Neural network architectures based on the classical XY model

Nikita Stroev[1] and Natalia G. Berloff[1,2,*]

[1]*Skolkovo Institute of Science and Technology, Bolshoy Boulevard 30, bld.1, Moscow 121205, Russian Federation*

[2]*Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge CB3 0WA, United Kingdom*

(Received 31 March 2021; revised 12 October 2021; accepted 1 November 2021; published 30 November 2021)

Classical XY model is a lattice model of statistical mechanics notable for its universality in the rich hierarchy of the optical, laser, and condensed matter systems. We show how to build complex structures for machine learning based on the XY model's nonlinear blocks. The final target is to reproduce the deep learning architectures, which can perform complicated tasks usually attributed to such architectures: speech recognition, visual processing, or other complex classification types with high quality. We developed a robust and transparent approach for the construction of such models, which has universal applicability (i.e., does not strongly connect to any particular physical system) and allows many possible extensions, while at the same time preserving the simplicity of the methodology.

DOI: 10.1103/PhysRevB.104.205435

I. INTRODUCTION

The growth of modern computers suffers from many limitations, among which is Moore's law [1,2], increased energy consumption connected with the growth of the performance, intrinsic issues arising from the architecture design like the so-called von Neumann bottleneck [3], etc. The latter refers to the internal data transfer process built according to the von Neumann architecture and usually denotes the idea of computer system throughput limitation due to the characteristic of bandwidth for data coming in and out of the processor. The von Neumann bottleneck problem was addressed in various ways [4,5] including managing multiple processes in parallel, different memory bus design, or even considering the conceptual "non–von Neumann" systems [6,7].

The inspirations for such information-processing architectures comes from different sources, artificial or naturally occurring. This concept is usually referred to as neuromorphic computation [8,9], which competes with state-of-the-art performance in many specialized tasks like speech and image recognition, machine translation, and others. Additionally, it has the property of distributed memory input and computation compared to the linear and conventional computational paradigm [10]. The artificial neural networks (NNs) demonstrated many advantages compared to classical computing [11] while offering a solution to the von Neumann bottleneck.

This alternative approach is to encode a particular computational task into the NN coefficients and minimize the specific cost function instead of performing the predefined sequence of computations (e.g., logical gates). The nodes' final configuration, obtained through a nonlinear evolution of the system, corresponds to the initial problem's solution. One of the familiar and famous models of the NNs is the Hopfield network [12], which serves as a content-addressable or associative memory system with binary nodes. Its mathematical description is similar to the random, frustrated Ising spin glass [13], for which finding the ground state is known to be an NP-hard problem [14]. Such correspondence reveals an interplay between the condensed matter and computer science areas while offering various physical realizations in optics, lasers, and beyond.

The Ising Hamiltonian can be treated as a degenerate XY Hamiltonian. The classical XY model, sometimes also called classical rotator or simply O(2) model, has been a subject of the extensive theoretical and numerical research and appears to be one of the basic blocks in the rich hierarchy of the condensed matter systems [15–19]. It has an essential property of universality, which allows one to describe a set of mathematical models that share a single scale-invariant limit under renormalization group flow with the consequent similar behavior [20]. Thus the model's properties are not influenced by whether the system is quantum or classical, continuous or on a lattice, and its microscopic details. These types of models are usually used to describe the properties of magnetic materials, superconducting, or superfluid states of matter [20]. Moreover, the classical XY model can be artificially reproduced through intrinsically different condensed matter systems. Among these systems are superconducting Josephson junction arrays [21,22] and other superconducting systems [23,24], ultracold bosonic quantum gases in optical lattices [25,26], and optical, laser, and exciton-polariton systems [27,28].

Reproducing the Ising, XY, or NN architecture model allows one to utilize a particular physical system's benefits. By constructing the specific hardware, we can additionally solve other computing problems while decreasing the energy consumption or volume inefficiency or even increasing the processing speed. The NN implementation has been previously discussed in the optical settings [29,30], all-optical realization of reservoir computing [31], or using silicon photonic integrated circuits [32] and other coupled classical

*Correspondence address: n.g.berloff@damtp.cam.ac.uk

oscillator systems [33]. Exciton-polariton condensates are one of such systems (together with many others, for instance, plasmon polaritons [34,35]) that lead to a variety of applications such as a two-fluid polariton switch [36], all-optical exciton-polariton router [37], polariton condensate transistor switch [38], and polariton transistor [39], which can also be realized at room temperatures [40], spin switches [41], and the XY-Ising simulators [27,28,42]. The coupling between different condensates can be engineered with high precision, while the phases of the condensate wave functions arrange themselves to minimize the XY Hamiltonian.

There has been recent progress in developing the direct hardware for the artificial NNs, emphasizing optical and polaritonic devices. Among them, there is the concept of reservoir computing in the Ginzburg-Landau lattices, applicable to the description of a broad class of systems, including exciton-polariton lattices [43], and the consequent experimental realization that demonstrated efficient recognition and high signal processing rates [44]. The implementation of the backpropagation mechanism using neurons in an all-optical training of optical NNs with the pump-probe passive elements resulted in the state-of-the-art performance [45]. The near-term experimental platform for realizing an associative memory based on spinful bosons coupled to a degenerate multimode optical cavity enjoyed advanced performance due to the system tuning possibilities [46].

In our paper we show how to perform the NN computation by establishing the correspondence between the XY model's nonlinear clusters that minimize the corresponding XY Hamiltonian and the basic mathematical operations, therefore, reproducing neural network transformations. Moreover, we solve an additional set of problems using the properties offered explicitly by the exciton-polariton systems. The ultrafast characteristic timescale for condensation (of the order of picoseconds) allows one to decrease computational time significantly. The intrinsic parallelism, based on working with multiple operational nodes, can enhance the previous benefits compared to the FPGA or tensor processing units. The nonlinear behavior can be used to perform the approximate nonlinear operations. This task is particularly computationally inefficient on the conventional computing platforms where a local approximation in evaluation is commonly used. There are many trivial ways of utilizing the correspondence between the Ising model and the Hopfield NN in terms of the discrete analog variables. In contrast, we use continuous degrees of freedom of the XY model. By moving to continuous variables, we address the volume efficiency problem since converting the deep NNs to the quadratic binary logic architecture meets significant overhead in the number of variables while increasing the range of the coupling coefficients. The former may be out of reach for the actual physical platforms, while the latter increases the computational errors.

With the motivation to transfer the deep learning (DL) architecture into the optical or condensed matter platform, we show how to build complex structures based on the XY model's nonlinear blocks that naturally behave in a nonlinear way in the process of reaching the equilibrium. The corresponding spin Hamiltonian is quite general because it can be engineered with many condensed matter systems as we previously discussed.

The key results we present are as follows.

(i) We show how to realize the basic numerical operations using some small-size XY networks, which minimize the XY Hamiltonian clusters' energy.

(ii) We obtain the complete set of operations sufficient to realize different combinations of the mathematical operations and to map the deep NN architectures into XY models.

(iii) We demonstrate that this approach can be applied to the existing special-purpose hardware capable of reproducing the XY models.

(iv) We present the general methodology based on simple nonlinear systems, which can be extended to other spin Hamiltonians, i.e., with different interactions or models with additional degrees of freedom and involving quantum effects.

This paper is organized as follows. Section II is devoted to describing our models' basic blocks and introducing notations used. Section III contains the demonstration of our approach's effectiveness in approximating simple functions. Section IV extends this approach to the small NN architectures. Section V considers the extensions to the deep architectures with a particular emphasis on various nonlinear functions' implementations. Section V A is dedicated to the particular exciton-polariton condensed matter system as a potential platform for implementing our approach. Conclusions and future directions are given in Sec. VI. The Appendix provides some technical details about the NN parameters, approximations, and corresponding optimization tasks.

II. BASIC XY EQUILIBRIUM BLOCKS

This section is devoted to the description of the basic blocks of the XY NN with the potential of its upscaling to the DL architecture. DL is usually defined as a part of the ML methods based on the artificial NN with representation learning and proven to be effective in many scientific domains [11,47–49] ranging from applied fields such as chemistry and material science to fundamental ones like particle physics and cosmology [50].

The DL is typically referred to as a black box [51,52] due to the lack of understanding behind its exceptional performance. When it comes to DL's application, one usually asks questions about adapting the DL architectures to new problems, how to interpret the results, and how to quantify the outcome errors reliably. Leaving these open problems behind, we formulate a more applied task. To build DL architectures, we want to transfer the pretrained parameters into a realization of a nonlinear computation. One of the mathematical core ideas in machine learning (ML) architectures is the ability to build hyperplanes on each neuron output. The union of such hyperplanes allows one to approximate the input data efficiently and adjust it to the output in the case of supervised learning (for example, in the classification tasks [53]). This procedure can be paraphrased as the feature engineering that before the modern DL approaches and the available computational resources was performed in a manual way [54]. Building hardware that performs the hyperplane transformation with a specific type of a nonlinear activation function with a given precision allows one to separate input data points and present building block operations for more complex tasks. The hierarchical structures

constructed with such blocks lead to even more complex architectures capable of performing more sophisticated tasks.

