Deep Machine Learning Investigation of Phase Transitions

Vladislav Chertenkov[1,2][0000−0002−4528−6730] ✉ satankow@yandex.ru, Evgeni Burovski[1,2][0000−0001−8149−0483] evgeny.burovskiy@gmail.com, and Lev Shchur[1,2][0000−0002−4191−1324] levshchur@gmail.com

[1] Landau Institute for Theoretical Physics, 142432 Chernogolovka, Russia
[2] HSE University, 101000 Moscow, Russia

Abstract. We explore the possibilities of using neural networks to study phase transitions. The main question is the level of accuracy which can be achieved for the estimates of the critical point and critical exponents of statistical physics models. We generate data for two spin models in two dimensions for which analytical solutions exist, the Ising model and Baxter-Wu model, which belong to the different universality classes. We applied six neural networks with three different architectures to the data and estimated the critical temperature and the correlation length exponent. We find that the accuracy of estimation does depend on the neural network architecture. The critical exponents of Baxter-Wu model are estimated by the deep machine learning technique for the first time.

Keywords: Ising model · Baxter-Wu model · Deep learning · Finite-size scaling · Resnet.

1 Introduction

Recent advances in deep learning—both algorithmic developments and efficient utilization of novel high-performance hardware—revolutionize a variety of engineering practice and academic research areas. One notable development is an observation that neural networks (NN) can successfully predict states of matter and phase transitions between them for both classical [1] and quantum [2] many-body systems. Various network architectures, learning protocols and NN observables are used [3–7] for several physical problems and using supervised or unsupervised learning.

It is clear that NNs trained on an equilibrium ensemble of microscopic states of a many-body system can learn and predict phase transitions between macroscopic states *in many situations*. The big open questions are “why” and “how”. A natural hypothesis is that a trained NN in some sense learns about the critical behavior of the model—if that is the case, some NN observables should display a universal critical behavior and obey finite-size scaling governed by the universality class of the learned many-body system.

To test this hypothesis, we consider two spin models, the two-dimensional Ising model [8] and two-dimensional Baxter-Wu (BW) model [9]. The models belong to different universality classes and their critical behavior is described by different sets of critical exponents. We apply a unified approach for both models. First, we generate ensembles of equilibrium spin configurations across a range of temperatures using conventional spin-flip Metropolis [10] Monte Carlo algorithms. We then train NNs on these datesets and extract critical exponents from the NN outputs. We consider several NN architectures (a fully connected network, a convolutional network and several members of the ResNet family) and compare their predictions to (i) the values of critical exponents extracted from the training data via conventional finite-size analysis, and (ii) the known values from exact solutions for these two models.

The rest of the paper is organized as follows. In Sec 2 we define the spin models and briefly discuss the Monte Carlo process for generating the data sets. In Sec 3 we detail the NN architectures and our training pipeline. In Sec 4 we discuss the NN observables and our data analysis procedure, which results in estimates for the correlation length critical exponents ν for both Ising model and BW model. Finally, we discuss our results in Sec 5.

2 Data Sets for Spin Models

In this section, we present briefly two spin models and describe some details of the data set generation.

Ising Model— The Ising model is defined on a square lattice. Each Ising spin $\sigma = \pm 1$ interacts with four of its neighbors (left, right, top, bottom). The Ising model Hamiltonian is $H_{\text{is}} = -J/2 \sum_{\langle i,j \rangle} \sigma_i \sigma_j$, where J is the coupling constant. The model belongs to the eponymous universality class and is described by the set of critical exponents presented in the second entry of Table 1.

BW Model— The Baxter-Wu model is defined on a triangular lattice. Each spin interacts with six of its neighbors. Summation is performed over 3 spins at the vertices of each triangular plaquettes (faces). Baxter-Wu Hamiltonian $H_{\text{bw}} = -J \sum_{\langle faces \rangle} \sigma_i \sigma_j \sigma_k, \quad \sigma = \pm 1$ The arrangement of spins on a triangular lattice is stored as a two-dimensional array. The model belongs to the 4-state Potts [11] universality class [12–14].

