New Journal of Physics
The open access journal at the forefront of physics

Deutsche Physikalische Gesellschaft DPG
IOP Institute of Physics

Published in partnership with: Deutsche Physikalische Gesellschaft and the Institute of Physics

OPEN ACCESS

RECEIVED
23 July 2020

REVISED
16 October 2020

ACCEPTED FOR PUBLICATION
23 October 2020

PUBLISHED
12 November 2020

PAPER

Phase detection with neural networks: interpreting the black box

Anna Dawid[1,2,*], Patrick Huembeli[2], Michal Tomza[1], Maciej Lewenstein[2,3] and Alexandre Dauphin[2]

[1] Faculty of Physics, University of Warsaw, Pasteura 5, 02-093 Warsaw, Poland
[2] ICFO - Institut de Ciències Fotòniques, The Barcelona Institute of Science and Technology, Av. Carl Friedrich Gauss 3, 08860 Castelldefels (Barcelona), Spain
[3] ICREA, Pg. Lluís Campanys 23, 08010 Barcelona, Spain
[*] Author to whom any correspondence should be addressed.

E-mail: Anna.Dawid@fuw.edu.pl

Keywords: interpretable machine learning, phase classification, neural networks

Abstract

Neural networks (NNs) usually hinder any insight into the reasoning behind their predictions. We demonstrate how influence functions can unravel the black box of NN when trained to predict the phases of the one-dimensional extended spinless Fermi–Hubbard model at half-filling. Results provide strong evidence that the NN correctly learns an order parameter describing the quantum transition in this model. We demonstrate that influence functions allow to check that the network, trained to recognize known quantum phases, can predict new unknown ones within the data set. Moreover, we show they can guide physicists in understanding patterns responsible for the phase transition. This method requires no *a priori* knowledge on the order parameter, has no dependence on the NN's architecture or the underlying physical model, and is therefore applicable to a broad class of physical models or experimental data.

1. Introduction

Machine learning (ML) influences everyday life in multiple ways with applications like text and voice recognition software, fingerprint identification, self-driving cars, and many others. These versatile algorithms, dealing with big and high-dimensional data, also have a noticeable impact on science, which harnessed neural networks (NNs) to solve problems of quantum chemistry, material science, and biology [1–4]. Physics is no different in exploring ML methods, encompassed already by astrophysics, high-energy physics, quantum state tomography, and quantum computing [5–11]. Especially abundant is the use of ML in phase classification. It is not surprising if one considers that determining the proper order parameters for unknown transitions is no trivial task, on the verge of being an art. It includes the search in the exponentially large Hilbert space and the examination of symmetries existing in the system, guided by the intuition and educated guess. The alternative route was shown, when NNs located the phase transitions for known models without *a priori* physical knowledge [12, 13]. Numerous works followed, studying classical [12, 14–17], quantum [13, 18–25], and topological phase transitions [26–30] as well as phases in experimental data [31, 32]. ML not only finds expected phases but also does it at a much lower computational cost, e.g., using fewer samples or smaller system sizes [20, 22].

On the other hand, there are still open questions of ML struggling with topological models and many-body localization (MBL). These problems include the need for pre-engineered features [33–35], disagreement of predicted critical exponents [20], and high sensitivity to hyperparameters describing the training process [22]. Moreover, even in the models described by Landau's theory, so far, these approaches have mostly enabled only the recovery of known phase diagrams or the location of phase transitions in qualitative agreement with more conventional methods based, for instance, on order parameters or theory of finite-size scaling. Most importantly, however, the resulting models are mostly black boxes, i.e., systems with internal logic not obvious at all to a user [36]. The missing key element is the model interpretability, i.e., the ability to be explained or presented to a human in understandable terms [37]. Without this property, we cannot learn anything new from the ML model when applying it to unknown physical systems, nor understand its problems with capturing the topological or MBL signatures. Physicists have already stressed this need for interpretation, but proposed methods are either restricted to linear and kernel models [17, 23, 38–41] or the particular NN's architecture [42, 43] or require pre-engineering of the data, being as a result specific to both the ML and physical model [44].

Hence, in this work, we address the need for interpretability of ML models used in physics on the example of the fundamental one-dimensional (1D) Fermi–Hubbard model. We follow a paradigm without relying on the *a priori* knowledge on the order parameter or the system itself, with an approach that is straightforwardly applicable to any physical model or experimental data with no dependence on the architecture of the ML model. We show how the interpretability method, called influence functions, can be used in the quantum phase classification to understand what characteristics are learned by an ML algorithm without, however, providing the order parameter explicitly. This universal approach unravels if an NN indeed learned a relevant physical concept or cannot be trusted. We also present how an interpretable NN can give additional information on the transition, not provided to the algorithm explicitly.

2. Methods

2.1. Supervised learning

We consider supervised learning problems with labeled training data $\mathcal{D} = \{z_i\}_{i=0}^{n}$, where $z_i = (x_i, y_i)$. The input data is coming from some input space $x_i \in \mathcal{X}$, and the model predicts the outputs coming from some output space $y_i \in \mathcal{Y}$. In our setup, the inputs x_i are the state vectors for a given physical system, and y_i are the corresponding phase labels. The model is determined by the set of parameters θ. In the training process, the parameters' space is being searched for the final parameters $\hat{\theta}_{\mathcal{D}} \equiv \hat{\theta}$ of the ML model, which minimize the training loss function $\mathfrak{L}(\mathcal{D}, \theta) = \frac{1}{n}\sum_{z\in\mathcal{D}}\mathcal{L}(z, \theta)$. The training data set size, n, tends to be of the order of thousands. After training, a model can make a prediction for an unseen test point, z_{test}, with the test loss function value, $\mathcal{L}(z_{\text{test}}, \hat{\theta})$, related to the model certainty of this prediction.

