

Design principles of deep translationally symmetric neural quantum states for frustrated magnets

Rajah P. Nutakki ^{*}, Ahmedeo Shokry , and Filippo Vicentini 

CPHT, CNRS, École Polytechnique, Institut Polytechnique de Paris, 91120 Palaiseau, France;

Collège de France, Université PSL, 11 place Marcelin Berthelot, 75005 Paris, France;

Inria Paris-Saclay, 91120 Palaiseau, France;

and LIX, CNRS, École Polytechnique, Institut Polytechnique de Paris, 91120 Palaiseau, France



(Received 13 May 2025; accepted 15 September 2025; published 24 October 2025)

Deep neural-network quantum states have emerged as a leading method for studying the ground states of quantum magnets. Successful architectures exploit translational symmetry, but the root of their effectiveness and differences between architectures remain unclear. Here, we apply the ConvNext architecture, designed to incorporate elements of transformers into convolutional networks, to quantum many-body ground states. We find that it is remarkably similar to the factored vision transformer, which has been employed successfully for several frustrated spin systems, allowing us to relate this architecture to more conventional convolutional networks. Through a series of numerical experiments, we design the ConvNext to achieve greatest performance at lowest computational cost, and then apply this network to the Shastry-Sutherland and $J_1 - J_2$ models, obtaining variational energies comparable to the state of the art, providing a blueprint for network design choices of translationally symmetric architectures to tackle challenging ground-state problems in frustrated magnetism.

DOI: [10.1103/ybgv-35jm](https://doi.org/10.1103/ybgv-35jm)

I. INTRODUCTION

Investigating the ground-state properties of frustrated quantum magnets remains a formidable challenge in computational physics, yet crucial for understanding what states of matter can be experimentally realized in solid-state and synthetic quantum platforms. Several methods have been developed over the years, but have their limitations. Tensor networks [1–5], for example, are a reliable tool when an efficient contraction scheme exists, while quantum Monte Carlo techniques [6–11] are generally well suited for investigating the finite temperature properties of sign-problem-free Hamiltonians.

Neural quantum states (NQSs) [12] have recently emerged as a promising family of variational wavefunctions for use within Variational Monte Carlo (VMC) [11], achieving state-of-the-art accuracy for 2D frustrated quantum spin systems [13,14], such as the $J_1 - J_2$ model on the square lattice [15–19], and offering promising results for the calculation of their dynamical properties [20–22]. Their recent success stems from the ability to optimize large networks with many variational parameters using stochastic reconfiguration [11,15,23–26], which is closely connected to imaginary time evolution.

Many neural-network wavefunctions rely on the same or slightly adapted *encoder* architectures used in several do-

main of Machine Learning, with only minor modifications to the final layers in order to output complex-valued amplitudes instead of a set of classification probabilities or text. Two successful encoder architectures, used to obtain the most accurate results for variational ground-state calculations, are the convolutional residual network (ResNet) [15] and the vision transformer with factored attention (fViT) [13,16,27,28]. Both are built out of translationally equivariant blocks, allowing one to efficiently enforce translational symmetries in the variational wavefunction, which appears to be instrumental in the success of the method. Evidence is emerging that more general attention mechanisms that are not translationally equivariant do not (significantly) improve performance [29,30]. However, the similarities and differences between the fViT and ResNet architectures have not been investigated, and it remains unclear how best to exploit their properties when studying different frustrated spin systems.

In this work, we implement the ConvNext [31,32] as an NQS, an architecture designed to elevate the performance of convolutional neural networks (CNNs) to that of transformers for computer vision tasks. Our motivation is to harness the power of transformers [33,34], while retaining the ability to efficiently impose translational symmetries, as well as the locality inherent in convolutional networks. Remarkably, we find that the ConvNext is similar to the fViT [27] (and can be made almost equivalent by specific choices of hyperparameters), showing the convergence of two directions in network design: making a CNN more transformerlike or making the attention in a ViT translationally equivariant. Thus, the ConvNext offers a way to understand translationally symmetric architectures in a clear and unified manner. First, we summarize the properties of translationally symmetric NQSs (Sec. III), putting their components into the embedding-encoder-readout framework of Ref. [13]. We then

^{*}Contact author: rajah.nutakki@polytechnique.edu

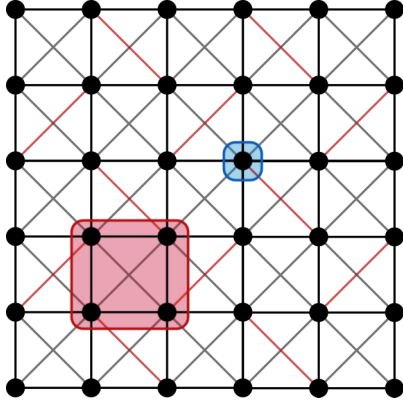


FIG. 1. The J_1 (black) and J_2 couplings of the (red) Shastry-Sutherland and (gray and red) $J_1 - J_2$ models. The sites in the unit cell are highlighted for the Shastry-Sutherland (red) and $J_1 - J_2$ (blue) models.

perform a series of numerical experiments (Sec. IV) on the ConvNext clarifying the effect of patching, network depth, kernel width, and the role of the readout block in light of this framework. Using our optimal design, we obtain variational energies for two benchmark problems, the $J_1 - J_2$ model and Shastry-Sutherland, which are comparable to the state of the art with NQS. We end by discussing (Sec. V) outstanding questions on translationally symmetric NQS and comparing to other types of architectures. Our work makes clear the design choices when employing translationally symmetric NQSs for ground states of frustrated spin systems, such that they can be employed to tackle challenging problems in frustrated magnetism and beyond.

II. THE SHASTRY-SUTHERLAND MODEL

We use the Shastry-Sutherland model [35] as our benchmark to evaluate the performance of different NQS architecture choices. We make this choice as the model has a multisite unit cell, like many in frustrated magnetism [36], and was recently studied using the fViT, so can compare with Ref. [13]. We also evaluate the performance of our final network design on the related $J_1 - J_2$ model, allowing us to compare to many other approaches that have been applied to the same model.

Both models are defined by the $S = 1/2$ Hamiltonian

$$\hat{H} = J_1 \sum_{\langle ij \rangle} \hat{\mathbf{S}}_i \cdot \hat{\mathbf{S}}_j + J_2 \sum_{(ij) \in J_2} \hat{\mathbf{S}}_i \cdot \hat{\mathbf{S}}_j, \quad (1)$$

where $\hat{\mathbf{S}}_i = (\hat{S}_i^x, \hat{S}_i^y, \hat{S}_i^z) = 1/2(\hat{\sigma}_i^x, \hat{\sigma}_i^y, \hat{\sigma}_i^z)$ with variables on a square grid. The J_1 term couples nearest-neighbor sites, whereas J_2 couples all ($J_1 - J_2$ model) or 1/4 (Shastry-Sutherland) of next-nearest neighbors, as shown in Fig. 1.

Both the Shastry-Sutherland [13,37–39] and $J_1 - J_2$ [40–43] models are thought to host a $\mathbb{Z}_2$ Dirac spin liquid [44,45] ground state, which were recently proposed to be part of the same phase [46]. The ground state in these parameter regimes is long-range correlated, has a nontrivial sign structure, and nearby competing phases, making computations challenging and therefore a useful benchmark.

For the remainder of this manuscript, we will restrict ourselves to $J_1 = 1$, $J_2 = 0.8$ of the Shastry-Sutherland and $J_1 = 1$, $J_2 = 0.5$ of the $J_1 - J_2$ model, within the putative spin liquid phase, taking the model on an $L \times L$ square geometry with periodic boundary conditions.

III. TRANSLATIONALLY SYMMETRIC NEURAL QUANTUM STATES

Sign structure and embedding. A major hurdle for NQS-based VMC applications to frustrated (or fermionic) ground states is learning their sign structure [47,48], which can even lead to an apparent exponential complexity emerging if a proper prior is not used [49]. While a clear cause is unknown, it might be related to vanishing amplitudes of the wavefunction, which increase the variance of the gradient estimator, emerging when the optimization attempts to flip the sign [50]. For models on bipartite lattices, the sign structure (Marshall sign rule) is known exactly [51]. For nonbipartite systems, like the $J_1 - J_2$ model, the sign rule may still serve as a good approximate prior [17,48]; however, for a general lattice this structure is not analytically known.

Starting from a wavefunction that already captures a reasonable approximation to the sign structure, either through prior knowledge or inductive bias, can significantly improve optimization and lead to more accurate ground-state calculations. For example, when signs depend on sublattice-dependent quantities, as is the case in models that (approximately) obey the Marshall sign rule, it is advantageous to design the network to operate on these quantities.

This can be done using an *embedding* layer at the beginning of the network, which partitions the lattice into P patches, with each of the sites within a patch assigned to a separate sublattice, as illustrated in Fig. 2. We denote the input configuration in the computational basis as $\sigma_{\mathbf{r},x}$, where $\mathbf{r}$ labels the position of the patch (the dashed squares in Fig. 2) and x labels the sites (sublattices) within the patch. The embedding layer f^s then applies a sublattice-dependent linear transformation W_x^μ to the spin variables within each patch. This embedding is written as (suppressing vector notation for clarity)

$$\tilde{\sigma}_r^\mu = f^s(\sigma)_r^\mu = \sum_x W_x^\mu \sigma_{\mathbf{r},x}, \quad (2)$$

where μ indexes the embedding dimension (features), of total size N_f . Because this transformation explicitly acts on the internal sublattice structure of each patch, it naturally biases the network to learn the sign structure of the ground state on a bipartite lattice, discussed further in Appendix B 1.

