Phase determination by Deep Learning for disordered systems

by

Djénabou Bayo
Thesis

Submitted to the University of Warwick

and

CY Cergy Paris Université

for the degree of

Doctor of Philosophy

Department of Physics

January 2024

Président du jury/ Head of the jury: Julie B. Staunton

André Muracciole
1941-2022

Thérèse Esquerré
1939-2023

Contents

List of Tables

List of Figures

Acknowledgments

A PhD offers the first real step into the world of research. Through these four years, notwithstanding a difficult beginning due to a global pandemic, I had the opportunity to mingle with scientists from around the world and exchange ideas. While conferences usually offered scope for some academic exchange, the Eutopia co-tutelle made it more a part of my routine. I am deeply grateful for this opportunity to realise my PhD journey at two universities – the University of Warwick and CY Cergy Paris University. This enriching experience allowed me to explore two different departments and apprehend the different local attitudes towards work, research and life.

First and foremost, I would like to express my sincere gratitude to my two supervisors, Prof. Rudolf A. Römer and Prof. Andreas Honecker. Both of them taught me how to be rigorous and determined. I only hope to become as good a scientist as my two supervisors.

I am grateful to Rudo for his continued support and guidance. Especially, at the beginning of my doctoral research, he helped me feel welcome at Warwick during the Covid-19 pandemic. He kept me motivated and interested in my research, even when the emptiness of the department was weighing down on me. I will always admire his curiosity and the versatility of his research subjects. I also have to acknowledge his determination to always push me to strive for greatness and to encourage me to present my work to others, even when I was more than reluctant.

I sincerely thank Andreas for always pushing me to aim higher even when I was a mere master's student at CY Cergy Paris Université. Despite my initial reluctance, he convinced me to recognize and tap my potential to succeed in my

PhD journey. He is the reason that I even thought of applying for a PhD. My regular discussions with him, over these years, helped me find and explore different perspectives on my research project.

I had the joy of evolving from a master's student to a PhD scholar at the LPTM. I would like to express my profound gratitude to the team who have known me since I was a master's student, particularly to Geneviève Rollet. She always believed in me, instilled confidence in myself, and pushed me to overcome my apprehensions. I also use this opportunity to thank Vita Ilakovac who was the first to introduce me to research through my master's internship at the LCPMR. I am also grateful to Prof. Tomi Ohstuki from Sophia University, for the extensive discussions, his patient help and support on issues concerning the Anderson model.

A PhD is never a solitary journey. I am grateful for having the opportunity to work with my group – my dear friend Burak, Jie, Syl, Quangminh, Gao and Kasia. Despite a move to the UK in 2020 in the middle of a global pandemic, I had the chance to meet wonderful people who helped me settle in Coventry. I am especially thankful to Leo, Leti, Eva, Stephen and Aiman for giving me a warm welcome into their social group in the department during these trying times.

I am grateful to Sharmila for providing me with a great model of what a Postdoctoral researcher should be and to Aaron for being a great colleague always eager to exchange about our respective research projects. I would like to thank both of them for being so patient with me during my thesis writing period and for coping with my questions. This is the kind of friendship that we cherish for life, I will always remember our tea breaks or our little trips fondly. Special thanks to Sourav for being the best storyteller that I had the opportunity of meeting. Our short breaks were always a great source of entertainment for me. For general support through the years, I also acknowledge my colleagues and friends across CY Cergy Paris University and the University of Warwick – a thank you to Mohamed, Branda, Quentin, Alexios, Jawad, Yunos, Benoit, Dipanjan, Elnaz and Ellen for being such good company.

I cannot *not mention* my two favourite housemates-turned-friends, Ashwani and Sam. Words cannot express my gratitude to them for entertaining me after a full day of work (and also for removing spiders from the house). I will always cherish our animated and weird conversations in the kitchen. Those memories will remain with me forever.

I also thank Eva-Rosa, a fellow EUTOPIA student who became a great friend through the year that we spent hunting for the best restaurant in town. I am grateful to my friends, Julie, Fanny and Lena – thank you for supporting me, even if none of you really seemed to remember what I am doing. To my dearest friend Charlène, I am so happy that we got to go through our respective PhD journeys at the same time. I will always be there to support you as you are supporting me and I wish you (and also Fanny) well for the success of your PhD. I am deeply indebted to my family, Fode, Mame Sophie, my brother, my sister and my parents without whom I would not have been able to achieve any of this. Finally, to Mamie and Tonton André, I only wish that the two people who shaped me into the person that I am could be here to see me pass this milestone. I dedicate this thesis to both of them.