2.2. Interpreting neural networks

An intuitive way of unraveling the logic learned by the machine is retraining the model after removing a single training point z_{r} (starting from the same minimum, if a non-convex problem is analyzed), and checking how it changes the prediction of a specific test point z_{test}. Such a leave-one-out training (LOO) [45] studies the change of the parameters θ, now shifted to a new minimum $\hat{\theta}_{\mathcal{D}\setminus\{z_{\text{r}}\}}$ of the loss function, as depicted in figure 1(a). The largest test loss changes, $\Delta\mathcal{L} \equiv \mathcal{L}(z_{\text{test}}, \hat{\theta}) - \mathcal{L}(z_{\text{test}}, \hat{\theta}_{\mathcal{D}\setminus\{z_{\text{r}}\}})$, indicate the most influential training points for a given test point z_{test} being the ones whose removal causes the largest change. Influential examples can be both helpful ($\Delta\mathcal{L} > 0$) and harmful ($\Delta\mathcal{L} < 0$). Such an analysis gives the notion of a similarity used by the machine in a given problem, as training points being the closest in the $\Delta\mathcal{L}$ space can be understood as the most similar. Once the most influential points are indicated, we can decode what characteristics are looked at by comparing 'similar' points in the machine's 'understanding'. It can be especially useful in phase classification problems where the analysis of $\Delta\mathcal{L}$ enables the recovery of patterns being crucial for distinguishing the phases. However, this technique is prohibitively expensive, as the model needs retraining for each removed z.

To circumvent this problem, one can make a Taylor expansion of the loss function $\mathcal{L}$ w.r.t. the parameters around the minimum $\hat{\theta}$ and approximate $\Delta\mathcal{L}$ resulting from the LOO training, as presented in figure 1(a). This method was proposed for regression problems already forty years ago [45–47] and named influence functions. This interpretability method is not only computationally feasible but also correctly treats a model as a function of training data. The influence function reads

$$\mathcal{I}(z_{\text{r}}, z_{\text{test}}) = \frac{1}{n}\nabla_\theta\mathcal{L}(z_{\text{test}}, \hat{\theta})^T H_\theta^{-1}(\hat{\theta})\nabla_\theta\mathcal{L}(z_{\text{r}}, \hat{\theta}),$$

and it estimates $\Delta\mathcal{L}$ for a chosen test point z_{test} after the removal of a chosen training point z_{r}. $\nabla_\theta\mathcal{L}(z_{\text{test}}, \hat{\theta})$ is the gradient of the loss function of the single test point, $\nabla_\theta\mathcal{L}(z_{\text{r}}, \hat{\theta})$ is the gradient of the loss function of the single training point whose removal's impact is being approximated, and $H_\theta^{-1}(\hat{\theta})$ is the inverse of Hessian, $H_{i,j}(\hat{\theta}) = \frac{\partial^2}{\partial_{\theta_i}\partial_{\theta_j}}\mathfrak{L}(\mathcal{D}, \theta)|_{\theta=\hat{\theta}}$. All derivatives are calculated w.r.t. the model parameters θ, evaluated at $\hat{\theta}$ corresponding to the minimum of the loss, $\mathfrak{L}(\mathcal{D}, \hat{\theta})$. We can only ensure the existence of the inverse of the Hessian if it is positive-definite. It is rarely the case with more sophisticated ML models such as NNs, whose loss landscape is highly non-convex and whose local minima are dominantly flat [48]. However, Koh *et al*

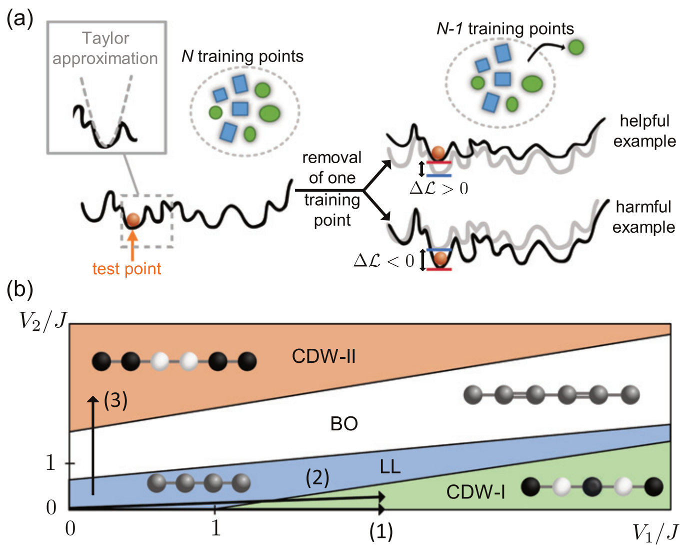

Figure 1. (a) A visual explanation of leave-one-out training and its approximation, the influence function. (b) Schematic phase diagram of the extended one-dimensional half-filled spinless Fermi–Hubbard model with the schemes of the corresponding states: LL—Luttinger liquid, BO—bond order, CDW-I and II—charge density wave type I and II. The arrows indicate the transitions studied in this work.

showed that this method can be generalized to such minima and therefore applied to ML [49, 50]. The example code can be found in Ref. [51].