In networks that do not include this embedding, it has been necessary to provide the sign rule as a prior to facilitate training and avoid convergence issues [15].

Symmetries and encoder. The core of a translationally symmetric NQS is the encoder, which processes the variables $\tilde{\sigma}_r^\mu$ through a series of translationally equivariant transformations,

$$\tilde{\sigma}_{ir}[l+1] = g^e(t^{-1} \tilde{\sigma}_r[l]), \quad (3)$$

where g^e is the encoder function, t is an element of the translation group, T , l labels the layers of the encoder, and $\tilde{\sigma}_r = (\tilde{\sigma}_r^0, \dots, \tilde{\sigma}_r^{N_f-1})$ [17], [52]. This encoder is typically

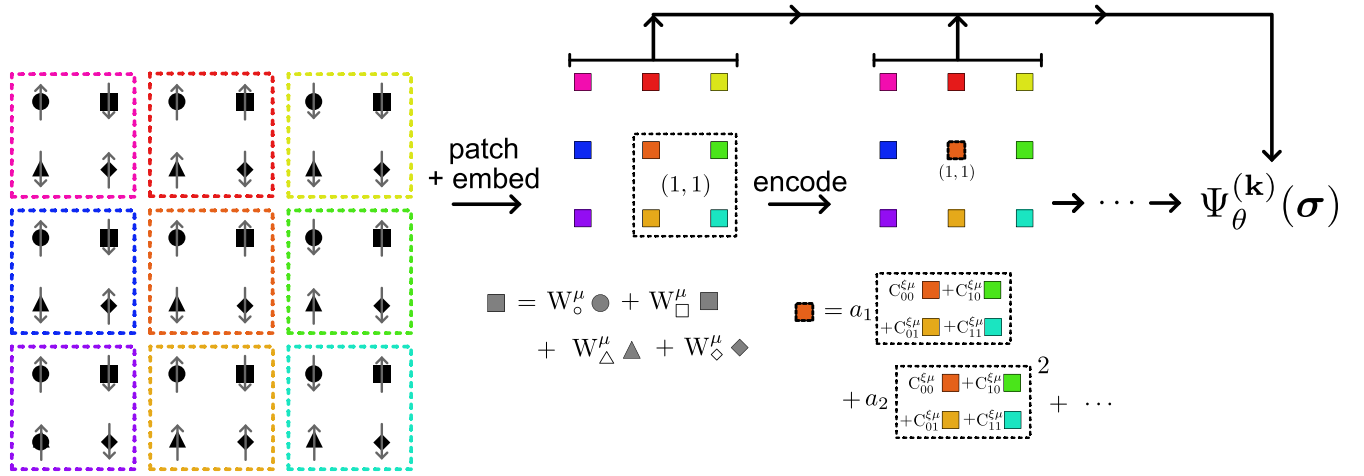


FIG. 2. Illustration of a patched translationally symmetric NQS. The patching and embedding of the input means the remainder of the network operates on sublattice-dependent quantities, a useful bias for learning the sign structure of ground states. Convolutionlike encoding blocks then construct a hidden representation out of (high-order) correlators of these quantities, before projection to a momentum, $\mathbf{k}$, and readout of a complex amplitude.

built from convolutional layers, combined with nonlinearities and skip connections.

The network can be later made translationally symmetric (up to the level of the patches) by projecting to a specific momentum $\mathbf{k}$,

$$\Psi_{\theta}^{(\mathbf{k})}(\sigma) = \frac{1}{|T|} \sum_{t \in T} e^{-i\mathbf{k} \cdot \mathbf{r}_t} \Psi_{\theta}(t^{-1}\sigma), \quad (4)$$

from a single evaluation of the network, where $\mathbf{r}_t$ is the shift in coordinates corresponding to t [53]. Point group and spin symmetries can be included with an overhead of $|G|$ network evaluations [Eq. (A11)], where $|G|$ is the number of elements in the corresponding group.

From a physical point of view, the encoder can be thought of as extracting σ_i^z correlators of all orders (see Appendix B 2). At each layer, the patched features $\tilde{\sigma}_r^{\mu}[l]$ are updated according to

$$\begin{aligned} \tilde{\sigma}_r[l+1] &= \tilde{\sigma}_r[l] + g^e \left(\tilde{\sigma}_{r+\begin{smallmatrix} \blacksquare & \blacksquare \\ \blacksquare & \blacksquare \end{smallmatrix}}[l], \tilde{\sigma}_{r+\begin{smallmatrix} \blacksquare & \blacksquare \\ \blacksquare & \blacksquare \end{smallmatrix}}[l], \dots, \tilde{\sigma}_{r+\begin{smallmatrix} \blacksquare & \blacksquare \\ \blacksquare & \blacksquare \end{smallmatrix}}[l] \right), \end{aligned} \quad (5)$$

where $\tilde{\sigma}_r[l]$ denotes the feature vector at patch r in layer l of the encoder, corresponding to the residual connection. The updated feature at the next layer, $\tilde{\sigma}_r[l+1]$, is computed as the sum of the current feature and a contribution from the neighboring patches within the receptive field. These neighbors are located at relative offsets $\{\begin{smallmatrix} \blacksquare & \blacksquare \\ \blacksquare & \blacksquare \end{smallmatrix}, \dots\}$ from patch r . In essence, the new representation at patch r aggregates information from its local neighborhood while retaining its original feature. The skip connection ensures that the original features (sublattice-dependent magnetizations) are preserved and passed along, enabling the network to encode higher-order structure on top without foregoing lower-order contributions. The recursive structure allows the encoder to build up combinations of the input variables, including products of σ_i^z values

at different sites, generating multispin correlators across the lattice.

The depth of the encoder and the size of the receptive field determine how nonlocal and how many different spin variables these correlators can contain. The encoder uses these correlators to construct a hidden representation, which a final readout layer aggregates, projecting to a desired momentum sector [Eq. (4)] and producing a complex scalar amplitude.

These elements can be broadly organized into an embedding-encoder-readout structure [13] (see, e.g., Fig. 3), which we find useful for discussing the various properties of the network in this article.

A. ConvNext quantum state

In this section, we discuss an adaptation of the ConvNext architecture, originally introduced for computer vision tasks [31,32], to represent quantum many-body states (see Fig. 3). The ConvNext combines structural elements of CNNs and transformers, and its use as an NQS provides valuable insights into designing translationally symmetric neural architectures.

The original ConvNext architecture [31] consists of multiple encoding stages, each progressively patching the input into smaller spatial dimensions while increasing the number of feature channels. As a result, the network outputs a highly compact representation with rich feature content at each spatial location. However, each additional patching stage partially breaks translational symmetry within the resulting patches, a trade-off typically acceptable or even beneficial in general machine learning applications [54].

For neural quantum states, however, preserving exact lattice translational symmetry is essential for accurate calculations. We therefore restrict ourselves to a single-stage architecture, patching only one time and therefore losing only a small number of symmetries (if the lattice has a single-site unit cell) that could be restored later. Under this constraint, the ConvNext encoder closely resembles the fViT architecture. Nevertheless, the hierarchical multistage structure could

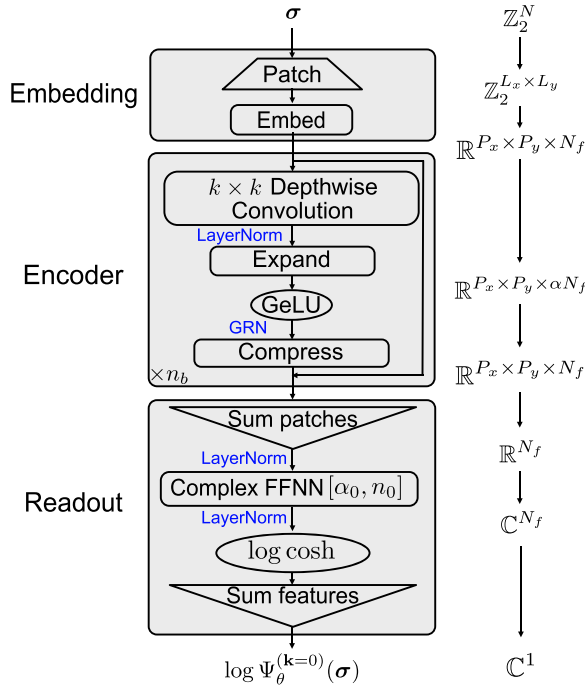


FIG. 3. The ConvNext architecture adapted for use as an NQS. The right-hand side displays the shape and type of the data as it passes through the network. The number of features, N_f , expansion ratio, α , number of encoder blocks, n_b , and convolutional kernel width, k , and size (n_o , α_o) of the feedforward neural network (FFNN) in the readout block are all hyperparameters. The number of patches along the Cartesian axes are denoted by P_x , P_y . The embedding and encoder have real-valued parameters, with a complex-valued FFNN in the readout head resulting in a complex output.

become advantageous when dealing with very large quantum systems consisting of thousands of lattice sites [55].

Patching and embedding. The first layer of a ConvNext stage is a stride- s convolution that reduces the spatial dimension of the input by a factor of s^2 and increases the number of features [31]. This operation is mathematically equivalent to first reshaping the input into $s \times s$ patches and then applying a learnable embedding, as done in the embedding layer (2) used by the fViT (see Fig. 4). Here, we use the patch-and-embed implementation, which we find to be more efficient for the relatively small input sizes encountered in NQS.

