arXiv:2303.15298

# The percolating cluster is invisible to image recognition with deep learning

Djénabou Bayo,[1,2,*] Andreas Honecker,[2,†] and Rudolf A. Römer[1,‡]

[1] *Department of Physics, University of Warwick, Coventry, CV4 7AL, United Kingdom*
[2] *Laboratoire de Physique Théorique et Modélisation, CNRS UMR 8089, CY Cergy Paris Université, Cergy-Pontoise, France*

(Dated: March 27, 2023)

We study the two-dimensional site-percolation model on a square lattice. In this paradigmatic model, sites are randomly occupied with probability $p$; a second-order phase transition from a non-percolating to a fully percolating phase appears at occupation density $p_c$, called percolation threshold. Through supervised deep learning approaches like classification and regression, we show that standard convolutional neural networks (CNNs), known to work well in similar image recognition tasks, can identify $p_c$ and indeed classify the states of a percolation lattice according to their $p$ content or predict their $p$ value via regression. When using instead of $p$ the spatial cluster correlation length $\xi$ as labels, the recognition is beginning to falter. Finally, we show that the same network struggles to detect the presence of a spanning cluster. Rather, predictive power seems lost and the absence or presence of a *global* spanning cluster is not noticed by a CNN with *local* convolutional kernel. Since the existence of such a spanning cluster is at the heart of the percolation problem, our results suggest that CNNs require careful application when used in physics, particularly when encountering less-explored situations.

## I. INTRODUCTION

Convolution neural nets (CNN) are a class of deep, i.e., multi-layered, neural nets (DNNs) in which spatial locality of data values is retained during training in a machine learning (ML) setting. When coupled with a form of residual learning [1], the resulting residual networks (ResNets) have been shown to allow astonishing precision when classifying images, e.g., of animals [2] and handwritten characters [3], or when predicting numerical values, e.g., of market prices [4]. In recent years, we also witnessed the emergence of DNN techniques in several fields of physics as a new tool for data analysis [5–8]. In condensed matter physics in particular, DNN and CNN proved to be performing well in identifying and classifying phases of material states [9–12].

Despite all these studies, the ML process in itself tends to be somewhat a black box, and it is yet not known what is allowing a DNN to correctly identify a phase. In order to gain further insight into this issue, we choose a well-known and well-studied system exhibiting perhaps the simplest of all second-order phase transitions, the *site-percolation* model in two spatial dimensions [13, 14]. In this model, a cluster spanning throughout the system emerges at an occupation probability $p_c$, leading to a non-spanning phase when $p < p_c$ while $p \geq p_c$ corresponds to the phase with at least one such spanning cluster [14]. Several ML studies on the percolation model have been already published, mostly using *supervised* learning in order to identify the two phases via ML classification [15, 16]. An estimate of the critical exponent, $\nu$, of the percolation transition has also been given [15]. The task of determining $p_c$ was further used to evaluate different ML regression techniques in Ref. [17]. For *unsupervised* and *generative* learning, less work has been done [16, 18]. While some successes have been reported [18], other works show the complexities involved when trying to predict percolation states [16].

In this work, we start by replicating some of the supervised DL analyses. We find that CNNs usually employed in image recognition ML tasks also work very well for classifying percolation states according to $p$ as well as for regression when determining $p$ from such states. The results are less convincing when instead of $p$, we use the spatial correlation lengths $\xi$ as an alternative means to characterize the phase boundary. We find that, even when correcting for probable difficulties due to non-balanced data availability for $\xi$, classification and regression tasks fail to give acceptably diagonal confusion matrices. Crucially, when analyzing in detail whether spanning clusters $< p_c$ or non-spanning clusters $> p_c$ are correctly identified, we find the CNNs that performed so well in the initial image recognition tasks now consistently fail to reflect the ground truth. Rather, it appears that the CNNs use $p$ as a proxy measure to inform their classification predictions — a strategy that is obviously false for the percolation problem. We confirm this conclusion by testing our networks with bespoke test sets that include artificially spanning clusters $< p_c$ or firebreak structures for $> p_c$.

## II. MODEL AND METHODS

### A. The percolation model

The percolation problem is well-known with a rich history across the natural sciences [13, 14, 19–22]. It provides the usual statistical characteristics across a

* Djenabou.Bayo@warwick.ac.uk
† Andreas.Honecker@cyu.fr
‡ R.Roemer@warwick.ac.uk

second-order transition such as, e.g., critical exponents, finite-size scaling, renormalization and universality [14]. Briefly, on a percolation lattice of size $L \times L$, individual lattice sites $\vec{x} = (x, y)$, $x, y \in [1, L]$, are randomly occupied with *occupation probability* $p$ such that the state $\psi$ of site $\vec{x}$ is $\psi(\vec{x}) = 1$ for occupied and $\psi(\vec{x}) = 0$ for unoccupied sites. We say that a connection between neighboring sites exists when these are side-to-side nearest-neighbors on the square lattice, while diagonal sites can never be connected. A group of these connected occupied sites is called a *cluster*. Such a cluster then *percolates* when it spans the whole lattice either vertically from the top of the square to the bottom or, equivalently, horizontally from the left to the right. Obviously, for $p = 0$, all sites are unoccupied and no spanning cluster can exist while for $p = 1$ the spanning cluster trivially extends throughout the lattice. In Fig. 1, we show examples of percolation clusters generated for various $p$ values. The *percolation threshold* is at $p = p_c(L)$, such that for $p < p_c(L)$ most clusters do not span while for $p > p_c(L)$ they do. This can be expressed as the *percolation probability* $P(p) = \langle s_L(p)/L^2 \rangle$, where $s_L(p)$ gives the size of the (largest) percolating cluster for size $L$ and $\langle \cdot \rangle$ denotes an average over many randomly generated realizations. Similarly, we can define a probability of non-percolating, $Q(p) = \langle (L^2 - s_L(p))/L^2 \rangle$ and $P(p) + Q(p) = 1$. For an infinite system ($L \to \infty$), one finds the emergence of an infinite spanning cluster at $p_c = 0.59274605079210(2)$. This estimate has been determined numerically evermore precisely over the preceding decades [23] while no analytical value is yet known [22]. Another quantity often used to characterize the percolation transition is the two-site correlation function $g(r) = \langle \psi(\vec{x})\psi(\vec{x} + \vec{r}) \rangle_{\vec{x},|\vec{r}|=r}$, where the $\langle \cdot \rangle_{\vec{x},|\vec{r}|=r}$ denotes the average over all $\vec{x}$ and directions $|\vec{r}|$. This $g(r)$ measures the probability to have two occupied sites separated by a distance $r$, in the same cluster [14]. The associated correlation length $\xi$ is determined through $\xi = \sqrt{\sum_r r^2 g(r)}/\sum_r g(r)$. In the infinite system $\xi$ diverges at $p_c$ as $|p - p_c|^{-\nu}$, where $\nu = 4/3$ is the critical exponent, determining the universality class of the percolation problem [14].