Decomposing the nonlinear expressions common in the ML, such as $\tanh(w_0x_0 + w_1x_1 + \cdots + w_nx_n + b)$, produces a set of mathematical operations, which we need to approximate with our system. These are nonlinear operation tanh, which is conventionally called an activation function, the multiplication of the input variables x_i by the constant (after training) coefficients w_i (also called weights of a NN), and the summation operation (with the additional constant b, called bias).

The activation function is an essential aspect of the deep NNs that brings nonlinearity into the learning process. The nonlinearity allows modern NNs to create complex mappings between the inputs and outputs that are vital for learning and approximating complex data with high dimensionality. Moreover, the nonlinear activation functions allow for backpropagation due to the smooth derivative of these functions. They normalize each neuron's output, allowing one to stack multiple layers of neurons to create a deep NN. The functional form of the nonlinear function is zero centered with the saturation effect, which mimics the binary decision process. We will see that approximating this operation is straightforward with the XY networks.

First, we introduce the list of simple blocks corresponding to the set of operations that are necessary for realization of the nonlinear activation function, which can be obtained by manipulating the small clusters of spins with underlying U(1) symmetry. These clusters minimize the XY Hamiltonian:

$$H = \sum_{i=1}^{N}\sum_{j=1}^{N} J_{ij}\cos(\theta_i - \theta_j), \tag{1}$$

where i and j go over N elements in the system and J_{ij} is the interaction strength between ith and jth spins represented by the classical phases $\theta_i \in [-\pi, \pi]$. If we take several spins as inputs θ_i in such a system and consider the others as outputs θ_k, then we can treat the whole system as a nonlinear function which returns $\arg\min_{\theta_k} H(\{\theta_i\}, \{\theta_k\})$ values due to the system equilibration into the steady state. In some cases the ground state is unique; other cases can produce multiple equilibrium states.

Above, we referred to the functional form Eq. (1) as the XY Hamiltonian; however, to be precise, this expression has the meaning of the configuration functional and does not define the dynamics of the system. Later on, we will assume that the couplings in Eq. (1) are directional and consider the systems that minimize this configurational functional.

It is useful to consider an analytical solution to such a kind of task, describing the function with one output and several input variables. We consider the system with N spins: θ_i, $i = 1, \ldots, N-1$ are input spins and θ_N is the output spin coupled with the input spins by the coupling coefficients $J_i \equiv J_{iN}$. The system configuration functional can be written as $H = \sum_i^{N-1} J_i \cos(\theta_i - \theta_N)$. By expanding H as $\sum_i^{N-1} J_i \cos(\theta_i - \theta_N) = \sum_i^{N-1} J_i \cos\theta_i \cos\theta_N + \sum_i^{N-1} J_i \sin\theta_i \sin\theta_N$ we can solve for the minimizer θ_N:

$$\begin{aligned}\theta_N &\equiv F((\theta_1|J_1), (\theta_2|J_2), \ldots, (\theta_{N-1}|J_{N-1})) \\ &= \arg(A + iB) - \pi = \arg C - \pi,\end{aligned} \tag{2}$$

where $A = \sum_i^{N-1} J_i \cos\theta_i$, $B = \sum_i^{N-1} J_i \sin\theta_i$, and $C = \sum_i^{N-1} J_i e^{i\theta_i} = A + iB$. Equation (2) has simple geometrical interpretation, identifying θ_N as the unit vector antiparallel to the weighted sum of unit vectors $\theta_1, \ldots, \theta_{N-1}$. The phase shift π disappears when the configurational functional is given with the minus sign, so that the direction given by the θ_N angle will be parallel to the weighted sum of unit vectors. We present several basic blocks in Fig. 1 and the outcomes of the functions' responses in Fig. 2.

We will use the notation introduced in Eq. (2) to describe both the activation function and the graph cluster of spins below. We will use the recurrent notation where the output of the first block serves as the input to the next one, for example, $F(F((\theta_1|J_1), (\theta_2|J_2))|J_3), (\theta_4|J_4))$.

To describe the iterative implementation of many (k) identical blocks, where the input is defined in terms of the output of the previous same block, we rewrite the recurrent formula for one argument as $(F_1 \circ F_1 \circ \cdots \circ F_1)(\theta_{\text{in}}) \equiv F_1(F_1(\ldots F_1(\theta_{\text{in}}))) \equiv F_1^k(\theta_{\text{in}})$,

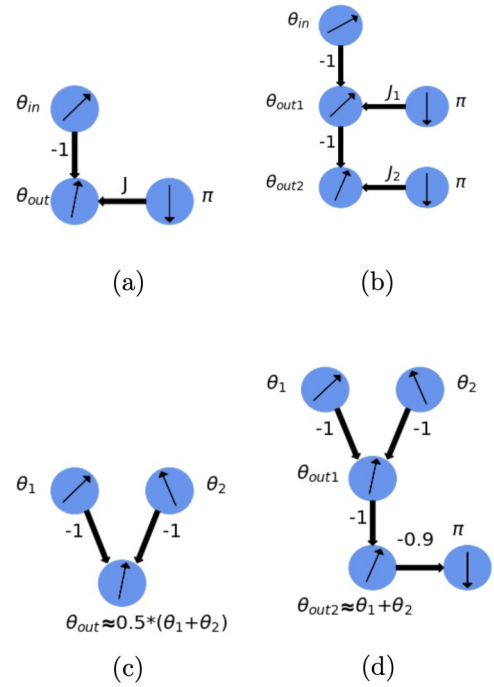

FIG. 1. Several examples of basic blocks and their combinations used in the XY NN architectures. (a) The block performing the function $F((\theta_{\text{in}}| - 1), (\pi|J))$ with one input spin and one reference/control spin with imposed π value, all coupled with the output by the ferromagnetic -1 and J ($J = 1$ is used on the picture). Depending on J we can realize the operations that approximate the multiplication by the constant k so that $\theta_{\text{out}} \approx 1.5\tanh(4\theta_{\text{in}}) + 0.5\theta_{\text{in}}$ with $J = -0.9$. (b) Two blocks representing $F(F((\theta_1| - 1), (\pi|J_1))| - 1), (\pi|J_2))$. (c) The block $F((\theta_1| - 1), (\theta_2| - 1))$ for the half sum of two variables θ_1 and θ_2. (d) Two blocks performing the function $F(F((\theta_1| - 1), (\theta_2| - 1))| - 1), (\pi| - 0.9))$. Some of the response functions for these blocks are presented in Fig. 2.

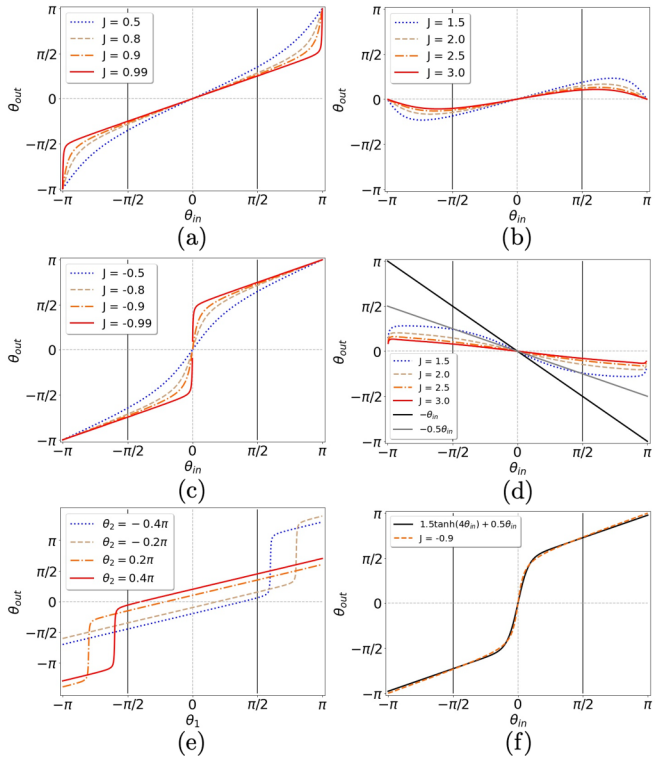

FIG. 2. Several examples of input-output relations for the basic blocks and their combinations used in the XY NN architectures. (a) The parametrized family of $F((\theta_{\text{in}}|-1),(\pi|J))$ functions, that corresponds to the basic blocks. Depending on the J coupling strength parameter we can realize the multiplication by arbitrary k. (b) The graphs of $F((\theta_{\text{in}}|-1),(\pi|J))$ functions for various values of J illustrating the the multiplication by small values of k. (c) The graphs of $F((\theta_{\text{in}}|-1),(\pi|J))$ functions for smaller values of $J<0$. (d) The graphs of $F_3^3(F_2(F_1^2(\theta_{\text{in}})|J))$ functions, where $F_1(\theta_{\text{in}}) = F((\theta_{\text{in}}|-1),(\pi|0.9))$, $F_2(\theta_{\text{in}}|J) = F((\theta_{\text{in}}|1),(\pi|J))$, and $F_3(\theta_{\text{in}}) = F((\theta_{\text{in}}|-1),(\pi|-0.2))$, showing different negative outputs. (e) The graphs of $F((\theta_1|-1),(\theta_2|-1))$, implementing the block shown on Fig. 1(c), which approximates the half sum of input variables. (d) The graphs of $1.5\tanh(4\theta_{\text{in}}) + 0.5\theta_{\text{in}}$ and $F((\theta_{\text{in}}|-1),(\pi|-0.9))$.

where $F_1(\theta_{\text{in}}) = F((\theta_{\text{in}}|J_1),(\theta_2|J_2))$ is a certain block with the predefined parameters. We separate all possible blocks of spins into several groups and consider them below in more detail.