We use periodic boundary conditions. Critical temperature $T_c = 2J/\log(1+\sqrt{2}) \approx 2.269(1)$, where $J = 1$.

Monte-Carlo Simulations— We generate equilibrium spin configurations using the Metropolis single flip Monte Carlo algorithm. The data represents spin configurations on the lattice, which can be viewed as red and blue pixels at the $(L \times L)$ square grid, where L is the number of vertices at horizontal and vertical directions. Each image (snapshot) is an instant lattice spin configuration, represented as blue (spin-down, $\sigma_i = -1$) and red (spin-up, $\sigma_i = 1$) pixels (Fig. 1).

Table 1: Values of the critical exponents.

Model	Universality class	α	β	γ	ν	γ/ν
BW	4-state Potts	2/3	1/12	7/6	2/3	7/4
Ising	Ising	0	1/8	7/4	1	7/4

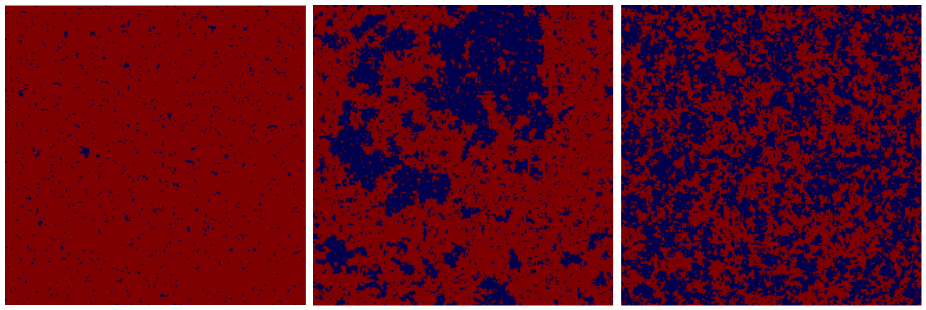

(a) $T = 1.8692$ (b) $T = 2.2697$ (c) $T = 2.7192$

Fig. 1: Ising spin configurations for $L = 216$ at different temperature T.

To generate a set of snapshots at each temperature, we perform relaxation from an initial random state ("hot start") to the state of thermodynamic equilibrium and then take a series of snapshots. We consider as a unit of measurement of computational complexity the Monte Carlo step (MCS). One MCS is equivalent to L^2 Metropolis local spin flips. For each flip we determine whether to change the value of the single spin to the opposite value or not:

– If changing the value of the spin lead to energy decrease by the amount ΔE, we accept it with probability $p^{acc} = 1$.
– Otherwise, we accept it with probability $p^{acc} = \exp(-\Delta E/T)$, with a random number rnd taken from a uniform distribution on $[0, 1]$ is less than or equal to p^{acc}.

Uncorrelated Snapshots— It is known that Markov chain Monte Carlo methods generate configurations which may be correlated especially around the critical point of second order phase transitions. In order to take the uncorrelated snapshots, we adopt recommendations [15] and start taking snapshots after $20 \cdot t_{corr}$ MCS, where t_{corr} is correlation time of magnetization. We take snapshots every $2 \cdot t_{corr}$ MCS to avoid correlations of images. We estimate relaxation time dependence from the lattice size t_{corr} equal to $L^{2.15}$ MCS in our previous study [16].

Data sets were generated for both models in the temperature interval $[T_c - 0.5; T_c + 0.5]$ for the Baxter-Wu model and $[T_c - 0.4; T_c + 0.4]$ for the Ising model. The Baxter-Wu data set contains 171 000 snapshots for each lattice size

(1500 images for each of 114 values of temperature). The maximum lattice size is $L = 243$. The Ising data set contains 189 000 images (1500 for each of 126 temperature points) and maximum lattice size is $L = 216$.