### B. Generating percolation states for training and validation

In order to facilitate the recognition of percolation with image recognition tools of ML, we have generated finite-sized $L \times L$, with $L = 100$, percolation states, denoted as $\psi_i(p)$, for the 31 $p$-values $0.1, 0.2, \ldots, 0.5, 0.55, 0.555, 0.556, \ldots, 0.655, 0.66, 0.7, \ldots, 0.9$. For each such $p$, $N = 10000$ different random $\psi_i(p)$ have been generated. Each state $\psi_i(p)$, $i = 1, \ldots, N$, is of course just an array of numbers with 0 denoting unoccupied and 1 occupied sites. Nevertheless, we occasionally use for convenience the term "image" to denote $\psi_i(p)$. The well-known Hoshen-Kopelman algorithm [24] is used to identify and label clusters from which we (i) compute $s(p)$, $g(r)$, and $\xi(p)$ as well as (ii) determine the presence or absence of a spanning cluster. In Fig. 1 we show examples of percolation states generated for various $p$ values as well as the extracted $P_{100}$, $Q_{100}$, $p_c(100)$ and $\langle \xi(p) \rangle$ estimates. The different gray scales used in Fig. 1(a) mark the different connected clusters. However, for the ML approach below, we shall only use the numerical values 0 and 1 corresponding to the state $\psi_i(p)$ [25]. This is visualized as the simple black and white version shown, e.g., for $p = 0.5$ in Fig. 1 (a). From Figs. 1 (b+c), we note that $P(p)$ and $\xi(p)$ behave qualitatively as expected [14], with $P(p) \lesssim 1$ for $p < p_c$ and $P(p) \gtrsim 0$ for $p > p_c$ and $\xi(p)$ maximal near $p_c$. Clearly, $p_c(L = 100) \sim 0.585(5) < p_c$. This latter behavior is as expected since $s_L(p) \leq s_\infty(p)$, i.e., a cluster that seemingly spans an $L \times L$ finite square might still not span on an infinite system.

We emphasize that in the construction, we took care to only construct states such that for each $p$, the number of occupied sites is exactly $N_{\rm occ} = p \times L^2$ and hence $p$ can be used as exact label for the supervised learning approach. We note that $p = N_{\rm occ}/L^2$ can therefore also be called the percolation *density*. For the ML results discussed below, it will also be important to note that the spacing between $p$ values reduces when $p$ reaches 0.5 with the next $p$ value given by 0.55 and then 0.555. Similarly, the $p$ spacing increases as 0.655, 0.66, 0.7. We will later see that this results in some deviations from perfect classification/regression. Last, we have also generated a similar test set with $L = 200$. As the ML training cycles naturally take much longer, we have not averaged these over ten independent trainings. We find that our results do not change significantly when using this much larger data set and hence we will refrain from showing any result for these larger states in the following.

### C. Supervised ML strategy for phase prediction

As discussed above, DL neural nets using the power of CNNs are among the preferred approaches when trying to identify phases in condensed matter systems [26–28]. Here, we shall use a ResNet18 [1] network with 17 convolutional and 1 fully-connected layers, pretrained on the ImageNet dataset [29]. As basis of our ML implementation we use the PyTorch suite of ML codes [30]. We train the ResNet18 on the percolation of 310000 states, using a 90%/10% split into training and validation data, $\mathcal{T}$ and $\mathcal{V}$, respectively; this corresponds to $N_{\rm train} = 279000$ and $N_{\rm val} = 31000$ samples, respectively. We concentrate on two supervised ML tasks. First, we *classify* percolation images according to (i) $p$, (ii) $\xi$ as well as (iii) spanning versus non-spanning. In the second task, we aim to predict $p$ and $\xi$ values via ML *regression*. In both tasks, the overall network architecture remains identical, we just adapt the last layer as usual [31]. For the classification the output layers have a number of neurons corresponding to the number of classes trained, i.e., for the classification by density the $\mathcal{C} = 31$ $p$-values

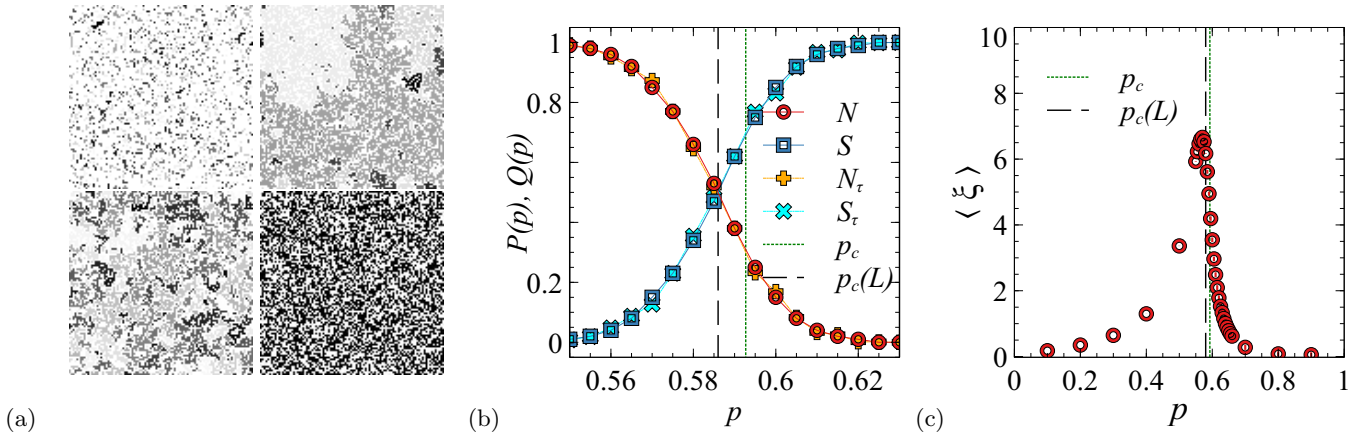


FIG. 1. (a) Examples of percolation clusters of size $L^2 = 100^2$, obtained for $p = 0.2 < p_c$, $0.6 > p_c$ in the top row and $p = 0.5$, i.e. close to $p_c$, in the bottom row. While individual clusters have been highlighted with different gray scales for the first three images, the bottom right image with $p = 0.5$ shows all occupied sites in black only, irrespective of cluster identity. This latter representation is used below for the ML approach. (b) Percolation probabilities $P(p)$ and $Q(p)$ of having a spanning (blue open squares) / non-spanning (red open circles) cluster close to the percolation threshold for dataset $\mathcal{T} \cup \mathcal{V}$. The percolation probability of having a spanning (cyan crosses) / non-spanning (orange plus) cluster close to the percolation threshold for dataset $\tau$. (c) Correlation length $\xi(p)$. In (b+c), the vertical lines indicate the estimates $p_c(100) \sim 0.585(5)$ (dotted) and $p_c$ (dashed).

given above, while for regression the output layer has only one neuron giving the numerical prediction. However, the loss functions are different. Let $\mathbf{w}$ denote the set of parameters (weights) of the ResNet and let $(\psi_i, \chi_i)$ represent a given image sample with $\chi_i$ its classification/regression target, i.e., classes $p$ or $\xi$, and also $\chi_i'$ the predicted values, $p'$ or $\xi'$. For classification of categorical data, the class names are denoted by a class index $c = 1, \ldots, \mathcal{C}$ and encoded as $\chi_{ck} = 1$ if $\chi_c = k$, 0 otherwise. Then, for the (multi-class) classification problem, we choose the usual cross-entropy loss function, $l_c(\mathbf{w}) = -\sum_{k=1}^{n} \sum_{c=1}^{\mathcal{C}} \chi_{ck} \log \chi_{ck}'(\mathbf{w}) + (1 - \chi_{ck}) \log[1 - \chi_{ck}'(\mathbf{w})]$, where $n$ is the number of samples, i.e., either $N_{\text{train}}$ or $N_{\text{val}}$ [26]. We use the AdaDelta optimizer [32] and find that a learning rate of $\ell_r = 0.001$ produces good convergence [33]. Another good metric for the classification task is the accuracy $a$, which is the ratio of correct predictions over $n$. The loss function for the regression problem is given by the mean-squared error $l_r(\mathbf{w}) = \frac{1}{n} \sum_{k=1}^{n} [\chi_k - \chi_k'(\mathbf{w})]^2$ while $\ell_r$ and the optimizer remain the same. When giving results for $l_c$ and $l_r$ below, we always present those after averaging over at least 10 independent training and validation cycles, i.e., with a different initial split of the data into mini-batches. We use the notation $\langle l_c \rangle$ and $\langle l_r \rangle$ to indicate this averaging. In the case of classification, we also represent the quality of a prediction by confusion matrices [26]. These graphically represent the predicted class labels as a function of the true ones in matrix form, with an error-free prediction corresponding to a purely diagonal matrix. For comparison of the classification and regression calculations, we use in both cases a maximum number of $\epsilon_{\max} = 20$ epochs.

