRNN) wave functions [18], where

$$\Psi(\boldsymbol{\sigma}) = \sqrt{P(\boldsymbol{\sigma})}\exp\left(\mathrm{i}\phi(\boldsymbol{\sigma})\right). \tag{6}$$

Here $\phi(\boldsymbol{\sigma})$ is computed as a sum of conditional phases ϕ_n, where each $\phi_n = \phi_{\boldsymbol{\theta}}(\sigma_n|\sigma_{<n})$ is the output of a softsign layer (SS), i.e.,

$$\phi_{\boldsymbol{\theta}}(\sigma_n|\sigma_{<n}) = \pi \text{ Softsign}\left(U\boldsymbol{h}_n + \boldsymbol{c}\right) \cdot \boldsymbol{\sigma}_n. \tag{7}$$

Note that $\text{Softsign}(x) = x/(1+|x|)$ is chosen such that the conditional phases $\phi_n \in (-\pi, \pi)$ [18].

In this paper, we use Gated Recurrent Units (GRU) to implement our one- and two-dimensional RNNs [18]. In the one-dimensional case, we use the standard implementation of GRUs provided in Ref. [10]. In the one-dimensional case, at each step n, the hidden state $\boldsymbol{h}_n$ is computed via a gating mechanism that interpolates between the previous hidden state $\boldsymbol{h}_{n-1}$ and a candidate state $\tilde{\boldsymbol{h}}_n$. This interpolation is governed by an update gate $\boldsymbol{u}_n$, which controls how much of the new candidate information is integrated. This gating mechanism helps mitigate the vanishing gradient problem in recurrent architectures [60, 49]. The GRU update equations are as follows:

$$\begin{aligned}
\boldsymbol{u}_n &= \text{sigmoid}\left(W_{\text{g}}[\boldsymbol{h}_{n-1}; \boldsymbol{\sigma}_{n-1}] + \boldsymbol{b}_{\text{g}}\right), \\
\boldsymbol{r}_n &= \text{sigmoid}\left(W_{\text{r}}[\boldsymbol{h}_{n-1}; \boldsymbol{\sigma}_{n-1}] + \boldsymbol{b}_{\text{r}}\right), \\
\tilde{\boldsymbol{h}}_n &= \tanh\left(\boldsymbol{r}_n \odot (W_{\text{h}}\boldsymbol{h}_{n-1} + \boldsymbol{b}_{\text{h}}) + W_{\text{in}}\boldsymbol{\sigma}_{n-1} + \boldsymbol{b}_{\text{in}}\right), \\
\boldsymbol{h}_n &= (1 - \boldsymbol{u}_n) \odot \boldsymbol{h}_{n-1} + \boldsymbol{u}_n \odot \tilde{\boldsymbol{h}}_n.
\end{aligned}$$

Here, '$\odot$' denotes the element-wise (Hadamard) product, and 'sigmoid' and 'tanh' refer to the standard activation functions. The reset gate $\boldsymbol{r}_n$ controls how much of the past information (i.e., $\boldsymbol{h}_{n-1}$) is used when computing the candidate hidden state. Note that the weight matrices $W_{\text{g}}, W_{\text{r}}, W_{\text{h}}, W_{\text{in}}$ and biases $\boldsymbol{b}_{\text{g}}, \boldsymbol{b}_{\text{r}}, \boldsymbol{b}_{\text{h}}, \boldsymbol{b}_{\text{in}}$ are trainable parameters of the one-dimensional GRU cell.

In the two dimensional case (2D RNN) [18], to compute the hidden state $\boldsymbol{h}_{i,j}$, we first construct a candidate hidden state $\tilde{\boldsymbol{h}}_{i,j}$ based on a summary of neighboring hidden states and inputs. An update gate $\boldsymbol{u}_{i,j}$ then determines how much of this candidate state is incorporated into the final hidden state versus how much of the neighboring hidden state information is retained. The two-dimensional recursion relation is defined as follows [19, 21]:

$$\begin{aligned}
\tilde{\boldsymbol{h}}_{i,j} &= \tanh\Big(W[\boldsymbol{h}'_{i,j}; \boldsymbol{\sigma}'_{i,j}] + \boldsymbol{b}\Big), \\
\boldsymbol{u}_{i,j} &= \text{sigmoid}\Big(W_{\text{g}}[\boldsymbol{h}'_{i,j}; \boldsymbol{\sigma}'_{i,j}] + \boldsymbol{b}_{\text{g}}\Big), \\
\boldsymbol{h}_{i,j} &= \boldsymbol{u}_{i,j} \odot \tilde{\boldsymbol{h}}_{i,j} + (1 - \boldsymbol{u}_{i,j}) \odot (U\boldsymbol{h}'_{i,j}).
\end{aligned}$$

