Variational Monte Carlo with large patched transformers

Check for updates

Kyle Sprague & Stefanie Czischek ✉

Large language models, like transformers, have recently demonstrated immense powers in text and image generation. This success is driven by the ability to capture long-range correlations between elements in a sequence. The same feature makes the transformer a powerful wavefunction ansatz that addresses the challenge of describing correlations in simulations of qubit systems. Here we consider two-dimensional Rydberg atom arrays to demonstrate that transformers reach higher accuracies than conventional recurrent neural networks for variational ground state searches. We further introduce large, patched transformer models, which consider a sequence of large atom patches, and show that this architecture significantly accelerates the simulations. The proposed architectures reconstruct ground states with accuracies beyond state-of-the-art quantum Monte Carlo methods, allowing for the study of large Rydberg systems in different phases of matter and at phase transitions. Our high-accuracy ground state representations at reasonable computational costs promise new insights into general large-scale quantum many-body systems.

The advent of artificial neural network quantum states marks a turn in the field of numerical simulations for quantum many-body systems[1–6]. Since then, artificial neural networks are commonly used as a general wavefunction ansatz to find ground states of a given Hamiltonian[1,7–9], to reconstruct quantum states from a set of projective measurements[2,3,10–16], or to model dynamics in open and closed quantum systems[1,17–21]. The powers and limitations of different network architectures, such as restricted Boltzmann machines[1–3,9,12,15,22,23], recurrent neural networks (RNNs)[7,8,13,24,25], or the PixelCNN[26], have been widely explored on several physical models. In addition, modified network architectures[27], the explicit inclusion of symmetries[13,28–30], and the pre-training on a limited amount of measurement data[8,31] have shown improved performances.

A particularly promising choice are autoregressive neural networks such as the PixelCNN[26] and RNNs[7,8,13,28], which can find ground states and reconstruct quantum states from data with high accuracies. These models consider qubit systems in sequential order, providing an efficient wavefunction encoding. However, these setups experience limitations for systems with strong correlations between qubits far apart in the sequence, which, for example, happens for two-dimensional qubit systems[7,8,26].

Similar to the RNN or PixelCNN approaches, transformer (TF) models[32] can be used as a wavefunction ansatz by considering a sequence of qubits[14,33–38] or for simulating quantum dynamics[39,40]. Due to their non-recurrent nature and the ability to highlight the influence of specific previous sequence elements, TF models perform better at covering long-range

correlations[32], promising to overcome the limitations of RNNs and PixelCNNs[34,35]. In this work, we analyze the performance of the TF wavefunction ansatz for variational ground state searches and observe improved accuracies in the representation of quantum states compared to the RNN approach.

Inspired by the introduction of the vision transformer, which enables the efficient application of TF models for image processing and generation tasks[41], and by previous works in the field[25,34,35], we study RNN and TF models that consider sequences of patches of qubits. This approach reduces the sequence length and thus the computational cost significantly, while accurately capturing correlations within the patch. For further improvements we introduce large, patched transformers (LPTF) consisting of a powerful patched TF model followed by a computationally efficient patched RNN that breaks large inputs into smaller sub-patches. This architecture allows for an efficient consideration of large patches in the input sequence of the TF network, further reducing the sequence length.

We benchmark the LPTF architecture on two-dimensional arrays of Rydberg atoms, whose recently demonstrated experimental controllability makes them promising candidates for high-performance quantum computation and quantum simulation[42–52]. Furthermore, quantum Monte Carlo methods can model Rydberg atom systems[52,53], and we use such simulations to determine the performance of different network models.

Analyzing different shapes and sizes of input patches, we demonstrate that LPTFs can represent ground states of Rydberg atom arrays with

Department of Physics, University of Ottawa, Ottawa, Ontario K1N 6N5, Canada.
✉e-mail: stefanie.czischek@uottawa.ca

accuracies beyond the RNN ansatz and traditional quantum Monte Carlo simulations, while requiring reasonable computational costs. Our results are consistent in different phases of matter and at quantum phase transitions. While we show that LPTFs can significantly improve numerical investigations of the considered Rydberg models, the introduced network model can similarly be applied to general qubit systems. The results presented in this work propose that the LPTF model can substantially advance numerical studies of quantum many-body physics.

Results

Rydberg atom arrays

Rydberg atoms, which we use as a qubit model to benchmark our numerical approaches, can be prepared in the ground state $|g\rangle$ and in a highly excited (Rydberg) state $|r\rangle$[42–46,48,49]. We specifically consider the atoms arranged on square lattices of different system sizes, as illustrated in Fig. 1a.

The system of $N = L \times L$ atoms is described by the Rydberg Hamiltonian[42,43],

$$\widehat{\mathcal{H}} = -\frac{\Omega}{2} \sum_{i=1}^{N} \hat{\sigma}_i^x - \delta \sum_{i=1}^{N} \hat{n}_i + \sum_{i,j} V_{i,j} \hat{n}_i \hat{n}_j, \qquad (1)$$

with the detuning δ and the Rabi oscillation with frequency Ω generated by an external laser driving. Here we use the off-diagonal operator $\hat{\sigma}_i^x = |g\rangle_i\langle r|_i + |r\rangle_i\langle g|_i$ and the occupation number operator $\hat{n}_i = |r\rangle_i\langle r|_i$. The last term in the Hamiltonian describes a van-der-Waals interaction between atoms at positions r_i and r_j, with $V_{i,j} = \Omega R_b^6/|r_i - r_j|^6$, and Rydberg blockade radius R_b. We further choose the lattice spacing $a = 1$. By tuning the free parameters in the Rydberg Hamiltonian, the system can be prepared in various phases of matter, separated by different kinds of phase transitions[46–48,51,52]. The Rydberg Hamiltonian is stoquastic[54], resulting in a positive and real-valued ground-state wavefunction[44,48,50]. More details on Rydberg atom arrays are provided in the Methods section.

Recurrent neural networks and transformers

Recurrent neural networks (RNNs) provide a powerful wavefunction ansatz that can variationally find ground state representations of quantum many-body systems[4,6–8,13,25,28,29]. For this, the possibility to naturally encode probability distributions in RNNs allows the representation of squared wavefunction amplitudes $|\Psi(\sigma)|^2$. Samples drawn from the encoded distribution correspond to state configurations that can be interpreted as outputs of projective measurements, as illustrated in Fig. 1d.

To represent the wavefunction amplitudes $\Psi(\sigma) = \langle\sigma|\Psi\rangle$ of a qubit system, such as an array of Rydberg atoms, a sequential order is defined over the system. Each atom is iteratively used as an input to the RNN cell, the core element of the network structure which we choose to be a Gated Recurrent Unit (GRU)[55] inspired by[6,7]. In addition, the RNN cell receives the state of internal hidden units as input. This state is adapted in each iteration and propagated over the input sequence, generating a memory effect. The network output $p_{RNN}(\sigma_i|\sigma_{<i}; \mathcal{W})$ at each iteration can be interpreted as the probability of the next atom σ_i being in either the ground or the Rydberg state, conditioned on the configuration of all previous atoms $\sigma_{<i}$ in the sequence, with variational weights $\mathcal{W}$ in the RNN cell. See Fig. 1d and the Methods section for more details. From this output, the state σ_i of the next atom is sampled and used autoregressively as input in the next RNN iteration, as illustrated in Fig. 1b[4,6,7]. We then train the RNN such that it approximates a target state $\Psi(\sigma)$,

$$\begin{aligned}\Psi_{RNN}(\sigma; \mathcal{W}) &= \sqrt{\prod_{i=1}^{N} p_{RNN}(\sigma_i|\sigma_{<i}; \mathcal{W})} \\ &= \sqrt{p_{RNN}(\sigma; \mathcal{W})} \\ &\approx \Psi(\sigma).\end{aligned} \qquad (2)$$

While we focus on positive, real-valued wave functions in this work, RNNs can represent general wave functions by including complex phases as a second network output[7]. The global phase of the encoded state is then expressed as the sum over single-qubit phases.

The RNN has shown high accuracies for representing ground states of various quantum systems. However, its sequential nature and the encoding of all information in the hidden unit state pose a challenge for capturing long-range correlations[4,6–8,13,28,29]. Here we refer to correlations between atoms that appear far from each other in the RNN sequence but not necessarily in the qubit system. Alternative autoregressive network models, such as the PixelCNN, experience similar limitations. These models cover correlations via convolutions with a kernel of a specific size. However, due to increasing computational costs, kernel sizes are commonly chosen rather small, so that the PixelCNN is as well limited to capturing only local correlations in qubit systems[26]. Specific RNN structures that better match the lattice structure in the considered model, such as two-dimensional RNNs for two-dimensional quantum systems, can overcome this limitation[7,25,28,29]. An alternative approach to improve the representation of long-range correlations is to use transformer (TF) architectures as a wavefunction ansatz[14,33–35]. These provide a similar autoregressive behavior but do not have a recurrent setup and naturally capture all-to-all interactions[32].

While autoregressively using the states of individual atoms as sequential input similar to the RNN, a masked self-attention layer in the TF setup provides trained connections to all previous elements in the sequence[32]. See Fig. 1e and the Methods section for more details. These trainable connections generate all-to-all interactions between the atoms in the system and allow the highlighting of high-impact connections or strong correlations. This setup thus proposes to represent strongly correlated quantum systems with higher accuracy than the RNN model[14,34,35]. As illustrated in Fig. 1c, the TF model outputs probability distributions which provide an autoregressive wavefunction ansatz $\Psi_{TF}(\sigma; \mathcal{W}) = \sqrt{p_{TF}(\sigma; \mathcal{W})}$[14,34,35], as further explained in the Methods section. Similarly to the RNN, the TF network can represent complex-valued wave functions by adding a second output representing the single-qubit phases[7].