Our ML calculations train with a batch size of 256 for classification and for regression. All runs are performed on NVIDIA Quadro RTX6000 cards.

### D. Generating test data sets

We generate a test data set, $\tau$, of 1000 states for each of the 31 $p$-values, such that in total we have $N_\tau = 31000$. This test set is used to make all the confusion matrices given below. By doing this, we ensure that the performance of the trained DL networks is always measured on unseen data [27].

In addition, we generate three special test data sets. These data sets have been constructed to allow testing for the existence of the spanning cluster. The first special data set, $\tau_{\text{sl}}$, is made for the 27 $p$-values $0.5, 0.55, 0.555, \ldots, 0.66, 0.7$ close to $p_c$ and again consists of 1000 states $\psi_i(p)$ for each $p$. After generating each $\psi_i(p)$, we add a *straight line* of occupied sites from top to bottom, while keeping $p$ constant by removing other sites at random positions. Obviously, every $\psi_i(p)$ in $\tau_{\text{sl}}$ therefore contains at least one spanning cluster by construction. As a consistency check to the performance of the ML networks, we also add two more $\psi_i$ without any connecting path for $p = 0.1$ and 0.2.

In the next set, $\tau_{\text{rw}}$, we start with the same 27 $p$-values for a new set of 27000 $\psi_i(p)$, but instead of the straight line, we add a directed *random walk* from top to bottom. As before, we conserve the overall density $p$ of occupied sites. Hence, every samples in $\tau_{\text{rw}}$ is spanning. We again add two $\psi_i$ for $p = 0.1$ and 0.2 without the connected random path.

Finally, the third special data set, $\tau_{\text{fb}}$, again contains 27000 lattices for the same previously mentioned

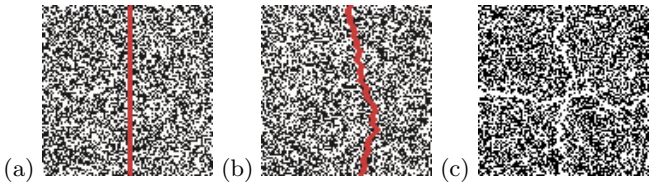

FIG. 2. Examples of percolation images from the three special test sets $\tau_S$ with (a) a percolating straight line from top to bottom, (b) a percolating random path from top to bottom and (c) a "firebreak"-like cross of empty sites preventing percolation. For the sake of visibility, in (a+b) the connected path is highlighted in red. In all three cases, $p = 0.5$.

27 $p$-values, but in each of the states we apply random *firebreak* paths, horizontally and vertically, of unoccupied sites. This set is clearly non-spanning. Following the same logic as for $\tau_{\rm sl}$ and $\tau_{\rm rw}$, we add two spanning test samples above $p_c$ without the firebreak, namely, for $p = 0.8$ and 0.9. In all three cases, despite the modification in the lattices we ensure that $N_{\rm occ} = p \times L^2$ and hence the occupation density is $p$. Examples of the three sets can be seen in Fig. 2.

## III. RESULTS

### A. Classification of states labeled with density $p$

We use the density $p$ values as labels for the ML task of image recognition with the DL implementation outlined in section II C. After ten trainings with all 310000 images for 20 epochs, we find on average a validation loss of $\langle l_{\rm c,val}\rangle = 0.052 \pm 0.009$ (corresponding to an accuracy of $\langle a_{\rm c,val}\rangle = 99.323\% \pm 0.003$). This is comparable to the very good image classification results shown on kaggle [34]. Fig. 3(a) gives the resulting averaged confusion matrix. The dependence of the training and validation losses, $\langle l_{\rm c,train}\rangle$ and $\langle l_{\rm c,val}\rangle$, respectively, on the number of epochs, $\epsilon$, is shown in Fig. 3(b). From the behavior of the loss functions, we can see that $\langle l_{\rm c,val}\rangle \geq \langle l_{\rm c,train}\rangle$ until $\epsilon = 15$ after which both losses remain similar. This suggests that $\epsilon_{\rm max} = 20$ for our DL approach is indeed sufficient and avoids over-fitting. Similarly, the confusion matrix is mostly diagonal with the exception of very few samples around the change of resolution in density, at $p \sim 0.555$ and 0.655, as commented before in section II A.

### B. Prediction of densities $p$ via regression

For the regression problem, we train the ResNet18 only for the nine evenly spaced densities $p = 0.1, 0.2, \ldots, 0.9$. After training and validation with $\mathcal{T}$ and $\mathcal{V}$, respectively, we examine the states in $\tau$ and predict their $p$ values. In Fig. 4, we present the results with (a) indicating the fidelity of the predictions

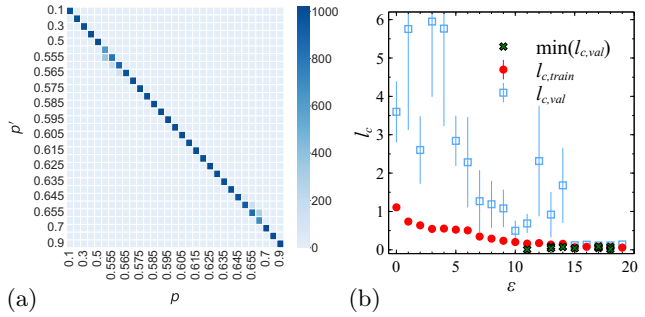


FIG. 3. (a) Average confusion matrix for *classification* according to $p$. The dataset used is the test data $\tau$ and the models used for predictions are those corresponding with a minimal $l_{\rm c,val}$. True labels for $p$ are indicated on the horizontal axis while the predicted labels are given on the vertical axis. The color scale represents the number of samples in each matrix entry. (b) Dependence of losses $l_{\rm c,train}$ and $l_{\rm c,val}$ averaged over ten independent training seeds, on the number of epochs $\epsilon$ for *classification* according to $p$. The squares (blue open) denote $l_{\rm c,train}$ while the circles (red solid) show $l_{\rm c,val}$. The green crosses indicate the minimal $l_{\rm c,val}$ for each of the ten trainings.

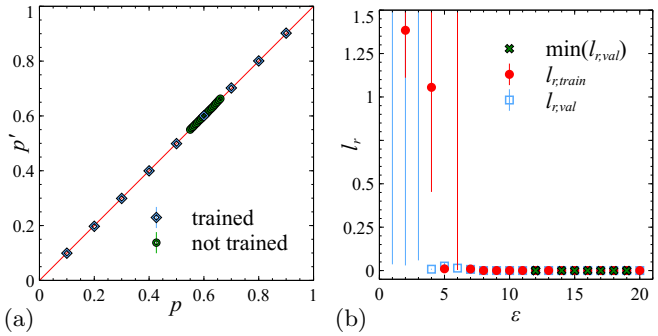


FIG. 4. (a) Average prediction curve obtained for *regression* according to $p$ at the minimal $l_{\rm r,val}$. The dataset used is the test data $\tau$ and the models used for predictions are those corresponding with a minimal $l_{\rm c,val}$. The blue open squares denote $p$-values that have been used during the training and the green open circle shows $p$-values that were not trained. (b) Dependence of losses $l_{\rm r,train}$ and $l_{\rm r,val}$ averaged as in Fig. 3 on the number of epochs $\epsilon$ for *regression* according to $p$. The squares (blue open) denote $l_{\rm r,train}$ while the circles (red solid) show $l_{\rm r,val}$. The green crosses show the minimal $l_{\rm r,val}$ for each of the ten trainings.

for each $p$-value and (b) showing good convergence of the losses $l_{\rm r,train}$ and $l_{\rm r,val}$. Clearly, the regression works very well for the nine trained $p$-values $p = 0.1, \ldots, 0.9$ as well as the untrained values $0.55, 0.555, \ldots, 0.0.655, 0.66$ close to $p_c(100)$. After reaching $\epsilon = 20$, we find that $\min_\epsilon[\langle l_{\rm r,train}\rangle] = 0.0003 \pm 0.0002$ and $\min_\epsilon[\langle l_{\rm r,val}\rangle] = (6.2 \pm 1.2) \times 10^{-5}$.