Here '$\odot$' denotes the element-wise (Hadamard) product. The vector $\boldsymbol{h}'_{i,j}$ is a concatenation of the neighbouring hidden states $\boldsymbol{h}_{i-(-1)^j,j}, \boldsymbol{h}_{i,j-1}$. The same definition also holds for $\boldsymbol{\sigma}'_{i,j}$. Note that the index $i-(-1)^j$ is used to ensure compatibility of the two-dimensional recursion relation with the zigzag sampling path. The weight matrices W, W_{g}, U and biases $\boldsymbol{b}, \boldsymbol{b}_{\text{g}}$ are trainable parameters of the two-dimensional GRU cell. We finally note that, before applying the Softmax layer, we apply a gated linear unit (GLU) layer [14, 48] on the hidden state as follows:

$$\mathbf{h}'_{i,j} = (W_1\mathbf{h}_{i,j} + \mathbf{b}_1) \odot \text{sigmoid}(W_2\mathbf{h}_{i,j} + \mathbf{b}_2),$$

where the weights $W_1, W_2 \in \mathbb{R}^{d_{\text{h}} \times d_{\text{model}}}$ and biases $\mathbf{b}_1, \mathbf{b}_2 \in\in \mathbb{R}^{d_{\text{model}}}$. Note that d_{model} is a hyperparameter that we choose as $d_{\text{model}} = d_{\text{h}}$.

B Hyperparameters

Tab. 2 summarizes the hyperparameters used for training the different models on all benchmark Hamiltonians. Note that we trained the Static and Adaptive RNNs for the same number of steps, except for the early stopping variant of the Adaptive scheme. The early stopping variant is trained until the criterion triggers a stop, i.e., it runs for a variable number of epochs.

Adam optimizer [23] is used as the standard parameter optimizer in all benchmarks. This choice requires maintaining momentum throughout the Adaptive training setup by carrying an optimizer state. To maintain the information from the smaller model faithfully, the information in the optimizer state is carried over to the new model. This step requires encapsulating this data into a higher-dimensional optimizer state. When progressing from one model in the sequence to the next, we carry over both the parameters and momentum states.

Benchmark	Model	Hyperparameter	Value
1D TFIM	Static	Architecture	1D pRNN with fixed d_{h}
		Number of samples	100
		Training iterations	$50,000$
		Learning rate	5×10^{-4}
		System Sizes	20, 40, 60, 80, 100
		d_{h}	256
	Adaptive	Architecture	1D pRNN with d_{model} and d_{h} doubling every 6250 steps
		Number of samples	100
		Training iterations	$50,000$
		Learning rate	5×10^{-3} until 25,000 steps, then 5×10^{-4}
		System Sizes	20, 40, 60, 80, 100
		Starting d_{h}	2
		Final d_{h}	256
2D Heisenberg	Static	Architecture	2D pRNN with fixed d_{model} and d_{h}
		Number of samples	500
		Training iterations	$200,000$
		Learning rate	$5 \times 10^{-4} \times \left(1 + \frac{t}{5000}\right)^{-1}$
		System Sizes	6×6
		d_{h}	256
	Adaptive	Architecture	2D pRNN with d_{h} doubling every 25,000 steps
		Number of samples	500
		Training iterations	200,000
		Learning rate	$5 \times 10^{-4} \times \left(1 + \frac{t-100,000}{5000} \times \left\lfloor \frac{t}{100,000} \right\rfloor\right)^{-1}$
		System Sizes	6×6
		Starting d_{h}	32
		Final d_{h}	256
	Early Stopping	Architecture	2D pRNN with d_{h} doubling from early stopping
		Number of samples	500
		Training iterations	Variable
		Learning rate	0.01 until $d_{\mathrm{h}} = 64$ then 0.0001
		Starting d_{h}	2
		Final d_{h}	256
		Early Stopping Criterion	Variance
		δ	$10^{-\frac{1}{2}\log_2(d_{\mathrm{h}})}$
		Patience	10000
Long-range TFIM	Static	Architecture	1D pRNN with fixed d_{h}
		Number of samples	500
		Training iterations	2,000
		Learning rate	10^{-3}
		d_{h}	256
	Adaptive	Architecture	1D pRNN with d_{h} doubling every 200 steps
		Number of samples	500
		Training iterations	2,000
		Learning rate	10^{-3}
		Starting d_{h}	2
		Final d_{h}	256
Cluster State	Static	Architecture	1D cRNN with fixed d_{h}
		Number of samples	100
		Training iterations	10,000
		Learning rate	10^{-4}
		d_{h}	256
	Adaptive	Architecture	1D cRNN with d_{h} doubling every 1,000 steps
		Number of samples	100
		Training iterations	10,000
		Learning rate	10^{-3}
		Starting d_{h}	2
		Final d_{h}	256

