View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Repositorio Universidad de Zaragoza

PHYSICAL REVIEW E **100**, 050102(R) (2019)

Rapid Communications

Learning a local symmetry with neural networks

A. Decelle,[1] V. Martin-Mayor,[2,3] and B. Seoane[4,5]

[1]*Laboratoire de Recherche en Informatique, TAU - INRIA, CNRS, Université Paris-Sud et Université Paris-Saclay, Bât. 660, 91190 Gif-sur-Yvette, France*
[2]*Departamento de Física Teórica, Universidad Complutense, 28040 Madrid, Spain*
[3]*Instituto de Biocomputación y Física de Sistemas Complejos (BIFI), 50018 Zaragoza, Spain*
[4]*Sorbonne Université, CNRS, IBPS, UMR 7238, Laboratoire de Biologie Computationnelle et Quantitative (LCQB), 75005 Paris, France*
[5]*Sorbonne Université, Institut des Sciences du Calcul et des Données (ISCD), 75005 Paris, France*

(Received 7 June 2019; published 6 November 2019)

We explore the capacity of neural networks to detect a symmetry with complex local and non-local patterns: the gauge symmetry Z_2. This symmetry is present in physical problems from topological transitions to quantum chromodynamics, and controls the computational hardness of instances of spin-glasses. Here, we show how to design a neural network, and a dataset, able to learn this symmetry and to find compressed latent representations of the gauge orbits. Our method pays special attention to system-wrapping loops, the so-called Polyakov loops, known to be particularly relevant for computational complexity.

DOI: 10.1103/PhysRevE.100.050102

The physics community is greatly excited by the possibilities offered by machine-learning tools, which have reached *superhuman* performance in tasks of significant complexity (think, for instance, of Go playing [1]). Indeed, deep (convolutional) neural networks (DCNN) [2,3], initially developed for classification and pattern-recognition tasks, have been applied to the identification of phases of matter [4–10], including glasses [11–13] and topological states [14], or even to seemingly for-humans-only tasks, such as finding real-space renormalization group transformations [15] (this is just a somewhat arbitrary selection of, literally, hundreds of applications to physics).

In this context, local (or gauge) symmetries pose a major challenge due to the absence of any local or global order parameter [16], which explains why only preliminary studies have been conducted [5,6]. In fact, thanks to their convolutional layers, DCNN successfully handle locally global translations and rotations: even if moved, DCNN still identify a previously learned image. The obvious next step for physicists is to consider more general symmetries for practical purposes.

The specific question we had in mind was whether DCNNs could be used to predict the *computational complexity* of a particular optimization problem instance. Spin-glasses represent the perfect playground to test this idea, because finding the ground state of a simple Hamiltonian, such as

$$\mathcal{H} = -\sum_{\langle \boldsymbol{x},\boldsymbol{y}\rangle} J_{\boldsymbol{xy}}\sigma_{\boldsymbol{x}}\sigma_{\boldsymbol{y}}\,,\ (\sigma_{\boldsymbol{x}} = \pm 1 \text{ for all sites } \boldsymbol{x})\,, \qquad (1)$$

is an NP-complete problem as soon as the underlying interaction graph is non-planar [17,18] (we consider statistically independent couplings $J_{\boldsymbol{xy}} = \pm 1$ with 50% probability). The classification problem is motivated because the computational difficulty of solving different problem instances of Eq. (1) spreads over several orders of magnitude [19–24], even for such a modest number of spins as $N \sim 500$.[1] In spite of the question's practical relevance, it is still unknown which features of the coupling-matrix $J_{\boldsymbol{xy}}$ cause this tremendous disparity of computational cost [21]. DCNNs would be an obvious choice to address the computational-cost classification problem, were it not for the gauge symmetry of Hamiltonian (1) (the $\epsilon_{\boldsymbol{x}} = \pm 1$ are arbitrary) [25]

$$J_{\boldsymbol{xy}} \to \tilde{J}_{\boldsymbol{xy}} = J_{\boldsymbol{xy}}\epsilon_{\boldsymbol{x}}\epsilon_{\boldsymbol{y}} \quad \text{and} \quad \sigma_{\boldsymbol{x}} \to \tilde{\sigma}_{\boldsymbol{x}} = \epsilon_{\boldsymbol{x}}\sigma_{\boldsymbol{x}}\,. \qquad (2)$$

All problem instances related by this transformation belong to the same *gauge orbit*. Now, the difficulty for solving problems from the same orbit is *identical*. Hence, our desired DCNN should first be able of telling us with certainty whether or not two problem instances belong to the same gauge orbit. For this task, one direct solution would be to embed the symmetry in the architecture of the neural network (as done in Refs. [26,27] to build a gauge equivariant DCNN). However, in this work, we will rather explore the difficulties standard DCNNs face when trying to discover a complex symmetry not pre-inserted on their structure, which is strongly related to the problem of automatically learning of phases of matter that has recently attracted lots of attention in physics.