Encoder. The ConvNext encoder transforms the embedded variables $\tilde{\sigma}_r^\mu$ through a series of convolutional blocks designed to efficiently encode spatial correlations. Previous convolutional wavefunctions built on top of ResNets [15,56] relied on layers of standard convolutions that mix all spatial and feature degrees of freedom. Those can be written as

$$\tilde{\sigma}_r^\xi[l+1] = g\left(\sum_{\mu} \sum_{\delta} C_{\delta}^{\xi,\mu} \tilde{\sigma}_{r+\delta}^\mu[l]\right) + \tilde{\sigma}_r^\xi[l], \quad (6)$$

where C is the convolutional kernel (a $k \times k \times N_f^2$ filter for 2D lattices) and g is some elementwise nonlinearity. While effective, these convolutions are relatively costly, as their memory usage and computational requirements scale quadratically with both k and N_f .

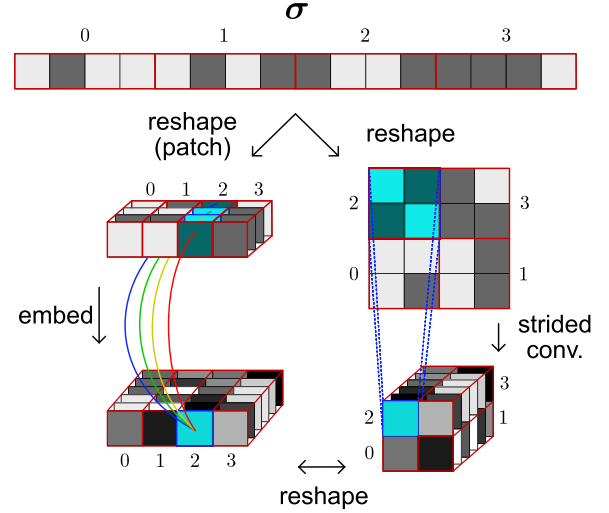


FIG. 4. Two different ways of patching and embedding the input. The numbers label the patches, which are made up of the variables in red boxes. (Left) The data are first reshaped (patched) to $\mathbb{Z}_2^{P \times N_x}$, where N_x is the number of sublattices created by the patching, before embedding to $\mathbb{R}^{P \times N_f}$. (Right) The data are reshaped to its lattice positions, before a strided convolution simultaneously performs the patching and embedding operations to $\mathbb{R}^{P_x \times P_y \times N_f}$. The two are mathematically equivalent.

The key insight behind the ConvNext [31] is to reduce this computational burden by splitting the convolution into two simpler steps. First, a depthwise convolution acts independently on each feature channel, mixing only spatial information:

$$\text{Conv}(\tilde{\sigma})_r^\xi = \sum_{\delta} K_{\delta}^{\xi} \tilde{\sigma}_{r+\delta}^{\xi}[l], \quad (7)$$

where K is a depthwise kernel of size $k \times k \times N_f$. This is followed by a feature-mixing nonlinear transformation $g(\cdot)$ that does not mix the spatial information. This nonlinearity is, in practice, implemented as a two-layer multilayer perceptron with a single nonlinearity,

$$g^{\xi}(\tilde{\sigma}_r) = \sum_{\nu=1}^{\alpha N_f} V^{\xi,\nu} \text{GeLU}\left(\sum_{\mu=1}^{N_f} W^{\nu,\mu} \tilde{\sigma}_r^{\mu}\right), \quad (8)$$

with matrices W and V having dimensions $(\alpha N_f, N_f)$ and $(N_f, \alpha N_f)$, respectively, where α controls the hidden layer density (also known as the *expansion factor*). Thus, the overall encoder layer becomes

$$\tilde{\sigma}_r^{\xi}[l+1] = \tilde{\sigma}_r^{\xi}[l] + g^{\xi}(\text{Conv}(\tilde{\sigma}[l])_r). \quad (9)$$

This factorization reduces the computational cost from quadratic to linear scaling in the embedding dimension, and it has been shown empirically that this does not substantially reduce the expressivity. In practice, α is set to between 1 and 4 and N_f is on the order of TENS to hundreds. Therefore, the vast majority of the variational parameters are found within this nonlinearity and not in the spatial convolutions.

Relationship to the fViT. We remark that the ConvNext, being inspired by a vision transformer, shares with it the same nonlinear activation g . The greatest difference between a ViT

and a ConvNext is in the attention mechanism: A standard ViT would not use a linear convolution to mix different spatial (patch) degrees of freedom but instead a dense matrix that depends on the input. Rende *et al.* [27] pointed out that this attention would not be translationally equivariant and proposed to replace it with a translationally equivariant factored attention that restores this equivariance [27], later showing further empirical evidence motivating this choice [57]. As detailed in Appendix B 3, this is mathematically equivalent to a depthwise convolution (as in the ConvNext) when employing as many heads as features and choosing a kernel size spanning the entire system.

Readout. Given the similarities between the ConvNext and fViT, we use a similar readout block to Ref. [28], first summing over positions (projecting to the $\mathbf{k} = 0$ sector) before applying a complex feed-forward network of depth n_o and hidden layers of size $\alpha_o N_f$. Note that this design does not allow one to project to $\mathbf{k} \neq 0$ sectors, so we also verify the performance of a modified version that does.

IV. NUMERICAL EXPERIMENTS

We perform a series of numerical experiments to identify the optimal design and hyperparameter choices for the ConvNext. As a benchmark, we decide to use the ground-state search of the Shastry-Sutherland model. We use, where possible, the same optimization protocol and sampling procedure, as described in Appendix C.

The main hyperparameters for the network are the depth of the encoder, n_b , kernel width of the convolution, k , and encoder expansion ratio, α , as well as the depths, n_o , and expansion factor, α_o , in the readout. In order to make our results comparable to those in Ref. [13], we fix $N_f = 72$. We specify a choice of hyperparameters with the notation $(n_b, k, \alpha)[n_o, \alpha_o]$. Since the unit cell of the model is a 2×2 plaquette (Fig. 1), we use these as the patches, so that we can easily enforce the translational symmetry of the model.

A. Patching

First, we compare the effect of patching versus not patching the input in the embedding layer. When not patching, we still project to the relevant translational-invariant symmetry subsector in the readout (details are provided in Appendix D), therefore removing the inductive bias of the patching as well as treating inter- and intra-unit-cell correlations on an equal footing, without modifying the symmetry of the wavefunction.

We perform a VMC calculation for an $L = 6$ system with a $(4, 3, 2)[1, 1]$ network with and without patching, with results shown in Fig. 5. We find that patching improves the accuracy of the final solution, reducing the relative error

$$\epsilon = \left| \frac{E - E_{\text{GS}}}{E_{\text{GS}}} \right|, \quad (10)$$

from 2.7×10^{-4} without patching to 8.0×10^{-5} with patching. Furthermore, the patching reduces the cost of evaluating the network by approximately a factor of 4 (when using an equal number of features; see Fig. 7).

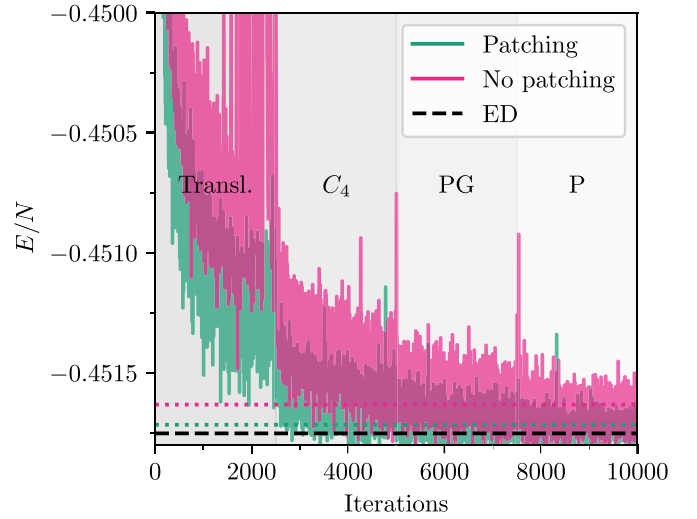


FIG. 5. Optimization of the $(4, 3, 2)[1, 1]$ ConvNext with and without patching for $L = 6$. The respective final energies [$E/N = -0.451717(6)$ and $-0.451633(6)$] are shown with dotted lines and the ground-state energy $E_{\text{GS}}/N = -0.4517531$ [13] with dashed lines. Relative errors along with measures of the computational cost are shown in Fig. 7. The annotations refer to the symmetrization stages of the optimization (see Appendix C).

In a model, such as the J_1 - J_2 , where 2×2 patches are larger than the unit cell, enforcing the correct translational symmetry using Eq. (4) requires multiple evaluations of the network, canceling the reduced computational cost. Nevertheless, lower energies are obtained with patching, likely due to the sublattice-dependent bias it provides, so proves to be a good design choice. In the following, we always use an embedding that patches the input.

B. Convolutional kernel size and encoder depth

We now turn to the encoder, first investigating how the convolutional kernel size k and the encoder depth n_b in $(n_b, k, 2)[1, 1]$ networks affect the accuracy of the calculation. We benchmark on multiple system sizes $L = 6, 8$, and 10 . For each system size, we consider a kernel size up to $k = L/2$, at which point the convolutional kernel spans the entire patched lattice, so can encode global features in a single encoder block, as in the fViT. Our results are summarized in Fig. 6(a).

Kernel size. For $L = 6$, the difference in energies obtained for $k = 2$ and 3 is comparable to numerical error from Monte Carlo (MC) estimation of the energy ($\approx 5 \times 10^{-6}$). For $L = 8$ and $L = 10$, we observe that the largest kernel sizes ($k = 4$ and 5 , respectively) perform worse than smaller kernel sizes as the networks become harder to optimize. For all system sizes, the ConvNext achieves a comparable accuracy to the fViT.

