Learning a local symmetry with neural networks

A. Decelle, V. Martin-Mayor, and B. Seoane

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

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

(1)

is an NP-complete problem as soon as the underlying interaction graph is non-planar [17,18] (we consider statistically independent couplings $J_{xy} = \pm 1$ 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 $N \sim 500$. In spite of the question’s practical relevance, it is still unknown which features of the coupling-matrix $J_{xy}$ 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 $\epsilon_x = \pm 1$ are arbitrary) [25]

$$J_{xy} \rightarrow \tilde{J}_{xy} = J_{xy} \epsilon_x \epsilon_y \quad \text{and} \quad \sigma_x \rightarrow \tilde{\sigma}_x = \epsilon_x \sigma_x.$$

(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...
tation immediately falls on the Wilson loops when studying a loop, see Fig. 1(a). However, two instances sharing the value computational complexity [23]. We improve over Ref. [5] by loops (the shortest Wilson loops wrapping the system through of every Wilson loop [29] [the product of couplings along a plaquette, but differing on the so-called Polyakov loop), as well as ˜\(J\) (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.

\textit{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 \(f_2: J^{cl_1} \rightarrow J^{cl_2}\) from any instance \(J\) from gauge orbit \(O_k\) to a single representative of it, \(\tilde{J}\). Thus, two instances are in the same orbit if, and only if, \(f_2(J) = f_2(J')\). We construct our mapping by changing the gauge: the \(\epsilon \equiv \{e_{kl}\}\) in Eq. (2) are chosen in such a way that \(\tilde{J}_{x,y} = 1\) for any horizontal coupling \(x - y = (\pm 1, 0)\) (but for \(J_{x=L-1,y}=0)\) which is equal to a gauge-invariant Polyakov loop), as well as \(\tilde{J}_{x=0,y=0,y+1}) = 1\) for \(0 \leq y < L - 2\). We call this transform, the \textit{comb-gauge}. We include a code performing this gauge-fixing in the Supplemental Material (SM) [32].

\textit{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 \(J_s\). Indeed, this classification problem is too easy. Most of the time, and this is what the DCNN learns, the pair of randomly chosen \(J_s\) 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 \(N_s\) pairs \(\{J, O(J')\}\) with \(J' = F(J)\), where \(O\) is a random gauge transform (i.e., with random \(\{e_{kl}\}\)) and \(F\) a simple transformation (see below and SM [32]). In 50\% of the cases, \(F\) is the identity mapping (and \(J = J'\)), and in the other half, some transformations that change only a small fraction of the \(J_{x,y}\).

In the so-called \(J' = R_{0}(J)\) transformation, a fraction \(q\) of randomly chosen \(J_{xy}\) is flipped.

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

\(^2\)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 \(J_s\), the probability of coincidence in \(k\) fixed, non-overlapping plaquettes falls as \(1/2^k\).
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 $N_i(p)$ be the size of the training-set needed to reach a target accuracy $p$. We see in Fig. 3 that $N_i(p)$ is much smaller in the training-set that in the test-set (problem instances in the test-set are new to the DCNN). Furthermore, $N_i(p)$ 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 $N_i$ 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.

\[ \text{any other transformation can be expressed as a combination of broken plaquette(s) and/or line(s).} \]

\[ \text{The output of our trained DCNN is a continuous value in between 0 and 1. During the training, the gradient-descent algorithm imposes that, given a pair of coupling-matrices, the output is 0 (or 1) if the two are in the same gauge orbit (or not). However, because of the minimization algorithm, the optimization is done on a continuous function (defined in [0,1]). In that case, the results can be interpreted formally as a probability (i.e., how confident the network is that two instances are in the same gauge orbit). If needed, the output can be mapped to [0, 1] by assigning all answers below 0.5 to 0, and to 1 the rest.} \]
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 $J$ 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 $x$ and maps it to a latent representation $f_E(x)$ (typically, $f_E(x)$ is of smaller dimensionality than $x$). A decoder generates a reconstructed vector from the latent representation afterwards, $x' = f_D(f_E(x))$. The weights of the encoder $f_E$ and the decoder $f_D$ functions are chosen to minimize a loss function (e.g., the $L_2$ distance between $x$ and $x'$).

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 $\hat{J}$ (the comb-gauge described above) for all the instances in a given gauge orbit.

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 $J$ 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

<table>
<thead>
<tr>
<th>$L$</th>
<th>$N_s$</th>
<th>$N_O$</th>
<th>$p^{J'}$</th>
<th>$p^J$</th>
<th>$p^{R_{q=0}(J)}$</th>
<th>$p^{L(J)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>100k</td>
<td>1k</td>
<td>0.020(6)</td>
<td>0.494(3)</td>
<td>0.410(6)</td>
<td>0.20(2)</td>
</tr>
<tr>
<td>6</td>
<td>400k</td>
<td>1k</td>
<td>0.028(8)</td>
<td>0.500(4)</td>
<td>0.422(2)</td>
<td>0.17(3)</td>
</tr>
<tr>
<td>8</td>
<td>800k</td>
<td>8k</td>
<td>0.044(14)</td>
<td>0.470(13)</td>
<td>0.394(11)</td>
<td>0.105(11)</td>
</tr>
</tbody>
</table>

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.

FIG. 4. Visualizing the 50-dimensional AE’s latent representation. (a) scatter-plot comparison for pairs of problem instances $\{J, O(J)\}$. We display 50 points for each pair $\{J, J'\}$, namely $(x_i', x_i'')$, where $x_i'$ and $x_i''$ are the $i$th coordinates of both latent representations. We consider pairs $\{J, O(J)\}$, with $J' = J$ (blue squares), $J' = R_{q=0.5}(J)$ (orange crosses), $J' = R_{q=0.1}(J)$ (red circles), and $J' = L(J)$ (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.
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. PGCC2018-094684-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).