Here, we present a machine-learning algorithm that solves the problem of gauge-orbit identification as formulated for spin-glasses on the square lattice. The same algorithm works in the cubic lattice, although we are limited to systems of smaller linear size due to memory and computational costs. Interestingly, all the standard DCNNs for image classification

[1]Actually, Refs. [19–21,23,24] attempted to find equilibrium configurations using a Parallel Tempering algorithm down to some minimal temperature $T_{\min}$. In order to compute the ground state, one needs to push $T_{\min} = 0$, as done in Ref. [22]. Unfortunately, the lower $T_{\min}$, the larger the spread over the samples of the computational hardness, see, e.g., Refs. [19,23,24].

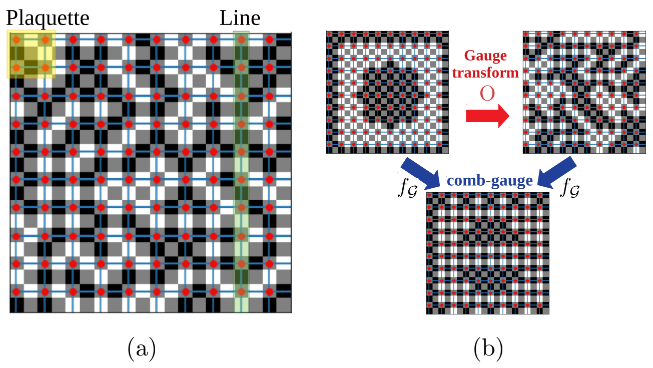


FIG. 1. (a) Chess transformation from the lattice (x, y), $0 \leqslant x, y \leqslant L-1$, to the image (x_1, x_2), $0 \leqslant x_1, x_2 \leqslant 2L-1$. Periodic boundary conditions (PBC) are implemented by two additional rows and columns framing the image (for clarity, we only show the additional row at $x_2 = -1$ and the additional column at $x_1 = 2L$). The spin at site $\boldsymbol{x} = (x, y)$ is assigned to the pixel $(x_1 = 2x, x_2 = 2y)$ in the image, depicted as dummy gray cells, which are set to zero when fed to the neural-network. The coupling $J_{\boldsymbol{x},\boldsymbol{y}}$ with $\boldsymbol{x} = (x, y)$ and $\boldsymbol{y} = (x+1, y)$ is in the pixel $(x_1 = 2x+1, x_2 = 2y)$ which is set to black if $J = 1$ (white if $J = -1$). Similarly, the pixel $(x_1 = 2x, x_2 = 2y+1)$ contains the coupling between (x, y) and $(x, y+1)$. The remaining pixels at the center of each plaquette, i.e., $(x_1 = 2x+1, x_2 = 2y+1)$, are also fixed as dummy gray pixels. We indicate with red dots the spin-pixels per site, while the blue edges are in the J-pixels joining neighboring spin-pixels. We also show the pixels necessary to compute a plaquette and a Polyakov loop [the (say) vertical line, which is a closed loop thanks to the PBC]. (b) A problem instance and one of its gauge transforms. Both instances lead to the same comb-gauge representation after gauge-fixing.

tried, including the ResNet [28], completely failed at this task. A careless posing of the problem could make it wrongly seem trivial. Indeed, instances from the same orbit share the value of every Wilson loop [29] [the product of couplings along a closed loop in the lattice, which is gauge invariant Eq. (2)]. Attention immediately falls on the *plaquette*, the shortest Wilson loop, see Fig. 1(a). However, two instances sharing the value of *every* plaquette, but differing on the so-called Polyakov loops (the shortest Wilson loops wrapping the system through the periodic boundary conditions), may have vastly different computational complexity [23]. We improve over Ref. [5] by teaching our machine to consider both local and non-local Wilson loops when studying a Z_2 gauge symmetry.