Additionally, we find that the network depth should be chosen such that its receptive field allows for correlations among all lattice variables. For a small network depth ($n_b = 2$), using a kernel size $k = 2$ limits the receptive field to 3×3 , which does not span the entire lattice and therefore the output amplitude does not depend on long-range correlations of the input variables. Increasing the kernel size to $k = 3$

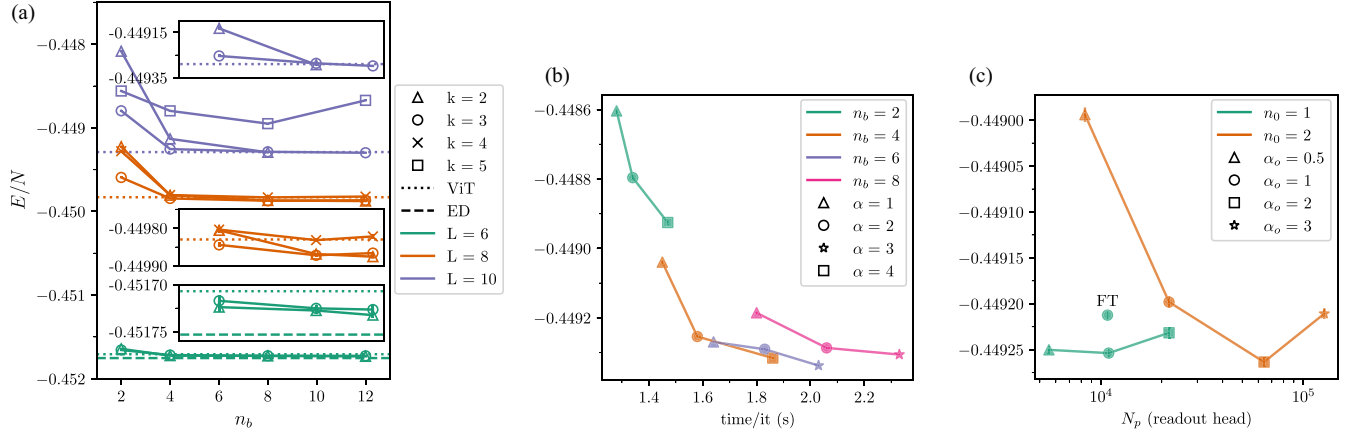


FIG. 6. Variational energies (a) obtained for various L with $(n_b, k, 2)[1, 1]$ networks, the dotted line is the energy obtained by the fViT in Ref. [13], and the insets show the same data on a narrower energy scale for $n_b = 4, 8$, and 12 . (b) For $L = 10$, $(n_b, 3, \alpha)[1, 1]$ networks, the time/iteration (x axis) is measured for the first symmetry stage of the optimization. (c) For $(4, 3, 2)[n_o, \alpha_o]$ networks (modifying the readout block), where N_p is the number of parameters in the readout. The number of parameters excluding the readout is $\sim 1 \times 10^5$. For the FT readout, the sum over patches is the last operation in the network, which allows one to project to $\mathbf{k} \neq 0$ sectors.

expands the receptive field to 5×5 , allowing the encoder to capture all-to-all correlations within the lattice sizes we study. Indeed, for system sizes $L = 8$ and $L = 10$, this increase in kernel size significantly improves the variational energy, highlighting the importance of these global correlations. Furthermore, when we increase the network depth to $n_b = 4$, all kernel sizes considered already have sufficiently large receptive fields to capture global correlations, which reduces the sensitivity of performance to the kernel size. Thus, a kernel size of $k = 3$ consistently achieves optimal performance, and we adopt this choice in subsequent experiments.

Depth. We also observe a significant improvement in the variational energies when increasing the network depth from $n_b = 2$ to 4 , indicating that deeper networks are more effective. Further increases in depth beyond $n_b = 4$ continue to yield improvements, although these gains quickly diminish, becoming marginal when moving from $n_b = 4$ to 8 and even smaller for greater depths.

C. Encoder expansion ratio

Next, we study the effect of modifying the expansion ratio α in the patchwise nonlinear feature transformation g of the encoder [Eq. (8)].

Using $(n_b, 3, \alpha)[1, 1]$ networks, we study the combined effect of modifying the encoder depth n_b and α , as summarized in Fig. 6(b). We again observe a significant improvement in the energy when increasing n_b from 2 to 4 . For $n_b \geq 4$, we find that it can be more efficient to increase the expansion ratio rather than the depth, achieving better energies at comparable computational cost. For $n_b = 6, \alpha = 4$ (not shown), our optimizations become unstable, whereas we deem the cost of an $n_b = 8, \alpha = 4$ network too high for our purposes. Therefore, we take a $(6, 3, 3)$ encoder as the optimal choice.

As discussed further in Appendix B 2, α and n_b play distinct roles in terms of how the hidden representation is constructed from correlators; α controls the number of different correlators computed for each encoder block and n_b the maximum number of recursions through the nonlinear

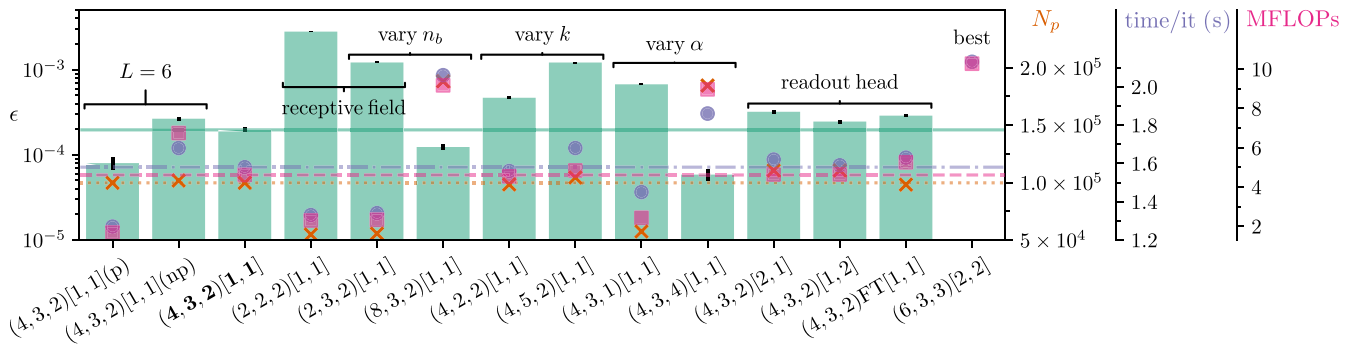


FIG. 7. Summary of the network changes and impact on performance. All results are for $L = 10$, except for the patching (p) and no patching (np) results that were obtained for $L = 6$. ϵ (bars) is measured relative to the ground-state energy ($L = 6$) or our best variational energy ($L = 10$; see Table I). The error bars are the MC errors in the evaluation of the energy after optimization. Also plotted are N_p (crosses), time/iteration of the optimization (circles), and million floating point operations for a single network evaluation (squares). The lines correspond to the values of these quantities for the reference $(4, 3, 2)[1, 1]$ network.

block to construct correlators. Therefore, one should tune both parameters in order to find a performant network design.

We note that these values are close to the optimal choices $n_b = 4$, $\alpha = 4$ found for computer vision tasks [31,32].

D. Readout size

Finally, we investigate the effect of modifying n_o and α_o in the readout feed-forward network, with results summarized in Fig. 6(c). By modifying the number of parameters in the readout, we are in principle modifying the representational power in mapping from the hidden representation constructed by the encoder to complex amplitudes.

We find that for a single layer readout, α_o makes little difference to the variational energy obtained, changing by only $\approx 2 \times 10^{-5}$ across all expansion ratios. On the other hand, for two layers, the final energy changes significantly with the expansion ratio, although requires a relatively large number of parameters to perform on par with a single layer readout. We also observe that the optimization becomes more challenging for $\alpha_o = 3$, resulting in higher variational energies. Nevertheless, improvements in the energy when modifying the readout block are small, suggesting that the power of the neural network to represent the ground state comes largely from the encoder block mapping to the hidden representation.

Despite this, the computational cost of increasing the number of parameters in the readout head is small, since it operates on input $X \in \mathbb{R}^{N_f}$ (see Fig. 7), so offers a computationally inexpensive way to improve variational energies. We therefore choose a [2, 2] readout.

We also tested a readout layer based on the Fourier transform (FT) that could be used to project to $\mathbf{k} \neq 0$ sectors (see Appendix B 4). The data corresponding to this approach, for the $\mathbf{k} = 0$ mode, are included in Figs. 6(c) and 7. We observe that the energy is slightly higher than the standard readout layer based on a dense layer. This is likely due to the fact that in the FT readout, the output is no longer explicitly a function of the sublattice magnetization (see Appendix B 1), which provides a useful bias for learning the sign structure of the ground state. Nevertheless, one does not generally know in which momentum sector the ground state lies, so a thorough variational calculation should check several momentum sectors.

E. System size dependence

In Fig. 6(a), the difference between the ConvNext and fViT variational energies decreases as system size is increased. This could be a result of different optimization procedures (e.g., we use a fixed number of iterations across all system sizes) or a result of differences in the architectures. Isolating the system size scaling of the various architectures would require further numerical experiments where computational cost and optimization protocols are made equal. Note that when running for $L > 10$, we found the optimization to be rather unstable, which could likely be remedied by using transfer learning from smaller system sizes.

TABLE I. Benchmark variational energies for the Shastry-Sutherland and $J_1 - J_2$ models, using deep translationally symmetric NQS. The corresponding number of parameters, N_p , are indicated.