Therefore we conclude that our CNN performs well for classification and regression tasks while $\mathcal{T}$, $\mathcal{V}$, and $\tau$ present appropriately structured data sets for these ML tasks in terms of data size.

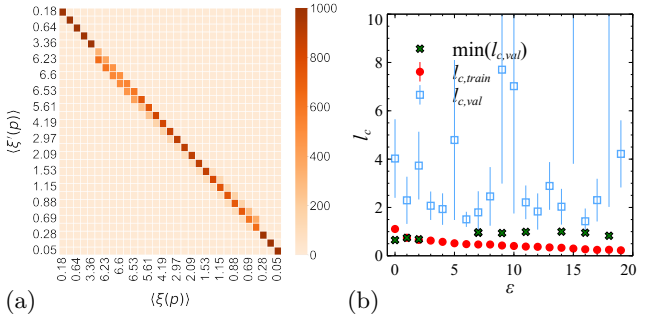


FIG. 5. (a) Average confusion matrix for *classification* according to $\langle\xi\rangle$. The dataset used is the test data $\tau$ and the models used for predictions are those corresponding with a minimal $l_{\mathrm{c,val}}$. (b) Dependence of losses $l_{\mathrm{c,train}}$ and $l_{\mathrm{c,val}}$ on the number of epochs $\epsilon$ for classification according to $\langle\xi\rangle$. We follow the same convention as in Fig. 3 and Fig. 7.

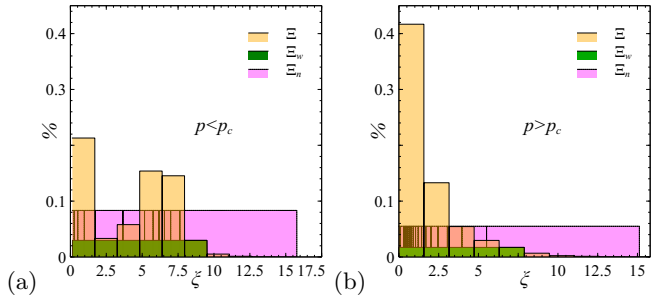


FIG. 6. Probability distributions for correlation lengths $\xi$ when (a) $p < p_c$ (with 12 $p$-values) and $p > p_c$ (18 $p$-values) with unbalanced $\Xi$ and the balanced counterparts $\Xi_n$ and $\Xi_w$ denoted by yellow, magenta and green, respectively. In each case, the distributions are normalized relative to the total number of $\xi$'s in each set, i.e. for (a) 120000 in $\Xi$ and $\Xi_n$ and $6 = 21360$ in $\Xi_w$ while for (b) there are 180000 in $\Xi$ and $\Xi_n$ and $5 \times 3077 = 15385$ in $\Xi_w$.

### C. Classification with correlation length $\xi$

We now turn our attention to studying image recognition when using the correlation lengths $\xi$, instead of $p$, as labels for the $\psi_i(p)$ states. One way to do this is to use $\langle\xi(p)\rangle$ as label. While for the classification by $p$ the label value was identical to the actual density $p$ of a given state, now each state is labeled by $\langle\xi(p)\rangle$. This means that the actual $\xi$ of the state might be different from the label assigned. Since $\langle\xi(p)\rangle$ can be uniquely identified by $p$, this strategy should be in fact equivalent to the previous situation and the CNN should give us similar classification results. The results of such a classification are shown in Fig. 5 where similarly to Fig. 3 we present in (a) the average confusion matrix for the 31 $\langle\xi(p)\rangle$ values (cf. also Fig. 1) and in (b) the evolution of losses during the training. We find a validation loss of $\min_\epsilon[\langle l_{\mathrm{c,val}}\rangle] = 0.38 \pm 0.07$ (corresponding to a maximal accuracy of $\max_\epsilon[\langle a_{\mathrm{c,val}}\rangle] = 87.12\% \pm 0.05$) and a highly diagonal confusion matrix, with only a small deviation that can be linked to the change in resolution in our data set above $p = 0.5$.

One might wish to interpret the above classification with $\langle\xi(p)\rangle$ as a success of the ML approach. However, let us reemphasize that it is fundamentally equivalent to simply changing labels while keeping the direct connection of the labels with $p$ unaltered. We now wish to obtain a classification of states via their $\xi$'s which is independent of the $p$'s. In Fig. 6 we show the distribution $\Xi$ of the $\xi$'s in $\mathcal{T} \cup \mathcal{V}$. Clearly, the number of small $\xi$ values is larger than the number of $\xi$ values close to the maximal value of $\max[\xi] = 15.771$ (cp. Fig. 1(c)). Hence simply using each $\xi$ as label for the corresponding $\psi_i$ would result in a biased dataset. We therefore reorganize the $\mathcal{T} \cup \mathcal{V}$ data set. This can be done in two ways. For the first reorganization we create bins a constant *number* of 10000 samples in each bin. We call this dataset $\Xi_n$. This results in a varying bin width. The second way to reorganize the data set is to keep the bin width constant while restricting the *number* of samples in each bin. We shall denote this reorganization as $\Xi_w$. Since $\xi(p)$ is non-monotonic in $p$, we split the reorganization into the case (i) $p < p_c$ with and (ii) $p > p_c$. We emphasize that the reorganized data sets consist of the same states as in $\mathcal{T} \cup \mathcal{V}$ but now have different labels according to the bin labels for $\Xi_w$ and $\Xi_n$. Furthermore, there is now no longer any direct connection of the new labels to the original $p$ densities.

In Fig. 7, we plot the resulting confusion matrices and losses. We see that the classification for $\Xi_w$ and $\Xi_n$ only results in large diagonal entries in the confusion matrices for small correlation lengths labels $\xi$. Overall, the classification for $\Xi_w$ is somewhat better than for $\Xi_n$ when away from $p_c(L)$. We attribute this to the larger number of states for the $\Xi_w$ for $p < p_c(L)$. Still, with overall 62.6% and 55.1% of states misclassified for $\Xi_w$ and $\Xi_n$, respectively, it seems clear that classification for correlation lengths must be considered unsatisfactory.

### D. Regression with correlation length $\xi$

For the regression task with $\xi$, we proceed analogously to section III B. Again, we train the CNN for the individual correlation length $\xi_i$ corresponding to each $\psi_i \in \mathcal{T}$ for the nine densities $p = 0.1, \ldots, 0.9$. We then compute the predictions of $\xi_i$ for all 31 densities in $\tau$. The results are shown in Fig. 8. We find that the network architecture which previously predicted the density quite accurately is now struggling to correctly predict $\xi$. A structure seems to exist in the predictions. By looking closely we notice that the network make use of the density for its predictions. Furthermore, by plotting the correlation length according to the density we retrieve the plot of $\xi$ Fig. 1.