Let us highlight two other aspects of this problem that machine-learning practitioners may find attractive: (i) a training set of (essentially) arbitrary size can be easily generated and (ii) we have a gauge-fixing algorithm that tells us unambiguously if two coupling matrices belong to the same gauge orbit. The computational cost of this algorithm scales as a power of the system size L.

Below, we present two different approaches to solve this classification problem using DCNN (we employed the Keras-tensorflow and scikit-learn libraries [30,31]). Our first algorithm tells us if two instances are in the same gauge orbit. Our second algorithm is an autoencoder, a DCNN capable of finding a latent representation of a gauge orbit by means of an approximate gauge fixing. Although the latent representation can be used for classification purposes, it further allows clustering instances by orbits.

For square lattices, it is natural to feed the coupling matrix $\boldsymbol{J}$ to the neural network as an image. After considering several alternatives, our choice was to map our physical square lattice of size L to a square image of size $2L$ through the *chess* transformation illustrated in Fig. 1(a) (the chess transformation generalizes to three dimensions). Although one pixel out of two is wasted in the resulting image, we found that the learning process and the interpretation of results were easier with the chess transformation than with less memory-demanding representations.

Gauge-fixing and the comb gauge—Gauge transformations O are also illustrated in Fig. 1(b): the naked eye can hardly tell whether or not the images corresponding to two coupling matrices belong to the same gauge orbit. This question can be answered by fixing the gauge,[2] that is, to use a map $f_{\mathcal{G}} : \boldsymbol{J}^{\mathcal{O}_k} \rightarrow \hat{\boldsymbol{J}}^{\mathcal{O}_k}$ from any instance $\boldsymbol{J}$ from gauge orbit $\mathcal{O}_k$ to a single representative of it, $\hat{\boldsymbol{J}}$. Thus, two instances are in the same orbit if, and only if, $f_{\mathcal{G}}(\boldsymbol{J}) = f_{\mathcal{G}}(\boldsymbol{J}')$. We construct our mapping by changing the gauge: the $\boldsymbol{\epsilon} \equiv \{\epsilon_{\boldsymbol{x}}\}$ in Eq. (2) are chosen in such a way that $\tilde{J}_{\boldsymbol{x},\boldsymbol{y}} = 1$ for any horizontal coupling $\boldsymbol{x} - \boldsymbol{y} = (\pm 1, 0)$ (but for $\tilde{J}_{\boldsymbol{x}=(L-1,y),\boldsymbol{y}=(0,y)}$ which is equal to a gauge-invariant Polyakov loop), as well as $\tilde{J}_{\boldsymbol{x}=(0,y),\boldsymbol{y}=(0,y+1)} = 1$ for $0 \leqslant y < L-2$. We call this transform, the *comb-gauge*. We include a code performing this gauge-fixing in the Supplemental Material (SM) [32].

Construction of the dataset—We found inconvenient for our purposes the approach used in Ref. [5] to detect the gauge symmetry, namely constructing a (balanced) dataset of pairs of systems, a group with pairs of instances from the same orbit and the other group with pairs of randomly chosen $\boldsymbol{J}$s. Indeed, this classification problem is too easy. Most of the time, and this is what the DCNN learns, the pair of randomly chosen $\boldsymbol{J}$s will be so different that one could tell that they do not belong to the same orbit just by looking at a very reduced number of plaquettes.[3] A DCNN trained in this way would completely miss situations in which just a few couplings changed, and it would be blind to extensive transformations that leave every plaquettes unaltered. Therefore, we need to ensure that in our dataset it is not enough to check one (or few) plaquette(s).

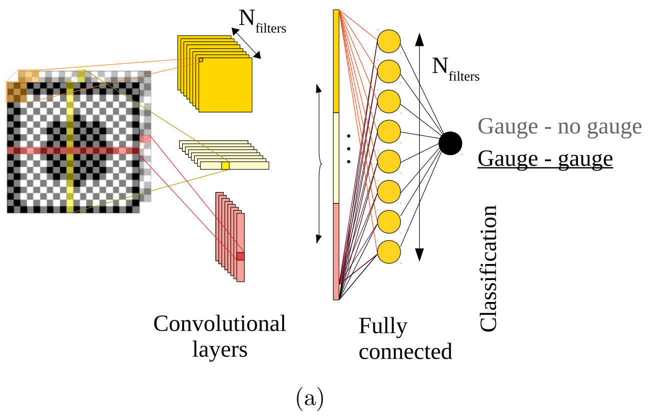
Specifically, our dataset is composed of $N_{\rm s}$ pairs $\{\boldsymbol{J}, \mathrm{O}(\boldsymbol{J}')\}$, with $\boldsymbol{J}' = \mathrm{F}(\boldsymbol{J})$, where O is a random gauge transform (i.e., with random $\{\epsilon_{\boldsymbol{x}}\}$) and F a simple transformation (see below and SM [32]). In 50% of the cases, F is the identity mapping (and $\boldsymbol{J} = \boldsymbol{J}'$), and in the other half, some transformations that change only a small fraction of the $J_{\boldsymbol{x},\boldsymbol{y}}$.