The phases θ_i are in $[-\pi,\pi]$; however, for an efficient approximation of the operations (summation, multiplication, and nonlinearity) we need to limit the domain to $[-\pi/2,\pi/2]$, which we refer to as the *working domain*. Additionally, we need to make sure that the values of the *working spins* (which are not fixed and are influenced by the system input, thus serving as analog variables) are located within the limits of the working domain. This will be implemented below. We made such a limitation due to several reasons. First, such mapping is monotonic. Extending $[-\pi/2,\pi/2]$ by a small value will ruin the monotonic properties of the available functions. Secondly, the set of operations expressed through the family of parametric functions Eq. (2) are defined on $[-\pi/2,\pi/2]$ domain without any additional adjustments; see Fig. 2.

Next, we consider the implementation of the elementary operations.

Multiplication by the constant value $k>0$. Connecting the input spin with the output spin by the "ferromagnetic" coupling $J=-1$ will lead to the input spin's replication. In this way, we can transmit the spin value from one block to another. Changing the value of the output spin can be achieved in many ways. The addition of another spin with a different value and coupling it to the output spin with a constant coupling J is one such possibility (for example, with imposed π value, which we will refer to as a *reference/control spin*). If J is in [0,1) (relative to -1 coupling between θ_{in} and θ_{out}), then the reference spin influences the output spin value with the effective "repulsion" and thus, depending on the relative coupling strength, decreases the output spin value [see Figs. 2(a) and 2(b) and the corresponding cluster configuration on Fig. 1(a)]. The resulting relation between the input and output spin values can be a good approximation to the multiplication by certain values lying in the [0,1] range. The block corresponding to the implementation of $F((\theta_{\text{in}}|-1),(\pi|1))$ has a peculiarity in the case of $\theta_{\text{in}}=0$, which allows the output to take any value due to the degeneracy of the ground state. To overcome this degeneracy, we choose $J=0.99$. For $J>0$, the linearized Eq. (2) leads to the following expression for $F((\theta_{\text{in}}|-1),(\pi|J))$:

$$\arg C-\pi = \arctan\frac{|B|}{|A|}+\pi-\pi = \arctan\frac{\sin\theta_{\text{in}}}{\cos\theta_{\text{in}}+J} = \frac{\theta_{\text{in}}}{J+1}-\frac{\theta_{\text{in}}^3(J^2-J)}{6(J+1)^3}+O\left(\theta_{\text{in}}^5\right), \tag{3}$$

which explains the relationship $J(k)\approx\frac{1}{k}-1$ for the approximation procedure and can be seen in Fig. 2 and Fig. 8. For $k>1$ we can use a ferromagnetic coupling $J<0$ [see Fig. 2(c) and the corresponding cluster configuration on Fig. 1(a)]. However, the positive values of J are more reliable for the implementation since the output functions have small approximation errors (see the Appendix for the exact values of this error and further clarification). We can replace the multiplication by a large factor by the multiplications by several smaller factors to reduce the final accumulated error. We can guarantee the uniqueness of the output since the clusters are small, and the output is defined by Eq. (2), which gives the unique solution.

Nonlinear activation function. The function $F((\theta_{\text{in}}|-1),(\pi|-0.9))$ is similar to the hyperbolic tanh function [see Fig. 2(c) and the Appendix for the exact difference]. There are two ways of using such a transformation as an activation function.

(1) We can use the similarity between the values of the $F((\theta_{\text{in}}|-1),(\pi|-0.9))$ and the function $1.5\tanh(4\theta_{\text{in}})+0.5\theta_{\text{in}}$ [see Fig. 2(f)]. We can train the NN initially with the $1.5\tanh(4\theta_{\text{in}})+0.5\theta_{\text{in}}$ function so that, in the final transfer, it will not be necessary to adjust the spin system to approximate the given function. Such a replacement of the activation function usually does not change the network's overall functionality as compared with the conventional one.

(2) We can use the similarity with the approximate hyperbolic tangent function within the XY spin cluster. In other words, to execute $\tanh(\theta_{\text{in}})$, we have to perform

$(F((0.25\theta_{\text{in}}|-1),(\pi|-0.9))-0.5\theta_{\text{in}})/1.5$ function using the spin block operations. This option will be used below.

Multiplication by the constant value $k=-1$. The main difficulty of this operation is in finding the set of parameters for the spin block where $\frac{\partial F}{\partial\theta_{\text{in}}}<0$. $F((\theta_{\text{in}}|1),(\pi|J))$ is one example of such a block. To perform the multiplication by $k=-1$, we need to embed the whole working domain into the region where the presented inequality is valid and return these values with the multiplication by $k>0$ factor. One final realization can be represented as $F_3^3(F_2(F_1^2(\theta_{\text{in}})))$ function, where $F_1(\theta_{\text{in}})=F((\theta_{\text{in}}|-1),(\pi|0.9))$, $F_2(\theta_{\text{in}})=F((\theta_{\text{in}}|1),(\pi|J))$, and $F_3(\theta_{\text{in}})=F((\theta_{\text{in}}|-1),(\pi|-0.2))$.

Summation. The function $F((\theta_1|-1),(\theta_2|-1))$ gives a good approximation to the half sum $(\theta_1+\theta_2)/2$. This block is presented in Fig. 1(c) and the cross sections of the surface defined by the function of two variables $F((\theta_1|-1),(\theta_2|-1))$ are plotted in Fig. 2(e). The plots show that the spin system realizes the half sum of two spin values with a minimum discrepancy compared to the target function on a working domain. One can multiply the final result by two using previously described multiplication to achieve an ordinary summation. In general, such a type of summation can be extended to multiple spins $N>2$, in a similar way by connecting them to the output spin, with the final value of $(\theta_1+\cdots+\theta_N)/N$. The linearized Eq. (2) leads to the following expression for $F((\theta_1|-1),(\theta_2|-1))$:

$$\arg C-\pi=\arctan\frac{\sin\theta_1+\sin\theta_2}{\cos\theta_1+\cos\theta_2}=\frac{\theta_1+\theta_2}{2}+O\left(\theta_1^{n_1}\theta_2^{n_2}\right), \tag{4}$$

where $n_1+n_2=5$ (with integer n_1,n_2) is the order of approximation is the order of approximation.

Summarizing, we presented a method of approximating the set of mathematical operations, necessary for performing the $\tanh(w_0x_0+w_1x_1+\cdots+w_nx_n+b)$ function, using the XY blocks described by Eq. (2). The output spin value of each block is formed when a global equilibrium is reached in the physical system with the speed that depends on a particular system and its parameters. We kept the model's universality, which allows one to implement this approach using various XY systems. We did not use any assumptions about the nature of the classical spins, their couplings, or the manipulation techniques; however, the forward propagation of information requires directional couplings. As the blocks corresponding to elementary operations are added one after another, the new output spins and new reference spins should not change the values of the output spins from the previous block. The directional couplings that affect the output spins of the next block but not the output spins of the previous block satisfy this requirement. Many systems can achieve directional couplings. For instance, in optical systems, the couplings are constructed by redirecting the light with either free-space optics or optical fibers to a spatial light modulator (SLM). At the SLM, the signal from each node is multiplexed and redirected to other nodes with the desired directional coupling strengths [55].

III. ONE-DIMENSIONAL FUNCTION APPROXIMATIONS

This section illustrates the efficiency of the proposed approximation method on one-dimensional functions of intermediate complexity by considering two examples of mathematical functions and their decomposition into the basis of nonlinear operations.

For illustration, we choose two nontrivial functions (one is monotonic and another is nonmonotonic). Extending these ideas to more complex functions in higher dimensions is straightforward. In the next section, we will apply this method to two-dimensional data toy problems.

We consider two functions

$$\mathcal{F}_1=0.125F_t(1.2x)+0.125F_t(0.5(x+1.4)) \tag{5}$$

and

$$\mathcal{F}_2=0.125F_t(0.5(x-1.2))-0.03125F_t(0.5(x+1.2)), \tag{6}$$

where $F_t(x)=1.5\tanh(4x)$. Note that arbitrary functions can be obtained using a linear superposition of scaled and translated basic functions $F_t(x)$.

The comparison of the XY blocks' approximations and the target functions are given in Figs. 3 and 4, demonstrating a good agreement in the working domain. We also plot the explicit structures of the corresponding XY spin clusters showing a rather small overhead on the number of spins used per operation.