In Fig. 2a, b, we compare the performance of RNNs (blue) and TFs (orange) when representing ground states of Rydberg arrays with $N = 8 \times 8$ (a) and $N = 16 \times 16$ (b) atoms. Here we fix $R_b = 7^{1/6} \approx 1.383$ and $\Omega = \delta = 1$, which brings us into the vicinity of the transition between the disordered and the striated phase[48]. We variationally train the network models by minimizing the energy expectation value, corresponding to a variational Monte Carlo method[4,6,23,56], see Methods section. If not stated otherwise, the energy expectation values in this work are evaluated on $N_s = 512$ samples generated from the network, which we consider in mini batches of $K = 256$ samples. We obtained satisfactory results with $d_H = 128$ hidden neurons in the RNN and the equivalent embedding dimension $d_H = 128$ in the TF model. To benchmark the performance of the two models, we show the difference between the ground state energies H_{QMC} obtained from quantum Monte Carlo (QMC) simulations at zero temperature[53], and the energy expectation value,

$$\langle E \rangle = \frac{1}{N_s} \sum_{s=1}^{N_s} H_{loc}(\sigma_s), \qquad (3)$$

extracted from network samples σ_s. Here we use the local energy,

$$H_{loc}(\sigma_s) = \frac{\langle\sigma_s|\widehat{\mathcal{H}}|\Psi_{\mathcal{W}}\rangle}{\langle\sigma_s|\Psi_{\mathcal{W}}\rangle}, \qquad (4)$$

with $|\Psi_{\mathcal{W}}\rangle$ denoting the wavefunction encoded in either the RNN or the TF network, as discussed in the Methods section. In the QMC simulations, we use the stochastic series expansion approach presented in[53] and evaluate the expectation value on $N_s = 7 \times 10^4$ samples generated from seven independent sample chains. Both system sizes show that TFs converge to the ground

state energy within fewer training iterations than the RNN. Additionally, for the larger system in Fig. 2b, TFs outperform RNNs significantly and reach higher accuracies in the ground state energy. This result demonstrates the expected improved performance.

We, however, also find that this enhancement comes at the cost of increased computational runtimes τ in hours (h) for 2×10^4 training iterations. With $\tau \approx 1.5$h and $\tau \approx 16$h for $N = 8 \times 8$ and $N = 16 \times 16$ atoms, RNNs process much faster than TFs with $\tau \approx 9.5$h and $\tau \approx 144$h, respectively. Figure 2a, b suggest stopping the TF training after fewer iterations due to the faster convergence, but the computational runtime is still too long to allow scaling to large system sizes.

We obtained QMC runtimes as $\tau \approx 18$h for $N = 8 \times 8$ and $\tau \approx 24$h for $N = 16 \times 16$ for a single run generating $N_s = 10^4$ samples, showing a more efficient scaling with system size than the network simulations. This behavior can be understood when considering the scaling of the computational cost for generating an individual sample, which is $\mathcal{O}(N)$ for the RNN and QMC, and $\mathcal{O}(N^2)$ for the TF. In addition, the network models need to evaluate energy expectation values in each training iteration, which comes at complexity $\mathcal{O}(N^2)$ for the RNN and at complexity $\mathcal{O}(N^3)$ for the TF, see Methods for more details. However, due to its non-recurrent setup, the TF enables a parallelization of the energy expectation value evaluation, which is not possible for the RNN ansatz, as further discussed in the Methods. The computational complexity for QMC scales as $\mathcal{O}(N)$ for both sampling and energy evaluation[53]. Thus, while the QMC requires longer runtimes than the RNN for small system sizes, it is expected to outperform both the RNN and the TF for larger systems.
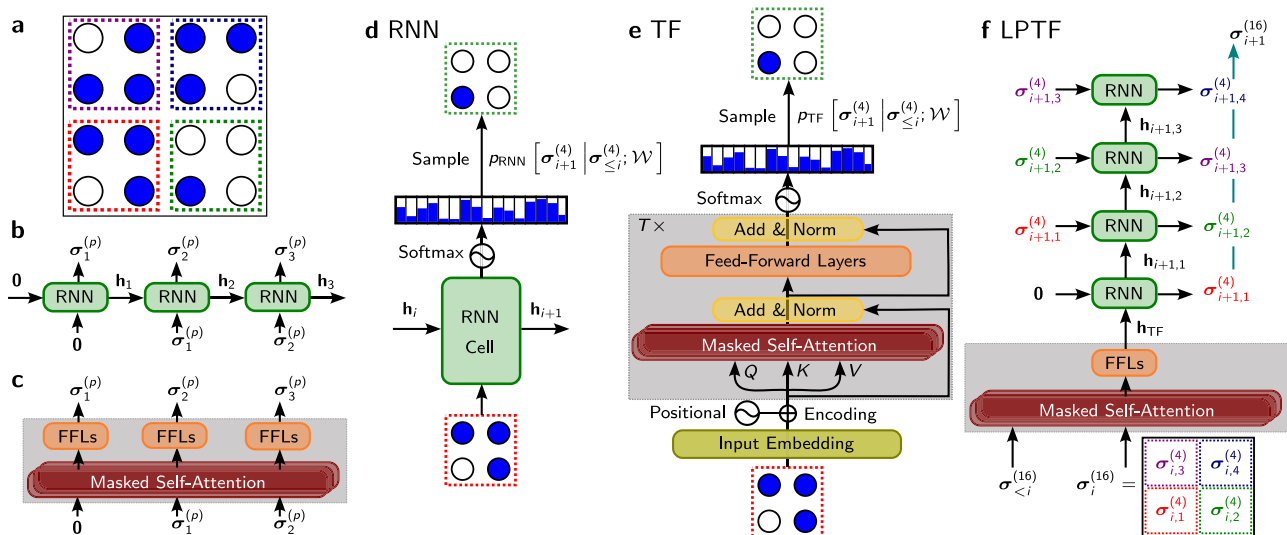
Patched inputs

To address the exceeding computational runtime of TF models, we take inspiration from the vision transformer[41] and consider patches of atoms as inputs to both considered network architectures, as illustrated in Fig. 1d, e. This reduces the sequence length to N/p elements for patch size p, leading to

a sampling complexity of $\mathcal{O}(N/p)$ for the patched RNN and $\mathcal{O}(N^2/p^2)$ for the patched TF model, as well as an energy evaluation complexity of $\mathcal{O}(N^2/p)$ and $\mathcal{O}(N^3/p^2)$, respectively.

We first use patches of $p = 2 \times 2$ atoms. The network output is then a probability distribution over the $2^p = 16$ states the atoms in the patch can take, from which the next patch is sampled and used autoregressively as input in the following iteration. As demonstrated in previous works[25,34,35], this significantly reduces the computational runtime due to the shorter sequence length. In addition, we expect it to capture correlations between neighboring atoms with higher accuracies by directly encoding them in the output probabilities. The patched models can also be modified to include complex phases as a second network output, which then correspond to the sum of phases of individual qubits in the patch[7].

Figure 2c and d show the results for the same $N = 8 \times 8$ and $N = 16 \times 16$ atom Rydberg array ground states as in panels a and b, using the patched RNN (green) and the patched TF setup (red) with $p = 2 \times 2$. The network hyperparameters are the same as in the RNN and the TF network in a and b. The computational runtime reduces significantly to $\tau \approx 0.5$h and $\tau \approx 3$h, using the patched RNN and the patched TF model for $N = 8 \times 8$ atoms, and to $\tau \approx 2$h and $\tau \approx 28$h, respectively, for $N = 16 \times 16$ atoms. Convergence further happens within fewer training iterations than for the unpatched networks, and all representations reach energy values within the QMC errors. We even observe energies below the QMC results, which always remain within the QMC uncertainties and thus do not violate the variational principle which we expect to be satisfied for the number of samples we use to evaluate energy expectation values and for the small variances we observe[56]. These energies propose that the patched networks find the ground state with better accuracy than the QMC simulations using $N_s = 7 \times 10^4$ samples. The QMC accuracy can be further increased by using more samples, where the uncertainty decreases as $\propto 1/\sqrt{N_s}$ for uncorrelated samples[53]. However, samples in a single QMC chain are correlated, resulting in an uncertainty scaling $\propto \sqrt{\tau_{auto}/N_s}$ with autocorrelation time τ_{auto} depending on the

Fig. 1 | Illustrating different network models. a Square lattice of $N = 4 \times 4$ Rydberg atoms randomly occupying the ground state (white) and the Rydberg state (blue). Dash-colored squares indicate patches used as network inputs. b Recurrent neural network (RNN) processing sequence. The RNN cell iteratively receives input sequence elements σ_i together with a hidden state. At each iteration, the output is used as the next input. The index (p) denotes the input size in the patched RNN. c Autoregressive transformer (TF) processing sequence, similar to the RNN in b. The multi-headed masked self-attention layer generates weighted connections to previous input sequence elements. For simplicity, we only include the feed-forward layers (FFLs) in the scheme. d Single patched RNN iteration on inputs of patch size $p = 2 \times 2$ [indicated with (4)]. A softmax function creates a probability distribution p_{RNN} over all possible patch states conditioned on previous sequence elements. The