Table 2: A summary of the Hyperparameters used on the four different benchmarks in this Paper.

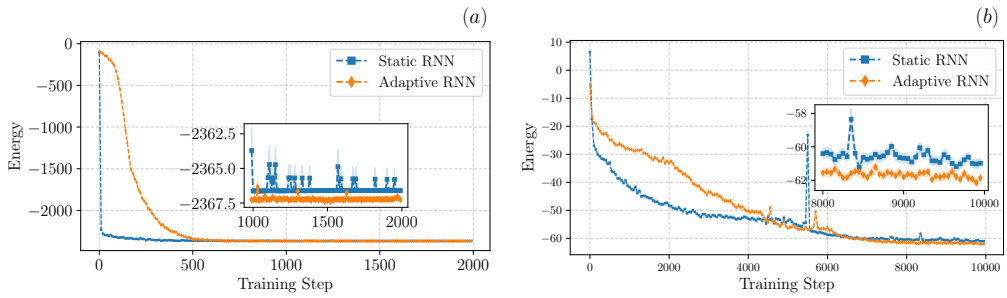

Figure 2: A comparison in terms of the variational energy between the Static RNN and the Adaptive RNN on two different models. (a) Long-range 1D TFIM with $N = 80$ spins and $\alpha = 0.1$. Here, the Adaptive RNN size is changed every 200 steps. (b) 1D Cluster state for $N = 64$ spins. Note that the Adaptive RNN size is changed every 1000 steps. For both panels, the lower the energy, the better. We observe a lower variational energy for the Adaptive RNN at convergence for the two Hamiltonians.

C Additional Results

In addition to the two benchmarks presented in this paper, we also explored the influence of our Adaptive scheme when training on long-range models.

C.1 Long-Range Transverse-field Ferromagnetic Ising Model

We start with the long-range TFIM given by the following Hamiltonian:

$$\hat{H}_{\mathrm{LR-TFIM}} = -\sum_{1\leq i<j\leq N} \frac{1}{|i-j|^{\alpha}} \hat{\sigma}_i^z \hat{\sigma}_j^z - \Gamma \sum_i \hat{\sigma}_i^x, \tag{8}$$

where α is a tunable parameter. This model has been experimentally realized using trapped ion quantum simulators, where the interaction strength decays with distance as a power law $J_{ij} \propto 1/|i-j|^{\alpha}$, with tunable α [58]. Additional realizations have been achieved using Rydberg atom arrays, allowing exploration of constrained and long-range Ising-type interactions [3]. Additionally, this model corresponds to the short-range 1D TFIM, introduced earlier, when $\alpha \to \infty$. In our simulations, we choose $\alpha = 0.1, \Gamma = 1$, and $N = 80$ spins as a playground for comparing our Adaptive RNN against the Static RNN.