IV. NEURAL NETWORKS BENCHMARKS

In this section, we test the XY NN architectures and check their effectiveness using typical benchmarks. For simple architectures, the classification of predefined data points perfectly suits this goal. We consider standard two-dimensional data sets, which are conventionally referred to as "moons" and "circles" and can be generated with *Scikit-learn* tools [56]. An additional useful property of such tasks is that they are easy for manual feature engineering.

First, we train a simple NN, the architecture of which consists of two neurons' input layer, one hidden layer with three neurons for each feature, and tanh activation function. The output layer consists of two neurons, which are transformed with SoftMax function. The corresponding weights for both cases are given in the Appendix section. The final performance demonstrates perfect accuracy in both data sets. Figures 5 and 6 show the decision boundaries with the given pretrained architectures and the landscape of one of the final neuron visualizations together with the data points.

Since we are focused on transferring the described architectures into the XY spin cluster system, we consider the basic architecture adjustment using the example of one feature. Suppose we have the expression $\tanh(w_1x+w_2y+b)$. To repeat the chosen strategy (2) of approximating *the nonlinear activation function*, we rewrite the coefficients w_1, w_2, and b as, say, $w_i\to(N/K)[(K/N)w_i]$, where K is a parameter chosen to increase the accuracy of each computation. We approximate the square brackets' operations using the building blocks from Sec. II. The factor $N=3$ will be canceled by the value $1/N$ during the summation of N spins, while the factor $K=4$ in the denominator will be taken into account during the operation of the function $F((\theta_{\text{in}}|-1),(\pi|-0.9))$ that approximates $\tanh(4\theta_{\text{in}})$. The resulting procedure achieves good performance depicted in Fig. 5, while the details are provided further in the Appendix section.

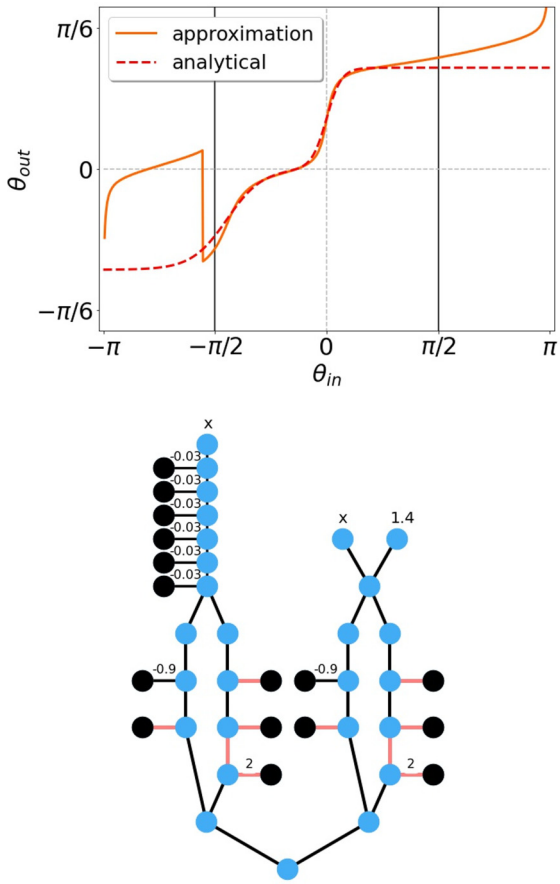

FIG. 3. Top: demonstration of the approximation quality obtained by using the nonlinear XY spin clusters. The monotonic analytical function (red dashed line) is given by Eq. (5) and the orange line is the approximation. Bottom: the graph structure representing the basic mathematical operation in Eq. (5) given by the blocks discussed in Sec. II. The input variables are mapped into the top spins, after which the cluster is equilibrated before performing the next operation. The blue empty nodes are working spins that change according to the variables at a higher block. The black nodes with the fixed π value are reference/control spins. The black edges without the notation have the default fixed relative strength -1; otherwise, they have the specified coupling coefficient explicitly written above. The red color of the edge represents the positive relative coupling strength 1 unless specified otherwise. The bottom spin gives the value of the coded function.

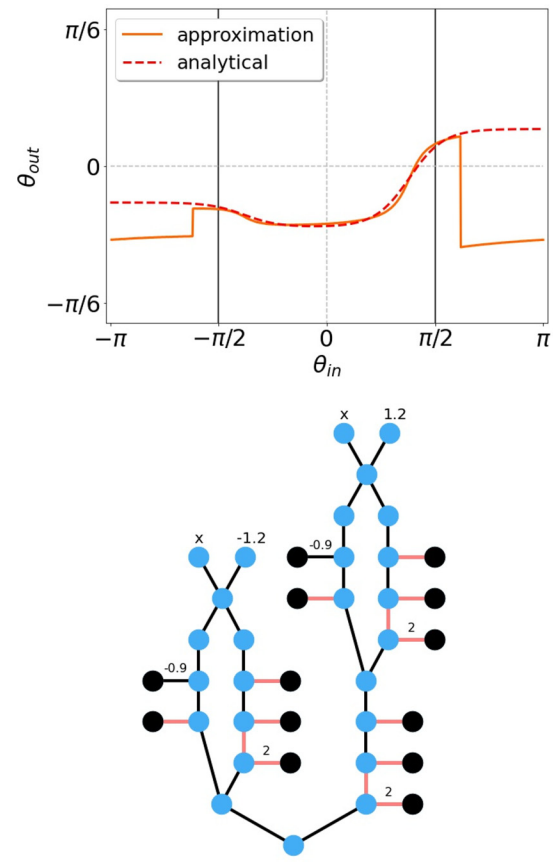

FIG. 4. Top: demonstration of the approximation quality, obtained by using the nonlinear XY spin clusters. The analytical nonmonotonic function (red dashed line) is given by Eq. (6), the orange line is the approximation, and the blue line is the linear identity relation. Bottom: the graph structure, representing the essential mathematical operation in Eq. (6), given by the blocks, discussed in Sec. II. The notation used for describing the graph parameters is the same as in Fig. 3.

The final SoftMax function in the original NN serves as the comparison function for two features to achieve the final decision boundary's smooth landscape. We can omit this function and replace it with a simpler expression $x - y$. To achieve the binary decision boundaries, one can exploit the block performing $F((\theta_{\text{in}}| - 1), (\pi| - 0.9))$ several times to place the final spin value either close to $\pi/2$ or $-\pi/2$. In this way, we adjusted architecture that performs the same functions as the described simple NN on a toy model. The final decision boundary of the XY NN approximation can be seen in Fig. 5, which is very close to the boundary of the standard trained NN architecture.

The case of the "moons" data set is a bit different. While the smooth functions are easy to approximate with the nonlinear XY blocks, it is quite complicated to reproduce "sharp" patterns with the high value of the function derivative. For this purpose, we adjust the NN coefficients to achieve good decision boundaries. The difference between the NN and its approximation and consequent results is shown in Fig. 6. Adjusted parameters of NN are given in the Appendix section. Figures 5 and 6 show the XY blocks of the spin architectures. The presented methodology allows us to upscale the XY blocks for even more complicated ML tasks.

V. TRANSFERRING DEEP LEARNING ARCHITECTURE

Deep NNs are surprisingly efficient at solving practical tasks [11,47,57]. The widely accepted opinion is that the key to this efficiency lies in their depth [58–60]. We can transfer the architecture's depth into our XY NN model without any significant loss of accuracy.

So far, we showed how to transfer the predefined architecture into the XY model. This section discusses the transition of more complex deep architectures, which are considered conventional across different ML fields without much emphasis on the details. To extend the method, we choose two

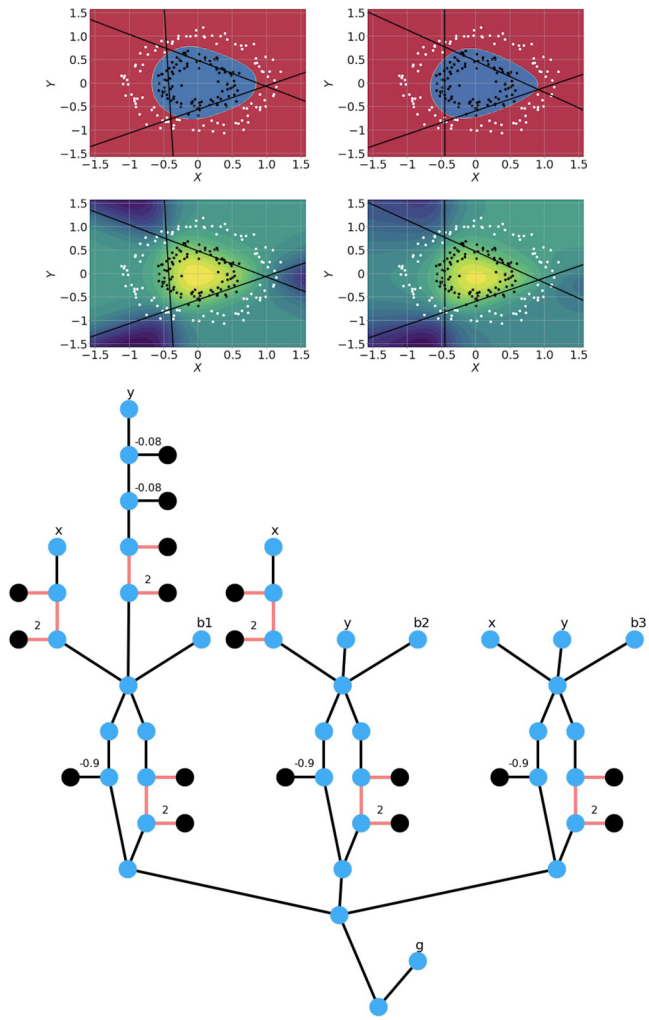

FIG. 5. Top row: decision boundaries of the simple (2,3,2) NN on the left and approximated ones for the XY NN on the right on the toy 2D circle data set. Black lines represent the bounds for automatically found features in the middle layer of classical NN and the approximated features are shown on the right picture. Middle row: the isosurface for the one particular chosen feature for typical NN and its corresponding matched XY NN last variable isosurface. The parameters of both NN and XY NN architectures can be found in the Appendix. Bottom: the corresponding graph structure.