Hamiltonian	Network	E/N	N_p
Shastry-Sutherland ($J_1 = 0.8, J_2 = 1$)	ConvNext	-0.449 342(4)	2.6×10^5
	fViT [13]	-0.449 329(1)	2.7×10^5
$J_1 - J_2$ ($J_1 = 1, J_2 = 0.5$)	ConvNext	-0.497 583(6)	2.6×10^5
	fViT [16]	-0.497 634(1)	2.7×10^5
	ResNet [15]	-0.497 6921(4)	1.07×10^6

F. Summary of design choices

The results from the previous sections are summarized in Fig. 7, comparing the variational energies obtained against various measures of computational cost: number of parameters, N_p , total floating point operations for a single evaluation of the network, and the time per iteration of the optimization.

Summarizing our findings on network design choices:

- (1) The receptive field should allow for all-to-all correlations; therefore, the depth should be $n_b = \text{ceil}(\frac{L_i/P_i - k}{k/2} + 1)$, where L_i is the linear system size and P_i the linear patch size.
- (2) Larger kernel sizes are in general harder to optimize; $k = 3$ works well and is a good compromise.
- (3) There is an optimal combination of depth and expansion ratio that can be used to increase the size of the network.
- (4) A relatively small readout block is sufficient, but the number of parameters can be increased at little cost.

G. Benchmark results

Bringing all of the previous results together, we choose a network of (6, 3, 3) [2, 2] to compute energies for the $L = 10$ Shastry-Sutherland model and the $L = 10$ $J_1 - J_2$ model. Our results are summarized in Table I, along with those from the ResNet in Ref. [15] and fViT [13,16]. Variational energies differ with a relative error of at most $\approx 10^{-4}$. These small differences can likely be accounted for by differing network sizes, number of samples, and optimization procedures, rather than any inherent differences between architectures. A more nuanced comparison would also compare total computational cost of the calculations, across several different Hamiltonians and lattice sizes.

V. DISCUSSION

As we have shown when introducing the ConvNext and in the results we obtain on benchmark problems, successful translationally symmetric NQSs employed so far in the literature operate under the same principles, with different architectures offering more or less efficiency or inductive bias for the problem at hand. For the two benchmark problems we study here, where translationally symmetric NQSs obtain state-of-the-art variational energies, the ground states are thought to be in the same phase (a $\mathbb{Z}_2$ Dirac spin liquid [46]), as well as having similar Hamiltonians, so it is unclear whether these architectures are specifically well suited to tackle these problems or can successfully be applied to other ground states and Hamiltonians. As we have mentioned,

the inductive bias of the embedding is well suited to learn the sign structure of states that (approximately) follow the Marshall sign rule, but for other models and lattice geometries the ground-state sign structure may be far from such a sublattice-dependent rule.

We have focused on translationally symmetric architectures as they can naturally include properties of condensed matter systems, translational symmetry, and locality, as well as allowing one to project to different momentum sectors without incurring a cost that scales as the size of the system. As seen in Fig. 5, including point group symmetries is also crucial for obtaining good variational energies with these networks, which is performed “externally” by quantum-number projection [58,59] (see Appendix A). Group convolutional neural networks offer a more natural way to handle both translational and point group symmetries within the network, yet on, for example, the J_1 – J_2 model are outperformed by translationally symmetric architectures. This may be due to the fact that imposing point group symmetries via quantum-number projection allows one to introduce point group symmetries over the course of the optimization rather than from the outset, reducing the computational cost and allowing one to optimize large networks. More speculatively, it may be that by not enforcing these symmetries from the outset, the larger Hilbert space available earlier in the training helps the network to ultimately find a better solution [54].

Lattice symmetry-agnostic architectures such as recurrent neural networks [60–63] and transformers [30,55] have advantages in principle such as direct sampling or a self-attention mechanism to learn relevant correlators, yet these do not appear to outperform translationally symmetric architectures on frustrated benchmark problems in practice.

More biased approaches may be used when sufficient information about the problem is known, for example, exploiting the emergent gauge symmetries of quantum spin liquids [64], yet may not be directly applicable when the nature of these emergent symmetries is not already known. One can also combine fermionic mean-field ansätze with NQS [42,65], which has yielded accurate variational energies, though computations are expensive. This cost could be partially reduced, and expressivity of the ansatz increased, by combining with translationally symmetric neural networks.

Overall, translationally symmetric architectures appear to provide useful inductive biases to study frustrated ground states, while allowing for the use of large networks and lattice symmetries to improve variational energies.

VI. CONCLUSION AND OUTLOOK

We have applied the ConvNext architecture as an NQS, using VMC for the Shastry-Sutherland model to optimize design choices, providing a blueprint for choosing network hyperparameters and clarifying the role of the various components of translationally symmetric architectures. Using our optimal network on both J_1 – J_2 and Shastry-Sutherland models, we find variational energies that are comparable to the lowest in the literature. We discuss how the ConvNext is related to other deep translationally equivariant NQSSs, providing a unified perspective on the fViT and convolutional networks, such as ResNets.

In future work, the ConvNext architecture could be readily applied to models of 2D frustrated magnets, with and without multisite unit cells, potentially helping to resolve cases where ground-state phase diagrams are debated. Furthermore, the architecture can be relatively straightforwardly generalized to three-dimensional systems, by replacing two-dimensional convolutions with three-dimensional ones. Although translationally equivariant architectures have yielded remarkable results on various benchmark problems, there is not yet a systematic understanding of what kinds of states and Hamiltonians they are suited to, which is important both for understanding their domain of applicability and for understanding how to modify architectures to tackle a broader range of problems.

Beyond ground states, using projection to nonzero momenta, translationally equivariant architectures can be used to perform a limited form of spectroscopy, whereby one can obtain an approximation of the lowest energy state in each symmetry sector of the Hamiltonian using VMC [15,17,42,59,66]. There has also been progress in using NQS to simulate dynamics [12,20,56,67,68] and obtaining spectroscopic quantities such as the dynamical structure factor [22], which can be probed in solid-state experiments using, for example, neutron scattering. Computing such quantities using NQS would be useful for reconciling experimental observations with predictions from theoretical models, providing a path to unambiguous experimental identification of exotic magnetic states of matter, such as quantum spin liquids.

Simulations were performed with NetKet [69,70], and at times parallelized with mpi4JAX [71]. This software is built on top of JAX [72] and Flax [73].

ACKNOWLEDGMENTS

We acknowledge extensive discussions with L. L. Viteritti and R. Rende. The authors acknowledge support by the French Agence Nationale de la Recherche through the NDQM project, Grant No. ANR-23-CE30-0018. This project was provided with computing HPC and storage resources by GENCI at IDRIS thanks to Grants No. 2023-AD010514908 and No. 2024-A0170515698 on the supercomputer Jean Zay’s V100/A100 partition.

DATA AVAILABILITY

Code for the implementation and optimization of the ConvNext is available at [74]. The data that support the findings of this article are openly available [75].

APPENDIX A: VARIATIONAL MONTE CARLO WITH NEURAL-NETWORK QUANTUM STATES

In order to approximate ground states of quantum many-body spin Hamiltonians, we use VMC [11], optimizing our ansatz with the SPRING algorithm [76], based on minSR [15,16].

Consider the quantum state $|\Psi_\theta\rangle$ with N_p parameters $\theta = (\theta_0, \dots, \theta_{N_p-1}) \in \mathbb{R}^{N_p}$. Expanding in the local $\hat{\sigma}^z$ basis,

$\sigma = (\sigma_0^z, \dots, \sigma_{N-1}^z) \in \mathbb{Z}_2^N$, for N degrees of freedom,

$$|\Psi_\theta\rangle = \sum_{\sigma} \langle \sigma | \Psi_\theta \rangle |\sigma\rangle = \sum_{\sigma} \Psi_\theta(\sigma) |\sigma\rangle. \quad (\text{A1})$$

The energy of the variational state

$$E[\Psi_\theta] = \frac{\langle \Psi_\theta | \hat{H} | \Psi_\theta \rangle}{\langle \Psi_\theta | \Psi_\theta \rangle} = \sum_{\sigma} \frac{|\Psi_\theta(\sigma)|^2}{\langle \Psi_\theta | \Psi_\theta \rangle} E_{\text{loc}}(\sigma) \quad (\text{A2})$$

is estimated via Monte Carlo sampling from $p(\sigma) = |\Psi_\theta(\sigma)|^2$,

$$\bar{E}[\Psi_\theta] = \frac{1}{N_s} \sum_{i=1}^{N_s} E_{\text{loc}}(\sigma_i), \quad (\text{A3})$$

where

$$v_i = E_{\text{loc}}(\sigma_i) = \sum_{\sigma'} \langle \sigma_i | \hat{H} | \sigma' \rangle \frac{\Psi_\theta(\sigma')}{\Psi_\theta(\sigma_i)}. \quad (\text{A4})$$

In VMC, we want to find the θ that minimizes the energy, thus finding an approximation of the ground state. We use the SPRING algorithm (without norm constraint) to optimize the parameters, which are updated at each step as $\theta_{k+1} = \theta_k + \Delta\theta_k$, with

$$\Delta\theta_k = \eta [X^T (X X^T + \lambda \mathbb{I})^{-1} \tilde{v} + p \Delta\theta_{k-1}], \quad (\text{A5})$$

where

$$\tilde{v}_i = \text{Re}(v_i) - p(X \Delta\theta_{k-1})_i, \quad (\text{A6})$$

$$\tilde{v}_{i+N_s} = \text{Im}(v_i) - p(X \Delta\theta_{k-1})_{i+N_s}, \quad (\text{A7})$$

$$X_{i,j} = \frac{\text{Re}(O_{i,j} - \bar{O}_i)}{\sqrt{N_s}}, \quad (\text{A8})$$

$$X_{i+N_s,j} = \frac{\text{Im}(O_{i,j} - \bar{O}_i)}{\sqrt{N_s}}, \quad (\text{A9})$$

with $\tilde{v} \in \mathbb{R}^{2N_s}$ and $X \in \mathbb{R}^{2N_s \times N_p}$ that are computed from the Jacobian

$$O_{i,j} = \frac{\partial \ln \Psi_\theta(\sigma_i)}{\partial \theta_j}. \quad (\text{A10})$$

Hyperparameters for the optimization are the learning rate, η , diagonal shift, λ , and momentum, p . For $p = 0$, the algorithm reduces to minSR.