Our results are summarized in Fig. 2(a). Here we observe that near convergence, the Adaptive RNN energy $-2367.244(2)$ is lower than the Static RNN energy $-2366.55(2)$ with a difference of about $0.69(2)$, even though the Static RNN converged faster in the initial phase of training. This result suggests that Adaptive RNNs are more effective at circumventing excited states compared to Static RNNs. One possible explanation for this result can be related to Adaptive RNNs starting with a small hidden dimension, reflecting a low entanglement structure. This property is an agreement with the ground state of Ising long range, for $\alpha = 0.1$, belonging to the mean-field universality class [24, 15]. Static RNNs, on the other hand, start with a large hidden dimension which are likely biased towards learning entangled states, as suggested by the RNN entanglement area law [27, 55]. This finding, along with the previous benchmarks, highlights the importance of starting with a small hidden dimension at the beginning of a variational calculation to learn simple quantum states in the initial phase of training.

C.2 1D Cluster State

Finally, we study ground states with a sign structure. In particular, we confirm a similar observation to the previous benchmark using a cRNN instead of a pRNN. To achieve this goal, we study the 1D Cluster State Hamiltonian:

$$\hat{H}_{\mathrm{Cluster}} = -\sum_{k=2}^{n-2} X_{k-1} Z_k X_{k+1} - Z_1 X_2 - X_{n-1} X_n - X_{n-2} Z_{n-1} Z_n.$$

This Hamiltonian is a prototypical model for measurement-based quantum computation, where entanglement is generated via multi-qubit stabilizer terms (e.g. $X_{k-1}Z_kX_{k+1}$) rather than dynamic evolution [42]. Its ground state belongs to a gapped, symmetry-protected topological (SPT) phase representative in 1D, with nontrivial edge modes and robustness under certain perturbations [16]. Note that this Hamiltonian is non-stoquastic. As a result, we use a cRNN to model the phase of its ground state [18].

We focus our comparison between the Static RNN and the Adaptive RNN on this Hamiltonian with $N = 64$ spins, adopted in Ref. [55], for a y-rotation angle $\theta = 0$, such that $R_y^\dagger(\theta)\hat{H}_{\text{Cluster}}R_y(\theta) = \hat{H}_{\text{Cluster}}$ where $R_y(\theta)$ is the y-rotation unitary. This point has a ground state with the largest conditional mutual information (CMI) [55], indicating long-range conditional correlations. As a result, $\theta = 0$ is the hardest point to learn by the RNN [55]. Similar to the long-range TFIM model, the runtime for both Adaptive and Static RNNs is similar (around 18 minutes for both), however we note that the Adaptive RNN takes only 13 minutes (around 8000 training steps) to outperform the Static RNN variational energies as illustrated in the inset of Fig. 2(b).

Our results, illustrated in Fig. 2(b), demonstrate a noticeable difference at the first decimal point between the Static RNN and the Adaptive RNN despite the faster convergence of the Static RNN. This result confirms once again the ability of the Adaptive RNN to better avoid local minima in the VMC optimization landscape. Additionally, even though we do not obtain the true ground state energy -64 [55], our Adaptive RNN energy is within a relative error of 3.3×10^{-2}, which is smaller than that of our Static RNN (5.0×10^{-2}) and less than half of the relative error obtained by the Static RNN in Ref. [55]. These results also highlight the advantage provided by the Adaptive scheme in the presence of a non-trivial sign structure in the ground state. The latter is known to induce a rugged optimization landscape [5], and our results suggest that the Adaptive training scheme is better equipped to navigate such landscapes.

References

[1] R. Assaraf and M. Caffarel. Zero-variance zero-bias principle for observables in quantum monte carlo: Application to forces. *The Journal of Chemical Physics*, 119(20):10536–10552, Nov. 2003. ISSN 1089-7690. doi: 10.1063/1.1621615. URL http://dx.doi.org/10.1063/1.1621615.

[2] F. Becca and S. Sorella. *Quantum Monte Carlo Approaches for Correlated Systems*. Cambridge University Press, 2017. doi: 10.1017/9781316417041.