2.3. Physical model

We apply influence functions to a small CNN (see appendix B for the architecture) trained to recognize phases in the extended Hubbard model, namely a 1D system consisting of spinless fermions at half-filling. The Hubbard models are of fundamental importance to the condensed-matter physics, with the two-dimensional Fermi–Hubbard model believed to describe the high-temperature superconductivity of cuprates [52]. The chosen 1D system has the advantage of being within the power of efficient numerical simulations. As a result, it has a rich and well-studied phase diagram [53, 54] and is a promising candidate to be simulated in quantum simulator [52]. As such, it is suitable to benchmark the influence functions (or any interpretability method) in phase classification problems. In this model, fermions hop between neighboring sites with amplitudes J and interact with nearest neighbors with strength V_1 and next-nearest neighbors with strength V_2

$$\hat{H} = -J\sum_{\langle i,j\rangle} c_i^\dagger c_j + V_1\sum_{\langle i,j\rangle} n_i n_j + V_2 \sum_{\langle\langle i,j\rangle\rangle} n_i n_j. \tag{1}$$

The competition between the system parameters J, V_1, and V_2 leads to four different phases: gapless Luttinger liquid (LL), two gapped charge-density-wave phases with density patterns 1010 (CDW-I) and 11001100 (CDW-II), and bond-order (BO) phase, as seen in figure 1(b). The order parameter describing the transition to the CDW-I (-II) phase is the average difference between (next-)nearest-site densities. Staggered effective hopping amplitudes characterize the BO phase. We feed the CNN with ground states expressed in the Fock basis, labeled with their appropriate phases, calculated for a 12-site system (see appendix A for the details). The hopping amplitude, J, is set to 1 throughout the paper.

3. Results

3.1. Transition between LL and CDW-I

We train a CNN to classify ground states into two phases: LL and CDW-I based on the transition line marked with the arrow (1) in figure 1(b) for $V_2 = 0$. We plot the influence functions of all training examples for a chosen test point (marked with orange line) in figure 2. The order parameter describing the transition here is the average difference between nearest-site densities, which is zero in the LL phase and non-zero (growing to one) in the CDW-I phase.

The panels (a) and (b) present how influential training points are for test points from the LL phase. The test state (a) is the ground state located deeply in the LL phase, while (b) is closer to the transition. If the CNN learns an order parameter, all training points, i.e., ground states from the LL phase exhibiting a zero

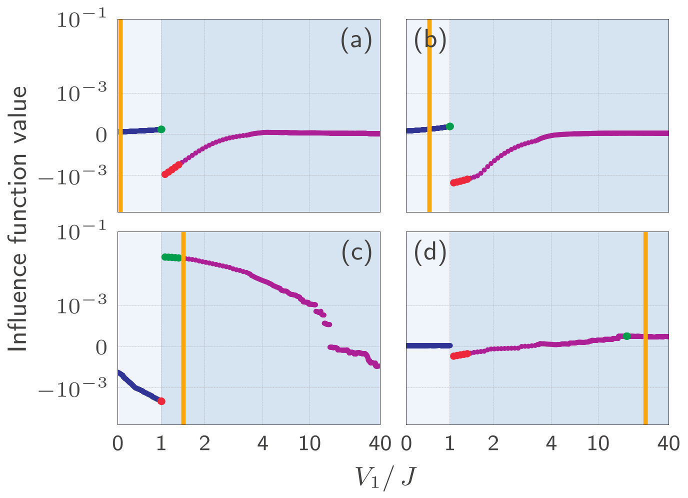

Figure 2. Influence functions of all training examples, i.e., ground states calculated for the transition line between LL and CDW-I for $V_2 = 0$, marked with dots, for chosen test points marked with an orange line. Blue (purple) dots are influence function values for training examples from the LL (CDW-I) phase. Larger green (red) dots are five the most influential helpful (harmful) training examples. Different background shades indicate two phases. (a) and (b) Blue training points from the LL phase are similarly influential to the classification of the test point from the same phase. They all are characterized by a zero order parameter. (c) and (d) The most helpful training examples for the classification of the test points from the CDW-I phase are the ones with the most similar order parameter. Note the use of symmetric log scale.

order parameter, should be similarly positively influential, and that is precisely what we observe. They form an almost flat line in panels (a) and (b). For both test points (a) and (b) from the LL phase, the most harmful training points are the ones closest to the transition, but on the CDW-I side. These states are the most similar (with the smallest order parameter value), but already labeled differently.

A careful reader can notice that if the CNN learns an order parameter, the training points from the LL phase, all exhibiting a zero order parameter, should be similarly influential and form a flat line in all the panels of figure 2. However, we see that in reality, their influence changes linearly, what panel (c) shows especially well. This divergence from expected behavior is mostly due to numerical reasons, and we discuss it in appendix A.