In addition to the translational invariance that we impose within the neural network, we also apply quantum-number projection [58,59,77] for the point group (PG) and spin-parity (P) symmetries, resulting in the symmetrized ansatz

$$\Psi_\theta(\sigma)_{\text{symm}} = \frac{1}{|G|} \sum_{g \in G} \chi_g \Psi_\theta(g^{-1}\sigma), \quad (\text{A11})$$

where G is the group made up of elements g , which is being symmetrized over, and, for simplicity, we restrict ourselves to the invariant irreducible representation, with character $\chi_g = 1, \forall g \in G$. Further details of the optimization procedure are provided in Appendix C.

APPENDIX B: COMPONENTS OF TRANSLATIONALLY SYMMETRIC NEURAL QUANTUM STATES

1. Sublattice bias from patching

Here, we discuss how the patched embedding constructs a hidden representation in terms of sublattice-dependent quantities, which, for example, could allow the network to learn the Marshall sign rule [48,51] on a bipartite lattice.

If we were to pass the output of the embedding directly to the readout block [see Eq. (2)], followed by a sum pooling over the patches, the hidden representation is given by

$$\tilde{\sigma}^\mu = \sum_x W_x^\mu m_x^z, \quad (\text{B1})$$

with the sublattice magnetization

$$m_x^z = \sum_r \sigma_{r,x}^z, \quad (\text{B2})$$

and the coefficients of W determine how much the magnetization contributes to that specific feature. Then passing through the remainder of the readout block, the full wavefunction encoded is

$$\Psi_\theta(\sigma) = g^o \left(\sum_x W_x^\mu m_x^z \right). \quad (\text{B3})$$

The Marshall sign rule specifies that the sign of the amplitude $\Psi(\sigma)$ is given by $(-1)^{n_{\uparrow,x}}$, where $n_{\uparrow,x}$ is the number of $\sigma^z = +1$ spins on the x sublattice, such that

$$n_{\uparrow,x} = \frac{m_x^z + N_x}{2}, \quad (\text{B4})$$

where N_x is the total number of sites on the x th sublattice. Thus, assuming the readout block is sufficiently expressive, the network can learn the sign rule, as a result of the sublattice-dependent bias introduced by the patching. In a network with a skip connection, the output of the embedding block is indeed passed directly to the readout with further corrections from the encoder, which explains why architectures that patch the input do not require the Marshall sign rule as a prior [13].

2. Generating higher-order correlators from convolutional layers

To understand how encoder blocks in translationally equivariant architectures construct hidden representations of the σ_i^z variables, let us consider how standard convolutional layers transform the σ_i^z variables. This will also allow us to understand the roles of the various components of the ConvNext encoder.

Consider a single-feature, feed-forward layer, which has a receptive field of n variables, $\tilde{\sigma}_i^z$. For simplicity, we also consider a system with a single spatial dimension, such that the effect of the layer is

$$y = f \left(\sum_{i=0}^{n-1} w_i \tilde{\sigma}_i^z + b \right) = f(\tilde{x} + b), \quad (\text{B5})$$

where f is a nonlinear activation function. Assuming f is analytic and has a convergent Taylor series, then expanding

around b to order $k_{\max}$ [78],

$$y = \sum_{k=0}^{k_{\max}} \tilde{x}^k \frac{f^{(k)}(b)}{k!} + R_{k_{\max}}(\tilde{x} + b), \quad (\text{B6})$$

where $R_{k_{\max}}$ is a remainder term and the $\tilde{x}^k$ terms are k th-order correlators

$$\tilde{x}^k \propto \tilde{\sigma}_{i_0}^z \tilde{\sigma}_{i_1}^z \cdots \tilde{\sigma}_{i_{k-1}}^z + \cdots. \quad (\text{B7})$$

Since a standard convolutional layer [Eq. (6)] mixes both feature and position degrees of freedom, the kernel weights select the relative contributions of the variables within the correlators, with access to all feature channels. The convolutional kernel width defines the receptive field, setting the maximum distance between variables that can be included. The recursion of convolutional layers allows these correlators to be further combined, while skip connections retain the correlators generated by previous layers.

In the ConvNext [Eq. (9)], the standard convolution is replaced by a depthwise convolution, K , which restricts which input variables each feature channel has access to. The expansion matrix, W , chooses linear combinations of variables within each patch from which the correlators should be constructed, with the expansion ratio, α , controlling the number of different correlators to be generated by the nonlinearity. Subsequently, the compression matrix, V , chooses linear combinations of the generated correlators to be carried forward.

3. Factored multihead attention versus depthwise convolutions

The key conceptual difference in the fViT compared to convolutional networks is that in its embedding block, the data are first reshaped (patched) to $\mathbb{R}^{P \times N_x}$, where $P = P_x P_y$ and N_x is the total number of sites in a patch. That is, the patches are flattened along a single dimension, and the data pass through the network with this shape. In this form, it is not so easy to see how the operations relate to convolutional networks, where one preserves the position dimensions. In the following, we translate the factored multihead attention (FMHA) block (see Fig. 8) to operation on $\tilde{\sigma} \in \mathbb{R}^{P_x \times P_y \times N_f}$ input, in order to make the connection to convolutions transparent. Consider a factored ViT where the number of heads $N_h = N_f$, then the factored multihead block is as follows. First, a dense layer is applied over the features,

$$\tilde{\sigma}_{i,j}^{\nu} = \sum_{\mu=0}^{N_f-1} V^{\nu,\mu} \tilde{\sigma}_{i,j}^{\mu}, \quad (\text{B8})$$

where V is an $N_f \times N_f$ matrix of parameters. Then, in order to preserve translational equivariance the factored attention mechanism [27] is used,

$$\tilde{\sigma}_{i_2,j_2}^{\mu} = \sum_{i_1,j_1} J_{(i_2,j_2),(i_1,j_1)}^{\mu} \tilde{\sigma}_{i_1,j_1}^{\mu}, \quad (\text{B9})$$

where

$$J_{(i_2,j_2),(i_1,j_1)}^{\mu} = J_{(i_2',j_2'),(i_1',j_1')}^{\mu},$$

$$\text{if } \mathbf{x}_{i_2} - \mathbf{x}_{i_1} + \mathbf{y}_{j_2} - \mathbf{y}_{j_1} = \mathbf{x}_{i_2'} - \mathbf{x}_{i_1'} + \mathbf{y}_{j_2'} - \mathbf{y}_{j_1'}, \quad (\text{B10})$$

that is, J is translationally invariant, it has the same value when mapping between input and output patches with the

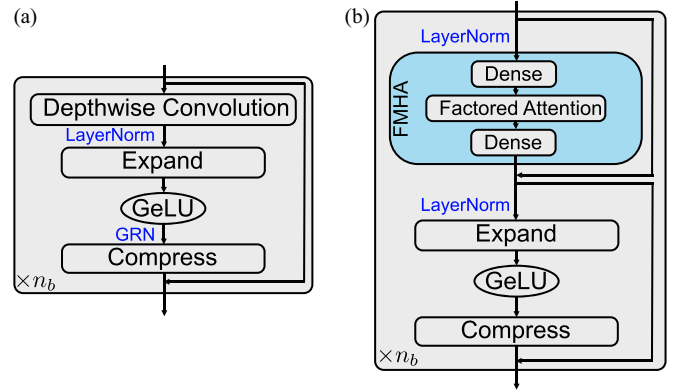


FIG. 8. Comparison of encoder blocks in the ConvNext [panel (a)] and fViT [panel (b)]; see Ref. [79]. In the ConvNext, correlations between different patches are created by a depthwise convolution, whereas in the fViT this is done by FMHA. When the number of heads in the fViT is equivalent to the number of features, the factored attention operation is equivalent to a depthwise convolution.

same relative displacement. This is followed by another dense layer in the form of Eq. (B8).

Due to this translationally invariant structure, this is equivalent to the depthwise convolution,

$$\tilde{\sigma}_{i,j}^{\mu} = \sum_{m,n} K_{m,n}^{\mu} \tilde{\sigma}_{i+m,j+n}^{\mu}, \quad (\text{B11})$$

where the kernel size is $P_x \times P_y$, since a convolutional kernel also has the property that weights connecting input and output positions are the same if the relative displacement between the two is the same.

Decreasing the number of heads $N_h < N_f$ in factored multihead attention (as is done in practice) introduces parameter sharing over the features, with N_h distinct $J_{(i_2,j_2),(i_1,j_1)}$ in Eq. (B9).

Since the ConvNext encoder uses a depthwise convolution followed by expand and compress operations, whereas the fViT encoder uses factored multihead attention also followed by expand and compress operations, we see that these architectures are similar. In our experiments, we find that kernel sizes that span the system, e.g., for $k = 5$ in Fig. 6(a), can be difficult to optimize, which is not the case for the fViT. This may be a result of parameter sharing across features in the attention or the dense layers before and after (see Fig. 8), helping to stabilize larger kernels (we verified that introducing two skip connections per encoder block in the ConvNext was not able to stabilize these large kernels).