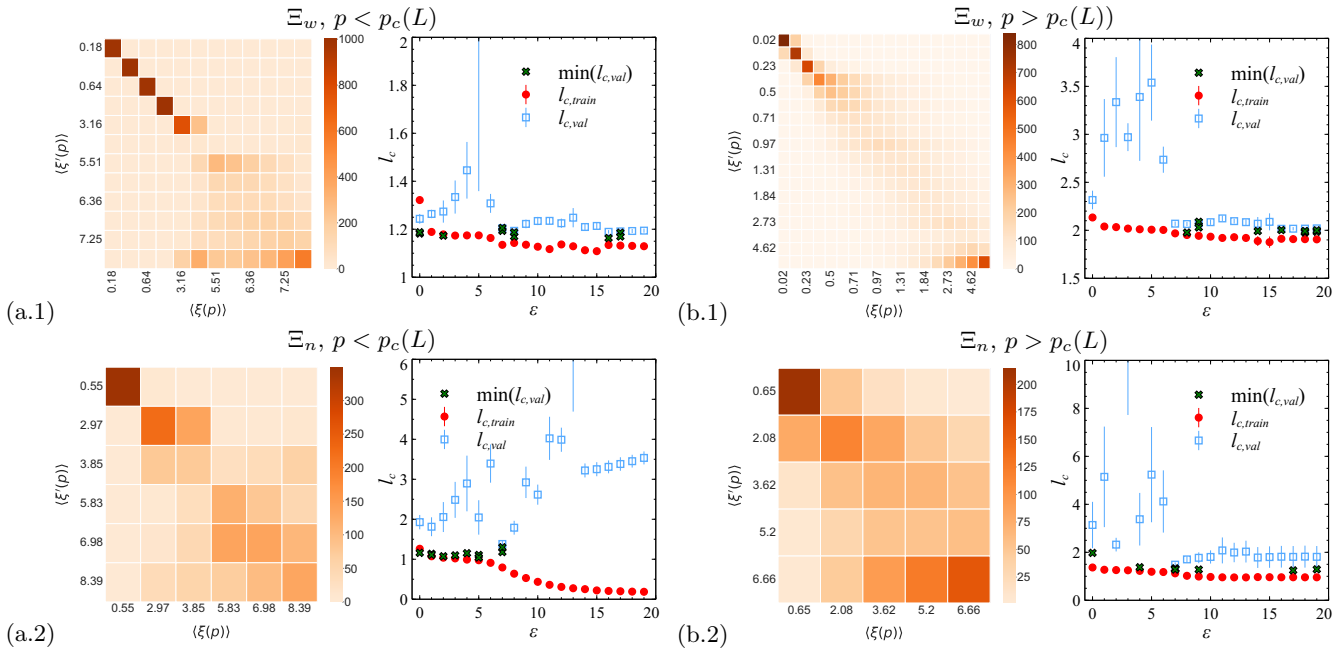


FIG. 7. Confusion matrices and losses $l_{\mathrm{r,train}}$ and $l_{\mathrm{r,val}}$ for the classification results when using the correlation-function-relabeled $\Xi_w$ and $\Xi_n$ data sets. The left column (a) shows the case $p < p_c$ while the right column (b) gives the outcome for $p > p_c$. The upper row corresponds to 10000 states for each class with 12 classes for $p < p_c$ and 18 at $p > p_c$ for $\Xi_w$. In the lower row, there are 3560 states for the 6 classes when $p < p_c$ and 3077 states for 5 classes when $p > p_c$.

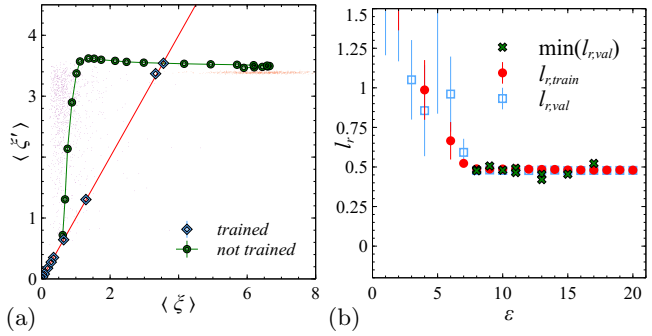


FIG. 8. (a) Average predictions for *regression* according to $\xi$. The dataset used is the test data $\tau$ and the models used for predictions are those corresponding with a minimal $l_{\mathrm{c,val}}$. (b) Dependence of losses $l_{\mathrm{r,train}}$ and $l_{\mathrm{r,val}}$ on the number of epochs $\epsilon$ for regression according to $\xi$. We follow the same convention as in Fig. 4.

### E. Classification with the spanning or non-spanning properties

As discussed earlier, the hallmark of the percolation transition is the existence of a spanning cluster which determines whether the system is percolating or not [14]. In the previous section, our DL approach has classified according to $p$ or $\xi$ values without testing whether spanning clusters actually exist. We now want to check this and label all states according to whether they are spanning or non-spanning. From Fig. 1, it is immediately clear that for finite-sized systems considered here, there are a non-negligible number of states with appear already spanning even when $p < p_c$ and, vice versa, are still non-spanning when $p > p_c$. Furthermore, we note that for such $L$, the difference between $p_c$ and $p_c(L)$ is large enough to be important and we hence use $p_c(L)$ as the appropriate value to distinguish the two phases.

Figure 9 shows the average results after $\epsilon = 20$ with an validation loss of $\min_\epsilon[\langle l_{\mathrm{c,val}}\rangle] = 0.165 \pm 0.001$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle a_{\mathrm{c,val}}\rangle] = 92.702\% \pm 0.001$). At first glance, the figure seems to indicate a great success: from the 31000 states present in $\tau$, 11510.6 have been correctly classified as non-spanning (i.e., $N \to N'$), and 17205.9 as spanning ($S \to S'$) while only 1223.1 are wrongly labeled as non-spanning ($S \to N'$) and 1059.41 as spanning ($N \to S'$) [35]. Overall, we would conclude that 92.6% of all test states are correctly classified while 7.4% are wrong. However, from the full percolation analysis for $\mathcal{T} \cup \mathcal{V}$ shown in Fig. 1, we know that there are 92.7% of states without a spanning cluster below $p_c(L)$ while 7.3% of states, equivalent to 876 samples, already contain a spanning cluster. Similarly, for $p > p_c(L)$, 94.8% of states, equivalent to 936 samples, are spanning and 5.2% are not. At $p_c(L) = 0.585$, we furthermore have 518 spanning and 482 non-spanning states. Hence in total, we expect 2812 wrongly classified states. Since the last number is decisively close to the actual number of 2282.5 of misclassified states, this suggests that it is precisely the spanning states below $p_c(L)$ and the non-spanning ones above $p_c(L)$ which the DL network is unable to recognize. Let us rephrase for clarity: it seems that the DL CNN, when

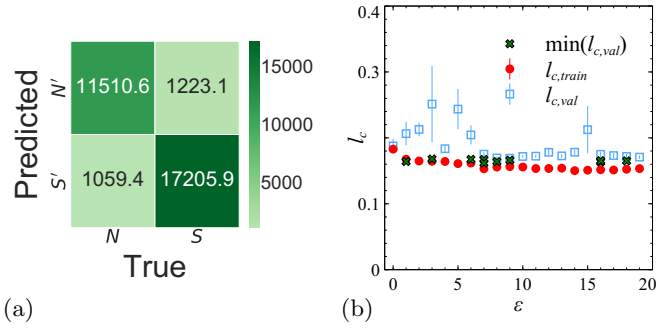


FIG. 9. (a) Average confusion matrix for *classification* according to spanning/non-spanning. The dataset used is the test data $\tau$ and the models used for predictions are those corresponding with a minimal $l_{\mathrm{c,val}}$. The true labels for $N$ and $S$, are indicated on the horizontal axis while the predicted labels are given on the vertical axis. (b) Dependence of losses $l_{\mathrm{c,train}}$ and $l_{\mathrm{c,val}}$ on the number of epochs $\epsilon$ for classification according to spanning/non-spanning. Again, we follow the same convention as for Figs. 3, 7, and 5