[3] H. Bernien, S. Schwartz, A. Keesling, H. Levine, A. Omran, H. Pichler, S. Choi, A. S. Zibrov, M. Endres, M. Greiner, V. Vuletić, and M. D. Lukin. Probing many-body dynamics on a 51-atom quantum simulator. *Nature*, 551(7682):579–584, Nov. 2017. ISSN 1476-4687. doi: 10.1038/nature24622. URL http://dx.doi.org/10.1038/nature24622.

[4] S. Bravyi, D. P. Divincenzo, R. Oliveira, and B. M. Terhal. The complexity of stoquastic local hamiltonian problems. *Quantum Info. Comput.*, 8(5):361–385, May 2008. ISSN 1533-7146. URL http://dl.acm.org/citation.cfm?id=2011772.2011773.

[5] M. Bukov, M. Schmitt, and M. Dupont. Learning the ground state of a non-stoquastic quantum hamiltonian in a rugged neural network landscape. *SciPost Physics*, 10(6), June 2021. ISSN 2542-4653. doi: 10.21468/scipostphys.10.6.147. URL http://dx.doi.org/10.21468/SciPostPhys.10.6.147.

[6] Z. Cai and J. Liu. Approximating quantum many-body wave functions using artificial neural networks. *Phys. Rev. B*, 97:035116, Jan 2018. doi: 10.1103/PhysRevB.97.035116. URL https://link.aps.org/doi/10.1103/PhysRevB.97.035116.

[7] L. Capriotti. Quantum Effects and Broken Symmetries in Frustrated Antiferromagnets, Dec. 2001. URL http://arxiv.org/abs/cond-mat/0112207. arXiv:cond-mat/0112207.

[8] G. Carleo and M. Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017. ISSN 0036-8075. URL https://www.jstor.org/stable/24918355. Publisher: American Association for the Advancement of Science.

[9] C. Casert, T. Vieijra, S. Whitelam, and I. Tamblyn. Dynamical large deviations of two-dimensional kinetically constrained models using a neural-network state ansatz. *Phys. Rev. Lett.*, 127:120602, Sep 2021. doi: 10.1103/PhysRevLett.127.120602. URL https://link.aps.org/doi/10.1103/PhysRevLett.127.120602.

[10] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. URL https://arxiv.org/abs/1406.1078.

[11] K. Choo, G. Carleo, N. Regnault, and T. Neupert. Symmetries and many-body excitations with neural-network quantum states. *Phys. Rev. Lett.*, 121:167204, Oct 2018. doi: 10.1103/PhysRevLett.121.167204. URL https://link.aps.org/doi/10.1103/PhysRevLett.121.167204.

[12] K. Choo, T. Neupert, and G. Carleo. Two-dimensional frustrated J_1-J_2 model studied with neural network quantum states. *Phys. Rev. B*, 100:125124, Sep 2019. doi: 10.1103/PhysRevB.100.125124. URL https://link.aps.org/doi/10.1103/PhysRevB.100.125124.

[13] B. A. Cipra. An introduction to the ising model. *The American Mathematical Monthly*, 94 (10):937–959, 1987. ISSN 00029890, 19300972. URL http://www.jstor.org/stable/2322600.

[14] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks, 2017. URL https://arxiv.org/abs/1612.08083.

[15] N. Defenu, T. Donner, T. Macrì, G. Pagano, S. Ruffo, and A. Trombettoni. Long-range interacting quantum systems. *Reviews of Modern Physics*, 95(3), Aug. 2023. ISSN 1539-0756. doi: 10.1103/revmodphys.95.035002. URL http://dx.doi.org/10.1103/RevModPhys.95.035002.

[16] A. C. Doherty and S. D. Bartlett. Identifying phases of quantum many-body systems that are universal for quantum computation. *Physical Review Letters*, 103(2), July 2009. ISSN 1079-7114. doi: 10.1103/physrevlett.103.020506. URL http://dx.doi.org/10.1103/PhysRevLett.103.020506.

[17] M. Ganahl, J. Beall, M. Hauru, A. G. Lewis, T. Wojno, J. H. Yoo, Y. Zou, and G. Vidal. Density matrix renormalization group with tensor processing units. *PRX Quantum*, 4:010317, Feb 2023. doi: 10.1103/PRXQuantum.4.010317. URL https://link.aps.org/doi/10.1103/PRXQuantum.4.010317.