On the side of the CDW-I phase, the influence pattern is significantly different. The curvature of influential points corresponds to the growth of the order parameter, and the most influential helpful points are the ones closest to the test point in the order parameter space, slightly shifted towards the transition point, as they provide more information. Panel (c) shows the influence functions of training points for the test states on the CDW-I side, close to the transition. The most harmful examples are, as in the previous test points, the ones closest to the transition, but on its other side. However, panel (d) presents a distinct behavior of the most harmful examples being in the same phase. All the training points are similarly influential with small values of influence functions resulting in the almost flat line. It is a signature of the CNN's high certainty regarding the prediction made in panel (d) manifesting with a small test loss function $\mathcal{L}(z_{\text{test}}, \hat{\theta})$. Also, the analyzed test point is deeply in the CDW-I phase, with all neighboring states being almost identical with the order parameter close to 1. The most harmful examples are the ones we label as the CDW-I phase, but very different, so the ones closest to the transition.

While analyzing the figures, it is vital to keep in mind that we do not explicitly provide any information on the nearest-neighbor interaction, V_1/J, present on the x-axis (or any physical parameters, in general). We provide the input states in the random order. Therefore, the smooth patterns created by the influence functions and resulting ordering of training points, especially on the CDW-I phase's side, is the sole consequence of the internal analysis of the states by the machine.

3.2. Transfer learning

With a similar approach, we validate the transfer learning to another transition line. We take the trained CNN from figure 2, and in figure 3 we apply it to test states coming from the transition line for $V_2 = 0.25V_1$, where the next-nearest-neighbor interaction shifts phase transition to higher values of V_1/J. Therefore the training and test states come from different transition lines, $V_2 = 0$ and $0.25V_1$, marked in figure 1(b) with the arrows (1) and (2), respectively. Notice the shift of the panels' backgrounds as compared to figure 2. They mark two phases of the test transition line, having a different transition point ($V_1/J = 1.85$) than the training transition line ($V_1/J = 1$).

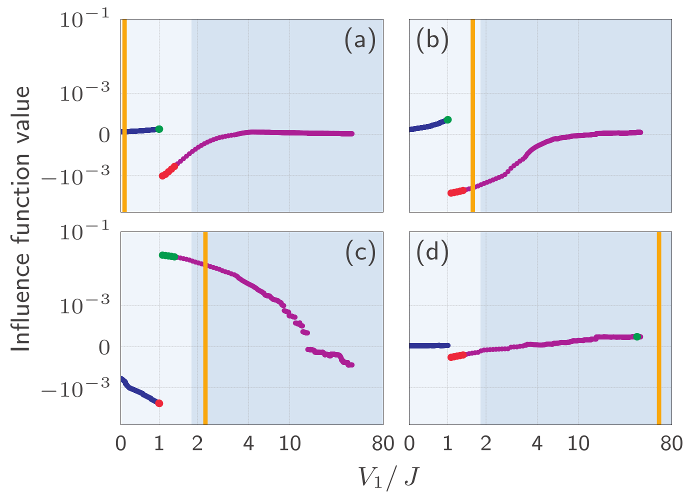

Figure 3. Influence functions of all training examples, i.e., ground states calculated for the transition line between LL and CDW-I for $V_2 = 0$, marked with dots, for chosen test states from transition line for $V_2 = 0.25V_1$ marked with an orange line. Blue (purple) dots are influence function values for training examples from the LL (CDW-I) phase for $V_2 = 0$. Larger green (red) dots are five the most influential helpful (harmful) training examples. Different background shades indicate the phase transition for $V_2 = 0.25V_1$ line, from which test states come from. Figure 3 shows very similar patterns as ones in figure 2, but shifted. It indicates that the similarity of test and training points is connected to their order parameters, as the order parameter of test points is shifted towards larger V_1/J values, compared to training points, due to coming from the $V_2 = 0.25V_1$ transition line. Note the use of symmetric log scale.

Panels (a) and (b) of figure 3 show the influence function values of training data for test states from the LL phase, while (c) and (d)—from the CDW-I phase. Panel (a) is identical to the panel (a) of figure 2, but already panel (b) shows an interesting divergence from the figure 2(b), being a result of a shifted transition point of the test line compared to the training line. No longer the same value of V_1/J, for which test and training states were calculated, yields the same order parameter for both of them. For example, the test state being in the LL phase, close to the transition point for $V_2 = 0.25V_1$ should be the most similar to the training points from the LL phase, close to the transition point $V_2 = 0$, and have the most similar order parameter. The ML algorithm follows this similarity with regards to the order parameter, what implies a successful transfer learning scheme. We see similar behavior in panel (c), where the most helpful points are also shifted as compared to figure 2(c).

3.3. Inferring the existence of the third phase

This time we analyze the transition line crossing three phases, LL, BO, and CDW-II, which is indicated by the arrow (3) in figure 1(b). Two order parameters describe this transition. One is the average difference of the next-nearest neighbor density, which equals zero in the LL and BO phases, and grows to 1 in the CDW-II phase. The other is the staggering of effective nearest-neighbor hoppings, being 0 in the LL phase, non-zero in the BO phase, and slowly decaying to 0 in the CDW-II phase. In the studied range of parameters, two phases (BO and CDW-II) co-exist (see appendix A for the details). It is crucial to note that in this section, we train on the mentioned transition line crossing three phases, but we label ground states only as belonging to one out of two phases.

In the first set-up, with results presented in the panels (a) and (b) of figure 4, we label ground states as belonging to the LL (blue dots, label 0) or the BO and CDW-II phases (purple dots, label 1). Independently on the test point location, notice two similarity regions within purple training points. Apparently the NN learns two different patterns (order parameters) to classify the data correctly. Therefore, it notices the existence of the third phase within the incorrectly labeled data. Inferring the third phase would be impossible without interpretability methods, which in this sense pave the way towards unknown phases detection.