state of the next patch is sampled and used as next input state. e Single patched TF iteration. The input patch is embedded into a state of dimension d_H, and a positional encoding keeps track of the sequence order. The signal is sent into the transformer cell (gray) which we generate similar to[32] and apply T times independently. The output of the transformer cells is used like in the patched RNN to sample the next input. f Single large, patched transformer (LPTF) iteration for patch size $p = 4 \times 4$ [indicated with (16)] and sub-patch size $p_s = 2 \times 2$. A patched TF model receives a large input patch, and the transformer cell output is propagated as a hidden state h_{TF} to a patched RNN. The patched RNN autoregressively constructs the input patch of the next LPTF iteration, reducing the output dimension. See Methods for more details on the network models.
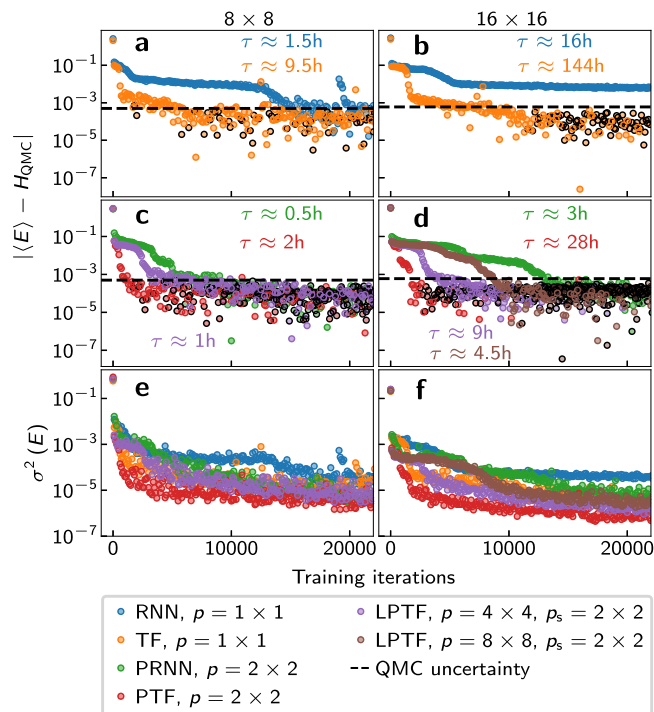
Fig. 2 | Performance of different network architectures on Rydberg atom arrays. **a, b** Absolute energy difference between $\langle E \rangle$ [Eq. (3)] from $N_s = 512$ recurrent neural network (RNN, blue) and transformer (TF, orange) samples and H_{QMC} from $N_s = 7 \times 10^4$ quantum Monte Carlo (QMC) samples as a function of network training iterations. The black dashed line denotes the QMC uncertainty and black-edged data points show the absolute value of energies below QMC results. τ is the total runtime in hours (h) for 2×10^4 training iterations using the network in the corresponding color (blue for RNN, orange for TF) on NVIDIA Tesla P100 GPUs. For the QMC, the total runtimes were obtained as $\tau = 18$h for $N = 8 \times 8$ atoms and $\tau = 24$h for $N = 16 \times 16$ atoms for $N_s = 10^4$ samples on a single CPU. **c, d** Same as **a, b** for the patched RNN (PRNN, green) and the patched TF model (PTF, red) with patch size $p = 2 \times 2$, and for the large, patched transformer (LPTF) approach (purple) with patch size $p = 4 \times 4$ and sub-patch size $p_s = 2 \times 2$. For $N = 16 \times 16$ atoms, we further show LPTF results with patch size $p = 8 \times 8$ (brown). **e, f** Variances $\sigma^2(E)$ of energy expectation values for the network architectures considered in **a-d**.

reducing the output of the LPTF model to the probability distribution over the $2^{p_s} = 16$ sub-patch states, as further discussed in the Methods. The sampling complexity for this model is reduced to $\mathcal{O}(N^2/p^2 + 2^{p_s}N/p_s)$ and the energy evaluation complexity takes the form $\mathcal{O}(N^3/p^2 + 2^{p_s}N^2/p_s)$, as derived in the Methods section. Generally, we can use both the patched RNN and the patched TF architecture as base network and subnetwork. We choose this setup here to combine the high accuracies reached with the patched TF network for large system sizes with the computational efficiency of the patched RNN, which can still accurately represent small systems (see Fig. 2a). Being a combination of a TF network and an RNN, the LPTF can similarly be modified to include complex phases as a second network output.
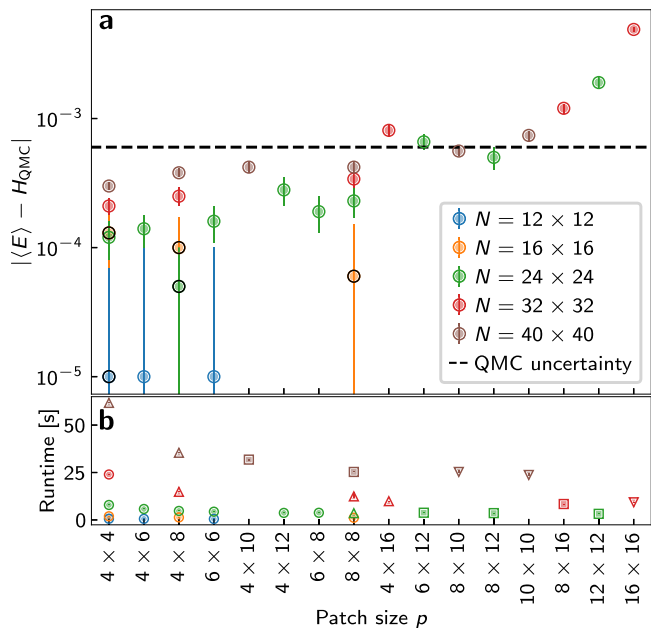
In Fig. 2c, d, we compare the performance of the LPTF model to the previously considered network architectures, where we choose $p = 4 \times 4$ (purple) and $p = 8 \times 8$ (brown), with $p_s = 2 \times 2$, using the same hyperparameters for all networks. These models require more training iterations than the patched TF architecture to converge but reach accuracies comparable to the patched RNN and the patched TF network. Even though more training iterations are required, the computational runtimes are reduced to $\tau \approx 1$h for $N = 8 \times 8$, $p = 4 \times 4$, as well as $\tau \approx 9$h and $\tau \approx 4.5$h for $N = 16 \times 16$ with $p = 4 \times 4$ and $p = 8 \times 8$, respectively. Thus, overall, we obtain convergence within shorter computational runtime.

The observed runtimes are also shorter than QMC runs, even though QMC is expected to outperform the network models for large system sizes due to the linear scaling of computational costs with N. However, QMC is based on the generation of a chain of correlated samples. For large system sizes, autocorrelation times between samples in the chain increase and the ergodicity of the sampling process is not necessarily guaranteed[53]. Since these limitations do not arise for the exact sampling process in autoregressive ANN methods[7,26], computationally efficient architectures such as the LPTF are still promising candidates for accurate studies of large quantum many-body systems.

Figure 2 e and f show the variances $\sigma^2(E)$ of the energy expectation values obtained with all considered network architectures. As expected[7], they decrease to zero when converging to the ground state energies. This behavior confirms the accurate ground state reconstruction, while the smoothness of all curves demonstrates stable training processes.

We can further increase the patch size p in the LPTF architecture, from which we expect even shorter runtimes. However, this also increases the patch size that needs to be reconstructed with the patched RNN. We thus expect the accuracy to decrease for large p if we keep $p_s = 2 \times 2$ fixed. Figure 3a shows the energy difference between QMC and LPTF simulations for ground states of Rydberg arrays with $N = 12 \times 12$ up to $N = 40 \times 40$ atoms. We keep the parameters at $R_b = 7^{1/6}$, $\delta = \Omega = 1$, and evaluate the QMC energies on $N_s = 7 \times 10^4$ samples from seven independent chains[53], where the computational cost for QMC scales as $\mathcal{O}(N)$ with the system size. Each LPTF data point corresponds to an average over training iterations 19,000 to 20,000 of ten independently trained networks with the same setup as for Fig. 2. We vary the input patch size between $p = 4 \times 4$ and $p = 16 \times 16$, where we also consider rectangular-shaped patches while fixing $p_s = 2 \times 2$. We ensure that the system size always divides by the input patch size.

As expected, the energy accuracies decrease with increasing patch size, which might result from the limited representational power of the patched RNN for large input p and small p_s and from the increased amount of information that is encoded in each network iteration. We find accuracies below the QMC uncertainty for up to $p = 8 \times 8$, which still proposes a significant speed-up compared to single-atom inputs in the TF model, see Fig. 2d. Figure 3b shows the computational runtimes of single training iteration steps for the different patch and system sizes. Each data point shows an average over 2×10^4 training iterations in a single network. We find a rapid decrease in computation times for small patches while we observe convergence to steady times for larger patches. This behavior results from the increased memory required by larger patch sizes, which forces us to decrease the mini-batch size K of samples for the energy evaluation, see

evaluated observable[53]. Even though the computational cost of QMC scales linearly with the sample chain size N_s and is thus more efficient than the RNN or the TF approach, which require the generation of N_s samples in each training iteration, we found that reaching higher QMC precisions comes at runtimes that exceed the patched RNN and the patched TF due to long autocorrelation times for large system sizes.

Large, patched transformers

Based on the results with $p = 2 \times 2$, we expect even shorter computational runtimes and higher representation accuracies from larger patch sizes. However, as illustrated in Fig. 1d, e, the network output dimension scales exponentially with the input patch size, encoding the probability distribution over all possible patch states. This output scaling leads to the sampling cost scaling as $\mathcal{O}(2^p N/p)$ for the patched RNN and as $\mathcal{O}(N^2/p^2 + 2^p N/p)$ for the patched TF network, as well as energy evaluation costs scaling as $\mathcal{O}(2^p N^2/p)$ and $\mathcal{O}(N^3/p^2 + 2^p N^2/p)$, respectively, see Methods. A hierarchical softmax approach is often used in image processing to efficiently address this exponential scaling[57]. Here we introduce large, patched transformers (LPTFs) as an alternative way to enable efficient patch size scaling.

As shown in Fig. 1f, the LPTF model uses a patched TF setup and passes the TF state into a patched RNN as the initial hidden state. The patched RNN splits the input patch into smaller sub-patches of size $p_s = 2 \times 2$,

Fig. 3 | Patch size scaling in large, patched transformers (LPTFs). a Absolute energy difference between $\langle E \rangle$ [Eq. (3)] evaluated with $N_s = 512$ large, patched transformer (LPTF) samples and H_{QMC} evaluated with $N_s = 7 \times 10^4$ quantum Monte Carlo (QMC) samples for different system sizes N (colors) as a function of the patch size p. Error bars denote the standard error of the sampling mean. The black dashed line indicates the QMC uncertainty, while black-edged data points show absolute values of energies below the QMC results. **b** Computational runtime per training iteration on NVIDIA Tesla P100 GPUs averaged over 2×10^4 iterations of LPTFs as in **a**. Different shapes denote different mini-batch sizes K, with $K = 256$ for circles, $K = 128$ for up-pointing triangles, $K = 64$ for squares, and $K = 32$ for down-pointing triangles, see Methods. Error bars are smaller than the data points.

Methods. Smaller mini-batch sizes lead to increased runtimes, which compete with the acceleration from the reduced sequence lengths.

We cannot find a conclusive dependence on the patch shape, with rectangular patches showing a similar behavior as squared patches. Thus, the only important factor is the overall patch size, and we conclude that input patches around $p = 8 \times 8$ atoms provide a good compromise with reduced computation times and high energy accuracies.