[18] M. Hibat-Allah, M. Ganahl, L. E. Hayward, R. G. Melko, and J. Carrasquilla. Recurrent neural network wave functions. *Physical Review Research*, 2(2), jun 2020. doi: 10.1103/physrevresearch.2.023358. URL https://doi.org/10.1103%2Fphysrevresearch.2.023358.

[19] M. Hibat-Allah, R. G. Melko, and J. Carrasquilla. Investigating topological order using recurrent neural networks. *Phys. Rev. B*, 108:075152, Aug 2023. doi: 10.1103/PhysRevB.108.075152. URL https://link.aps.org/doi/10.1103/PhysRevB.108.075152.

[20] M. Hibat-Allah, R. G. Melko, and J. Carrasquilla. Supplementing recurrent neural network wave functions with symmetry and annealing to improve accuracy, 2024. URL https://arxiv.org/abs/2207.14314.

[21] M. Hibat-Allah, E. Merali, G. Torlai, R. G. Melko, and J. Carrasquilla. Recurrent neural network wave functions for rydberg atom arrays on kagome lattice, 2024. URL https://arxiv.org/abs/2405.20384.

[22] M. Hibat-Allah, E. Merali, G. Torlai, R. G. Melko, and J. Carrasquilla. Recurrent neural network wave functions for rydberg atom arrays on kagome lattice, 2024. URL https://arxiv.org/abs/2405.20384.

[23] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014.

[24] T. Koffel, M. Lewenstein, and L. Tagliacozzo. Entanglement entropy for the long-range ising chain in a transverse field. *Physical Review Letters*, 109(26), Dec. 2012. ISSN 1079-7114. doi: 10.1103/physrevlett.109.267203. URL http://dx.doi.org/10.1103/PhysRevLett.109.267203.

[25] H. Lange, A. Van de Walle, A. Abedinnia, and A. Bohrdt. From architectures to applications: a review of neural quantum states. *Quantum Science and Technology*, 9(4):040501, sep 2024. doi: 10.1088/2058-9565/ad7168. URL https://dx.doi.org/10.1088/2058-9565/ad7168.

[26] H. Lange, G. Bornet, G. Emperauger, C. Chen, T. Lahaye, S. Kienle, A. Browaeys, and A. Bohrdt. Transformer neural networks and quantum simulators: a hybrid approach for simulating strongly correlated systems. *Quantum*, 9:1675, Mar. 2025. ISSN 2521-327X. doi: 10.22331/q-2025-03-26-1675. URL https://doi.org/10.22331/q-2025-03-26-1675.

[27] Y. Levine, O. Sharir, N. Cohen, and A. Shashua. Quantum entanglement in deep learning architectures. *Phys. Rev. Lett.*, 122:065301, Feb 2019. doi: 10.1103/PhysRevLett.122.065301. URL https://link.aps.org/doi/10.1103/PhysRevLett.122.065301.

[28] Z. C. Lipton, J. Berkowitz, and C. Elkan. A critical review of recurrent neural networks for sequence learning, 2015. URL https://arxiv.org/abs/1506.00019.

[29] Z. Liu and E. Manousakis. Variational calculations for the square-lattice quantum antiferromagnet. *Physical Review B*, 40(16):11437–11440, Dec. 1989. ISSN 0163-1829. doi: 10.1103/PhysRevB.40.11437. URL https://link.aps.org/doi/10.1103/PhysRevB.40.11437.

[30] D. Luo, Z. Chen, K. Hu, Z. Zhao, V. M. Hur, and B. K. Clark. Gauge-invariant and anyonic-symmetric autoregressive neural network for quantum lattice models. *Phys. Rev. Res.*, 5:013216, Mar 2023. doi: 10.1103/PhysRevResearch.5.013216. URL https://link.aps.org/doi/10.1103/PhysRevResearch.5.013216.

[31] M. Mareschal. The early years of quantum monte carlo (1): the ground state. *The European Physical Journal H*, 46(1), 2021. ISSN 2102-6459. doi: 10.1140/epjh/s13129-021-00017-6.

[32] W. Marshall. Antiferromagnetism. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 232(1188):48–68, 1955. ISSN 0080-4630. URL https://www.jstor.org/stable/99682. Publisher: The Royal Society.