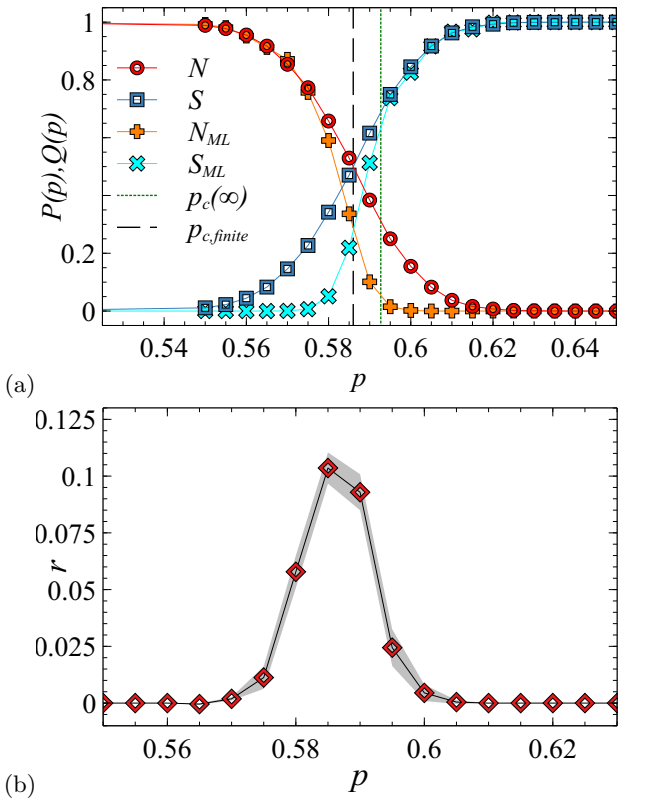


FIG. 10. (a) The blue curve (red curve) shows the probability to have a spanning (non-spanning) sample in the training dataset. The cyan (orange) curve gives us the prediction of probability to have a spanning (non-spanning) sample, according to the trained network. (b) Dependence of the Pearson correlation coefficient $r$ on the density $p$ for classification according to spanning/non-spanning. The confidence interval is indicated in gray. In both (a) and (b), The lines connecting the symbols are only a guide to the eye.

trained in whether a cluster is spanning or non-spanning, completely disregards this information in its classification outputs.

### F. Density-resolved study of spanning/non-spanning close to $p_c(L)$

In order to understand the behavior observed in the last section, we now reexamine the result of Fig. 9 by analyzing the ML-predicted probabilities, $P_{\mathrm{ML}}(p)$. In Fig. 10, we show both $P_{\mathrm{ML}}(p)$ as well as $P(p)$; the latter having been obtained by the Hoshen-Kopelman algorithm, cf. Fig. 1(a). While the $P(p)$ and $P_{\mathrm{ML}}(p)$ curves — and of course also the corresponding $Q(p)$ and $Q_{\mathrm{ML}}(p)$ — appear qualitatively similar, they are nevertheless not identical and the slopes of $P_{\mathrm{ML}}(p)$, $Q_{\mathrm{ML}}(p)$ are different. We emphasize that the slopes are important for determining the universality class of a second-order phase transition via finite-size scaling [36]. Since we know for each image whether it percolates or not, we can also check how well the ML predictions worked by considering the covariance. Let $\zeta(\psi_i(p)) = 0$ when there is no percolating cluster in the state $\psi_i(p)$ while $\zeta(\psi_i(p)) = 1$ if there is. Similarly, we define $\zeta_{\mathrm{ML}}(\psi_i(p))$ for the prediction by the DL network. Then $\mathrm{cov}(\zeta, \zeta_{\mathrm{ML}})(p)$ measures the covariance of states being found to span by percolation *and* by ML for given $p$. In Fig. 10(b) we show the normalized result, i.e., the Pearson coefficient $r_{\zeta,\zeta_{\mathrm{ML}}}(p) = \mathrm{cov}(\zeta, \zeta_{\mathrm{ML}})(p)/[\sigma_\zeta(p)\sigma_{\zeta_{\mathrm{ML}}}(p)]$, where $\sigma_\zeta$ and $\sigma_{\zeta_{\mathrm{ML}}}$ are the standard deviations of the percolation results and the ML predictions. We see that in the transition region, $r_{\zeta,\zeta_{\mathrm{ML}}} \lesssim 0.12$ which is very far from the maximally possible value 1. This suggest that while the ML predictions are not simply random, they are also not very much better than random.

Let us now study the classification into spanning/non-spanning states in detail for each $p$. Figure 11 and Table I show a comparison of the classification for the ten $p$ values 0.56 to 0.605. We see, e.g., that for $p = 0.56, 0.565, 0.57, 0.575 < p_c(L) \sim 0.58, 0.585$, 485 of 487 samples, which are already spanning, have been misclassified as non-spanning. Similarly, for $p = 0.59, 0.595, 0.6, 0.605 > p_c(L)$, 7545.9 of in total 864 still non-spanning samples are classified as spanning. These results are similar whether one considers a typical sample or the averaged result. Hence, contrary to the supposed success of Fig. 9, we now find that the seemingly few misclassified states of Fig. 9 are indeed precisely those which represent the correct physics. Saying it differently, the ML process seems to have led to a DL network which largely disregards the characteristic of spanning clusters and just uses the overall density of occupied vs. non-occupied sites to ascertain the phases. Of course, this is the wrong physics when considering percolation.

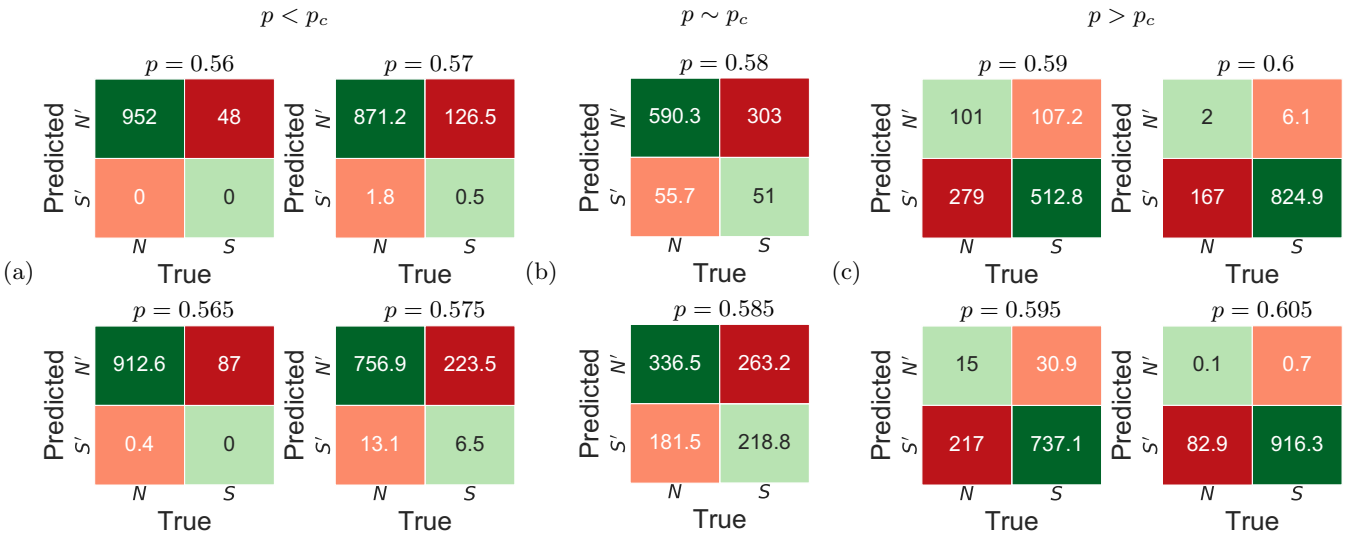


FIG. 11. Confusion matrices showing the predictions of the trained network Fig. 9 in a region $p = [0.56, 0.605]$ comprising $p_c$, with (a) for predictions made before the percolation threshold, (b) in the threshold region and (c) after the percolation threshold. Each confusion matrix is an average of the predictions made by the 10 trained models shown in Fig. 9.