Generating data for lattice size 243 took approximately $3 \cdot 10^{15}$ spin flips. The total simulation time for each lattice size is given by formula:

$$\begin{aligned} N_{flips} &= 20 \cdot L^2 \cdot t_{corr} \cdot N_{T_{points}} + L^2 \cdot 2t_{corr} \cdot N_{images} \\ &= 20 \cdot L^{4.15} \cdot N_{T_{points}} + 2 \cdot L^{4.15} \cdot N_{images} \end{aligned}$$

3 Machine Learning

In this section, we describe the deep learning approach we use for the analysis of data sets described in the previous section.

3.1 Basics

Neural network (NN) architectures differ in the sequence of layers and building blocks that form them. Each layer has an input, an output, and can contain linear and non-linear operators. To build our NNs we use essential blocks like fully connected layers, convolutional layers [17], poolings and activation functions.

The fully connected layer (fc) consists of neurons (perceptrons [18]). Each neuron applies the scalar product of the weight vector to the output vector of neurons from the previous layer. An activation function is applied to the result and the output of the neuron is passed to the connected neurons of the next layer. The most common activation functions are sigmoid, ReLU (rectified linear unit), and SoftMax (normalized exponential function) [19]. The latter is commonly used in the output layer to normalize output values if the NN output has more than one neuron. SoftMax output values have a Boltzmann distribution.

The convolutional layer (conv) is a set of sliding windows (kernels). Each kernel, given dimensions and adjustable weights, applies the dot product to the input matrix and moves along it with a fixed stride.

Pooling (pool) is similar to convolution moving windows with a fixed kernel size and step, but it uses a function to aggregate the values in the input matrix. We use pooling with the function of **max**imum to take the highest value - max pooling (max pool).

3.2 Architectures

FCNN— Our fully connected NN (FCNN) architecture takes images reshaped into a 1-dimensional array of size L^2. An input is forwarded to a hidden fully connected layer of 100 neurons and sigmoid activation function. The output of the network is two neurons with a SoftMax activation function.

ConvNN— Our convolutional neural network (ConvNN) takes images as a 2-dimensional array ($L \times L$). An input is forwarded to 2-D convolutional layer (2x2) with 1 input channel and 64 output channels, stride 2 and padding 0. Convolutional layer output falls into max pooling with kernel size (2x2) and stride 2. After max pooling the array is reshaped into a 1-D vector of size $L/4 \cdot L/4 \cdot 64$. The vector is forwarded to a fully connected layer with the same number of neurons and ReLU activation function. The output of the network is two neurons with a SoftMax activation function.

ResNet— We use four types of deep convolutional residual network (ResNet) [20] architecture with different hidden layer depths. Our ResNet implementation has minor changes, as only one input channel is used in the first convolutional layer, unlike the classical ResNet designed for images with three color channels (RGB). ResNet takes images as a 2-dimensional array ($L \times L$) as an input. The NN forwarding is the same for all types. The output is also two neurons with a SoftMax activation function. We have described used ResNet types in a Table 2.

Table 2: ResNet architecture configurations.

10-layer	18-layer	34-layer	50-layer
7x7, 64, stride 2			
3x3 max pool, stride 2			
$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$ x1	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$ x2	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$ x3	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix}$ x3
$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$ x1	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$ x2	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix}$ x4	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix}$ x4
$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$ x1	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix}$ x2	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix}$ x6	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix}$ x6
$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$ x1	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$ x2	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix}$ x3	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix}$ x3
avg pool, 2-d fc, SoftMax			

3.3 Training Pipeline

Our problem is a binary classification, and two classes represent ferromagnetic (low temperature) phase and paramagnetic (higher temperature) phase. The NN is trained using the backward propagation of errors algorithm (backprop) to update NN adjustable parameters (weights). We chose Adam optimizer (Adaptive Moment Estimation [21]) and fixed the learning rate at 10^{-4}. The binary cross-entropy (BCE) is a loss function to measure errors when training.