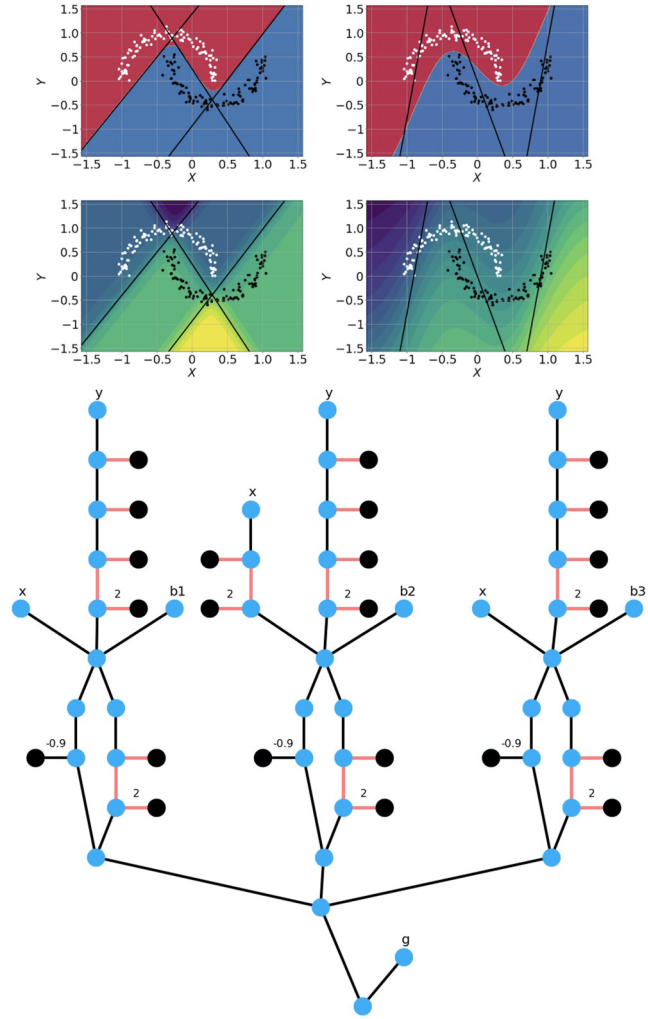

FIG. 6. Decision boundaries of the simple (2,3,2) NN on the left and approximated ones for the XY NN on the right on the toy 2D moons data set. Black lines represent the bounds for automatically found features in the middle layer of classical NN; the approximated features adjusted for this specific task are located in the right picture. Middle row: the isosurface for the one particular chosen feature for typical NN and its corresponding matched XY NN last variable isosurface. The parameters of both NN and XY NN architectures can be found in the Appendix. We can state that the XY model can give a good approximation of the classical NN architectures with not ideally smooth representations, but requires small adjustments of their parameters, due to difficulties with representing sharp geometric figures (which in general can be multidimensional). Bottom: the corresponding graph structure.

conventional image recognition task models (the architecture details are given in the Appendix section).

The focus of the architecture adjustment will be on operations, which were not previously discussed. We list some of these such operations: Conv2D, ReLU activation function, MaxPool2D, and SoftMax [47,61,62]. The Conv2d is a simple convolution operation and does not present any significant difficulty, since it factorizes into the operations previously discussed. ReLU activation function can be replaced with its rough approximation $\approx e^{\theta_{\text{in}}}/3$, similar to the so-called shifted exponential linear unit ELU [63]. Using its similarity with the analytical expression $1.5(1+\tanh 0.8(\theta_{\text{in}}-1.5))$ allows one to obtain the following set of explicit transformations: $z = F(F((\theta_{\text{in}}|-1),(-1.5|-1))|-1),(\pi|1.5))$ and the summation of the three terms 1.5, while $F((z|-1),(\pi|-0.9))$ and $-0.5z$ give a good approximation for ReLU. We will denote the overall function as $G(\theta_{\text{in}})$.

MaxPool2D relies on $\max(x,y)$ realization. We use the following relationship in further description: $\max(x,y) = [\max(0,x-y)+y+\max(0,y-x)+x]/2$. Therefore, it is convenient to use two similar architectures of spin value transmission, which are symmetrical with respect to variables x and y. The first one consists of $G(x-y)$ operation, described previously, and summation with the y variable, while the second architecture interchanges x and y and consists of $G(y-x)$

and $+x$ operations. Summing the results of each architecture $2F(G(x-y)+y|-1), (G(y-x)-y|-1))$ will give us the required value of $\max(x, y)$.

The SoftMax $e^{z_i}/\sum_{j=1}^{K} e^{z_j}$ is usually used as the final layer in the NN architectures (see the Appendix) in order to compare the corresponding values of e^{z_i} among each other. One can avoid the complicated approximation by comparing the arguments directly.

A. Exciton-polariton setting

In this subsection, we discuss the implementation of the proposed technique using a system of exciton-polariton condensates. As we discussed in the Introduction, it is possible to reproduce the XY configurational functional with a variety of systems from superconductors and superfluids to optical and laser systems. Here we present one possible realization using exciton polaritons [27,28], which are hybrid light-matter quasiparticles that are formed in the strong coupling regime in semiconductor microcavities [64]. The condensates of exciton polaritons can be formed. They are described by the density and phase degrees of freedom, serving as the analog variables for the loss minimization. The exciton-polariton condensate is a gain-dissipative, nonequilibrium system due to the finite quasiparticle lifetimes. Polaritons decay emitting photons. Such emission carries all necessary information of the corresponding system state and can serve as the readout mechanism. Redirecting photons from one condensate to another using an SLM allows one to couple the condensates in a lattice directionally [55].

The system of condensates maximizes the total occupation of condensates by arranging their relative phases so as to minimize the losses [27]. The exciton-polariton platform allows one to manipulate several parameters, such as coupling strengths between the condensates, which makes it possible to set up the particular mathematical operation or to fix the phase of the condensate (through the combination of resonant and nonresonant pumping; see [65]), and thus create reference/control spins. Each input spin in the whole system can be controlled via two fixed couplings with the two reference/control spins of different values; see, for example, one of the blocks from Fig. 2. Fixing the coupling coefficients between spatially located elements is required to perform the necessary operation and establish the XY network. It can be further upscaled to approximate a particular ML architecture, with the final output spin being the readout target.

One additional note is that the same system can be exploited differently by introducing the spin self-locking mechanism. It consists of saving the spin value in the system without coupling connections with the external elements. This mechanism can be achieved by coupling the local output with another element(s) with high negative coupling strength and decoupling it from the previous units. The self-locking allows one to save the local output and use it for the consequent operations without significant overhead on elements from the previously established operations. We demonstrate the difference in Fig. 7. The presented alternative, requiring a self-locking mechanism, operates with fewer spins by performing each action at the same cluster. Hence it is volume efficient, which is noticeable in the scaling of elements per operation.

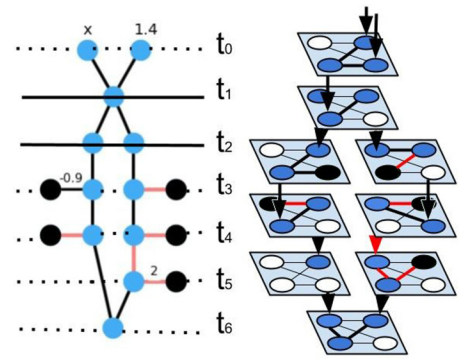

FIG. 7. Left: graph representation of the $F_t(0.5(x+1.4))$ operation defined in the text. The final spin values are established through six time units and the measure equals the local characteristic equilibration time of the XY spin cluster. Right: the alternative architecture, which performs each operation at the same cluster while saving the local output spin value and transferring it the next time through the self-locking mechanism. The color correspondence is the same as in XY graphs: blue nodes are the working spins and green are reference/control spins with π values.

Figure 7 shows the same operation with 16 spins (without external nodes) and 8 total spins with the self-locking mechanism. The advantage is even more pronounced for larger spin networks.