[33] W. L. McMillan. Ground state of liquid he^4. *Phys. Rev.*, 138:A442–A451, Apr 1965. doi: 10.1103/PhysRev.138.A442. URL https://link.aps.org/doi/10.1103/PhysRev.138.A442.

[34] M. Medvidović and J. R. Moreno. Neural-network quantum states for many-body physics. *The European Physical Journal Plus*, 139(7):631, Jul 2024. ISSN 2190-5444. doi: 10.1140/epjp/s13360-024-05311-y. URL https://doi.org/10.1140/epjp/s13360-024-05311-y.

[35] R. G. Melko and J. Carrasquilla. Language models for quantum simulation. *Nature Computational Science*, 4(1):11–18, Jan 2024. ISSN 2662-8457. doi: 10.1038/s43588-023-00578-0. URL https://doi.org/10.1038/s43588-023-00578-0.

[36] A. M. Mood. *Introduction to the Theory of Statistics (Theorem 2).* McGraw-hill, 1950.

[37] M. S. Moss, R. Wiersema, M. Hibat-Allah, J. Carrasquilla, and R. G. Melko. Leveraging recurrence in neural network wavefunctions for large-scale simulations of heisenberg antiferromagnets: the square lattice, 2025. URL https://arxiv.org/abs/2502.17144.

[38] M. S. Moss, R. Wiersema, M. Hibat-Allah, J. Carrasquilla, and R. G. Melko. Leveraging recurrence in neural network wavefunctions for large-scale simulations of heisenberg antiferromagnets: the triangular lattice, 2025. URL https://arxiv.org/abs/2505.20406.

[39] Y. Nomura. Helping restricted boltzmann machines with quantum-state representation by restoring symmetry. *Journal of Physics: Condensed Matter*, 33(17):174003, Apr. 2021. ISSN 1361-648X. doi: 10.1088/1361-648x/abe268. URL http://dx.doi.org/10.1088/1361-648X/abe268.

[40] Y. Nomura and M. Imada. Dirac-type nodal spin liquid revealed by refined quantum many-body solver using neural-network wave function, correlation ratio, and level spectroscopy. *Phys. Rev. X*, 11:031034, Aug 2021. doi: 10.1103/PhysRevX.11.031034. URL https://link.aps.org/doi/10.1103/PhysRevX.11.031034.

[41] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada. Restricted boltzmann machine learning for solving strongly correlated quantum systems. *Phys. Rev. B*, 96:205152, Nov 2017. doi: 10.1103/PhysRevB.96.205152. URL https://link.aps.org/doi/10.1103/PhysRevB.96.205152.

[42] R. Raussendorf and H. Briegel. Computational model underlying the one-way quantum computer, 2002. URL https://arxiv.org/abs/quant-ph/0108067.

[43] R. Rende, L. L. Viteritti, L. Bardone, F. Becca, and S. Goldt. A simple linear algebra identity to optimize large-scale neural network quantum states. *Communications Physics*, 7(260), 2024. doi: 10.1038/s42005-024-01732-4. URL https://www.nature.com/articles/s42005-024-01732-4.

[44] C. Roth. Iterative Retraining of Quantum Spin Models Using Recurrent Neural Networks, Mar. 2020. URL http://arxiv.org/abs/2003.06228. arXiv:2003.06228.

[45] A. W. Sandvik and J. Kurkijärvi. Quantum Monte Carlo simulation method for spin systems. *Physical Review B*, 43(7):5950–5961, Mar. 1991. ISSN 0163-1829, 1095-3795. doi: 10.1103/PhysRevB.43.5950. URL https://link.aps.org/doi/10.1103/PhysRevB.43.5950.

[46] M. Schmitt and M. Heyl. Quantum many-body dynamics in two dimensions with artificial neural networks. *Physical Review Letters*, 125(10), Sept. 2020. ISSN 1079-7114. doi: 10.1103/physrevlett.125.100503. URL http://dx.doi.org/10.1103/PhysRevLett.125.100503.

[47] U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, Jan. 2011. ISSN 0003-4916. doi: 10.1016/j.aop.2010.09.012. URL http://dx.doi.org/10.1016/j.aop.2010.09.012.