We divide data sets in two unequal parts: **2/3** of generated data used for training and validation and **1/3** used for testing. The validation data is 10% of the training. Training stage takes place in epochs. During training the data is divided into batches of size 36. We measure the mean loss function on the validation data for each epoch to make sure there is no overfitting. It took us 10 epochs to ensure we are not facing overfitting and the learning curve has plateaued. Additionally, we use augmentation heuristic to increase NN generalization ability by 4 times increasing training data size: each image is rotated by $\pi/2, \pi, 3\pi/2$ radians.

Before the test data inference, we choose the best NN parameters by the minimum loss function on the epoch for each lattice size. For all architectures, the BCE on the validation data does not exceed 0.36 (Ising), 0.4 (BW) and for the maximum lattice sizes it drops to the level of 0.16 (Ising), 0.05 (BW). The minimum value of the accuracy metric is 84% (Ising), 82% (BW) for lattice size 48 and reaches 92% (Ising), 98% (BW) for the maximum lattice sizes 216 and 243, respectively. The use of the accuracy metric is justified due to the balance of classes. With such values of the quality metric, we can conclude that the NN works better than random or constant (50% accuracy) classifier.

For input image at fig. 1a the NN would output a correct prediction − 1 for ferromagnetic phase and 0 for paramagnetic phase. For image at fig. 1c, the network is also output correct predictions but an opposite values 0 and 1 respectively.

All neural network models are implemented in Python using the Pytorch [22] library. NNs training was carried out on 1x NVIDIA Tesla V100-SXM2 32 GB configuration. The Table 3 summarizes the data on the training computational cost in seconds per epoch (*time*) and the quantity of NN adjustable parameters (*# parameters*) for Ising model with lattice size $L = 48$.

Table 3: Ising model computational summary.

NN type	# parameters	Time, s/ep.
ConvNN	590 336	108(5)
FCNN	230 702	66.0(3)
ResNet-10	4 900 546	534(4)
ResNet-18	11 171 266	1200(72)
ResNet-34	21 279 426	2369(111)
ResNet-50	23 505 858	2590(145)

4 Exponents Estimation

We obtain critical exponents using the conventional method from Monte Carlo data and then compare them to the NN method.

MC Analysis— We evaluate the mean energy per spin E and magnetization per spin M to construct the heat capacity C and magnetic susceptibility χ. The set of equations used is as follows

Baxter-Wu model:

$$\begin{aligned} e &= \frac{1}{L^2} H_{\mathrm{bw}}(\{\sigma\}) \\ m &= \frac{1}{L^2} \sqrt{\sum_{lat=1}^{3} m_{lat}^2} \\ C &= \frac{L^2}{T^2} \left(\langle e^2 \rangle - \langle e \rangle^2\right) \\ \chi &= \frac{L^2}{T} \left(\langle m^2 \rangle - \langle m \rangle^2\right) \end{aligned} \tag{1}$$

Ising model:

$$\begin{aligned} e &= \frac{1}{L^2} H_{\mathrm{is}}(\{\sigma\}) \\ m &= \frac{1}{L^2} \left| \sum_{i=1}^{L^2} \sigma_i \right| \\ C &= \frac{L^2}{T^2} \left(\langle e^2 \rangle - \langle e \rangle^2\right) \\ \chi &= \frac{L^2}{T} \left(\langle m^2 \rangle - \langle m \rangle^2\right) \end{aligned} \tag{2}$$

We use finite-size analysis and extract exponents analysing dependence from the lattice size of the values of C_{max} and χ_{max}, the position of the maxima, and the width of the $C(T)$ and $\chi(T)$ curves.