In the so-called $\boldsymbol{J}' = \mathrm{R}_q(\boldsymbol{J})$ transformation, a fraction q of randomly chosen $J_{\boldsymbol{xy}}$ is flipped.

In the (horizontal) line-transformation $\boldsymbol{J}' = L(\boldsymbol{J})$, $\boldsymbol{J}'$ is obtained from $\boldsymbol{J}$ by flipping the couplings joining $\boldsymbol{x} = (0, y)$ and $\boldsymbol{y} = (1, y)$ for any y [vertical transformation: $\boldsymbol{x} = (x, 0)$

[2]In this work we deal with an Abelian gauge group which makes gauge-fixing simple (difficulties arise for non-Abelian gauge groups, see, e.g., Ref. [41]).

[3]For two randomly chosen $\boldsymbol{J}$s, the probability of coincidence in k fixed, non-overlapping plaquettes falls as $1/2^k$.



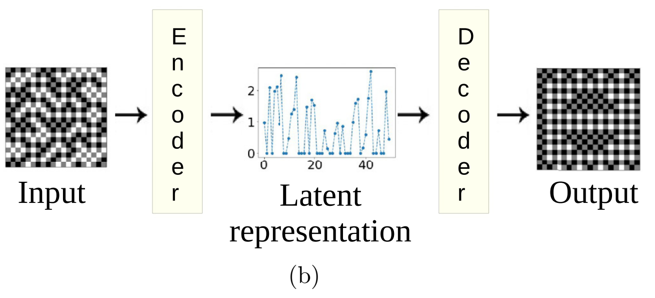


FIG. 2. (a) The typical architecture used for detecting the gauge symmetry between pairs of systems. It is important to scan both square-like kernels for the plaquettes and full-line kernels for the Polyakov loops. (b) Schematic representation of the autoencoder. The encoder is very similar to the architecture above, the decoder is typically made of upsampling layers (increasing the size of the input) and of convolutional layers.

and $\boldsymbol{y} = (x, 1)$, for all x]. The line transformation preserves the value of each plaquette, but the sign of all the horizontal (vertical) Polyakov loops is reversed. These line transformations,[4] are important when assessing the computational hardness [23].

In our dataset, when $\boldsymbol{J} \neq \boldsymbol{J}'$, we choose with 1/3 probability $\boldsymbol{J}' = \mathrm{L}(\boldsymbol{J})$ or, with probability 2/3, $\boldsymbol{J}' = \mathrm{R}_q(\boldsymbol{J})$. Line transformations are equally likely to be horizontal or vertical. If the chosen transformation is R_q, in order to force the scan of every plaquette, we pick $q \sim 1/L^2$ with 50% probability (we invert randomly 1–5 couplings), or $q = q_\mathrm{R}$ where q_R is a uniform random number with $1/(2L^2) \leqslant q_\mathrm{R} < 1/4$.

Construction of the DCNN—We build a DCNN that inputs the chess-transformed [see Fig. 1(a)] images representing a pair of coupling matrices $\{\boldsymbol{J}, \mathrm{O}(\boldsymbol{J}')\}$ and outputs the probability that the two instances belong to the same gauge-orbit.[5]

The Euclidean geometry of our problem suggests to use convolutional neural networks (CNN) [33–35], which are well adapted to translational symmetry. Specifically, we combine in parallel three CNNs that scan simultaneously all the pixels that would be necessary to compute the plaquettes [i.e., the pixels inside the square in Fig. 2(a)], and the Polyakov loops [that is, the pixels on the horizontal and vertical $1 \times L$ slabs in Fig. 2(a)]. We stress here that we feed the CNNs only the value of the pixels within these regions, and not the the product of the couplings along the loops, which means that, even with help, the machine still needs to learn alone how to compute gauge-invariant quantities to succeed. The first CNN allows us to find quickly small defects in the gauge symmetry, while the other two search for non-local defects. These three CNNs serve as feature detectors before a fully-connected layer that performs the classification. We illustrate in Fig. 2(a) the general architecture of our DCNN. Additional details, as well as sample programs, can be found in the SM [32].