[48] N. Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

[49] H. Shen. Mutual information scaling and expressive power of sequence models, 2019. URL https://arxiv.org/abs/1905.04271.

[50] J. A. Sobral, M. Perle, and M. S. Scheurer. Physics-informed transformers for electronic quantum states, 2024. URL https://arxiv.org/abs/2412.12248.

[51] K. Sprague and S. Czischek. Variational monte carlo with large patched transformers. *Communications Physics*, 7(1), Mar. 2024. ISSN 2399-3650. doi: 10.1038/s42005-024-01584-y. URL http://dx.doi.org/10.1038/s42005-024-01584-y.

[52] S. R. White and A. L. Chernyshev. Neél Order in Square and Triangular Lattice Heisenberg Models. *Physical Review Letters*, 99(12):127004, Sept. 2007. ISSN 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.99.127004. URL https://link.aps.org/doi/10.1103/PhysRevLett.99.127004.

[53] D. Wu, R. Rossi, F. Vicentini, and G. Carleo. From tensor-network quantum states to tensorial recurrent neural networks. *Physical Review Research*, 5(3), July 2023. ISSN 2643-1564. doi: 10.1103/physrevresearch.5.l032001. URL http://dx.doi.org/10.1103/PhysRevResearch.5.L032001.

[54] D. Wu, R. Rossi, F. Vicentini, N. Astrakhantsev, F. Becca, X. Cao, J. Carrasquilla, F. Ferrari, A. Georges, M. Hibat-Allah, M. Imada, A. M. Läuchli, G. Mazzola, A. Mezzacapo, A. Millis, J. R. Moreno, T. Neupert, Y. Nomura, J. Nys, O. Parcollet, R. Pohle, I. Romero, M. Schmid, J. M. Silvester, S. Sorella, L. F. Tocchio, L. Wang, S. R. White, A. Wietek, Q. Yang, Y. Yang, S. Zhang, and G. Carleo. Variational benchmarks for quantum many-body problems. *Science*, 386(6719):296–301, 2024. doi: 10.1126/science.adg9774. URL https://www.science.org/doi/abs/10.1126/science.adg9774.

[55] T.-H. Yang, M. Soleimanifar, T. Bergamaschi, and J. Preskill. When can classical neural networks represent quantum states?, 2024. URL https://arxiv.org/abs/2410.23152.

[56] R. Zen and S. Bressan. Transfer learning for larger, broader, and deeper neural-network quantum states. In C. Strauss, G. Kotsis, A. M. Tjoa, and I. Khalil, editors, *Database and Expert Systems Applications*, pages 207–219, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86475-0.

[57] R. Zen, L. My, R. Tan, F. Hébert, M. Gattobigio, C. Miniatura, D. Poletti, and S. Bressan. Transfer learning for scalability of neural-network quantum states. *Physical Review E*, 101(5), May 2020. ISSN 2470-0053. doi: 10.1103/physreve.101.053301. URL http://dx.doi.org/10.1103/PhysRevE.101.053301.

[58] J. Zhang, G. Pagano, P. W. Hess, A. Kyprianidis, P. Becker, H. Kaplan, A. V. Gorshkov, Z.-X. Gong, and C. Monroe. Observation of a many-body dynamical phase transition with a 53-qubit quantum simulator. *Nature*, 551(7682):601–604, Nov. 2017. ISSN 1476-4687. doi: 10.1038/nature24654. URL http://dx.doi.org/10.1038/nature24654.

[59] Y.-H. Zhang and M. Di Ventra. Transformer quantum state: A multipurpose model for quantum many-body problems. *Physical Review B*, 107(7), Feb. 2023. ISSN 2469-9969. doi: 10.1103/physrevb.107.075147. URL http://dx.doi.org/10.1103/PhysRevB.107.075147.

[60] G.-B. Zhou, J. Wu, C.-L. Zhang, and Z.-H. Zhou. Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing*, 13(3):226–234, Jun 2016. ISSN 1751-8520. doi: 10.1007/s11633-016-1006-2. URL https://doi.org/10.1007/s11633-016-1006-2.