| | | $N$ | | $S$ | | $(S \to S')$ | | $(S \to N')$ | | $(N \to S')$ | | $(N \to N')$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p$ | $n$ | # | % | # | % | $\langle\#\rangle$ | $\langle\%\rangle$ | $\langle\#\rangle$ | $\langle\%\rangle$ | $\langle\#\rangle$ | $\langle\%\rangle$ | $\langle\#\rangle$ | $\langle\%\rangle$ |
| 0.56 | 1000 | 952 | 95.2 | 48 | 4.8 | 0.0 | 0.0 | 48.0 | 4.8 | 0.0 | 0.0 | 952.0 | 95.2 |
| 0.565 | 1000 | 913 | 91.3 | 87 | 8.7 | 0.0 | 0.0 | 87.0 | 8.7 | 0.4 | 0.0 | 912.6 | 91.3 |
| 0.57 | 1000 | 873 | 87.3 | 127 | 12.7 | 0.5 | 0.1 | 126.5 | 12.7 | 1.8 | 0.2 | 871.2 | 87.1 |
| 0.575 | 1000 | 770 | 77.0 | 230 | 23.0 | 6.5 | 0.7 | 223.5 | 22.4 | 13.1 | 1.3 | 756.9 | 75.7 |
| 0.58 | 1000 | 646 | 64.6 | 354 | 35.4 | 51.0 | 5.1 | 303.0 | 30.3 | 55.7 | 5.6 | 590.3 | 59.0 |
| 0.585 | 1000 | 518 | 51.8 | 482 | 48.2 | 218.8 | 21.9 | 263.2 | 26.3 | 181.5 | 18.2 | 336.5 | 33.6 |
| 0.59 | 1000 | 380 | 38.0 | 620 | 62.0 | 512.8 | 51.3 | 107.2 | 10.7 | 279.0 | 27.9 | 101.0 | 10.1 |
| 0.595 | 1000 | 232 | 23.2 | 768 | 76.8 | 737.1 | 73.7 | 30.9 | 3.1 | 217.0 | 21.7 | 15.0 | 1.5 |
| 0.60 | 1000 | 169 | 16.9 | 831 | 83.1 | 824.9 | 82.5 | 6.1 | 0.6 | 167.0 | 16.7 | 2.0 | 0.2 |
| 0.605 | 1000 | 83 | 8.3 | 917 | 91.7 | 916.3 | 91.6 | 0.7 | 0.1 | 82.9 | 8.3 | 0.1 | 0.0 |

TABLE I. Predictions of the trained network on the test data set $\tau$ for $p \in [0.56, 0.605]$. $N$ and $S$ denote the respectively the number of non-spanning and the number of spanning samples in $\phi$. The four following columns $(S \to S')$, $(S \to N')$, $(N \to S')$, and $(N \to N')$ gives the averaging of 10 independent prediction runs.

### G. Testing the accuracy of the DL network

The difficulties that the trained DL network has with recognizing whether a state contains a percolating cluster or not can be made more explicit. In section II D, we had generated three test sets $\tau_S$ for this purpose. Namely, percolating states even for $p < p_c(L)$ by adding (i) a straight line and (ii) a random walk of connecting sites as well as for $p > p_c(L)$ (iii) the firebreak states of percolation-prohibiting random unoccupied sites. We now use these sets and feed them independently as test sets to the DL network. Figure 12 shows the three confusion matrices obtained when classifying for spanning vs. non-spanning. In Fig. 12(a+b), we see that the network completely misclassifies the spanning datasets $\tau_{\rm sl}$ and $\tau_{\rm rw}$. The two correctly identified non-spanning images are just the two such states added to each of the data sets to show that the network is still performing. Similarly, in Fig. 12(c), we see that this time the network cannot correctly identify the non-spanning samples in $\tau_{\rm fb}$. Again, the two samples correctly identified are the ones without the firebreak.

## IV. CONCLUSIONS

Let us briefly summarize what we have achieved thus far. We showed that when looking at $p$, classification and regression techniques for percolation states allow us to obtain good recognition with near-perfect $\langle a_{c,val}\rangle = 99.323\% \pm 0.003)$ for classification and near-zero $\langle l_{r,val}\rangle = 0.000062 \pm 0.000012$ average mean-square loss for regression. Confusion matrices are heavily diagonal for classification while prediction curves for regression are similarly convincing. These results are in good agreement with previous similar such studies [15, 16, 37], which should not come as a surprise: the information about $p$ is of course directly enclosed in each state. We emphasize that

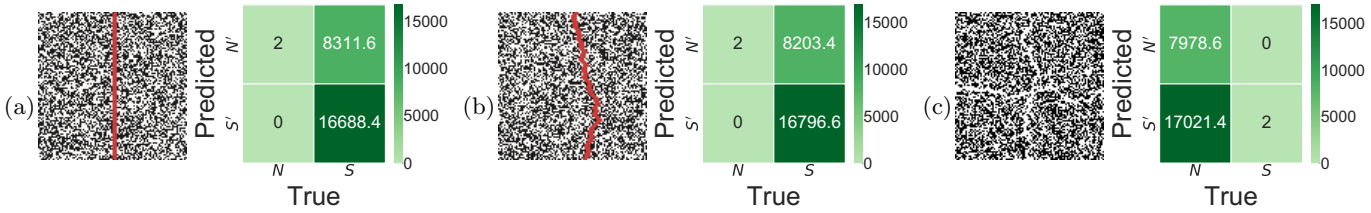


FIG. 12. Sample states for the three special test sets (a) $\tau_{\rm sl}$ with added straight spanning lines, (b) $\tau_{\rm rw}$ with spanning random walks and (c) $\tau_{\rm fb}$ with the non-spanning firebreaks. In each case, the right plots gives the confusions matrices obtained from the DL model previously trained in a spannin vs. non-spanning classification. In all cases, the density is strictly $p = 0.5$.

our approach is already somewhat more challenging than these previous works since instead of just asking the DL network to identify the two phases $p < p_c$ and $p > p_c$, we also successfully identify all 31 distinct densities $p$. Using $\langle\xi(p)\rangle$ instead of $p$ to identify the 31 densities also works quite well, but is again expected since $\langle\xi(p)\rangle$ merely acts as a new set of supervised labels.

Problems emerge when we use the computed correlation length $\xi$ for each state and try classification and regression with these correlation lengths. Having thus explicitly removed any connection with the $p$ values, we nevertheless find that the resulting confusion matrices, fidelity curves and losses are all of much less quality than before. Instead, it seems that the density $p$ information is still the overriding measure used by the DL network to arrive at its outputs (cf. Figs. 7 and 8) [38]. Rather, as we show in sections III E–G, the DL network completely ignores whether a cluster is spanning or non-spanning, essentially missing the underlying physics of the percolation problem — it seems to still use $p$ as its main ordering measure.