Results for the classifying DCNN—For our dataset, we manage to obtain almost 100% of accuracy on sizes of $L = 5, 10$. Of course, the accuracy needed is application-specific. Consider, for instance, the problem of chaos (either in temperature or couplings) in spin-glasses, which is the physical origin of the variable computational hardness of different samples discussed in the introduction. We know that the fraction of links (hence of plaquettes) that can be randomly flipped without generating a catastrophic effect scales with L as $1/L^b$, with an exponent $b \approx 1$, both in $D = 2$ [36] and in $D = 3$ [37] (D is the space dimension). This scaling-law sets a scale: If the approximate algorithm fails to recognize the Gauge transformation when the fraction of flipped plaquettes goes above $1/L$, it will be useless for tasks related to the investigation of chaotic effects. We note that we have shaped our first algorithm to detect even one flipped plaquette. Our criterion was to stop the training when the accuracy reached the value 0.995 (it becomes extremely slow at this point). Within this setup, we reach accuracies on the test-set of 0.9938(3) for $L = 5$ and 0.9947(10) $L = 10$. In other words, even for our very exigent dataset, the DCNN learns to tell whether or not two problem instances really are the same orbit.

However, let $N_s(p)$ be the size of the training-set needed to reach a target accuracy p. We see in Fig. 3 that $N_s(p)$ is much smaller in the training-set that in the test-set (problem instances in the test-set are new to the DCNN). Furthermore, $N_s(p)$ grows significantly with L. We repeat exactly the same process but replacing our DCNN by the ResNet DCNN [28]. Being this DCNN more complex, it always overfits the data (it does not classify correctly unseen data). We observe that even the $L = 5$ case, in the range of N_s studied (see Fig. 3), the accuracy on the test-set remains fixed to that of a purely random guess. It is important to stress that training the ResNet is much costlier, and GPUs were necessary.

[4]Any other transformation can be expressed as a combination of broken plaquette(s) and/or line(s).

[5]The output of our trained DCNN is a continuous value in between 0 and 1. During the training, the gradient-descent algorithm imposes that, given a pair of coupling-matrices, the output is 0 (or 1) if the two are in the same gauge orbit (or not). However, because of the minimization algorithm, the optimization is done on a continuous function (defined in [0,1]). In that case, the results can be interpreted formally as a probability (i.e., how confident the network is that two instances are in the same gauge orbit). If needed, the output can be mapped to {0, 1} by assigning all answers below 0.5 to 0, and to 1 the rest.

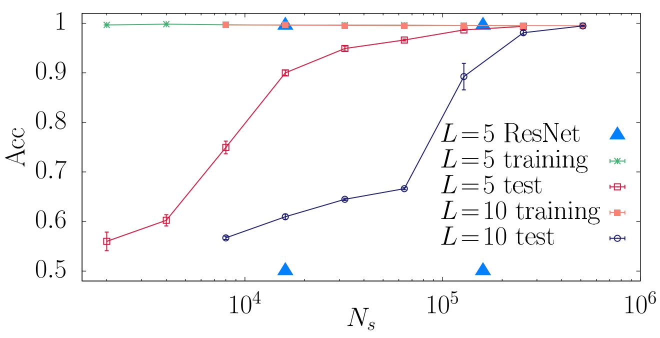


FIG. 3. General accuracy of the classification task of the pairs of coupling matrices in our dataset (both for the training and the test-set), as computed for lattices of sizes $L = 5$ and 10 using our DCNN, and for $L = 5$ using the ResNet DCNN. Data and errors are computed from averages over 5 independent learning runs and datasets.

We have found that the difficulty of the problem is largely caused by the Polyakov-loop flipping line transformations. More details on this analysis can be found in the SM [32].