Phases of matter in Rydberg atom arrays

We now explore the performance of LPTFs at different points in the Rydberg phase diagram by varying the detuning from $\delta = 0$ to $\delta = 3$ and fixing $R_b = 3^{1/6} \approx 1.2$, $\Omega = 1$. With this, we drive the system over the transition between the disordered and the checkerboard phase[46,48]. The order parameter for the checkerboard phase is given by the staggered magnetization[46],

$$\sigma^{stag} = \left\langle \left| \sum_{i=1}^{N} (-1)^i \frac{n_i - 1/2}{N} \right| \right\rangle, \quad (5)$$

where i runs over all $N = L \times L$ atoms and $n_i = |r\rangle_i \langle r|_i$ is the occupation number operator acting on atom i. The expectation value denotes the average over sample configurations generated via QMC or from the trained network.

Figure 4 shows the staggered magnetization when tuning δ over the phase transition, where we compare LPTF and QMC simulations. The QMC data points show the average of $N_s = 7 \times 10^5$ samples generated from seven independent chains[53]. The LPTF data is averaged over training iterations 11,000 to 12,000 of five independently trained networks. We look at systems with $N = 8 \times 8$ and $N = 16 \times 16$ atoms, choosing patch sizes $p = 4 \times 4$ and $p = 8 \times 8$, respectively, with $p_s = 2 \times 2$.
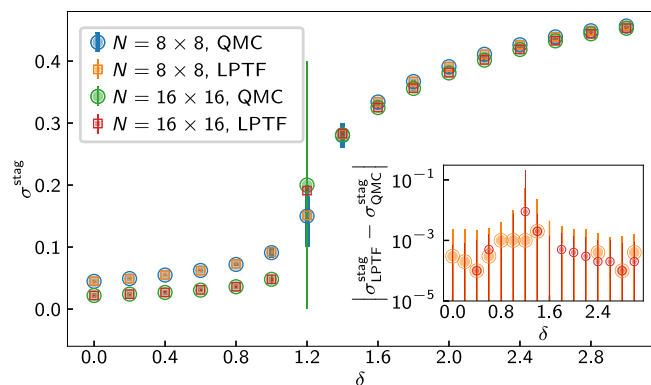
Fig. 4 | Staggered magnetization as order parameter. Staggered magnetization σ^{stag} [Eq. (5)] obtained with large, patched transformers (LPTFs, squares) and with quantum Monte Carlo (QMC, circles) for Rydberg arrays with $N = 8 \times 8$ (blue, orange) and $N = 16 \times 16$ (green, red) atoms when driving the detuning δ across the transition between the disordered ($\delta \lesssim 1.2$) and the checkerboard ($\delta \gtrsim 1.2$) phase. For $N = 8 \times 8$, we use patch size $p = 4 \times 4$, while $p = 8 \times 8$ for $N = 16 \times 16$. LPTF data is averaged over training iterations 11,000 to 12,000 of five independent networks with $N_s = 512$ samples. QMC data is evaluated on $N_s = 7 \times 10^5$ samples. Error bars denote the standard error of the sampling mean, where autocorrelation times are considered in the QMC samples[53]. *Inset:* Absolute difference between the LPTF and QMC data in the main plot for $N = 8 \times 8$ (orange) and $N = 16 \times 16$ (red). Error bars denote the standard error of the sampling mean.

The LPTF model captures the phase transition accurately for both system sizes, overlapping closely with the QMC results for all δ and showing small uncertainties. In the inset in Fig. 4, we plot the absolute difference between the staggered magnetizations obtained with QMC and LPTFs for both system sizes. The most challenging regime to simulate is at $\delta \approx 1.2$, where we find the phase transition in the main panel. Here the observed difference is $\sim 10^{-2}$, demonstrating the high accuracies reachable with the LPTF approach. In the vicinity of the phase transition, the QMC uncertainties increase. This behavior is related to long autocorrelation times τ_{auto} in the individual sample chains and the uncertainty scaling as $\propto \sqrt{\tau_{auto}/N_s}$[53]. The errors in the LPTF simulations remain small here, demonstrating a consistent and accurate outcome in all independent networks.

Discussion

We explored the power of transformer (TF) models[32] in representing ground states of two-dimensional Rydberg atom arrays of different sizes by benchmarking them on quantum Monte Carlo simulations[53]. Our work provides a careful performance comparison of TF models with a recurrent neural network (RNN) wavefunction ansatz[4,6,7], showing that TFs reach higher accuracies, especially for larger system sizes, but require longer computational runtimes. We accelerate the network evaluation using patches of atoms as network inputs inspired by the vision transformer[41] and demonstrate that these models significantly improve computational runtime and reachable accuracies.

Based on the obtained results, we introduce large, patched transformers (LPTFs), which consist of a patched TF network whose output is used as the initial hidden unit state of a patched RNN. This model enables larger input patch sizes which are broken down into smaller patches in the patched RNN, keeping the network output dimension at reasonable size.

The LPTF models reach accuracies below the QMC uncertainties for ground states obtained with a fixed number of samples, while requiring significantly reduced computational runtimes compared to traditional neural network models. We are further able to scale the considered system sizes beyond most recent numerical studies, while keeping the accuracies high and computational costs reasonable[8,46,52,53]. These observations promise the ability to study the scaling behavior of Rydberg atom arrays to large system sizes, allowing an in-depth exploration of the underlying phase

diagram. While such studies go beyond the scope of this proof-of-principle work, we leave it open for future follow-up works.

Our results show that the LPTF model performs similarly well in different phases of matter in the Rydberg system and accurately captures phase transitions. While we focus on Rydberg atom arrays, the introduced approach can be applied to general quantum many-body systems, where complex-valued wave functions can be represented by adding a second output to the autoregressive network architecture as in[7]. While we expect the inclusion of complex phases to make the training process harder[7], modifications of the LPTF setup can be explored in future works to study more complex or larger qubit systems. Such modifications include larger network models with more transformer cells, or higher embedding dimensions, which increase the network expressivity[58]. Additionally, larger input patch sizes can be achieved by including multiple patched RNN and patched TF components in the LPTF architecture, which successively reduce the sub-patch sizes. We further expect that the performance of LPTFs can be enhanced with a data-based initialization, as discussed in[8,31].

Our results and possible future improvements promise high-quality representations of quantum states in various models and phases of matter at affordable computational costs. This prospect proposes significant advances in the modeling of quantum many-body systems, promising insightful follow-up works exploring new physical phenomena.

Methods
Rydberg atom arrays

We apply our numerical methods on Rydberg atom arrays as an example for qubit systems. In state-of-the-art experiments, Rydberg atoms are individually addressed via optical tweezers that allow for precise arrangements on arbitrary lattices in up to three dimensions[44,45,48,49]. Fluorescent imaging techniques are then used to perform projective measurements in the Rydberg excitation basis. Such accurate and well-controlled experimental realizations are accompanied by intensive numerical investigations, which have unveiled a great variety of phases of matter, separated by quantum phase transitions, in which Rydberg atom systems can be prepared[46–48,51,52]. The atoms on the lattice interact strongly via the Rydberg many-body Hamiltonian in Eq. (1)[42,43]. The Rydberg blockade radius $R_{\rm b}$ defines a region within which simultaneous excitations of two atoms are penalized.

The ground states of this Rydberg Hamiltonian are fully described by positive, real-valued wavefunctions so that the outcomes of measurements in the Rydberg occupation basis provide complete information about ground state wavefunctions[44,48,50]. We can thus model ground state wavefunctions with real-valued neural network model architectures[6,7]. In this work, we choose $\Omega = 1$ and describe the system in terms of the detuning δ and the Rydberg blockade radius $R_{\rm b}$. We further consider square lattices of $N = L \times L$ atoms with lattice spacing $a = 1$ and open boundary conditions.

Recurrent neural network quantum states

Recurrent neural networks (RNNs) are generative network architectures that are optimized to deal with sequential data[55,59]. They naturally encode a probability distribution and enable efficient sample data generation. As illustrated in Fig. 1b, d, the RNN input is given by individual elements σ_i of dimension $d_{\rm I}$ from a given data sequence $\boldsymbol{\sigma}$, and a hidden state $\boldsymbol{h}_i$ of dimension $d_{\rm H}$. We use the initial input states $\boldsymbol{\sigma}_0 = \boldsymbol{0}$ and $\boldsymbol{h}_0 = \boldsymbol{0}$. Throughout this work, we choose $d_{\rm H} = 128$. The input is processed in the RNN cell, where non-linear transformations defined via variational parameters $\mathcal{W}$ are applied. Here we use the Gated Recurrent Unit (GRU)[55] as RNN cell, which is applied at each iteration with shared weights[59].

We then apply two fully connected projection layers on the hidden state, the first followed by a rectified linear unit (ReLU) activation function and the second followed by a softmax activation function (not layers are not shown in Fig. 1d). This setup generates an output vector of dimension $d_{\rm O}$ which is interpreted as a probability distribution over all possible output values[7]. The hidden state configuration is propagated over the input sequence encoding information of previous inputs and generating a memory effect. This setup conditions the output probability

on all previous sequence elements, $p_{\rm RNN}(\sigma_{i+1}|\sigma_i, \ldots, \sigma_1; \mathcal{W})$, from which an output state is sampled. Here, we use this output as the input element σ_{i+1} of the next iteration, running the network in an auto-regressive manner. In this case, the joint probability of the generated sequence is given by $p_{\rm RNN}(\boldsymbol{\sigma}; \mathcal{W}) = \prod_i p_{\rm RNN}(\sigma_i|\sigma_{i-1}, \ldots, \sigma_1; \mathcal{W})$[7].