4. Projection to momentum sectors

In order to project the wavefunction to $\mathbf{k} \neq 0$ sectors, the sum over patches (see Fig. 3) is moved to the end of the readout head and generically replaced with (see Eq. (4) and [15])

$$\ln \Psi_{\theta}^{\mathbf{k}}(\sigma) = \ln \sum_{i=0}^{N_p-1} e^{-i\mathbf{k} \cdot \mathbf{r}_i} \Phi_{\theta,i}(\sigma), \quad (\text{B12})$$

where $\Phi_{\theta,i}(\sigma)$ is the output of the network for the patch i at position $\mathbf{r}_i$. We find that for the $\mathbf{k} = 0$ sector, this slightly increases the energy obtained, likely due to the fact that the output of the network is not explicitly a function of m_x^z (see Appendix B 1). However, it is generally desirable to obtain results in different $\mathbf{k}$ sectors, as the ground state need not generically be in the $\mathbf{k} = 0$ sector and also gives information on the low-energy spectrum of the model.

APPENDIX C: OPTIMIZATION PROCEDURE

Our optimization protocol makes use of different symmetry stages [16], as shown in Fig. 5, projecting to the invariant representation of all symmetries. We begin the optimization with the bare network, which internally is made to be translationally invariant. We then introduce point group and spin parity externally using Eq. (A11). First, we symmetrize over the rotational elements of the point group, that is C_4 rotations. We then symmetrize over the full point group, adding reflection in the line $x = y$ to the group to be symmetrized over. Finally, we add spin-parity symmetry, under $\hat{P} = \prod_i \hat{S}_i^x$, to the symmetrization. The advantage of this approach over symmetrizing at the outset is two fold; optimization is computationally cheaper the fewer symmetries we enforce externally and is more stable as well.

For each iteration, we use the SPRING algorithm described in Appendix A to update the parameters. We found that using $p = 0.9$ led to faster convergence when comparing different p over the first 1000 iterations, so we use this value. In the numerical experiments for designing the network, we typically perform 2500 iterations in each symmetry stage, with a cosine decaying learning rate [80] from 10^{-2} to 5×10^{-3} and a cosine decaying diagonal shift from 10^{-2} to 10^{-6} in each stage. We use a total of 4092 samples per iteration, with 512 MC chains on each of 8 A100 GPUs (1 sample per chain), each chain performing a single MC sweep of N MC steps.

At the end of the optimization, we freeze the learning rate and diagonal shift, performing 50 additional optimization

steps with more samples per chain, discarding half for thermalization. We use 32 chains per GPU and 8196 total samples. To compute an expectation value for the energy, we then load in the parameters that give the minimum energy over these 50 iterations and use 6.4×10^5 samples on 32 chains on a single GPU (half of the samples are discarded for thermalization).

APPENDIX D: UNPATCHED ARCHITECTURE

In order to assess the impact of patching on network performance, we modify the ConvNext architecture to remove patching while still maintaining translational invariance up to the unit cell (of the Shastry-Sutherland model). To this end, we remove the embedding block and increase the kernel size of the first convolution of the encoder, in order to have the same receptive field in both cases. Therefore, the data passing through the encoder are of form $\mathbb{R}^{L_x \times L_y \times N_f}$, with spatial dimensions corresponding to the sites of the lattice.

In the readout head, we perform a sum over the b^2 basis sites in the unit cells

$$\tilde{\sigma}'_{i,j}{}^\mu = \sum_{n,m=0}^{L/b-1} \tilde{\sigma}_{i+nb,j+mb}^\mu \quad (\text{D1})$$

to take the encoder output to the form $\mathbb{R}^{b \times b \times N_f}$. We then flatten the spatial dimensions and perform a dense layer

$$\tilde{\sigma}'^\mu = \sum_{i=0}^{b^2-1} W_i \tilde{\sigma}_i^\mu, \quad (\text{D2})$$

equivalent to a strided convolution of stride b , taking to the form $\mathbb{R}^{N_f}$, after which the data pass through the remainder of the usual readout block. If the problem at hand is a lattice with a single-site unit cell, the appropriate translational symmetry can be enforced by replacing the above two operations with pooling over sites or using a readout block that can project to $\mathbf{k} \neq 0$ sectors (Appendix B 4).

-
- [1] S. R. White, Density matrix formulation for quantum renormalization groups, *Phys. Rev. Lett.* **69**, 2863 (1992).
 - [2] S. R. White, Density-matrix algorithms for quantum renormalization groups, *Phys. Rev. B* **48**, 10345 (1993).
 - [3] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, *Ann. Phys.* **326**, 96 (2011).
 - [4] E. M. Stoudenmire and S. R. White, Studying two-dimensional systems with the density matrix renormalization group, *Annu. Rev. Condens. Matter Phys.* **3**, 111 (2012).
 - [5] S.-J. Ran, E. Tarrito, C. Peng, X. Chen, L. Tagliacozzo, G. Su, and M. Lewenstein, *Tensor Network Contractions: Methods and Applications to Quantum Many-Body Systems*, Lecture Notes in Physics (Springer International Publishing, Cham, 2020), Vol. 964.
 - [6] N. V. Prokof'ev, B. V. Svistunov, and I. S. Tupitsyn, Exact, complete, and universal continuous-time worldline Monte Carlo approach to the statistics of discrete quantum systems, *J. Exp. Theor. Phys.* **87**, 310 (1998).
 - [7] A. W. Sandvik, Stochastic series expansion method with operator-loop update, *Phys. Rev. B* **59**, R14157 (1999).
 - [8] O. F. Syljuåsen and A. W. Sandvik, Quantum Monte Carlo with directed loops, *Phys. Rev. E* **66**, 046701 (2002).
 - [9] H. G. Evertz, The loop algorithm, *Adv. Phys.* **52**, 1 (2003).
 - [10] K. V. Houcke, E. Kozik, N. Prokof'ev, and B. Svistunov, Diagrammatic Monte Carlo, *Phys. Proc.* **6**, 95 (2010).
 - [11] F. Becca and S. Sorella, *Quantum Monte Carlo Approaches for Correlated Systems*, 1st ed (Cambridge University Press, Cambridge, UK, 2017).
 - [12] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks, *Science* **355**, 602 (2017).
 - [13] L. L. Viteritti, R. Rende, A. Parola, S. Goldt, and F. Becca, Transformer wave function for two dimensional frustrated magnets: Emergence of a spin-liquid phase in the Shastry-Sutherland model, *Phys. Rev. B* **111**, 134411 (2025).