NN Analysis— The NN output neurons return estimates of Bayesian posterior probability [23] that the input image belongs to ferromagnetic $p_i^f(T)$ and paramagnetic $p_i^p(T)$ phases, $p_i^f(T) + p_i^p(T) = 1$. The test data set consists with $N_{test} = 500$ snapshots at each value of temperature. The average value $F(T)$ and variance $V(T)$ of the NN ferromagnetic output neuron $p_i^f(T)$ for each temperature are

$$F(T) = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} p_i^f(T) \tag{3}$$

$$V(T) = \left(\frac{1}{N} \sum_{i=1}^{N_{test}} (p_i^f(T))^2 \right) - \left(\frac{1}{N} \sum_{i=1}^{N_{test}} p_i^f(T) \right)^2 . \tag{4}$$

The critical exponents can be obtained by fitting the curve $V(T)$ using the probability density function (**pdf**) of the Gaussian distribution multiplied by k

$$\begin{aligned} V_{gauss}(k,\mu,\sigma) &= k \cdot \mathrm{pdf}(\mu,\sigma) \\ &= \frac{k}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right). \end{aligned} \tag{5}$$

As a result, we obtain estimates of peak positions μ and standard deviation σ. We extract the critical exponent ν observing how the peak position $\mu(L)$ tends to the analytically known critical temperature $\mu(L) - T_c \sim L^{-1/\nu}$.

We found that the width $\sigma(L)$ of $V(T)$ changes as the lattice size increases $\sigma(L) \sim L^{-1/\nu}$.

One can expect that the width of the curve $V(T)$ in the low-temperature $V(T < T_c)$ and high-temperature $V(T > T_c)$ phases may not be the same. Therefore, we also fit separately the curves to the right and to the left of the $V(T)$ peak maximum with a power law.

We construct the dependencies of these parameters on the lattice size L.

Tables 5 and 4 summarize our results both using conventional finite-size analysis of Monte Carlo data and those obtained from the analysis of NN output.

Table 4: Estimates of exponents for the Baxter-Wu model with different methods.

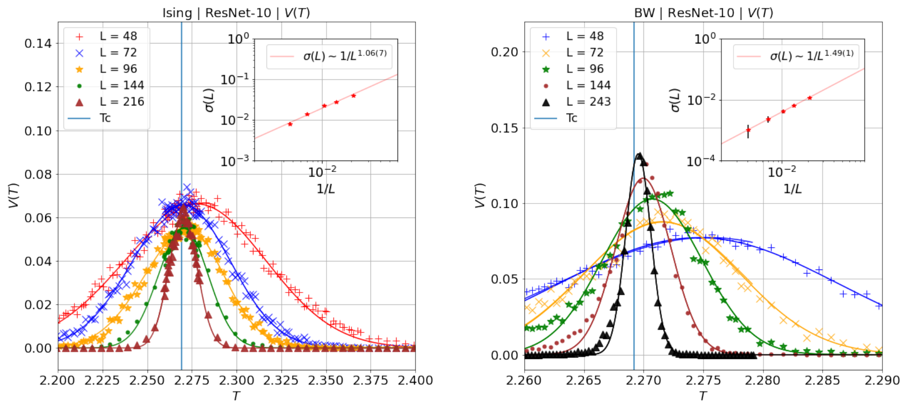
Method	$\left(\frac{1}{\nu}\right)_\mu$	$\left(\frac{1}{\nu}\right)_\sigma$	$\left(\frac{\gamma}{\nu}\right)_{peak}$
MC (C)	1.54(2)	1.48(1)	-1.01(2)
MC (χ)	1.52(3)	1.30(6)	-1.83(3)
ConvNN	1.36(24)	1.49(2)	-0.22(2)
FCNN	1.49(6)	1.46(2)	-0.29(4)
ResNet-10	1.62(7)	1.49(1)	-0.32(3)
ResNet-18	1.48(14)	1.50(3)	-0.32(3)
ResNet-34	1.55(10)	1.50(2)	-0.30(3)
ResNet-50	1.35(7)	1.52(1)	-0.35(6)

5 Discussion

We are comparing two methods of extraction of critical exponents using finite-size scaling analysis—a FSS analysis of the Monte-Carlo data, and a FSS analysis of the fluctuations of outputs of trained neural networks. We find that the NN output fluctuations $V(T)$ contain an information about the exponent correlation length exponent, ν. We estimate the critical exponent ν using finite-size scaling from the NN output fluctuations. The best estimate of ν for the Baxter-Wu

Table 5: Estimates of exponents for the Ising model with different methods.