To use the RNN as a wavefunction ansatz to represent quantum states, we consider the quantum system as a sequence of qubits, sampling the state of one qubit at a time and using it as the input in the next RNN iteration[4,6,7]. The hidden state propagation captures correlations in the qubit system by carrying information about previously sampled qubit configurations. We then interpret the probability distribution encoded in the RNN as the squared wavefunction amplitude of the represented quantum state, $p_{\rm RNN}(\boldsymbol{\sigma}; \mathcal{W}) = |\langle\boldsymbol{\sigma}|\Psi_\mathcal{W}\rangle|^2 = |\Psi_{\rm RNN}(\boldsymbol{\sigma}; \mathcal{W})|^2$. This ansatz can model the complete information of ground states in the considered Rydberg Hamiltonian, Eq. (1). Samples generated from the RNN then correspond to outcomes of projective measurements in the computational basis and can be used to estimate expectation values of general observables $\hat{\mathcal{O}}$[1,2,4,6,7,23],

$$
\begin{aligned}
\langle\Psi_\mathcal{W}|\hat{\mathcal{O}}|\Psi_\mathcal{W}\rangle &= \sum_{\{\boldsymbol{\sigma},\boldsymbol{\sigma}'\}} \Psi^*_{\rm RNN}(\boldsymbol{\sigma}; \mathcal{W})\Psi_{\rm RNN}(\boldsymbol{\sigma}'; \mathcal{W}) \times \langle\boldsymbol{\sigma}|\hat{\mathcal{O}}|\boldsymbol{\sigma}'\rangle \\
&= \sum_{\{\boldsymbol{\sigma}\}} |\Psi_{\rm RNN}(\boldsymbol{\sigma}; \mathcal{W})|^2 \mathcal{O}_{\rm loc}(\boldsymbol{\sigma}; \mathcal{W}) \\
&\approx \frac{1}{N_{\rm s}} \sum_{\substack{\boldsymbol{\sigma}_s \propto \\ p_{\rm RNN}(\boldsymbol{\sigma};\mathcal{W})}} \mathcal{O}_{\rm loc}(\boldsymbol{\sigma}_s; \mathcal{W}),
\end{aligned}
\tag{6}
$$

where we introduce the local observable,

$$
\begin{aligned}
\mathcal{O}_{\rm loc}(\boldsymbol{\sigma}_s; \mathcal{W}) &= \frac{\langle\boldsymbol{\sigma}_s|\hat{\mathcal{O}}|\Psi_\mathcal{W}\rangle}{\langle\boldsymbol{\sigma}_s|\Psi_\mathcal{W}\rangle} \\
&= \sum_{\{\boldsymbol{\sigma}'\}} \langle\boldsymbol{\sigma}_s|\hat{\mathcal{O}}|\boldsymbol{\sigma}'\rangle \frac{\Psi_{\rm RNN}(\boldsymbol{\sigma}'; \mathcal{W})}{\Psi_{\rm RNN}(\boldsymbol{\sigma}_s; \mathcal{W})}.
\end{aligned}
\tag{7}
$$

This local observable is evaluated and averaged over $N_{\rm s}$ samples $\boldsymbol{\sigma}_s$ generated from the RNN. To find the ground state representation of a given Hamiltonian in the RNN, we use a gradient descent training algorithm to minimize the energy expectation value $\langle E \rangle = \langle \hat{\mathcal{H}} \rangle$, which can be similarly evaluated using samples from the RNN as stated in Eq. (3) and Eq. (4)[4,6,7,23,56]. We train the RNN using the Adam optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and learning rate $\Delta = 0.0005$.

The GRU cell has three internal weight matrices of dimension $d_{\rm I} \times d_{\rm H}$, with input dimension $d_{\rm I}$ and hidden unit dimension $d_{\rm H}$, and three internal weight matrices of dimension $d_{\rm H} \times d_{\rm H}$. It furthermore has six internal bias vectors of size $d_{\rm H}$, and we add two fully connected layers with weight matrices of dimension $d_{\rm H} \times d_{\rm H}$ and $d_{\rm H} \times d_{\rm O}$ and biases of size $d_{\rm H}$ and $d_{\rm O}$, respectively, to obtain the desired RNN output vector with output dimension $d_{\rm O}$[7,55]. Single-qubit inputs give $d_{\rm I} = 1$ and $d_{\rm O} = 2$ as we use a one-hot encoded output. Together with $d_{\rm H} = 128$ as chosen in this work, this leads to a total of

$$
3(d_{\rm I} \times d_{\rm H}) + 4(d_{\rm H} \times d_{\rm H}) + 7d_{\rm H} + (d_{\rm H} \times d_{\rm O}) + d_{\rm O} = 67,074
\tag{8}
$$

trainable network parameters.

Transformer quantum states

Transformer (TF) models can be applied to sequential data similarly to RNNs. Such models do not include a recurrent behavior but are based on self-attention, which provides access to all elements in the sequence and enables the dynamical highlighting of salient information. We use the TF model as introduced in[32] and restrict it to the encoder part only[14,33–35].

As illustrated in Fig. 1e, the TF model first embeds the given input vector. This embedding corresponds to a linear projection of the input vector of dimension $d_{\rm I}$ to a vector of embedding dimension $d_{\rm H}$ with trainable parameters $W^{\rm I}$. As a next step, the positional encoding matrix is
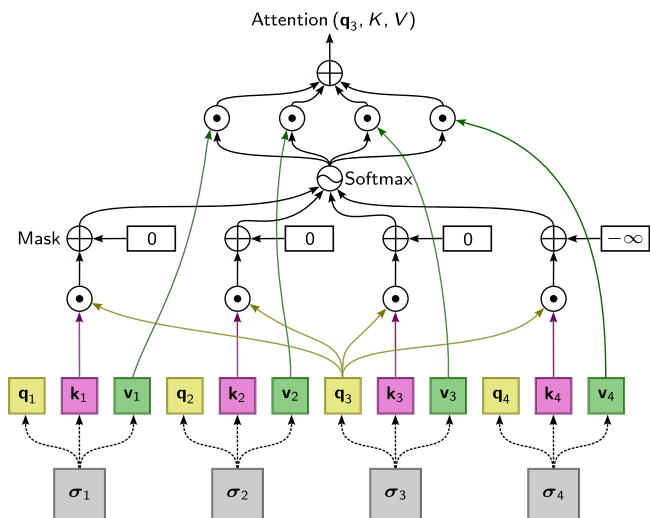
Fig. 5 | Illustration of the attention mechanism applied to sequence element σ_3. Each sequence element σ_j is projected onto a query (q_j), key (k_j), and value (v_j) vector, and the dot product of q_3 is taken with each key vector k_j, $j \in \{1, 2, 3, 4\}$. Adding a mask matrix eliminates the influence of all elements later in the sequence by adding $m_{i,j > i} = -\infty$, while contributions of previously sampled sequence elements remain unchanged with $m_{i,j \leq i} = 0$, ensuring the autoregressive behavior. A softmax activation function is applied to all signals before a dot product is taken with the corresponding value vectors. The sum of all resulting signals provides the attention outcome for the input σ_3. This algorithm is the same as introduced in[32].

evaluated and added to the embedded input vector to include information about the positions of the input elements in the sequence[32]. To keep this information when propagating the signal through the TF structure, the overall embedding dimension d_H of internal states is conserved. Throughout this work, we choose $d_H = 128$.

The embedded input with positional encoding is passed to the transformer cell, where the query, key, and value matrices are generated, and multi-headed self-attention is applied[32]. We use a masked self-attention mechanism to ensure the TF model is autoregressive, like the RNN. The output vector of the masked self-attention is then added to the embedded input state, and the sum is normed before being fed into two feed-forward layers. The normalization is a trained process to improve the network training stability and adds $2d_H$ trainable parameters. Similar to[32], we apply one feed-forward layer with a ReLU activation function, followed by a linear feed-forward layer, where the weights and biases of both layers are trainable. The first feed-forward layer projects the input into a vector of size $d_{FF} = 2048$, while the second feed-forward layer projects it back to size d_H. The output of the feed-forward layers is again added to the output vector of the self-attention cell, and the sum is normalized, see Fig. 1e.

The entire transformer cell, including the self-attention and feed-forward layers, as well as the add-and-norm operations, can be applied multiple times independently to improve the network expressivity. We obtain satisfying results with $T = 2$.

To represent quantum states similarly to the RNN ansatz, we apply two fully connected layers with trainable weights to the output of the transformer cell. The first layer conserves the dimension d_H and is followed by a ReLU activation function. The second layer projects the output to a vector of output dimension d_O and is followed by a softmax activation function. These two layers are not shown in the diagram in Fig. 1e. After the softmax activation function, the output can be treated the same way as the RNN output, and it can be interpreted as a probability distribution from which the next qubit state in the sequence can be sampled[32]. We train the TF model the same way as the RNN, using the Adam optimizer to minimize the energy expectation value via gradient descent. The energy expectation value is obtained in the same way as in the RNN, see Eq. (3) and Eq. (4), and we

choose the same values as in the RNN approach for all hyperparameters involved in the training process.

Positional encoding. Since the TF model does not include a recurrent behavior as the RNN, it does not provide any information about the order of the sequence elements by default. A positional encoding algorithm is used to include information about the position of each input. We use the algorithm as proposed in[32], which creates a matrix of dimension $L \times d_H$ with sequence length L and embedding dimension d_H. The individual elements are calculated via

$$P(l, 2i) = \sin\left[\frac{l}{10000^{2i/d_H}}\right], \tag{9}$$

$$P(l, 2i + 1) = \cos\left[\frac{l}{10000^{2i/d_H}}\right], \tag{10}$$

with $0 \leq l < L$ indexing the sequence elements, and $0 \leq i < d_H/2$ the column indices of the output matrix. The resulting matrix is added to the embedded input element of the TF setup, which linearly projects the input vector to a vector of dimension d_H using trainable weights. This operation gives each element a unique information about its position in the sequence.

The self-attention mechanism. The self-attention mechanism, as introduced in[32] and illustrated in Fig. 5, projects each embedded sequence element σ_i of dimension d_H to a query vector q_i, a key vector k_i, and a value vector v_i of dimensions d_H,

$$q_i = \sum_{l=1}^{d_H} W_{i,l}^q \sigma_{i,l}, \quad k_i = \sum_{l=1}^{d_H} W_{i,l}^k \sigma_{i,l}, \quad v_i = \sum_{l=1}^{d_H} W_{i,l}^v \sigma_{i,l}, \tag{11}$$

with trainable weight matrices W^q, W^k, W^v of dimension $d_H \times d_H$. Query, key, and value vectors of all input elements can be summarized in the corresponding matrices,

$$Q = \begin{bmatrix} q_1 \\ \vdots \\ q_L \end{bmatrix}, \quad K = \begin{bmatrix} k_1 \\ \vdots \\ k_L \end{bmatrix}, \quad V = \begin{bmatrix} v_1 \\ \vdots \\ v_L \end{bmatrix}, \tag{12}$$

with sequence length L.