VI. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper, we introduced a robust and transparent approach for approximating the standard feedforward NN architectures utilizing the set of nonlinear functions arising from the classical XY spin behavior and discussed the possible extensions to other architectures. The number of additional spins required per operation scales linearly. The best-case scenario has two spin elements per multiplication and nonlinear operation (not considering the multiplication by a negative factor), making the general framework quite practical. Some operation approximations used in this work allow additional improvements such as the reduction of cumulative errors between the initial architecture and its nonlinear approximation (see the Appendix).

The entire spectrum of the benefits dramatically depends on a particular type of optical or condensed matter platform. The presented approach has universal applicability and, at the same time, has a certain degree of flexibility. It preserves basic blocks' simplicity, and overall structure works with the intermediate complexity architectures capable of solving toy model data scientific tasks. The upscaling to reproduce DL architectures was discussed.

Our work's side product is a correspondence that allows us to adjust every hardware aiming at minimization of the XY configurational functional into the ML task. It becomes possible to build DL hardware from scratch and readjust the existing special-purpose hardware to solve the XY model's minimization task.

Finally, we would like to mention the alternative of using a hybrid architecture. Instead of transferring the operations used in the conventional NN model, we can introduce the nonlinear

TABLE I. NN (2,3,2) parameters used for the toy data set "circles."

w_{11}	w_{12}	b_1	
−1.3465	−2.4191	1.1582	
−3.5880	−0.1474	−1.5228	
−1.3565	2.6776	1.5239	
w_{21}	w_{22}	w_{23}	b_2
−17.8809	16.4797	−18.2794	23.6661
17.3684	−17.0227	18.4634	−23.3256

TABLE II. XY NN (2,3,1) parameters used to approximate the standard NN for the toy data set "circles."

w_{11}	w_{12}	b_1	
−0.5	−0.75	0.35	
1.0	0.0	0.45	
−0.5	1.0	0.6	
w_{21}	w_{22}	w_{23}	b_2
1.0	1.0	1.0	−0.31

blocks coming from the presented XY model into the working functional given by a particular ML library. For example, we can change the activation function into the one that comes from the system operation and therefore is easily reproducible by that system. This would result in the architecture and the transfer processes that do not require additional adjustments from the hardware perspective.

The question about the implementation of the backpropagation mechanism, i.e., computing of the gradient of the NN weights in a supervised manner, usually with the gradient descent method, is still under consideration because we limited the scope of our work with the transfer of the predefined, pretrained architecture.

Several possible extensions of our work are possible, such as extending to k-local Hamiltonians or to other models with additional degrees of freedom and controls, simplifying different mathematical operations and approximations of the basic functions using many-body clusters in a particular model, and increasing the presented approach's functionality (for instance, adding the backpropagation mechanism).

APPENDIX

Here we present the parameters of the NNs used in this work and details of the training and approximations for the XY graphs. We also present two DL architectures mentioned in the main text emphasizing their nonlinear operations. Finally, we discuss the calculations and estimations of the approximation quality.

For training simple classical (2,3,2) feedforward NN architectures on "moons" and "circle" data sets we used the Pytorch library [66] and Adam optimizer [67] with batch size 32 and learning rate 0.01 value. The expected learning procedure passed without the problems on data sets consisting of 200 points generated with the small noise of magnitude 0.1. The final performance gives perfect expected accuracy in both cases.

For the circles data set, the NN parameters are presented in Table I.

The first row of mathematical approximations gives us similar coefficients. We can rewrite in the same manner the NN parameters with minor adjustments for demonstrative purposes (see the main text for the detailed analysis of possible assumptions and the improvements such as the multiplication by the scaling coefficients).

The presented approximation architecture given in Table II was adjusted for better representation of one final feature, which is general enough to mark the decision boundaries for this particular task, while getting rid of the unnecessary parameters. Another approximation stage leads us to the final architecture, which is shown in Fig. 5. Let f_i denote the ith feature and $R(x) = F((F_1(x)| - 1), (F_3(F_2(x))| - 1))$, where $F_1(x) = F((x| - 1), (\pi| - 0.9))$, $F_2(x) = F((x| - 1), (\pi|1))$, and $F_3(x) = F((x|1), (\pi|2))$ represent the approximation of the activation function with the reduced accuracy; then $x_{11} = F((F((x_1| - 1), (\pi|0.99))|1), (\pi|2))$, $x_{12} = F((F((x_{11}| - 1), (\pi|0.99))|1), (\pi|2))$, $y_{11} = F((F((y_1| - 1), (\pi|0.99))|1), (\pi|2))$, $y_{12} = F((F((y_{11}| - 1), (\pi| - 0.08))| - 1), (\pi| - 0.08))$, $f_1 = F((x_{12}| - 1), (y_{12}| - 1), (b_1 = 0.35| - 1))$, $f_{11} = R(f_1)$; $x_{21} = F((F((x_2| - 1), (\pi|0.99))|1), (\pi|2))$, $f_2 = F((x_{21}| - 1), (y_2| - 1), (b_2 = 0.6| - 1))$, $f_{22} = R(f_2)$; $f_3 = F((x_3| - 1), (y_3|0), (b_3 = 0.45| - 1))$, $f_{33} = R(f_3)$; $G_0 = F((f_{11}| - 1), (f_{22}| - 1), (f_{33}| - 1))$, $G = F((G_0| - 1), (g = -0.1033| - 1))$.

This structure is represented in Fig. 5.

The NN parameters for the moons data set are presented in Table III.

The first row of the approximations' parameters is given in Table IV.

The presented architecture was adjusted for better representation of one final feature. For the case of moons, the additional adjustment has been added since the presented XY architecture has lower expressivity for the case of sharp boundaries. Another approximation stage leads us to the final architecture, which can be found in Fig. 6: $y_{11} = F((F((y_1| - 1), (\pi|0.99))| - 1), (\pi|0.99))$, $y_{12} = F((F((y_{11}| - 1), (\pi|0.99))|1), (\pi|2))$, $f_1 = F((x_1| - 1), (y_{12}| - 1), (b_1 = -0.95| - 1))$, $f_{11} = R(f_1)$; $y_{21} = F((F((y_2| - 1), (\pi|0.99))| - 1), (\pi|0.99))$, $y_{22} = F((F((y_{21}| - 1), (\pi|0.99))|1), (\pi|2))$, $f_2 = F((x_1| - 1), (y_{22}| - 1), (b_1 = 0.9| - 1))$, $f_{22} = R(f_2)$; $x_{31} = F((F((x_3| - 1), (\pi|0.99))|1), (\pi|2))$, $y_{31} = F((F((y_3| - 1), (\pi|0.99))| - 1), (\pi|0.99))$, $y_{32} = F((F((y_{31}| - 1), (\pi|0.99))|1), (\pi|2))$, $f_3 = F((x_{31}| - 1), (y_{32}| - 1), (b_1 =$

TABLE III. NN (2,3,2) parameters used for the toy data set "moons."

w_{11}	w_{12}	b_1	
6.2888	−3.2930	−3.0992	
−3.5880	−4.2940	5.9965	
−6.1958	−2.7684	0.6882	
w_{21}	w_{22}	w_{23}	b_2
−6.1143	−6.8860	−6.8151	−0.0621
6.6825	6.2848	6.8381	−0.0325

TABLE IV. XY NN (2,3,1) parameters used to approximate the standard NN for the toy data set "moons."

w_{11}	w_{12}	b_1	
1.0	−0.125	−0.9	
1.0	−0.125	0.9	
−0.5	−0.125	0	
w_{21}	w_{22}	w_{23}	b_2
1.0	1.0	1.0	0.065

$0|-1))$, $f_{33}=R(f_3)$; $G_0=F((f_{11}|-1),(f_{22}|-1),(f_{33}|-1))$, $G=F((G_0|-1),(g=0.0216|-1))$.

The presented structure follows the graph structure given in Fig. 6.

The presented DL architectures are defined with the Pytorch library's help in the following Table V.

In Table V, we present the details of two DL architectures that were discussed in the main text, emphasizing their nonlinear operations.

Finally, we show that the initial task of approximating a particular set of mathematical operations by the parametrized family of nonlinear functions can be done more rigorously with a potential for the accumulated error estimation through the layers of NN.

The discrepancy between the target function and its approximation can be estimated with the $L^1([-\pi/2,\pi/2])$ norm on the working domain:

$$L^1=\int_{-\pi/2}^{\pi/2}\Bigg|-\operatorname{sgn}B(x,\{J_i\})\times\left(\frac{\pi}{2}+\arcsin\frac{A(x,\{J_i\})}{\sqrt{A(x,\{J_i\})^2+B(x,\{J_i\})^2}}\right)-f(x)_{\text{target}}\Bigg|dx. \tag{A1}$$

TABLE V. Examples of the simple DL architectures used to represent nonlinear/unique functions. (a) NN for simple 10 classes digit recognition. (b) NN for CIFAR10 data set classification.