We believe that the root cause of the failure to identify the spanning clusters, or their absence, lies in the fundamentally local nature of the CNN: the filter/kernels employed in the ResNets span a few *local* sites only [39]. Hence it is not entirely surprising that such a CNN cannot correctly identify the essentially *global* nature of spanning clusters. But it is of course exactly this global percolation that leads to the phase transition.

The reader might wonder why previous CNN studies of phases in other models, such as, e.g., the Ising-type models [9], the three-dimensional Anderson model and its topological variants [10], have failed to find similar such issues. We think that this is because in the Ising case, the majority rule for spin alignment is not concerned with any globally spanning domain [40], while in the Anderson-type models, it is the (typical) local density of states which can serve as order parameter [41]. In short, in these models a local property is indeed sufficient to distinguish their phases.

Of course ML aficionados might now want to suggest that extensions of local kernels are possible in CNNs. Indeed, one might, e.g., want to use CNNs in which large dilution parameters are employed to effectively make filters/kernels of manageable size while still spanning across a sizeable portion of the $L \times L$ size of each percolation state [42]. But while this might solve the issue for fixed $L$ in the percolation case — we have not tested it — it does imply knowing that a global property is important to start with. This would rather diminish the relevance of DL as a tool for unbiased discovery in physics.

## ACKNOWLEDGMENTS

We thank B. Çivitcioğlu, M. Hilke, and T. Ohtsuki for discussions. D.B. is grateful for co-tutelle funding via the EUtopia Alliance. We gratefully acknowledge the University of Warwick Research Technology Platform (RTP Scientific Computing) and the Sulis Tier 2 HPC platform hosted by the RTP. Sulis is funded by EPSRC Grant EP/T022108/1 and the HPC Midlands+ consortium.

[1] K. He, X. Zhang, S. Ren, and J. Sun, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 2016-Decem (IEEE, 2016) pp. 770–778.
[2] M. A. Tabak, M. S. Norouzzadeh, D. W. Wolfson, S. J. Sweeney, K. C. Vercauteren, N. P. Snow, J. M. Halseth, P. A. Di Salvo, J. S. Lewis, M. D. White, B. Teton, J. C. Beasley, P. E. Schlichting, R. K. Boughton, B. Wight, E. S. Newkirk, J. S. Ivan, E. A. Odell, R. K. Brook, P. M. Lukacs, A. K. Moeller, E. G. Mandeville, J. Clune, and R. S. Miller, Methods in Ecology and Evolution **10**, 585 (2019).
[3] R. Zhang, Q. Wang, and Y. Lu, in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)* (IEEE, 2017) pp. 25–29.
[4] Y. Zhao and M. Khushi, in *2020 International Conference on Data Mining Workshops (ICDMW)*, Vol. 2020-Novem (IEEE, 2020) pp. 385–391.
[5] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada, Phys. Rev. B **96**, 205152 (2017).
[6] W.-J. Rao, J. Phys.: Condens. Matter **30**, 395902 (2018).
[7] Y. Zhang and E.-A. Kim, Phys. Rev. Lett. **118**, 216401 (2017).

[8] E. M. Stoudenmire and D. J. Schwab, Advances in Neural Information Processing (2017).
[9] J. Carrasquilla and R. G. Melko, Nature Physics **13**, 431 (2017).
[10] T. Ohtsuki and T. Mano, J. Phys. Soc. Jpn. **89**, 022001 (2020).
[11] K. Ch'ng, J. Carrasquilla, R. G. Melko, and E. Khatami, Phys. Rev. X **7**, 031038 (2017).
[12] J. Venderley, V. Khemani, and E.-A. Kim, Phys. Rev. Lett. **120**, 257204 (2018).
[13] S. R. Broadbent and J. M. Hammersley, Mathematical Proceedings of the Cambridge Philosophical Society **53**, 629 (1957).
[14] D. Stauffer and A. Aharony, *Introduction to Percolation Theory*, 2nd ed. (Taylor & Francis Group, 1991).
[15] W. Zhang, J. Liu, and T.-C. Wei, Phys. Rev. E **99**, 032142 (2019).
[16] J. Shen, W. Li, S. Deng, and T. Zhang, Phys. Rev. E **103**, 052140 (2021).
[17] S. Patwardhan, U. Majumder, A. D. Sarma, M. Pal, D. Dwivedi, and P. K. Panigrahi, "Machine learning as an accurate predictor for percolation threshold of diverse networks," (2022), arXiv:2212.14694 [physics.soc-ph].
[18] W. Yu and P. Lyu, Physica A **559**, 125065 (2020).
[19] R. J. Elliott, B. R. Heap, D. J. Morgan, and G. S. Rushbrooke, Phys. Rev. Lett. **5**, 366 (1960).
[20] P. J. Flory, *Principles of Polymer Chemistry* (Cornell University Press, 1953).
[21] B. Derrida and D. Stauffer, Journal de physique Paris **46**, 1623 (1985).
[22] G. Grimmett, *Percolation* (Springer Verlag, Berlin, 1989).
[23] J. L. Jacobsen, J. Phys. A: Math. Theor. **47**, 135001 (2014).
[24] J. Hoshen and R. Kopelman, Phys. Rev. B **14**, 3438 (1976).
[25] We have checked that our results do not change when using the gray scales. Furthermore, the full cluster information is already coded in the spatial distribution of these gray intensities, hence giving much information to the DNNs.
[26] P. Mehta, M. Bukov, C. H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, Phys. Rep. **810**, 1 (2019).
[27] A. Dawid, J. Arnold, B. Requena, A. Gresch, M. Podzie, K. Donatella, K. A. Nicoli, P. Stornati, R. Koch, M. Btner, R. Okua, G. Muoz-Gil, R. A. Vargas-Hernndez, A. Cervera-Lierta, J. Carrasquilla, V. Dunjko, M. Gabri, P. Huembeli, E. van Nieuwenburg, F. Vicentini, L. Wang, S. J. Wetzel, G. Carleo, E. Greplov, R. Krems, F. Marquardt, M. Tomza, M. Lewenstein, and A. Dauphin, "Modern applications of machine learning in quantum sciences," (2022), arXiv:2204.04198 [quant-ph].
[28] E. Bedolla, L. C. Padierna, and R. Castaeda-Priego, J. Phys.: Condens. Matter **33**, 053001 (2020).
[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, 2009 IEEE Conference on Computer Vision and Pattern Recognition , 248 (2009).
[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, Advances in Neural Information Processing Systems **32** (2019).
[31] "SciKit-Learn 1.2.2: 1.17. Neural network models (supervised)," (2023).
[32] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," (2012), arXiv:1212.5701 [cs.LG].
[33] We also optimized other hyperparameters and tested other optimizers such as ADAM but found no significant performance improvement.
[34] "Kaggle competition "Dogs vs. Cats": Create an algorithm to distinguish dogs from cats," (2012).
[35] We note that these numbers are not integers since they are computed as averages over the 10 independent training runs as mentioned in section II C.
[36] T. R. Kirkpatrick and D. Belitz, J. Phys.: Condens. Matter **4**, L37 (1992).
[37] S. Cheng, F. He, H. Zhang, K.-D. Zhu, and Y. Shi, "Machine learning percolation model," (2021), arXiv:2101.08928 [cond-mat.dis-nn].
[38] We emphasize that we have spent considerable effort at making sure that this result is not due to erroneous information leakage [27].
[39] In addition to the ResNet18 used in the main text, we have also checked that ResNet34 yields a similar outcome.
[40] B. M. McCoy and T. T. Wu, *The Two-Dimensional Ising Model* (Harvard University Press, 1973).
[41] V. Dobrosavljević, A. A. Pastor, and B. K. Nikolić, Europhys. Lett. **62**, 76 (2003).
[42] M. Al-Shabi, H. K. Lee, and M. Tan, IEEE Access **7**, 178827 (2019).