The attention mechanism then maps the queries and key-value pairs to an output for each sequence element, allowing for highlighting of connections to sequence elements with important information. For each sequence element σ_i, the dot product of the query vector q_i with the key vector k_j for all $j \in \{1, \ldots, L\}$ is evaluated. We then add a masking term $m_{i,j}$ to the signal, which is given by,

$$m_{i,j} = \begin{cases} 0 & \text{if } i \leq j, \\ -\infty & \text{otherwise.} \end{cases} \tag{13}$$

This ensures that the self-attention only considers previous elements in the sequence and does not look at later elements that still need to be determined in the autoregressive behavior. Applying a softmax activation function to all signals after adding the mask ensures that the contributions of all later sequence elements with $m_{i,j} = -\infty$ are driven to zero. We then take the dot product of each signal with the corresponding value vector v_j and sum all signals to generate the output of the attention mechanism. The complete attention formalism can thus be summarized as,

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_H/h}} + M\right)V, \tag{14}$$

where the mask matrix M has entries $m_{i,j}$ as in Eq. (13). Here we further use multi-headed attention, as discussed in[32]. This approach linearly projects each query, key, and value vector to h vectors with individually trainable projection matrices. We thus end up with h heads with modified query, key, and value vectors on which the attention mechanism is applied, where we choose $h = 8$ throughout this work, as we find it to yield satisfying results. The linear projection further reduces the dimension of the query, key, and value vectors to d_H/h, so that the outputs of the individual heads can be concatenated to yield the total output dimension d_H of the attention mechanism. We then scale the outcome of each query-key dot product with the factor $1/\sqrt{d_H/h}$ in Eq. (14)[32]. The output of the multi-headed attention formalism is given by,

$$\text{Multihead}(Q, K, V) = \text{Concat}(\hat{y}_1, \ldots, \hat{y}_h) W^O, \tag{15}$$

$$\hat{y}_l(Q, K, V) = \text{Attention}(Q W_l^Q, K W_l^K, V W_l^V), \tag{16}$$

with output weight matrix W^O and query, key, and value weight matrices W_l^Q, W_l^K, and W_l^V for head l.

A TF model given an input of dimension d_I has an embedding matrix W_I of dimension $d_I \times d_H$, with embedding dimension d_H. The weight matrices in the multi-headed self-attention mechanism then have dimensions $d_H \times d_H/h$ for W_l^Q, W_l^K, and W_l^V, and $d_H \times d_H$ for W^O. Each weight matrix also comes with a bias whose size equals the column-dimension. The two feed-forward layers in the transformer cell have weight matrices of $d_H \times d_{FF}$ and $d_{FF} \times d_H$ with corresponding biases, and the two norm operations add $4d_H$ trainable parameters. The transformer cell, containing the attention mechanism, the feed-forward layers, and the norm operations, is applied T times with independent variational parameters. After the transformer cell we add two fully connected layers with weight matrices of dimensions $d_H \times d_H$ and $d_H \times d_O$ for output dimension d_O. Both layers come with corresponding biases.

Single-qubit inputs give $d_I = 1$ and $d_O = 2$, using one-hot encoded output. With $d_H = 128$, $d_{FF} = 2048$, and $T = 2$, as chosen throughout this work, the TF architecture has a total of

$$
\begin{aligned}
(d_I \times d_H) + d_H &+ T\big[4(d_H \times d_H) + 9d_H + 2(d_{FF} \times d_H) + d_{FF}\big] \\
&+ (d_H \times d_H) + d_H + (d_H \times d_O) + d_O \\
&= 1,203,074
\end{aligned}
\tag{17}
$$

trainable variational parameters.

Patched network models
The bottleneck of the RNN and TF wavefunction ansatz is the iteration of the network cell over the entire qubit sequence. This computationally expensive step needs to be done for each sample that is generated, as well as each time a wavefunction $\Psi(\sigma; \mathcal{W})$ is calculated, which is required to evaluate non-diagonal observables, see Eq. (7). We reduce the number of iterations per network call by shortening the input sequence and in return increasing the dimension of the input vector.

As illustrated in Fig. 1, for two-dimensional Rydberg atom arrays, we consider patches of p qubits arranged in squares or rectangles. We flatten these patches into binary input vectors of dimension $d_I = p$. This modification increases the network input dimension, which is, however, projected to the unaffected hidden state dimension in the RNN cell or the embedding dimension in the TF model. Thus, the computational cost of evaluating the network cell is barely affected by the increased input dimension, but the shorter sequence length leads to significantly reduced computational runtimes. In addition to this expected speed-up, we expect the patched network models to capture local correlations in the system with higher accuracy. Neighboring qubit states are now considered at the same iteration, and their information is not encoded in the network state.

The network output uses one-hot encoding of the patched quantum states, so that the output vector is of dimension $d_O = 2^p$. Each entry

represents one possible state of the qubit patch, see Fig. 1d, e. This output dimension, and with this the computational cost of evaluating the softmax function, thus scales exponentially with the patch size. In this work, we only consider patches up to $p = 2 \times 2$ for the patched network models and introduce large, patched TFs to deal with larger patch sizes.

The patched RNN with $p = 2 \times 2$ has input dimension $d_I = 4$ and output dimension $d_O = 16$, so Eq. (8) leads to $70,032$ trainable network parameters. For the same input and output dimension, the patched TF model has $1,203,406$ trainable network parameters, according to Eq. (17).

Large, patched transformer models
In the large, patched transformer (LPTF) model, we apply the TF network to a patch of p qubits. However, we abort the TF model in Fig. 1e right after the transformer cell and do not apply the fully connected layers and the softmax activation function. Instead, we use the generated output state of the transformer cell as an input hidden state to a patched RNN with the hidden-unit dimension matching the embedding dimension d_H of the TF model. This patched RNN breaks up the large input patch into smaller sub-patches of size p_s, where we always choose a sub-patch size of $p_s = 2 \times 2$ in this work. We then use the patched RNN model to iteratively sample the quantum states of these sub-patches in the same way as when applying the patched RNN to the full system size. The only difference is that the initial hidden state is provided by the TF output, $h_0 = h_{TF}$, see Fig. 1f for an illustration.

The total number of trainable network parameters in the LPTF setup is then given by a combination of Eq. (8) and Eq. (17), where the two fully connected layers at the TF output are subtracted,

$$
\begin{aligned}
&\underbrace{(p \times d_H) + T\big[4(d_H \times d_H) + 9d_H + 2(d_{FF} \times d_H) + d_{FF}\big]}_{\text{patched TF}} \\
&+ \underbrace{3(p_s \times d_H) + 4(d_H \times d_H) + 7d_H + (d_H \times d_O) + d_O}_{\text{patched RNN}}.
\end{aligned}
\tag{18}
$$

We use the input dimension $d_I = p$ for the patched TF and $d_I = p_s$ for the patched RNN, as well as the output dimension $d_O = 2^{p_s}$. In this work, we choose $p_s = 2 \times 2$, which yields $1,256,208 + 128p$ trainable parameters with $d_H = 128$, $d_{FF} = 2048$, and $T = 2$. For the choice $p = 8 \times 8$, we thus get $1,264,400$ variational network parameters.

Computational complexity
The process of finding ground state representations with ANNs can be divided into two steps, the generation of samples from the network and the evaluation of energy expectation values according to Eq. (6) and Eq. (7) in each training iteration. We start with analyzing the sample complexity for the different network architectures. As we choose the hidden and the embedding dimension d_H fixed and equal for all architectures, we consider it as a constant in the complexity analysis.

Generating a single sample σ from $p_{RNN}(\sigma; \mathcal{W})$ encoded in an RNN requires N executions of the RNN cell, leading to a computational cost of $\mathcal{O}(N)$. By considering the patched RNN, we reduce the sequence length from N to N/p, so that only N/p RNN cells are evaluated. However, the output dimension in this case is 2^p, so each evaluation of the RNN cell requires 2^p products to evaluate the outcome probability distribution. This leads to an overall sampling complexity of $\mathcal{O}(N/p 2^p)$ for the patched RNN.

In order to generate a single sample σ from $p_{TF}(\sigma; \mathcal{W})$ encoded in the TF network, we similarly need to evaluate the transformer cell N times. However, the attention algorithm itself requires the computation of N multiplications that need to be evaluated in each pass through the network. Additionally, the transformer cell contains a projection of the embedded state to a vector of size $d_{FF} \gg d_H$, which requires significantly more multiplication operations than an RNN cell evaluation. Thus, drawing a sample from the TF model comes at computational complexity $\mathcal{O}(N[N + d_{eff}])$. When introducing the patched TF model, we similarly reduce the sequence length to N/p, so that the full network needs to be evaluated N/p times and

the attention mechanism requires N/p multiplications. Also here the output scales as 2^p, leading to a computational cost of $\mathcal{O}\left(N/p\left[N/p + d_{\mathrm{FF}} + 2^p\right]\right)$.

In the LPTF, the transformer cell is followed by a patched RNN with p/p_s cells. Since each LPTF iteration requires the evaluation of one such RNN, we evaluate the transformer cell and p/p_s RNN cells N/p times to generate a single sample $\boldsymbol{\sigma}$. While the TF network output is kept at embedding dimension d_{H}, the RNN output is of dimension 2^{p_s}, leading to a computational complexity of

$$\mathcal{O}\left(\frac{N}{p}\left[\frac{N}{p} + d_{\mathrm{FF}}\right] + \frac{N}{p}\frac{p}{p_s}2^{p_s}\right) = \mathcal{O}\left(\frac{N^2}{p^2} + \frac{N}{p}d_{\mathrm{FF}} + \frac{N}{p_s}2^{p_s}\right). \quad (19)$$

This shows a significant reduction of the sampling complexity compared to the patched TF model and explains the observed efficiency of our introduced LPTF architecture.