(a) NN layer
5 × 5 Conv2D(3,6)
2 × 2 MaxPool2D
5 × 5 Conv2D(6,16)
Linear (400,120)
Linear (120,84)
Linear (84,10)
(b) NN layer
5 × 5 Conv2D(1,10)
2 × 2 MaxPool2d
ReLU
Dropout(0.5)
5 × 5 Conv2D(10,20)
2 × 2 MaxPool2D
ReLU
Flatten
Linear (320,50)
ReLU
Linear (50,10)
SoftMax

Starting with the multiplication operation, one can calculate Eq. (A1) with $f(x,k)_{\text{target}}=kx$ and obtain the expression [depending on the (J,k) parameters] for one block of spins. Further minimization of Eq. (A1) leads to the expression for $J(k)$. Evaluating Eq. (A1) analytically can be done in a simpler way by replacing the expression involving the arcsin function with the one with arccot, so that initial integral (in terms of the argument of the complex parameter $C=\sum_i^{N-1}J_ie^{i\theta_i}$) contains the following expression for one input and one control/reference spin:

$$I=\int_{-\pi/2}^{\pi/2}\operatorname{arccot}\frac{B((x|-1),(\pi|J))}{A((x|-1),(\pi|J))}dx=\int_{-\pi/2}^{\pi/2}\operatorname{arccot}\frac{\sin(x)}{J+\cos(x)}dx. \tag{A2}$$

We evaluate this to

$$\begin{aligned}I=x\operatorname{arccot}\frac{\sin(x)}{J+\cos(x)}+\frac{1}{4}\Bigg(&x^2+2i\ \operatorname{sgn}(J^2-1)\\&\times\left\{i\left[\operatorname{Li}_2\left(\frac{D(1-E)}{1+E}\right)+\operatorname{Li}_2\left(\frac{D^*(1-E)}{1+E}\right)\right]\right.\\&+2x\ \operatorname{arctanh}(E^{-1})-G\ \operatorname{arctanh}(E)\\&+[G-2i\ \operatorname{arctanh}(E)]\ln\frac{2J(1+E)}{I_1[\tan(x/2)-i]}\\&+[G+2i\operatorname{arctanh}(E)]\ln\frac{2J(1+E)}{I_2[\tan(x/2)+i]}\\&+\ln H\,e^{-ix/2}[2i\ \operatorname{arctanh}(E)-2i\ \operatorname{arctanh}(E^{-1})+G]\\&\left.+\ln H\,e^{ix/2}[2i\ \operatorname{arctanh}(E^{-1})-2i\ \operatorname{arctanh}(E)+G]\right\}\Bigg)\Bigg|_{-\pi/2}^{\pi/2},\end{aligned} \tag{A3}$$

with variables $D(J)=(J^2+1+|J^2-1|)/2J$, $E(x,J)=\frac{i|J^2-1|}{(J+1)^2}\tan x/2$, $C(J)=\arccos(-\frac{J^2+1}{2J})$, $H(J)=\frac{i|J^2-1|}{2\sqrt{J}\sqrt{J^2+2J\cos x+1}}$, $I_1(J)=2i(J-1)$, if $J^2>1$; $2iJ(1-J)$, otherwise, $I_2(J)=2iJ(J-1)$, if $J^2>1$; $2J(J-1)$, otherwise, and arccot, arctanh, and arccos denoting the inverse for tangent, hyperbolic tangent, and cosine functions, respectively, with $\operatorname{Li}_s(x)=\sum_{k=1}^{\infty}x^k/k^s$ being the polylogarithm function and $*$ denoting the complex conjugate operation. One can simplify the given formula by the contraction of the complex pair terms:

$$\begin{aligned}I=x\operatorname{arccot}\frac{\sin(x)}{J+\cos(x)}+\frac{1}{4}\Bigg\{&x^2+2i\ \operatorname{sgn}(J^2-1)\\&\times\left[i\left(\sum_{k=1}^{\infty}\frac{2\cos(k\phi)}{k^2}\right)+2G\ln H+(2x-G)\operatorname{arctanh}(E)\right.\\&+G\ln\frac{J(1+E)^2}{E^2(J+1)^2-(J-1)^2}-2i\ \operatorname{sgn}(J^2-1)\\&\left.\times\operatorname{arctanh}(E)\ln\frac{J[E(J+1)-(1-J)]}{E(J+1)-(J-1)}\right]\Bigg\}\Bigg|_{-\pi/2}^{\pi/2},\end{aligned} \tag{A4}$$

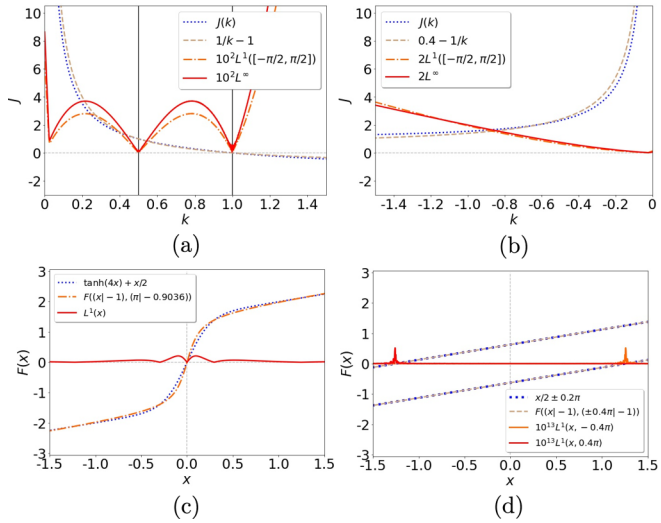

FIG. 8. Top: (a) function $J(k)$ that minimizes Eq. (A1) with $F((|x|-1),(\pi|J))$, $f(x)_{\text{target}} = kx$ and a positive factor k (blue dotted line), its approximation (light brown dashed line), and the fitted formula $\approx 1/k - 1$. The supporting plots depict $10^2 L^1$ (dash-dotted orange line) and $10^2 L^\infty$ (red) for each value of k. Vertical black lines denote the points with the minimal accumulated error. (b) Function $J(k)$ that minimizes Eq. (A1) with $F((x|1),(\pi|J))$, $f(x)_{\text{target}} = kx$ and a negative factor k (blue dotted line), its approximation (light brown dashed line), and the fitted formula $\approx 0.4 - 1/k$. The supporting plots depict $2L^1$ (dash-dotted orange line) and $2L^\infty$ (orange) for each value of k. Bottom: (c) function $\tanh(4x) + x/2$ (blue dotted line) and its approximation with the function $F((|x| - 1),(\pi| - 0.9036))$ (dash-dotted orange line) with the optimized parameter J. The supporting red plot represents the error at each point on the x axis. (d) The half sums for x and two variables $y = -0.2\pi, 0.2\pi$ (blue dotted lines), which coincide with their approximations $F((|x| - 1),(y| - 1))$ (light brown dashed lines) giving insignificant approximating error $10^{13} L^1$ (red and orange lines).

where we added a new variable $\phi = \arg[D(1-E)/(1+E)]$.

To shorten the description of the dependence of the coupling strength J on the multiplication factor k and avoid overcomplicated analytical expressions, we present the plot of its approximation, which alternatively can be calculated numerically and can be approximated by an expression $1/k - 1$ with good accuracy. Additionally, we calculated $L^1([-\pi/2, \pi/2])$ according to Eq. (A1) for each value of k. An additional good measure of the approximation quality is L^∞, which is the maximal discrepancy between the functions $F((|x| - 1),(\pi|J))$ and $f(x)_{\text{target}}$, which has a similar behavior as the original norm. Figure 8(a) shows all the plots corresponding to the multiplication by a positive factor $k > 0$ with the special points of the minimal error at $k = 1, 0.5$ and $k = 0$. These graphs explain why the lesser factors are more reliable for the multiplication and why multiplying by larger factors without factorization leads to worse performance.

The same task of multiplication by a negative factor $k < 0$ is illustrated in Fig. 8(b). The $J(k)$ function can be approximated with reasonable accuracy by an expression $0.4 - 1/k$. Since the general error has a much higher factor $\approx 10^2$ for the negative values, one has to accompany this block with an additional linear embedding to achieve a good approximation. The nonlinear function $3/2\ \tanh(4x) + x/2$ and its approximation with the optimized function $F((|x| - 1),(\pi| - 0.9036))$ is depicted in Fig. 8(c). The lowest accuracy is observed near the origin.

The final example of the half-sum approximation is illustrated in Fig. 8(d). The surprisingly good agreement (with an error of 10^{-13} of the magnitude) between the initial function and its XY representation $F((|x| - 1),(|y| - 1))$ can be explained with the help of the Taylor expansion of Eq. (2) near zeros. It gives the linear coefficient of $1/2$ accurate to the fourth order of approximation. With all the presented information, one can estimate the maximal discrepancy between an arbitrary NN and its transferred XY analog, which will be a good measure of the approximation quality and adequacy of the transfer procedure.

[1] G. E. Moore, Electronics **38**, 8 (1965).

[2] G. E. Moore, M. D. Hill, N. P. Jouppi, and G. S. Sohi, *Readings in Computer Architecture* (Morgan Kaufmann Publishers Inc., San Francisco, CA, 2000), pp. 56–59.

[3] J. Von Neumann, IEEE Ann. History Computing **15**, 27 (1993).

[4] J. Edwards and S. O'Keefe, *2016 IEEE Symposium Series on Computational Intelligence (SSCI)* (IEEE, New York, 2016), pp. 1–5.