Learning to fix the gauge—Gauge-fixing may be regarded as an algorithm to reduce the dimensionality of the coupling-matrix $\boldsymbol{J}$ with no information loss. Hence, it is natural to ask ourselves if a particular type of DCNN, an autoencoder (AE) [38,39], may learn to fix the gauge. Indeed, an AE takes an input vector $\boldsymbol{x}$ and maps it to a latent representation $f_{\mathcal{E}}(\boldsymbol{x})$ [typically, $f_{\mathcal{E}}(\boldsymbol{x})$ is of smaller dimensionality than $\boldsymbol{x}$]. A decoder generates a reconstructed vector from the latent representation afterwards, $\boldsymbol{x}' = f_{\mathcal{D}}(f_{\mathcal{E}}(\boldsymbol{x}))$. The weights of the encoder $f_{\mathcal{E}}$ and the decoder $f_{\mathcal{D}}$ functions are chosen to minimize a loss function (e.g., the L_2 distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$).

At variance with the traditional approach, we will not ask our AE to reconstruct the input but to fix the gauge, that is to reconstruct a unique $\hat{\boldsymbol{J}}$ (the comb-gauge described above) for all the instances in a given gauge orbit.

TABLE I. The autoencoder as a classifier. Fraction of not-trivially-one couplings that are different in the "comb-gauge" output of the AE as applied to two instances $\{\boldsymbol{J}, \mathrm{O}(\boldsymbol{J}')\}$ from: $\boldsymbol{J} = \boldsymbol{J}'$ [$p^{J,J}$], $\boldsymbol{J}' = \mathrm{R}_{q=0.5}(\boldsymbol{J}')$ [$p^{J,J'}$], $\boldsymbol{J}' = \mathrm{R}_{q=0.1}(\boldsymbol{J})$ [$p^{J,\mathrm{R}_q(J)}$], or $\boldsymbol{J}' = \mathrm{L}(\boldsymbol{J})$ [$p^{J,\mathrm{L}(J)}$] [where the exact value is $L/(L^2+1)$]. The AE was trained with N_s instances, randomly extracted from N_O orbits. The results were computed from 1000 pairs $\{\boldsymbol{J}, \mathrm{O}(\boldsymbol{J}')\}$, with $\boldsymbol{J}$ extracted from orbits outside the training-set, and five independent learnings.

L	N_s	$N_\mathcal{O}$	$p^{J,J}$	$p^{J,J'}$	$p^{J,\mathrm{R}_{q=0.1}(J)}$	$p^{J,\mathrm{L}(J)}$
5	100k	1k	0.020(6)	0.494(3)	0.410(6)	0.20(2)
6	400k	1k	0.028(8)	0.500(4)	0.422(2)	0.17(3)
8	800k	8k	0.044(14)	0.470(13)	0.394(11)	0.105(11)

Our encoder will essentially share the architecture of our classifying DCNN (namely, the three CNNs of Fig. 2 without the classification layer). The decoder takes the encoder's output, and pipes it to an upsampling layer, followed by our three feature-detector CNNs and by a last CNN from which we take the output (more details can be found in the SM [32]). The output from a given coupling-matrix $\boldsymbol{J}$ is an attempted reconstruction of its comb-gauge representation [Fig. 1(b)].

The AE can be used as a classifier simply by comparing the "comb-gauge" obtained from two problem instances. As shown in Table I, only pairs of instances from the same orbit have a similar "comb-gauge" (the performance does not deteriorate when the system size increases).

We can gain some understanding by visualizing the latent representation, see Fig. 4. Indeed the AE's latent representation clusters problem instances belonging to the same orbit. Furthermore, not only the representation for two problems from the same orbit is nearly identical: changing a few links