- [14] T. Duric, J. H. Chung, B. Yang, and P. Sengupta, Spin-1/2 kagome Heisenberg antiferromagnet: Machine learning discovery of the spinon pair density wave ground state, *Phys. Rev. X* **15**, 011047 (2025).
- [15] A. Chen, Empowering deep neural quantum states through efficient optimization, *Nat. Phys.* **20**, 1476 (2024).
- [16] 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, *Commun. Phys.* **7**, 260 (2024).
- [17] C. Roth, A. Szabó, and A. H. MacDonald, High-accuracy variational Monte Carlo for frustrated magnets with deep neural networks, *Phys. Rev. B* **108**, 054410 (2023).
- [18] M. Li, J. Chen, Q. Xiao, F. Wang, Q. Jiang, X. Zhao, R. Lin, H. An, X. Liang, and L. He, Bridging the gap between deep learning and frustrated quantum spin system for extreme-scale simulations on new generation of sunway supercomputer, *IEEE Trans. Parallel Distrib. Syst.* **33**, 2846 (2022).
- [19] X. Liang, M. Li, Q. Xiao, J. Chen, C. Yang, H. An, and L. He, Deep learning representations for quantum many-body systems on heterogeneous hardware, *Mach. Learn.: Sci. Technol.* **4**, 015035 (2023).
- [20] L. Gravina, V. Savona, and F. Vicentini, Neural projected quantum dynamics: A systematic study, *Quantum* **9**, 1803 (2025).
- [21] L. Mauron, Z. Denis, J. Nys, and G. Carleo, Predicting topological entanglement entropy in a Rydberg analog simulator, *Nat. Phys.* **21**, 1332 (2025).
- [22] T. Mendes-Santos, M. Schmitt, and M. Heyl, Highly resolved spectral functions of two-dimensional systems with neural quantum states, *Phys. Rev. Lett.* **131**, 046501 (2023).
- [23] S. Sorella, Green function Monte Carlo with stochastic reconfiguration, *Phys. Rev. Lett.* **80**, 4558 (1998).
- [24] S.-I. Amari, Natural gradient works efficiently in learning, *Neural Comput.* **10**, 251 (1998).
- [25] J. Martens, New insights and perspectives on the natural gradient method, *J. Mach. Learn. Res.* **21**, 1 (2020).
- [26] S. Dash, L. Gravina, F. Vicentini, M. Ferrero, and A. Georges, Efficiency of neural quantum states in light of the quantum geometric tensor, *Commun. Phys.* **8**, 92 (2025).
- [27] R. Rende, F. Gerace, A. Laio, and S. Goldt, Mapping of attention mechanisms to a generalized Potts model, *Phys. Rev. Res.* **6**, 023057 (2024).
- [28] L. L. Viteritti, R. Rende, and F. Becca, Transformer variational wave functions for frustrated quantum spin systems, *Phys. Rev. Lett.* **130**, 236401 (2023).
- [29] R. Rende and L. L. Viteritti, Are queries and keys always relevant? A case study on transformer wave functions, *Mach. Learn.: Sci. Technol.* **6**, 010501 (2024).
- [30] A. Chen, V. D. Naik, and M. Heyl, Convolutional transformer wave functions, [arXiv:2503.10462](https://arxiv.org/abs/2503.10462).
- [31] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, A ConvNet for the 2020s, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, New Orleans, LA, 2022) pp. 11966–11976.
- [32] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I. S. Kweon, and S. Xie, ConvNeXt V2: Co-designing and scaling ConvNets with masked autoencoders, in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Vancouver, BC, Canada, 2023), pp. 16133–16142.
- [33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, Attention is all you need, in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2017), Vol. 30.
- [34] S. Islam, H. Elmekki, A. Elsebai, J. Bentahar, N. Drawel, G. Rjoub, and W. Pedrycz, A comprehensive survey on applications of transformers for deep learning tasks, *Expert Syst. Appl.* **241**, 122666 (2024).
- [35] B. Sriram Shastry and B. Sutherland, Exact ground state of a quantum mechanical antiferromagnet, *Physica B+C* **108**, 1069 (1981).
- [36] Introduction to frustrated magnetism, in *Springer Series in Solid-State Sciences*, edited by C. Lacroix, P. Mendels, and F. Mila (Springer, Heidelberg, 2011), Vol. 164.
- [37] J. Yang, A. W. Sandvik, and L. Wang, Quantum criticality and spin liquid phase in the Shastry-Sutherland model, *Phys. Rev. B* **105**, L060409 (2022).
- [38] L. Wang, Y. Zhang, and A. W. Sandvik, Quantum spin liquid phase in the Shastry-Sutherland model detected by an improved level spectroscopic method, *Chin. Phys. Lett.* **39**, 077502 (2022).
- [39] A. Keleş and E. Zhao, Rise and fall of plaquette order in the Shastry-Sutherland magnet revealed by pseudofermion functional renormalization group, *Phys. Rev. B* **105**, L041115 (2022).
- [40] L. Wang and A. W. Sandvik, Critical level crossings and gapless spin liquid in the square-lattice spin-1/2 $J_1 - J_2$ Heisenberg antiferromagnet, *Phys. Rev. Lett.* **121**, 107202 (2018).
- [41] F. Ferrari and F. Becca, Gapless spin liquid and valence-bond solid in the $J_1 - J_2$ Heisenberg model on the square lattice: Insights from singlet and triplet excitations, *Phys. Rev. B* **102**, 014417 (2020).
- [42] 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 (2021).
- [43] W.-Y. Liu, S.-S. Gong, Y.-B. Li, D. Poilblanc, W.-Q. Chen, and Z.-C. Gu, Gapless quantum spin liquid and global phase diagram of the spin-1/2 $J_1 - J_2$ square antiferromagnetic Heisenberg model, *Sci. Bull.* **67**, 1034 (2022).
- [44] W.-J. Hu, F. Becca, A. Parola, and S. Sorella, Direct evidence for a gapless Z_2 spin liquid by frustrating Néel antiferromagnetism, *Phys. Rev. B* **88**, 060402 (2013).
- [45] F. Ferrari and F. Becca, Spectral signatures of fractionalization in the frustrated Heisenberg model on the square lattice, *Phys. Rev. B* **98**, 100405 (2018).
- [46] A. Maity, F. Ferrari, J. Y. Lee, J. Potten, T. Müller, R. Thomale, R. Samajdar, and Y. Iqbal, Evidence for a Z_2 Dirac spin liquid in the generalized Shastry-Sutherland model, [arXiv:2501.00096](https://arxiv.org/abs/2501.00096).
- [47] 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 (2019).
- [48] A. Szabó and C. Castelnovo, Neural network wave functions and the sign problem, *Phys. Rev. Res.* **2**, 033075 (2020).
- [49] Z. Denis, A. Sinibaldi, and G. Carleo, Comment on “can neural quantum states learn volume-law ground states?” *Phys. Rev. Lett.* **134**, 079701 (2025).
- [50] A. Sinibaldi, C. Giuliani, G. Carleo, and F. Vicentini, Unbiasing time-dependent variational Monte Carlo by projected quantum evolution, *Quantum* **7**, 1131 (2023).

- [51] W. Marshall, Antiferromagnetism, *Proc. R. Soc. Lond. Ser. A* **232**, 48 (1955).
- [52] The translation of the input gives a corresponding translation of the output.
- [53] The encoder is not equivariant, and the whole network not translationally symmetric, under subpatch translations. For example, considering the square lattice with 2×2 patches of Fig. 5, the network would not transform correctly under translations by ONE site along the x or y axes.
- [54] A. G. Wilson, Position: Deep learning is not so Mysterious or different, in *Forty-second International Conference on Machine Learning Position Paper Track* (ICML, 2025).
- [55] K. Sprague and S. Czischek, Variational Monte Carlo with large patched transformers, *Commun. Phys.* **7**, 90 (2024).
- [56] M. Schmitt and M. Heyl, Quantum many-body dynamics in two dimensions with artificial neural networks, *Phys. Rev. Lett.* **125**, 100503 (2020).
- [57] R. Rende, S. Goldt, F. Becca, and L. L. Viteritti, Fine-tuning neural network quantum states, *Phys. Rev. Res.* **6**, 043280 (2024).
- [58] T. Mizusaki and M. Imada, Quantum-number projection in the path-integral renormalization group method, *Phys. Rev. B* **69**, 125110 (2004).
- [59] Y. Nomura, Helping restricted Boltzmann machines with quantum-state representation by restoring symmetry, *J. Phys.: Condens. Matter* **33**, 174003 (2021).
- [60] M. Hibat-Allah, M. Ganahl, L. E. Hayward, R. G. Melko, and J. Carrasquilla, Recurrent neural network wave functions, *Phys. Rev. Res.* **2**, 023358 (2020).
- [61] M. Hibat-Allah, R. G. Melko, and J. Carrasquilla, Supplementing recurrent neural network wave functions with symmetry and annealing to improve accuracy, in *35th Annual Conference on Neural Information Processing Systems* (NeurIPS, Vancouver, BC, Canada, 2021).
- [62] M. Hibat-Allah, R. G. Melko, and J. Carrasquilla, Investigating topological order using recurrent neural networks, *Phys. Rev. B* **108**, 075152 (2023).
- [63] 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 anti-ferromagnets: The square lattice, *arXiv:2502.17144*.
- [64] D. S. Kufel, J. Kemp, D. Vu, S. M. Linsel, C. R. Laumann, and N. Y. Yao, Approximately-symmetric neural networks for quantum spin liquids, *Phys. Rev. Lett.* **135**, 056702 (2025).
- [65] 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 (2017).
- [66] S. Morita, R. Kaneko, and M. Imada, Quantum spin liquid in spin $1/2 J_1 - J_2$ Heisenberg model on square lattice: Many-variable variational Monte Carlo study combined with quantum-number projections, *J. Phys. Soc. Jpn.* **84**, 024720 (2015).
- [67] A. Sinibaldi, D. Hendry, F. Vicentini, and G. Carleo, Time-dependent neural Galerkin method for quantum dynamics, *arXiv:2412.11778*.
- [68] A. V. de Walle, M. Schmitt, and A. Bohrdt, Many-body dynamics with explicitly time-dependent neural quantum states, *Mach. Learn.: Sci. Technol.* **6**, 045011 (2025).
- [69] F. Vicentini, D. Hofmann, A. Szabó, D. Wu, C. Roth, C. Giuliani, G. Pescia, J. Nys, V. Vargas-Calderón, N. Astrakhantsev, and G. Carleo, NetKet 3: Machine learning toolbox for many-body quantum systems, *SciPost Phys. Codebases* **7** (2022).
- [70] G. Carleo, K. Choo, D. Hofmann, J. E. T. Smith, T. Westerhout, F. Alet, E. J. Davis, S. Efthymiou, I. Glasser, S.-H. Lin, M. Mauri, G. Mazzola, C. B. Mendl, E. van Nieuwenburg, O. O'Reilly, H. Théveniaut, G. Torlai, F. Vicentini, and A. Wietek, NetKet: A machine learning toolkit for many-body quantum systems, *SoftwareX* **10**, 100311 (2019).
- [71] D. Häfner and F. Vicentini, mpi4jax: Zero-copy MPI communication of JAX arrays, *J. Open Source Softw.* **65**, 3419 (2021).
- [72] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, JAX: Composable transformations of Python+NumPy programs, GitHub (2018), <https://github.com/google/jax-ml/jax>.
- [73] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, Flax: A neural network library and ecosystem for JAX, GitHub (2024), <http://github.com/google/flax>.
- [74] Convnext neural quantum states and lattice symmetries, GitHub (2025), <https://github.com/NeuralQXLab/convnextnnqs>.
- [75] R. P. Nutakki, A. Shokry, and F. Vicentini, Data for “Design principles of deep translationally symmetric neural quantum states for frustrated magnets”, Zenodo (2025), <https://doi.org/10.5281/zenodo.17157844>.
- [76] G. Goldshlager, N. Abrahamsen, and L. Lin, A Kaczmarz-inspired approach to accelerate the optimization of neural network wavefunctions, *J. Comput. Phys.* **516**, 113351 (2024).
- [77] M. Reh, M. Schmitt, and M. Gärtner, Optimizing design choices for neural quantum states, *Phys. Rev. B* **107**, 195115 (2023).
- [78] Z. Denis and G. Carleo, Accurate neural quantum states for interacting lattice bosons, *Quantum* **9**, 1772 (2025).
- [79] Vision transformer wave function (2025), <https://netket.readthedocs.io/en/stable/tutorials/ViT-wave-function.html>.
- [80] I. Loshchilov and F. Hutter, SGDR: Stochastic gradient descent with warm restarts, in *International Conference on Learning Representations* (ICLR, 2017).