Next, we consider the complexity of evaluating energy expectation values. While the evaluation of the diagonal part is given by a linear average over all N_s samples and thus scales linearly with the system size N, evaluating the off-diagonal part for each sample $\boldsymbol{\sigma}_s$ according to Eq. (7) requires the evaluation of $\Psi(\boldsymbol{\sigma}'; \mathcal{W})$ for all $\boldsymbol{\sigma}'$ corresponding to $\boldsymbol{\sigma}_s$ with one atom flipped. This leads to N evaluations of $\Psi(\boldsymbol{\sigma}'; \mathcal{W})$ for each sample, which is obtained by passing $\boldsymbol{\sigma}'$ through the network architecture and obtaining the output probability $p_{\mathrm{RNN}}(\boldsymbol{\sigma}'; \mathcal{W})$ or $p_{\mathrm{TF}}(\boldsymbol{\sigma}'; \mathcal{W})$.

Thus, the patched RNN with N/p network cells needs to be evaluated N times for each sample, leading to a computational complexity of $\mathcal{O}(N_s 2^p N^2/p)$ for obtaining the energy expectation value. Similarly, the patched TF model with N/p iterations and N/p multiplications in the attention mechanism is evaluated N times for each sample, leading to a computational cost of $\mathcal{O}(N_s[N^3/p^2 + d_{\mathrm{FF}}N^2/p + 2^p N^2/p])$. For the LPTF we accordingly obtain $\mathcal{O}(N_s[N^3/p^2 + d_{\mathrm{FF}}N^2/p + 2^{p_s}N^2/p_s])$. The required memory scaling behaves similarly for the discussed network architectures. This scaling can be reduced using optimized implementation algorithms as discussed in the next section. While the TF and LPTF show a worse scaling than the RNN, the evaluation of the off-diagonal energy terms can be parallelized for these two models. Since no autoregressive sampling is required for this task, all N/p masked self-attention layers can be evaluated in parallel, significantly reducing the computational runtime. This parallelization is not possible for the RNN due to its recurrent nature, where the hidden state needs to be evaluated for each individual RNN cell before it can be passed to the next iteration.

The generation of samples with QMC is of computational complexity $\mathcal{O}(VN)$, with average interaction strength V over the system. The energy estimation also scales as $\mathcal{O}(VN)$ and only needs to be done at the end of the run after all samples have been generated, which is in contrast to the neural network approach that requires the evaluated energy in each training iteration[53]. While QMC thus shows much more promising computational complexity than all three neural network methods when considering the scaling to large system sizes, we observe that it requires longer computational runtimes than the LPTF for the system sizes considered in this work. Considering the uncertainties of expectation values, we find that QMC simulations require far more samples ($N_s = 7 \times 10^4$ in Figs. 2, 3, and $N_s = 7 \times 10^5$ in Fig. 4) than the ANN approaches ($N_s = 512$). However, the ANN approaches require the generation of N_s samples in each training iteration, so that overall more samples are generated.

The higher uncertainties in the QMC simulations are caused by correlations in the generated sample chains, where autocorrelation times grow with increasing system sizes. Furthermore, ergodicity in the QMC sampling process is not guaranteed for large systems[53]. These problems do not arise in the exact and independent autoregressive sampling of the neural network algorithms, explaining the lower uncertainties observed for smaller sample sizes. At the same time, these observations limit accurate QMC simulations to small system sizes.

Implementation details

We train the network to find the ground state of the Rydberg Hamiltonian by minimizing the energy expectation value, which we evaluate using Eq. (6) and Eq. (7) with the Hamiltonian operator $\hat{\mathcal{H}}$,

$$\begin{aligned}\langle E\rangle &= \langle \Psi_{\mathcal{W}}|\hat{\mathcal{H}}|\Psi_{\mathcal{W}}\rangle \\ &= \sum_{\{\boldsymbol{\sigma}, \boldsymbol{\sigma}'\}} \Psi^*(\boldsymbol{\sigma}; \mathcal{W})\,\Psi(\boldsymbol{\sigma}'; \mathcal{W})\langle \boldsymbol{\sigma}|\hat{\mathcal{H}}|\boldsymbol{\sigma}'\rangle,\end{aligned} \quad (20)$$

where $\Psi(\boldsymbol{\sigma}; \mathcal{W})$ denotes a wavefunction encoded in one of the discussed network models. To optimize the variational network parameters, we use the gradient descent algorithm the same way as discussed in[6,7], with gradients

$$\partial_{\mathcal{W}_i}E \approx \frac{2}{N_s}\sum_{s=1}^{N_s}\partial_{\mathcal{W}_i}\Psi^*(\boldsymbol{\sigma}_s; \mathcal{W})\left[E_{\mathrm{loc}}(\boldsymbol{\sigma}_s; \mathcal{W}) - \langle E\rangle\right], \quad (21)$$

and local energy

$$E_{\mathrm{loc}}(\boldsymbol{\sigma}_s; \mathcal{W}) = \sum_{\{\boldsymbol{\sigma}'\}}\langle \boldsymbol{\sigma}_s|\hat{\mathcal{H}}|\boldsymbol{\sigma}'\rangle\frac{\Psi(\boldsymbol{\sigma}'; \mathcal{W})}{\Psi(\boldsymbol{\sigma}_s; \mathcal{W})}. \quad (22)$$

The training process requires the evaluation of the gradients of $\Psi(\boldsymbol{\sigma}; \mathcal{W})$. To reduce the necessary amount of memory, we always generate a batch of N_s samples from the network without evaluating the gradients. We then pass each sample through the network again to obtain the wavefunction amplitude with the corresponding gradients. This approach requires $2N_s$ network passes instead of N_s, but evaluating gradients on a given input sequence is less memory-consuming than evaluating gradients on an autoregressive process in PyTorch[60]. We can further reduce the required memory by dividing the total batch of N_s samples into mini batches of K samples each, which are evaluated in separate processes. This reduces the memory scaling by a factor K/N_s per pass, while requiring N_s/K network passes instead of one. Thus, the smaller we choose the mini batches, the less memory is required, but the longer the computational runtime. If not stated otherwise, we choose $K = 256$ in this work.

Considering the off-diagonal term in the Rydberg Hamiltonian, its contribution to the energy expectation value is given by $E_{\mathrm{off}} = \sum_{i=1}^N \langle \hat{\sigma}_i^x\rangle$. Thus, calculating the local energy $E_{\mathrm{loc}}(\boldsymbol{\sigma}_s; \mathcal{W})$ using Eq. (7) requires for each sampled state $\boldsymbol{\sigma}_s$ to evaluate $\Psi(\boldsymbol{\sigma}'; \mathcal{W})$ for all $\boldsymbol{\sigma}'$ that correspond to the sampled state with one atom flipped from the ground to the excited state or vice versa[6,7]. Instead of passing N states through the network for each sample we generate, we can reduce the required memory and accelerate our algorithm by splitting the atom sequence into D equally sized parts. In each part, the wavefunction amplitude is evaluated for the states where each of the N/D atoms is flipped. When calculating the wavefunction amplitudes using sequential networks, the flipping of one atom only affects the calculation for atoms that appear later in the sequence. We store the network outcome for the original state $\boldsymbol{\sigma}_s$ after each N/D atoms and evaluate each group starting from this stored value. We then only need to pass the sequence from the first atom of the group to the last atom in the system to the network. This ansatz can be used since no gradient needs to be evaluated on this off-diagonal term and corresponds to using mini batches of atoms. This way we reduce the amount of required memory by a factor D/N per pass, while requiring N/D network passes instead of only one. While we now evaluate the network N/D times, the sequence length at iteration d is reduced from N/p to $(N - [d - 1]D)/p$, leading to an accelerated evaluation of the local energies since the input sequence length is reduced in most cases. In this work, we always split the sequence of atoms into $D = N/p$ parts, with p patch size in the patched network models.

We base our simulations on PyTorch[60] and NumPy[61], and use Matplotlib[62] to visualize our results.

Data availability

All presented data can be reproduced with the publicly available source code. It is further available upon request to the corresponding author.

Code availability

The source code used to generate the data in this work is available on https://github.com/APRIQuOt/VMC_with_LPTFs.git. It is based on PyTorch[60] and NumPy[61] and we used Matplotlib[62] for visualizing our results.