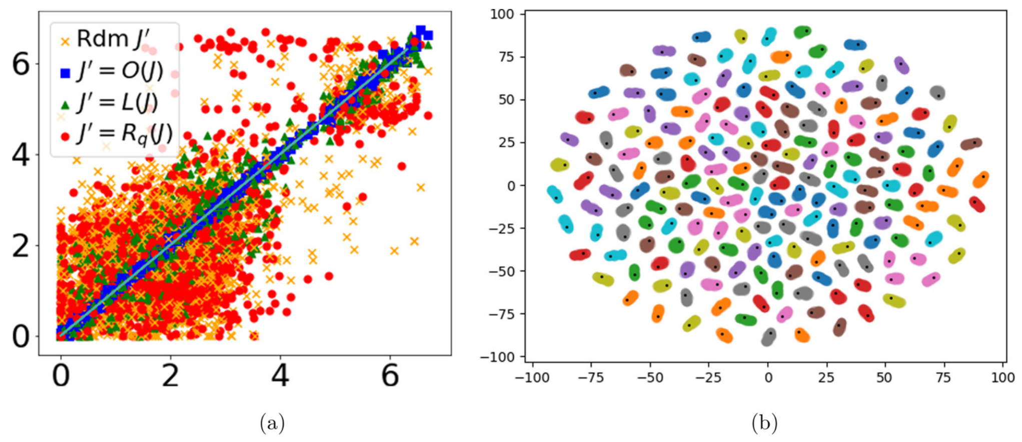


FIG. 4. Visualizing the 50-dimensional AE's latent representation. (a) scatter-plot comparison for pairs of problem instances $\{\boldsymbol{J}, \mathrm{O}(\boldsymbol{J}')\}$. We display 50 points for each pair $\{\boldsymbol{J}, \boldsymbol{J}'\}$, namely $(x_i^{J}, x_i^{J'})$, where x_i^{J} and $x_i^{J'}$ are the ith coordinates of both latent representations. We consider pairs $\{\boldsymbol{J}, \mathrm{O}(\boldsymbol{J}')\}$, with $\boldsymbol{J}' = \boldsymbol{J}$ (blue squares), $\boldsymbol{J}' = \mathrm{R}_{q=0.5}(\boldsymbol{J})$ (orange crosses), $\boldsymbol{J}' = \mathrm{R}_{q=0.1}(\boldsymbol{J})$ (red circles), and $\boldsymbol{J}' = L(\boldsymbol{J})$ (green triangles). The plot contains data from 50 pairs of each type. (b) two-dimensional t-sne representation [40] of the latent representation as obtained for 20000 instances randomly extracted from 200 (unrelated) gauge orbits. Instances from the same orbit are represented by points of the same color (some orbits share color, due to our limited palette) forming the bigger regions of the same color since they clusterize around the same position. The black points inside each cluster are the t-sne coordinates for the latent representation obtained for the gauge-comb representative of each of these orbits.

or performing a line transformation results into a significantly different latent representation. Theoretically, we know that the minimal number of pixels necessary to encode the orbit would be $L^2 + 1$ (the pixels that do not belong to the comb in the comb-gauge). Here, we found practical to speed up the learning, to consider latent representations of twice this value.

Conclusions—We have demonstrated a successful machine-learning approach to detect whether or not two spin-glass instances are mutually related by a gauge transformation. This problem is particularly challenging for neural networks due to the absence of an order parameter. In fact, we have checked the failure of the standard DCNNs for image classification, such as pre-trained DCNNs, no matter the size of the training set. Our results underline the necessity of carefully choosing the learning dataset, if we want the DCNN to learn the full symmetry (which includes global Wilson loops). We show that our DCNNs are able to learn the gauge symmetry and even to find a latent representation that can be used to fix the gauge. This success comes at the cost of very large training datasets, whose size need to grow with the system size. Now that we have in our hands DCNNs able to identify gauge symmetries, we will approach our original question, namely *what makes certain problem instances far more computationally costly than others?*

Acknowledgments—We thank L. A. Fernández for encouraging discussions and Marco Baity-Jesi for his careful reading of the manuscript. This work was partially supported by Ministerio de Economía, Industria y Competitividad (MINECO) (Spain) and by EU's FEDER program through Grants No. FIS2015-65078-C2-1-P and No. PGC2018-094684-B-C21 and by the LabEx CALSIMLAB (public Grant No. ANR-11-LABX-0037-01 constituting a part of the "Investissements d'Avenir" program - reference No. ANR-11-IDEX-0004-02).