The second set-up consists of labeling the same data as belonging to the LL and BO phases (blue dots, label 0) or the CDW-II phase (purple dots, label 1). The influence functions' values, resulting from this classification, are in the panels (c) and (d) of figure 4. The pattern they form is starkly different. First of all, there is no additional similarity region within training points from the LL and BO. The behavior is then more similar to the one seen in figure 2 with the transition between LL and CDW-I. It is not identical, though, as in the phase LL + BO the most helpful training points are always distributed randomly, but deep in the LL phase, avoiding the BO phase. The most helpful points on the CDW-II side are deep in the CDW-II phase in contrast to figure 2, where they mostly follow the test point. Consider, that the deeper the

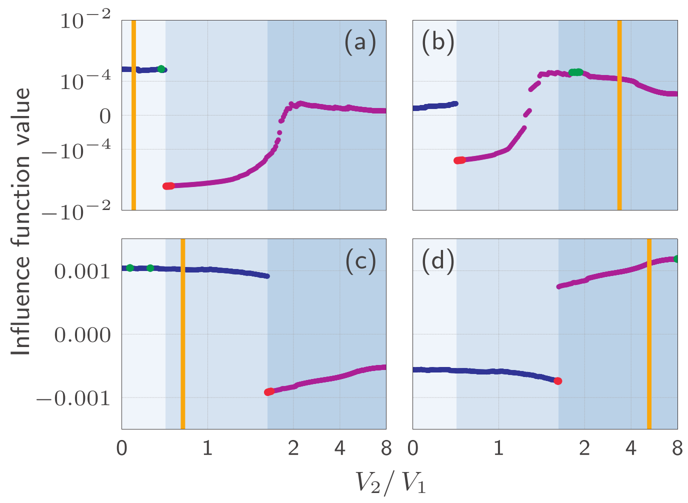

Figure 4. Influence functions of all training examples, i.e., ground states calculated for the transition line crossing LL, BO, and CDW-II for $V_1/J = 1$, for chosen test points marked with an orange line. Training examples are marked with dots and labeled differently within two rows, i.e., as (a) and (b) LL—not LL and (c) and (d) CDW-II—not CDW-II. Blue dots are influence function values for training examples from (a) and (b) the LL phase, (c) and (d) the LL and BO phases, while purple ones (a) and (b) from the BO and CDW-II phases, (c) and (d) from the CDW-II phase. Larger green (red) dots are five the most influential helpful (harmful) training examples. Different background shades indicate phase transitions. (a) and (b) CNN classifies states as belonging to the LL phase or not and detects two similarity regions in the 'not-LL phase'. It effectively indicates the existence of an additional phase. (c) and (d) The model exhibits overfitting. Note the use of a symmetric log scale, except for the linear y axis in panels (c) and (d).

CDW-II phase, the smaller the BO order parameter, what makes CDW-II predictions easier. The observed pattern is the example of NN not learning correctly the order parameter and potentially overfitting.

Finally, we trained a CNN on the same data, but with three labels correctly corresponding to all three phases. The influence patterns resemble those seen in figure 2 and panels (c) and (d) of figure 4, indicating that CNN correctly learns both appropriate order parameters.

4. Conclusions

We used the interpretability method called influence functions on the CNN trained in a supervised way to classify ground states of the extended 1D half-filled spinless Fermi–Hubbard model. We provided strong evidence that the ML algorithm learned a relevant order parameter describing the quantum phase transition. If no knowledge on the actual order parameter were available, influence functions' values would guide the search for patterns responsible for phase transition and help extract a relevant order parameter, however not providing it explicitly. We showed that the influence functions, applied to the trained NN, were able to detect an unknown phase. Two aspects impacted which training points were the most important for a given test point: how similar they were to the test state and how unique within the training data set. Together they gave a notion of distance or similarity used by the CNN in the phase classification problem and indicated that the patterns relevant for the predictions coincided with the order parameters.

Our approach may be used to address open problems of topological models and MBL with NNs, whose logic can be finally discovered by influence functions. They may be easily applied to any physical model in general. Influence functions should be very successful at distinguishing between types of phase transitions. In particular, the curvature of the line drawn by influence functions' values should be different for the transitions characterized by continuous and discontinuous change of the order parameter. Moreover, this tool proved to be very sensitive to outliers existing in the data set and may serve for anomaly detection. Finally, along with unsupervised learning techniques, it can serve as the first search for unknown phases and order parameters in experimental data.

Acknowledgments

AnD acknowledges the financial support from the National Science Centre, Poland, within the Preludium Grant No. 2019/33/N/ST2/03123. This project has also received funding from the European Unions Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 665884 (PH). MT acknowledges the financial support from the Foundation for Polish Science within the

Homing and First Team programmes co-financed by the EU Regional Development Fund. We (ML group) also acknowledge the Spanish Ministry of Economy and Competitiveness (Plan National FISICATEAMO and FIDEUA PID2019-106901GB-I00/10.13039 / 501100011033, "Severo Ochoa" program for Centres of Excellence in R&D (CEX2019-000910-S), FPI, FIS2020-TRANQI), European Social Fund, Fundació Privada Cellex, Fundació Mir-Puig, Generalitat de Catalunya (AGAUR Grant No. 2017 SGR 1341, CERCA program, QuantumCAT U16-011424, co-funded by ERDF Operational Program of Catalonia 2014-2020), ERC AdG NOQIA, MINECO-EU QUANTERA MAQS (funded by State Research Agency (AEI) PCI2019-111828-2/10.13039/501100011033), and the National Science Centre, Poland-Symfonia Grant No. 2016/20/W/ST4/00314. AlD acknowledges the Juan de la Cierva program (IJCI-2017-33180) and the financial support from a fellowship granted by la Caixa Foundation (ID 100010434, fellowship code LCF/BQ/PR20/11770012).