Method	$\left(\frac{1}{\nu}\right)_{\sigma}$	$\left(\frac{\gamma}{\nu}\right)_{peak}$
MC (C)	0.95(1)	-
MC (χ)	1.02(5)	-1.77(2)
ConvNN	1.10(9)	-0.025(126)
FCNN	0.71(5)	0.043(46)
ResNet-10	1.06(7)	0.087(85)
ResNet-18	1.03(7)	-0.324(354)
ResNet-34	1.07(8)	0.073(108)
ResNet-50	1.09(11)	0.098(142)



(a) Ising $1/\nu$ from $V(T)$

(b) Baxter-Wu $1/\nu$ from $V(T)$

Fig. 2: Estimating $1/\nu$ for Ising and BW models from $V(T)$, Eq. (4). See text for discussion.

model is extracted (cf. Fig. 2b) with the same accuracy for both MC and NN approaches, and the results agree within the combined errorbars. Likewise, for the Ising model (cf. Fig. 2a) the MC and NN estimates have similar accuracy and are consistent within the combined errorbars.

While the width of the NN output fluctuation, $V(T)$, displays finite-size scaling consistent with the correlation length exponent ν, the scaling of the height of the peak of $V(T)$ is not clear. Specifically, the peak height for the BW model scales according to a power law $L^{0.30(3)}$ but the extracted values are not similar to any known combination of exponents for this model. The peak height for Ising model displays no finite-size scaling at all. Note that our results thus do not agree to the results of Ref. [3] which reported the ratio of exponents $\gamma/\nu = 1.78$ from the scaling of the peak height.

We obtain a smaller error bars and a smaller coefficient of variation for critical exponent ν on the level of 5% comparing with the result of paper [1] on the level of 30%. Our numerical results for the peak position for the Baxter-Wu model are roughly consistent with the $1/\nu$ scaling. Given that the Ising model does not display a shift at all, more work is needed to reliably assess whether the peak position is a reliable estimator for the critical exponent.

We have tested ResNet family with different depths from 10 to 50 layers and found no evidence that the quality of the critical exponent extraction depends on the number of convolutional layers.

6 Conclusions and Outlook

We investigate the applicability of deep learning techniques to studying critical phenomena. We consider two classical models, the two-dimensional Ising model, and a Baxter-Wu model, and benchmark the NN approach on the exact solutions. We use supervised learning, where the NNs are trained on ensembles of spin configurations generated by spin-flip Metropolis algorithm Monte Carlo simulations. The samples are labeled by a binary variable, whether a given is generated at a temperature above or below the critical temperature of the ferromagnetic phase transition. We also compare the results of the NN analysis to a traditional finite-size analysis of the Monte Carlo data sets.

We find that the fluctuation of the NN output as a function of temperature has a characteristic Gaussian shape. The parameters of the Gaussian depend on the lattice size. The width of the Gaussian displays a power-law dependence on the lattice size, which is consistent with the correlation length exponent ν of the corresponding spin model. We can thus conclude that the NN learns not only the location of the phase transition, but also (some) critical exponents of the universality class of the model.

We consider three different NN architectures: the fully connected network, a simple convolutional network and several members of the ResNet family. We find that for both models, the ResNet NNs achieve best accuracy of the estimates of the correlation length critical exponent. On the other hand, the quality of the ResNet estimates only weakly depends on the depth of the network: ResNet50 does not significantly improve on predictions of ResNet10. On the other hand, both reliability and accuracy of estimates of the critical exponent is clearly improved by ResNet NNs, as compared to both FCNN and a simple convolutional NN. This observation should be compared with Ref [5], which reported that shallow networks perform better than deep ones for the Ising model near criticality.