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, Nature **524**, 484 (2016).
[2] Y. LeCun, Y. Bengio, and G. Hinton, Nature **521**, 436 (2015).
[3] J. Schmidhuber, Neural Networks **61**, 85 (2015).
[4] G. Torlai and R. G. Melko, Phys. Rev. B **94**, 165134 (2016).
[5] J. Carrasquilla and R. G. Melko, Nat. Phys. **13**, 431 (2017).
[6] S. J. Wetzel and M. Scherzer, Phys. Rev. B **96**, 184410 (2017).
[7] E. van Nieuwenburg, Y.-H. Liu, and S. Huber, Nat. Phys. **13**, 435 (2017).
[8] L. Wang, Phys. Rev. B **94**, 195105 (2016).
[9] T. Ohtsuki and T. Ohtsuki, J. Phys. Soc. Jpn. **86**, 044708 (2017).
[10] M. J. S. Beach, A. Golubeva, and R. G. Melko, Phys. Rev. B **97**, 045207 (2018).
[11] S. S. Schoenholz, E. D. Cubuk, D. M. Sussman, and E. Kaxiras, Nat. Phys. **12**, 469 (2016).
[12] S. S. Schoenholz, E. D. Cubuk, E. Kaxiras, and A. J. Liu, Proc. Natl. Acad. Sci. U.S.A. **114**, 263 (2017).
[13] E. D. Cubuk, S. S. Schoenholz, J. M. Rieser, B. D. Malone, J. Rottler, D. J. Durian, E. Kaxiras, and A. J. Liu, Phys. Rev. Lett. **114**, 108001 (2015).
[14] D.-L. Deng, X. Li, and S. Das Sarma, Phys. Rev. B **96**, 195145 (2017).
[15] M. Koch-Janusz and Z. Ringel, Nat. Phys. **14**, 578 (2018).
[16] S. Elitzur, Phys. Rev. D **12**, 3978 (1975).
[17] F. Barahona, J. Phys. A: Math. Gen. **15**, 3241 (1982).
[18] S. Istrail, in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing - STOC '00* (ACM Press, New York, NY, 2003), pp. 87–96.
[19] R. Alvarez Baños, A. Cruz, L. A. Fernandez, J. M. Gil-Narvion, A. Gordillo-Guerrero, M. Guidetti, A. Maiorano, F. Mantovani, E. Marinari, V. Martin-Mayor, J. Monforte-Garcia, A. Muñoz Sudupe, D. Navarro, G. Parisi, S. Perez-Gaviro, J. J. Ruiz-Lorenzo, S. F. Schifano, B. Seoane, A. Tarancon, R. Tripiccione, and D. Yllanes (Janus Collaboration), J. Stat. Mech. (2010) P06026.
[20] L. A. Fernández, V. Martín-Mayor, G. Parisi, and B. Seoane, Europhys. Lett. **103**, 67003 (2013).
[21] A. Billoire, J. Stat. Mech. (2014) P04016.
[22] V. Martín-Mayor and I. Hen, Sci. Rep. **5**, 15324 (2015).
[23] L. Fernandez, E. Marinari, V. Martin-Mayor, G. Parisi, and D. Yllanes, J. Stat. Mech.: Theory Exp. (2016) 123301.
[24] A. Billoire, L. A. Fernandez, A. Maiorano, E. Marinari, V. Martin-Mayor, J. Moreno-Gordo, G. Parisi, F. Ricci-Tersenghi, and J. J. Ruiz-Lorenzo, J. Stat. Mech.: Theory Exp. (2018) 033302.
[25] G. Toulouse, Commun. Phys. **2**, 115 (1977).
[26] T. N. Kipf and M. Welling, arXiv:1609.02907 (2016).
[27] T. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling, in *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov (PMLR, 2019), pp. 1321–1330.
[28] K. He, X. Zhang, S. Ren, and J. Sun, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, Piscataway, NJ, 2016), pp. 770–778.
[29] I. Montvay and G. Münster, *Quantum Fields on a Lattice* (Cambridge University Press, Cambridge, 1997).
[30] F. Chollet *et al.*, "Keras", https://keras.io (2015).
[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, J. Mach. Learn. Res. **12**, 2825 (2011).
[32] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevE.100.050102 for the code with additional details about implementation.
[33] K. Fukushima, Biol. Cybern. **36**, 193 (1980).

[34] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, Neural Comput. **1**, 541 (1989).

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, in *Advances in Neural Information Processing Systems 25*, edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Curran Associates, Inc., Lake Tahoe, 2012), pp. 1097–1105.

[36] F. Krzakala and J. P. Bouchaud, Europhys. Lett. **72**, 472 (2005).

[37] H. G. Katzgraber and F. Krzakala, Phys. Rev. Lett. **98**, 017201 (2007).

[38] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning Internal Representations by Error Propagation, Technical Report, California University, San Diego La Jolla Institute for Cognitive Science, 1985.

[39] D. H. Ballard, in *Proceedings of the Sixth National Conference on Artificial Intelligence - Volume 1, AAAI'87* (AAAI Press, Seattle, Washington, 1987), pp. 279–284.

[40] L. van der Maaten and G. Hinton, J. Mach. Learn. Res. **9**, 2579 (2008).

[41] E. Marinari, C. Parrinello, and R. Ricci, Nucl. Phys. B **362**, 487 (1991).