References

1. Carleo, G. & Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science* **355**, 602–606 (2017).
2. Torlai, G. et al. Neural-network quantum state tomography. *Nat. Phys.* **14**, 447–450 (2018).
3. Torlai, G. & Melko, R. G. Latent space purification via neural density operators. *Phys. Rev. Lett.* **120**, 240503 (2018).
4. Dawid, A. et al. Modern applications of machine learning in quantum sciences, arXiv:2204.04198 [cond-mat] (2022).
5. Carrasquilla, J. Machine learning for quantum matter. *Adv. Phys.-X* **5**, 1797528 (2020).
6. Carrasquilla, J. & Torlai, G. How To Use Neural Networks To Investigate Quantum Many-Body Physics. *PRX Quantum* **2**, 040201 (2021).
7. Hibat-Allah, M., Ganahl, M., Hayward, L. E., Melko, R. G. & Carrasquilla, J. Recurrent neural network wave functions. *Phys. Rev. Res.* **2**, 023358 (2020).
8. Czischek, S., Moss, M. S., Radzihovsky, M., Merali, E. & Melko, R. G. Data-enhanced variational Monte Carlo simulations for Rydberg atom arrays. *Phys. Rev. B* **105**, 205108 (2022).
9. Viteritti, L. L., Ferrari, F. & Becca, F. Accuracy of restricted Boltzmann machines for the one-dimensional $J_1 - J_2$ Heisenberg model. *SciPost. Phys.* **12**, 166 (2022).
10. Neugebauer, M. et al. Neural-network quantum state tomography in a two-qubit experiment. *Phys. Rev. A* **102**, 042604 (2020).
11. Schmale, T., Reh, M. & Gärttner, M. Efficient quantum state tomography with convolutional neural networks. *npj Quant. Inf.* **8**, 115 (2022).
12. Torlai, G. et al. Integrating neural networks with a quantum simulator for state reconstruction. *Phys. Rev. Lett.* **123**, 230504 (2019).
13. Morawetz, S., De Vlugt, I. J. S., Carrasquilla, J. & Melko, R. G. U(1)-symmetric recurrent neural networks for quantum state reconstruction. *Phys. Rev. A* **104**, 012401 (2021).
14. Cha, P. et al. Attention-based quantum tomography. *Mach Learn: Sci. Technol.* **3**, 01LT01 (2022).
15. Carrasquilla, J., Torlai, G., Melko, R. G. & Aollita, L. Reconstructing quantum states with generative models. *Nat. Mach. Intell* **1**, 155–161 (2019).
16. Torlai, G., Mazzola, G., Carleo, G. & Mezzacapo, A. Precise measurement of quantum observables with neural-network estimators. *Phys. Rev. Res.* **2**, 022060 (2020).
17. Schmitt, M. & Heyl, M. Quantum many-body dynamics in two dimensions with artificial neural networks. *Phys. Rev. Lett.* **125**, 100503 (2020).
18. Nagy, A. & Savona, V. Variational quantum monte carlo method with a neural-network ansatz for open quantum systems. *Phys. Rev. Lett.* **122**, 250501 (2019).
19. Vicentini, F., Biella, A., Regnault, N. & Ciuti, C. Variational neural-network ansatz for steady states in open quantum systems. *Phys. Rev. Lett.* **122**, 250503 (2019).
20. Hartmann, M. J. & Carleo, G. Neural-network approach to dissipative quantum many-body dynamics. *Phys. Rev. Lett.* **122**, 250502 (2019).
21. Reh, M., Schmitt, M. & Gärttner, M. Time-dependent variational principle for open quantum systems with artificial neural networks. *Phys. Rev. Lett.* **127**, 230501 (2021).
22. Czischek, S., Gärttner, M. & Gasenzer, T. Quenches near ising quantum criticality as a challenge for artificial neural networks. *Phys. Rev. B* **98**, 024311 (2018).
23. Melko, R. G., Carleo, G., Carrasquilla, J. & Cirac, J. I. Restricted Boltzmann machines in quantum physics. *Nat. Phys.* **15**, 887–892 (2019).
24. Hibat-Allah, M., Inack, E. M., Wiersema, R., Melko, R. G. & Carrasquilla, J. Variational neural annealing. *Nat. Mach. Intell.* **3**, 952–961 (2021).
25. Hibat-Allah, M., Melko, R. G. & Carrasquilla, J. Investigating topological order using recurrent neural networks. *Phys. Rev. B* **108**, 075152 (2023).
26. Sharir, O., Levine, Y., Wies, N., Carleo, G. & Shashua, A. Deep autoregressive models for the efficient variational simulation of many-body quantum systems. *Phys. Rev. Lett.* **124**, 020503 (2020).
27. Valenti, A., Greplova, E., Lindner, N. H. & Huber, S. D. Correlation-enhanced neural networks as interpretable variational quantum states. *Phys. Rev. Res.* **4**, L012010 (2022).
28. Hibat-Allah, M., Melko, R. G. & Carrasquilla, J. Supplementing Recurrent Neural Network Wave Functions with Symmetry and Annealing to Improve Accuracy, arXiv:2207.14314 [cond-mat] (2022).
29. Ahsan Khandoker, S., Munshad Abedin, J. & Hibat-Allah, M. Supplementing recurrent neural networks with annealing to solve combinatorial optimization problems. *Mach. Learn: Sci. Technol.* **4**, 015026 (2023).
30. Luo, Di et al. Gauge-invariant and anyonic-symmetric autoregressive neural network for quantum lattice models. *Phys. Rev. Res.* **5**, 013216 (2023).
31. Bennewitz, E. R., Hopfmueller, F., Kulchytskyy, B., Carrasquilla, J. & Ronagh, P. Neural error mitigation of near-term quantum simulations. *Nat. Mach. Intell.* **4**, 618–624 (2022).
32. Vaswani, A. et al. Attention Is All You Need, arXiv:1706.03762 [cs] (2017).
33. Zhang, Y.-H. & Di Ventra, M. Transformer Quantum State: A Multi-Purpose Model for Quantum Many-Body Problems. *Phys. Rev. B* **107**, 075147 (2023).
34. Viteritti, L. L., Rende, R. & Becca, F. Transformer variational wave functions for frustrated quantum spin systems. *Phys. Rev. Lett.* **130**, 236401 (2023).
35. Sharir, O., Chan, G. K.-L. & Anandkumar, A. Towards Neural Variational Monte Carlo That Scales Linearly with System Size, arXiv:2212.11296 [quant-ph] (2022).
36. Ma, H., Sun, Z., Dong, D., Chen, C. & Rabitz, H. Tomography of Quantum States from Structured Measurements via quantum-aware transformer https://doi.org/10.48550/arXiv.2305.05433, arXiv:2305.05433 [quant-ph] (2023).
37. An, Z., Wu, J., Yang, M., Zhou, D. L. & Zeng, B. Unified quantum state tomography and Hamiltonian learning: A language-translation-like approach for quantum systems. *Phys. Rev. Appl.* **21**, 014037 (2024).
38. von Glehn, I., Spencer, J. S. & Pfau, D. A self-attention ansatz for ab-initio quantum chemistry. https://doi.org/10.48550/arXiv.2211.13672, arXiv:2211.13672 [physics.chem-ph] (2022).
39. Carrasquilla, J. et al. Probabilistic simulation of quantum circuits using a deep-learning architecture. *Phys. Rev. A* **104**, 032610 (2021).
40. Luo, D., Chen, Z., Carrasquilla, J. & Clark, B. K. Autoregressive neural network for simulating open quantum systems via a probabilistic formulation. *Phys. Rev. Lett.* **128**, 090501 (2022).
41. Dosovitskiy, A. et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, arXiv:2010.11929 [cs] (2021).
42. Jaksch, D. et al. Fast Quantum Gates for Neutral Atoms. *Phys. Rev. Lett.* **85**, 2208–2211 (2000).

43. Lukin, M. D. et al. Dipole Blockade and Quantum Information Processing in Mesoscopic Atomic Ensembles. *Phys. Rev. Lett.* **87**, 037901 (2001).

44. Endres, M. et al. Atom-by-atom assembly of defect-free one-dimensional cold atom arrays. *Science* **354**, 1024–1027 (2016).

45. Barredo, D., Lienhard, V., de Léséleuc, S., Lahaye, T. & Browaeys, A. Synthetic three-dimensional atomic structures assembled atom by atom. *Nature* **561**, 79–82 (2018).

46. Samajdar, R., Ho, W. W., Pichler, H., Lukin, M. D. & Sachdev, S. Complex Density Wave Orders and Quantum Phase Transitions in a Model of Square-Lattice Rydberg Atom Arrays. *Phys. Rev. Lett.* **124**, 103601 (2020).

47. Samajdar, R., Ho, W. W., Pichler, H., Lukin, M. D. & Sachdev, S. Quantum phases of Rydberg atoms on a kagome lattice. *PNAS* **118**, e2015785118 (2021).

48. Ebadi, S. et al. Quantum phases of matter on a 256-atom programmable quantum simulator. *Nature* **595**, 227–232 (2021).

49. Scholl, P. et al. Quantum simulation of 2D antiferromagnets with hundreds of Rydberg atoms. *Nature* **595**, 233–238 (2021).

50. Xu, W. et al. Fast Preparation and Detection of a Rydberg Qubit Using Atomic Ensembles. *Phys. Rev. Lett.* **127**, 050501 (2021).

51. Miles, C. et al. Machine learning discovery of new phases in programmable quantum simulator snapshots. *Phys. Rev. Res.* **5**, 013026 (2023).

52. Kalinowski, M. et al. Bulk and Boundary Quantum Phase Transitions in a Square Rydberg Atom Array. *Phys. Rev. B* **105**, 174417 (2022).

53. Merali, E., De Vlugt, I. J. S. & Melko, R. G. Stochastic Series Expansion Quantum Monte Carlo for Rydberg Arrays, arXiv:2107.00766 [cond-mat] (2023).

54. Bravyi, S., DiVincenzo, D. P., Oliveira, R. I. & Terhal, B. M. The Complexity of Stoquastic Local Hamiltonian Problems. *Quant. Info. Comput.* **8**, 361–385 (2008).

55. Cho, K. et al. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, in https://doi.org/10.3115/v1/D14-1179 *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* 1724–1734 (Association for Computational Linguistics, Doha, Qatar, 2014).

56. Becca, F. & Sorella, S. https://doi.org/10.1017/9781316417041 *Quantum Monte Carlo Approaches for Correlated Systems*, 1st ed. (Cambridge University Press, 2017).

57. Morin, F. & Bengio, Y. Hierarchical probabilistic neural network language model, in https://proceedings.mlr.press/r5/morin05a.html *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, (eds Cowell, R. G. & Ghahramani, Z.) Vol. R5, 246–252 (PMLR, 2005).

58. Zhai, X., Kolesnikov, A., Houlsby, N. & Beyer, L. https://doi.org/10.48550/arXiv.2106.04560 Scaling vision transformers, arXiv:2106.04560 [cs.CV] (2021).

59. Hochreiter, S. & Schmidhuber, J. Long Short-Term Memory. *Neural. Comput.* **9**, 1735–1780 (1997).

60. Paszke, A. et al. Pytorch: An imperative style, high-performance deep learning library, in http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf *Advances in Neural Information Processing Systems 32* 8024–8035 (Curran Associates, Inc., 2019).

61. Harris, C. R. et al. Array programming with NumPy. *Nature* **585**, 357–362 (2020).

62. Hunter, J. D. Matplotlib: A 2d graphics environment. *Comput. Sci. Eng.* **9**, 90–95 (2007).

Acknowledgements

Author contributions

The fundamental ideas of the introduced approach were developed by K. Sprague who further implemented and organized the used Python code. S. Czischek used the provided code to obtain the presented results with support by K. Sprague. The manuscript was written by S. Czischek with valuable feedback by K. Sprague.

Competing interests

The authors declare no competing interests. S. Czischek is a guest editor for *Communications Physics*, but was not involved in the editorial review of, or the decision to publish this article.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42005-024-01584-y.

Correspondence and requests for materials should be addressed to Stefanie Czischek.

Peer review information This manuscript has been previously reviewed at another Nature Portfolio journal. The manuscript was considered suitable for publication without further review at *Communications Physics*.

Reprints and permissions information is available at http://www.nature.com/reprints

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.