For future work, an important question is the accuracy and reliability of transfer learning: whether and to what accuracy an NN trained on one model, predicts critical properties of a different model in the same universality class. One other big question is whether NN learns only the correlation length exponent, or if other critical exponents can be extracted from the NN outputs.

Acknowledgements This work is supported by the grant 22-11-00259 of the Russian Science Foundation. Simulations were done using the computational resources of HPC facilities at HSE University [24].

References

[1] Juan Carrasquilla and Roger G Melko. "Machine learning phases of matter". In: *Nature Physics* 13.5 (2017), pp. 431–434.

[2] Giuseppe Carleo and Matthias Troyer. "Solving the quantum many-body problem with artificial neural networks". In: *Science* 335 (2017), p. 602. URL: https://www.science.org/doi/abs/10.1126/science.aag2302.

[3] Dimitrios Bachtis, Gert Aarts, and Biagio Lucini. "Mapping distinct phase transitions to a neural network". In: *Physical Review E* 102.5 (2020), p. 053306.

[4] E. van Nieuwenburg, YH. Liu, and S. Huber. "Learning phase transitions by confusion". In: *Nature Physics* 13 (2017), pp. 435–439.

[5] Alan Morningstar and Roger G. Melko. "Deep Learning the Ising Model Near Criticality". In: *Journal of Machine Learning Research* 18.163 (2018), pp. 1–17. URL: http://jmlr.org/papers/v18/17-527.html.

[6] Tom Westerhout et al. "Generalization properties of neural network approximations to frustrated magnet ground states". In: *Nature Communications* 11 (2020), p. 1593.

[7] Nicholas Walker and Ka Ming Tam. "InfoCGAN Classification of 2-Dimensional Square Ising Configurations". In: *arXiv preprint arXiv:2005.01682* (2020). URL: https://arxiv.org/abs/2005.01682.

[8] Lars Onsager. "Crystal statistics. I. A two-dimensional model with an order-disorder transition". In: *Physical Review* 65.3-4 (1944), p. 117.

[9] Rodney J Baxter and FY Wu. "Ising model on a triangular lattice with three-spin interactions. I. The eigenvalue equation". In: *Australian Journal of Physics* 27.3 (1974), pp. 357–368.

[10] Nicholas Metropolis et al. "Equation of state calculations by fast computing machines". In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[11] Renfrey Burnard Potts. "Some generalized order-disorder transformations". In: *Mathematical proceedings of the cambridge philosophical society*. Vol. 48. 1. Cambridge University Press. 1952, pp. 106–109.

[12] MPM Den Nijs. "A relation between the temperature exponents of the eight-vertex and q-state Potts model". In: *Journal of Physics A: Mathematical and General* 12.10 (1979), p. 1857.

[13] Robert B Pearson. "Conjecture for the extended Potts model magnetic eigenvalue". In: *Physical Review B* 22.5 (1980), p. 2579.

[14] Bernard Nienhuis. "Critical behavior of two-dimensional spin models and charge asymmetry in the Coulomb gas". In: *Journal of Statistical Physics* 34.5 (1984), pp. 731–761.

[15] Alan Sokal. "Monte Carlo methods in statistical mechanics: foundations and new algorithms". In: *Functional integration*. Springer, 1997, pp. 131–192.

[16] Vladislav Chertenkov and Lev Shchur. "Universality classes and machine learning". In: *Journal of Physics: Conference Series*. Vol. 1740. 1. IOP Publishing. 2021, p. 012003.

[17] Kunihiko Fukushima and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[18] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6 (1958), p. 386.

[19] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[20] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[21] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[22] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[23] Michael D Richard and Richard P Lippmann. "Neural network classifiers estimate Bayesian a posteriori probabilities". In: *Neural computation* 3.4 (1991), pp. 461–483.

[24] PS Kostenetskiy, RA Chulkevich, and VI Kozyrev. "HPC resources of the higher school of economics". In: *Journal of Physics: Conference Series*. Vol. 1740. 1. IOP Publishing. 2021, p. 012050.