Appendix A. Phase diagram of the extended one-dimensional half-filled spinless Fermi–Hubbard model

We study the one-dimensional system consisting of spinless fermions at half-filling with hopping between neighboring sites with amplitudes J, interacting with nearest neighbors with strength V_1 and next-nearest neighbors with strength V_2, described with Hamiltonian 1. The model exhibits four different phases, two of them co-exist in the limited range of parameters. Without the next-nearest-neighbor interaction, V_2, the system can follow only patterns of the gapless liquid Luttinger (algebraic) phase (LL) or the charge-density wave of type I (CDW-I) with the degenerated density pattern 101 010. The CDW-I order parameter describing this transition reads $O_{\text{CDW-I}} = \frac{1}{L}\sum_{\langle i,j\rangle}|n_i - n_j|$, where $\langle\rangle$ symbolizes nearest neighbors. The next-nearest-neighbor interaction, V_2 competes with V_1, so for non-zero V_2 but still smaller than V_1 the transition between LL and CDW-I shifts towards bigger V_1. For sufficiently strong V_2 the BO phase emerges with the order parameter $O_{\text{BO}} = \frac{1}{L}\sum_i(-1)^i B_i$, where $B_i = \left\langle c_i^\dagger c_{i+1} + c_{i+1}^\dagger c_i\right\rangle$. It turns into the charge-density wave of the type II (CDW-II) with the degenerated density pattern 11 001 100 for large V_2 values, with $O_{\text{CDW-II}} = \frac{1}{L}\sum_{\langle\langle i,j\rangle\rangle}|n_i - n_j|$, where $\langle\langle\rangle\rangle$ symbolizes next-nearest neighbors.

To calculate the ground states and order parameters of the model, we use QuSpin package [55] to write the Hamiltonian for a 12-site system in the Fock basis, resulting in 924 basis states. We assume periodic boundary conditions. We perform the exact diagonalization with the SciPy package [56]. The ground states belonging to BO, CDW-I, and II phases are degenerated. To lift the degeneracy of the ground state, we apply symmetry breaking (guiding) fields favoring one of the patterns.

This approach results in the order parameters in the LL being not exactly constant and equal to zero. Instead, their values are growing very slowly when approaching the transition points. Therefore, there is no exact transition point, so we define it as such parameters of the system that correspond to the order parameter being ten times bigger than the corresponding symmetry breaking fields. Due to the guiding fields of values 10^{-7}, 10^{-5}, and 10^{-4} for 101 010 and 11 001 100 density patterns and 1010 hopping pattern, respectively, the order parameters of values 10^{-6}, 10^{-4}, and 10^{-3} signal the transition to the CDW-I, CDW-II, and BO phase, respectively.

The non-zero order parameter in the uniform phase and numerical arbitrariness of choosing the transition points are the main reasons why the influence functions' values in the LL phase, seen in figures 2 and 3, and panels (a) and (b) of figure 4 of the manuscript are not precisely the same. The third reason is the finite-size effects. As the order parameters in the LL phase are growing very slowly, finally, the most helpful points are the ones near the transition—they are also the most unique from the training points labeled as LL, and the information they provide is the most valuable. In the perfect scenario (observed, for example, for training on states obtained from mean-field calculations), the five most influential points randomly distribute over the whole LL phase.

It is interesting to note that the results presented in this work stay the same without the symmetry-breaking fields and do not depend on the size of the system.

Within this work, we train the convolutional NN on three transition lines indicated with arrows (1)–(3) in figure 1(b). The first transition line leads from the LL to the CDW-I phase. We calculate it for a constant $V_2 = 0$ and $V_1/J = \langle 0, 40\rangle$. It is a source of training data for both figures 2 and 3, and test data for figure 2. It is symbolized in figure 1(b) with the arrow (1), and the values of corresponding order parameter $O_{\text{CDW-I}}$ are plotted in figure 5(a). The transition, defined as above, occurs for $V_1/J = 1$. The second transition line is calculated for $V_2 = 0.25V_1$ and $V_1/J = \langle 0, 80\rangle$. Indicated with the arrow (2), it is the source of test data for figure 3 of the main manuscript. We plot the corresponding order parameter CDW-I in figure 5(b), and the transition takes place for $V_1/J = 1.85$. The final transition line cuts three phases: LL, BO, and CDW-II. It is marked with the arrow (3) and provides both training and test data for figure 4 of the main