[5] M. Naylor and C. Runciman, *Symposium on Implementation and Application of Functional Languages* (Springer, New York, 2007), pp. 129–146.

[6] C. S. Lent, K. W. Henderson, S. A. Kandel, S. A. Corcelli, G. L. Snider, A. O. Orlov, P. M. Kogge, M. T. Niemier, R. C. Brown, J. A. Christie, *et al.*, *2016 IEEE International Conference on Rebooting Computing (ICRC)* (IEEE, New York, 2016), pp. 1–7.

[7] D. Shin and H.-J. Yoo, Proc. IEEE **108**, 1245 (2019).

[8] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, arXiv:1705.06963.

[9] S. Furber, J. Neural Eng. **13**, 051001 (2016).

[10] H. Sompolinsky, Phys. Today **41**, 70 (1988).

[11] Y. LeCun, Y. Bengio, and G. Hinton, Nature (London) **521**, 436 (2015).

[12] J. J. Hopfield and D. W. Tank, Science **233**, 625 (1986).

[13] D. L. Stein and C. M. Newman, *Spin Glasses and Complexity* (Princeton University Press, Princeton, NJ, 2013), Vol. 4.

[14] F. Barahona, J. Phys. A: Math. Gen. **15**, 3241 (1982).

[15] P. M. Chaikin, T. C. Lubensky, and T. A. Witten, *Principles of Condensed Matter Physics* (Cambridge University Press, Cambridge, UK, 1995), Vol. 10.

[16] J. Kosterlitz, J. Phys. C **7**, 1046 (1974).

[17] R. Gupta, J. DeLapp, G. G. Batrouni, G. C. Fox, C. F. Baillie, and J. Apostolakis, Phys. Rev. Lett. **61**, 1996 (1988).

[18] P. Gawiec and D. R. Grempel, Phys. Rev. B **44**, 2613 (1991).

[19] J. M. Kosterlitz and N. Akino, Phys. Rev. Lett. **82**, 4094 (1999).

[20] B. V. Svistunov, E. S. Babaev, and N. V. Prokof'ev, *Superfluid States of Matter* (CRC Press, Boca Raton, FL, 2015).

[21] P. Martinoli and C. Leemann, J. Low Temp. Phys. **118**, 699 (2000).

[22] J. Affolter, M. Tesei, H. Pastoriza, C. Leemann, and P. Martinoli, Phys. C (Amsterdam, Neth.) **369**, 313 (2002).

[23] M. J. P. Gingras and D. A. Huse, Phys. Rev. B **53**, 15193 (1996).

[24] M. Franz and A. P. Iyengar, Phys. Rev. Lett. **96**, 047007 (2006).

[25] J. Struck, M. Weinberg, C. Ölschläger, P. Windpassinger, J. Simonet, K. Sengstock, R. Höppner, P. Hauke, A. Eckardt, M. Lewenstein *et al.*, Nat. Phys. **9**, 738 (2013).

[26] A. Kosior and K. Sacha, Phys. Rev. A **87**, 023602 (2013).

[27] N. G. Berloff, M. Silva, K. Kalinin, A. Askitopoulos, J. D. Töpfer, P. Cilibrizzi, W. Langbein, and P. G. Lagoudakis, Nat. Mater. **16**, 1120 (2017).

[28] P. G. Lagoudakis and N. G. Berloff, New J. Phys. **19**, 125008 (2017).

[29] G. Montemezzani, G. Zhou, and D. Z. Anderson, Opt. Lett. **19**, 2012 (1994).

[30] D. Psaltis, D. Brady, X.-G. Gu, and S. Lin, *Landmark Papers on Photorefractive Nonlinear Optics* (World Scientific, Singapore, 1995), pp. 541–546.

[31] F. Duport, B. Schneider, A. Smerieri, M. Haelterman, and S. Massar, Opt. Express **20**, 22783 (2012).

[32] Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund *et al.*, Nat. Photon. **11**, 441 (2017).

[33] G. Csaba and W. Porod, Appl. Phys. Rev. **7**, 011302 (2020).

[34] R. Frank, Phys. Rev. B **85**, 195463 (2012).

[35] R. Frank, Ann. Phys. (NY) **525**, 66 (2013).

[36] M. De Giorgi, D. Ballarini, E. Cancellieri, F. M. Marchetti, M. H. Szymanska, C. Tejedor, R. Cingolani, E. Giacobino, A. Bramati, G. Gigli, and D. Sanvitto, Phys. Rev. Lett. **109**, 266407 (2012).

[37] F. Marsault, H. S. Nguyen, D. Tanese, A. Lemaître, E. Galopin, I. Sagnes, A. Amo, and J. Bloch, Appl. Phys. Lett. **107**, 201115 (2015).

[38] T. Gao, P. S. Eldridge, T. C. H. Liew, S. I. Tsintzos, G. Stavrinidis, G. Deligeorgis, Z. Hatzopoulos, and P. G. Savvidis, Phys. Rev. B **85**, 235102 (2012).

[39] D. Ballarini, M. De Giorgi, E. Cancellieri, R. Houdré, E. Giacobino, R. Cingolani, A. Bramati, G. Gigli, and D. Sanvitto, Nat. Commun. **4**, 1778 (2013).

[40] A. V. Zasedatelev, A. V. Baranikov, D. Urbonas, F. Scafirimuto, U. Scherf, T. Stöferle, R. F. Mahrt, and P. G. Lagoudakis, Nat. Photon. **13**, 378 (2019).

[41] A. Amo, T. Liew, C. Adrados, R. Houdré, E. Giacobino, A. Kavokin, and A. Bramati, Nat. Photon. **4**, 361 (2010).

[42] K. P. Kalinin and N. G. Berloff, Phys. Rev. Lett. **121**, 235302 (2018).

[43] A. Opala, S. Ghosh, T. C. H. Liew, and M. Matuszewski, Phys. Rev. Appl. **11**, 064029 (2019).

[44] D. Ballarini, A. Gianfrate, R. Panico, A. Opala, S. Ghosh, L. Dominici, V. Ardizzone, M. De Giorgi, G. Lerario, G. Gigli *et al.*, Nano Lett. **20**, 3506 (2020).

[45] X. Guo, T. D. Barrett, Z. M. Wang, and A. Lvovsky, Photon. Res. **9**, B71 (2021).

[46] B. P. Marsh, Y. Guo, R. M. Kroeze, S. Gopalakrishnan, S. Ganguli, J. Keeling, and B. L. Lev, Phys. Rev. X **11**, 021048 (2021).

[47] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning* (MIT Press, Cambridge, MA, 2016), Vol. 1.

[48] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, New York, 2015), pp. 1–9.

[49] K. He, X. Zhang, S. Ren, and J. Sun, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, New York, 2016), pp. 770–778.

[50] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Rev. Mod. Phys. **91**, 045002 (2019).

[51] L. Zdeborová, Nat. Phys. **16**, 602 (2020).

[52] L. Deng and D. Yu, Found. Trends Signal Process. **7**, 197 (2014).

[53] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, Art. Intel. Rev. **26**, 159 (2007).

[54] C. R. Turner, A. Fuggetta, L. Lavazza, and A. L. Wolf, J. Syst. Softw. **49**, 3 (1999).

[55] K. P. Kalinin, A. Amo, J. Bloch, and N. G. Berloff, Nanophotonics **9**, 4127 (2020).

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, J. Mach. Learning Res. **12**, 2825 (2011).

[57] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, *Proceedings of the IEEE International Conference on Computer Vision* (IEEE, New York, 2017), pp. 843–852.

[58] K. Kawaguchi, J. Huang, and L. P. Kaelbling, Neural Comput. **31**, 1462 (2019).

[59] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, edited by D. Precup and Y. W. Teh (PMLR, 2017), Vol. 70, pp. 2847–2854.

[60] R. Eldan and O. Shamir, *29th Annual Conference on Learning Theory*, Proceedings of Machine Learning Research, edited by V. Feldman, A. Rakhlin, and O. Shamir (PMLR, New York, 2016), Vol. 49, pp. 907–940.

[61] X. Glorot, A. Bordes, and Y. Bengio, Deep sparse rectifier neural networks, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (JMLR Workshop and Conference Proceedings, 2011), pp. 315–323.

[62] V. Nair and G. E. Hinton, *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Omnipress, Madison, WI, USA, 2010), pp. 807–814.

[63] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, *ICLR (Poster)*, edited by Y. Bengio and Y. LeCun (2016).

[64] C. Weisbuch, M. Nishioka, A. Ishikawa, and Y. Arakawa, Phys. Rev. Lett. **69**, 3314 (1992).

[65] H. Ohadi, Y. del Valle-Inclan Redondo, A. Dreismann, Y. G. Rubo, F. Pinsker, S. I. Tsintzos, Z. Hatzopoulos, P. G. Savvidis, and J. J. Baumberg, Phys. Rev. Lett. **116**, 106403 (2016).

[66] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison *et al.*, Pytorch: An imperative style, high-performance deep learning library, in *Advances in neural information processing systems* (Curran Associates, Inc., 2019), Vol. 32, pp. 8026–8037.

[67] D. P. Kingma and J. Ba, arXiv:1412.6980.