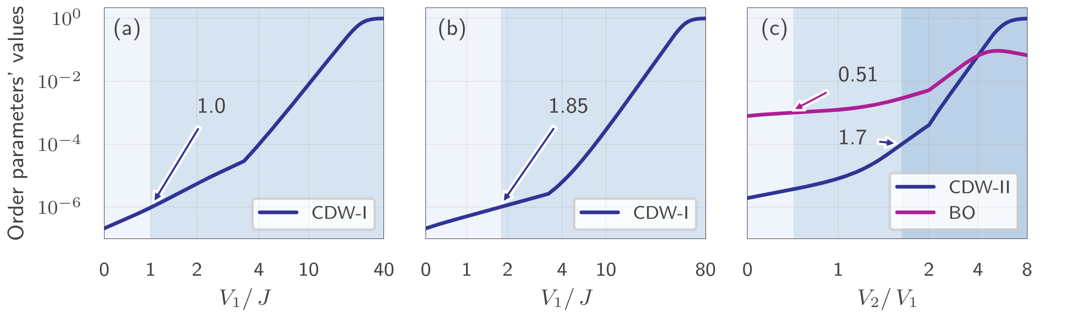

Figure 5. Corresponding order parameters' values for three transition lines studied within this work, indicated with arrows (1)–(3) in figure 1(b). (a) and (b) CDW-I order parameter for the transition line between the LL and the CDW-I phase for $V_2 = 0$ and $0.25V_1$, respectively. (c) CDW-II and BO order parameters for the transition line between LL, BO, and CDW-II for $V_1 = 1J$. Note the logarithmic scale of y-axis, and the symmetric log scale of x-axis with threshold points chosen to be 3, 3, and 2, respectively. Cusps in the lines are artificial and result from the symmetric log scale of x-axis.

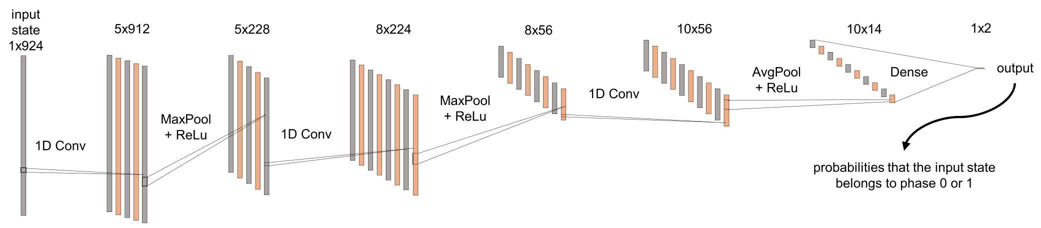

Figure 6. Scheme of used architecture. Length scale does not apply.

manuscript. It is calculated for constant $V_1 = 1/J$ and $V_2 = \langle 0, 8\rangle V_1$. Transition between LL and BO occurs for $V_2 = 0.51V_1$, and between BO and CDW-II for $V_2 = 1.7V_1$. It is important to notice that for the chosen range of parameters $V_2 = \langle 1.7, 8\rangle V_1$, two phases co-exist what can be seen in figure 5(c).

Appendix B. Convolutional neural network

We use a NN (see figure 6) consisting of 3 one-dimensional convolutional layers with five filters on the input vector, eight filters on the first hidden layer, and ten filters for the last convolution layer. After the first two convolutions, we apply a max-pooling layer to reduce the dimension, and the last convolutional layer is followed by an average pooling layer. Finally we have one fully connected layer with two output neurons that predict the labels. When designing the architecture, we make sure that the convolutional part contains a large part of the NN's parameters. For the training of the NN, we use state vectors from each phase as input and label them with 0 or 1 for each phase. We obtain the state vectors via the exact diagonalization of the Hamiltonian 1.

We use L_2 regularization during the training to effectively decrease the certainty of the NN's predictions. The undertrained NN with imperfect accuracy can provide better intuition behind the problem than overtrained one, whose predictions are impacted by overfitting. Used CNNs had accuracy between 89 and 96%.

ORCID iDs

Anna Dawid https://orcid.org/0000-0001-9498-1732
Patrick Huembeli https://orcid.org/0000-0001-7047-6897
Michal Tomza https://orcid.org/0000-0003-1792-8043
Maciej Lewenstein https://orcid.org/0000-0002-0210-7800
Alexandre Dauphin https://orcid.org/0000-0003-4996-2561

References

[1] Behler J and Parrinello M 2007 *Phys. Rev. Lett.* **98** 146401
[2] Ward L and Wolverton C 2016 *Curr. Opin. Solid State Mater. Sci.* **21** 167
[3] Christiansen E M *et al* 2018 *Cell* **173** 792
[4] Wong D and Yip S 2018 *Nature* **555** 446
[5] Carleo G, Cirac I, Cranmer K, Daudet L, Schuld M, Tishby N, Vogt-Maranto L and Zdeborová L 2019 *Rev. Mod. Phys.* **91** 045002
[6] Naul B, Bloom J S, Pérez F and van der Walt S 2018 *Nat. Astron.* **2** 151
[7] Baldi P, Sadowski P and Whiteson D 2014 *Nat. Commun.* **5** 4308
[8] Torlai G, Mazzola G, Carrasquilla J, Troyer M, Melko R and Carleo G 2018 *Nat. Phys.* **14** 447
[9] Carrasquilla J, Torlai G, Melko R G and Aolita L 2019 *Nat. Mach. Intell.* **1** 155
[10] Torlai G and Melko R G 2018 *Phys. Rev. Lett.* **120** 240503
[11] Bukov M, Day A G R, Sels D, Weinberg P, Polkovnikov A and Mehta P 2018 *Phys. Rev.* X **8** 031086
[12] Carrasquilla J and Melko R G 2017 *Nat. Phys.* **13** 431
[13] van Nieuwenburg E P L, Liu Y-H and Huber S D 2017 *Nat. Phys.* **13** 435
[14] Schäfer F and Lörch N 2019 *Phys. Rev.* E **99** 062107
[15] Tanaka A and Tomiya A 2017 *J. Phys. Soc. Jpn.* **86** 063001
[16] Li C-D, Tan D-R and Jiang F-J 2018 *Ann. Phys., NY* **391** 312
[17] Wang L 2016 *Phys. Rev.* B **94** 195105
[18] Liu Y-H and van Nieuwenburg E P L 2018 *Phys. Rev. Lett.* **120** 176401
[19] Broecker P, Carrasquilla J, Melko R G and Trebst S 2017 *Sci. Rep.* **7** 8823
[20] Huembeli P, Dauphin A, Wittek P and Gogolin C 2019 *Phys. Rev.* B **99** 104106
[21] Ch'ng K, Vazquez N and Khatami E 2018 *Phys. Rev.* E **97** 013306
[22] Théveniaut H and Alet F 2019 *Phys. Rev.* B **100** 224202
[23] Wetzel S J 2017 *Phys. Rev.* E **96** 022140
[24] Kottmann K, Huembeli P, Lewenstein M and Acin A 2020 *Phys. Rev. Lett.* **125** 170603
[25] Vargas-Hernández R A, Sous J, Berciu M and Krems R V 2018 *Phys. Rev. Lett.* **121** 255702
[26] Deng D-L, Li X and Das Sarma S 2017 *Phys. Rev.* X **7** 021021
[27] Huembeli P, Dauphin A and Wittek P 2018 *Phys. Rev.* B **97** 134109
[28] Zhang P, Shen H and Zhai H 2018 *Phys. Rev. Lett.* **120** 066401
[29] Tsai Y-H, Yu M-Z, Hsu Y-H and Chung M-C 2020 *Phys. Rev.* B **102** 054512
[30] Greplova E, Valenti A, Boschung G, Schäfer F, Lörch N and Huber S D 2020 *New J. Phys.* **22** 045003
[31] Rem B S, Käming N, Tarnowski M, Asteria L, Fläschner N, Becker C, Sengstock K and Weitenberg C 2019 *Nat. Phys.* **15** 917–20
[32] Khatami E, Guardado-Sanchez E, Spar B M, Carrasquilla J F, Bakr W S and Scalettar R T 2020 *Phys. Rev.* A **102** 033326
[33] Zhang Y and Kim E-A 2017 *Phys. Rev. Lett.* **118** 216401
[34] Beach M J S, Golubeva A and Melko R G 2018 *Phys. Rev.* B **97** 045207
[35] Richter-Laskowska M, Khan H, Trivedi N and Maśka M M 2018 *Condens. Matter Phys.* **21** 33602
[36] Guidotti R, Monreale A, Ruggieri S, Turini F, Pedreschi D and Giannotti F 2018 arXiv:1802.01933
[37] Doshi-Velez F and Kim B 2017 arXiv:1702.08608
[38] Ponte P and Melko R G 2017 *Phys. Rev.* B **96** 205146
[39] Zhang W, Wang L and Wang Z 2019 *Phys. Rev.* B **99** 054208
[40] Greitemann J, Liu K, Jaubert L D C, Yan H, Shannon N and Pollet L 2019 *Phys. Rev.* B **100** 174408
[41] Greitemann J, Liu K and Pollet L 2020 arXiv:2007.01685
[42] Wetzel S J and Scherzer M 2017 *Phys. Rev.* B **96** 184410
[43] Wetzel S J, Melko R G, Scott J, Panju M and Ganesh V 2020 *Phys. Rev. Res.* **2** 033499
[44] Zhang Y, Ginsparg P and Kim E-A 2020 *Phys. Rev. Res.* **2** 023283
[45] Cook R D 1977 *Technometrics* **19** 15
[46] Cook R D and Weisberg S 1980 *Technometrics* **22** 495
[47] Cook R D and Weisberg S 1982 *Residuals and Influence in Regression* (London: Chapman and Hall)
[48] Sagun L, Bottou L and LeCun Y 2017 arXiv:1611.07476v2
[49] Koh P W and Liang P 2017 arXiv:1703.04730
[50] Koh P W, Ang K-S, Teo H H K and Liang P 2019 arXiv:1905.13289
[51] Dawid A, Huembeli P, Tomza M, Lewenstein M and Dauphin A 2020 Shmoo137/Interpretable-Phase-Classification: Journal submission (Version arXiv1.1) Zenodo http://doi.org/10.5281/zenodo.3759432
[52] Dutta O, Gajda M, Hauke P, Lewenstein M, Lühmann D-S, Malomed B A, Sowiński T and Zakrzewski J 2015 *Rep. Prog. Phys.* **78** 066001
[53] Hallberg K, Gagliano E and Balseiro C 1990 *Phys. Rev.* B **41** 9474
[54] Mishra T, Carrasquilla J and Rigol M 2011 *Phys. Rev.* B **84** 115135
[55] Weinberg P and Bukov M 2017 *SciPost Phys.* **2** 003
[56] Virtanen P *et al* 2020 *Nat. Methods* **17** 261