Declarations

This thesis is submitted to the University of Warwick and CY Cergy Paris Université in the framework of an EUTOPIA co-tutelle as part of my application for the degree of Doctor of Philosophy in Physics. The first year of the PhD was spent at the host institution CY Cergy Paris Université in France, and the remainder was spent at the Home institution at the University of Warwick. This thesis contains work composed by myself and my collaborators and has not been submitted in any previous application for any degree or other university. I declare that this thesis is my own work except where indicated. Part of the work presented in this thesis is based on published articles. Chapters 3 and 4 are based upon:

- D. Bayo, A. Honecker, and R. A. Römer, "Machine learning the 2D percolation model," Journal of Physics: Conference Series, vol. 2207, p. 012057, 3 2022 [1]
- D. Bayo, A. Honecker, and R. A. Römer, "The percolating cluster is invisible to image recognition with deep learning", New Journal of Physics, vol. 25, p. 113041, 11 2023 [2]

Abstract

In the last few years, we have witnessed a growing number of machine learning publications in the field of condensed matter and statistical physics. In particular, machine learning (ML) methods seem to perform well in tasks such as the identification of phases of matter. In this thesis, we study machine learning through the scope of two models. Our first model is the two-dimensional site percolation. 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 explore the ability of convolutional neural networks (CNNs) to predict the density of occupation p of percolation states, the correlation length ξ, as well as the presence of a spanning cluster. We find that image recognition tools such as CNN, which are not naturally tailored for physics, successfully identify p. However, when dealing with parameters like ξ or the presence of a spanning cluster, these same techniques fail to provide quantitative results.

The second model is the three-dimensional Anderson model of localisation. This model is characterised by a localisation of the wavefunctions above a critical disorder W_c. We begin by reproducing previous work done on phase classification, and perform several new studies with classification and regression methods, to identify individual disorders in both phases. Throughout our investigation, multiple parameters such as the size of the system or the nature of the input are studied to observe their influence on the performance of the model. Via the study of these two models and the use of several ML methods, we will display the successes and limitations that one might be confronted with when using ML for phase recognition.

Abstract en français

Ces dernières années, nous avons vu l'émergence d'un grand nombre de publications de machine learning (ML) dans les domaines de la physique de la matière condensée et de la physique statistique. En particulier, les outils de ML apparaissent comme des méthodes valides pour l'identification de phase. Dans cette thèse, nous étudions le ML sous le spectre de deux modèles. Le premier est le modèle de percolation de site en deux dimensions. Dans ce modèle paradigmatique, les sites sont occupés avec une probabilité p; une transition de phase du second ordre d'une phase non-percolante à une phase percolante apparait à une probabilité d'occupation p_c, appelé seuil de percolation. A l'aide de méthodes d'apprentissage supervisé telles que la classification et la régression, nous explorons les capacités des réseau de neurones convolutifs (CNNs) à prédire la densité d'occupation p, la longueur de corrélation ξ, ainsi que la présence d'amas percolant. Nous constatons que les CNNs, qui ne sont à la base pas pensés pour la physique arrivent à prédire p. Cependant pour ξ ou la présence d'amas percolant, ces mêmes techniques ne parviennent pas à donner de résultats satisfaisants. Le second modèle est le modèle de localisation d'Anderson en trois dimensions. Ce modèle se caractérise par une localisation de la fonction d'onde au-delà d'un désordre critique W_c. Nous commençons par reproduire des résultats obtenus précédemment en classification de phase, et réalisons par la suite des études dans le but d'identifier plusieurs valeurs de désordres dans les deux phases. Au cours de nos recherches, nous étudions l'influences de la taille du système ou la nature de l'entrée sur la performance du réseau. Au travers de l'étude de ces deux modèles, nous montrons les points forts et les limitations auxquels il est possible d'être confrontés en utilisant le ML pour la reconnaissance de phase.

List of Abbreviations

AI Artificial Intelligence. 1, 2

ANN Artificial Neural Network. 2, 3, 35

BCE Binary Cross-Entropy. 46

CCE Categorical Cross-Entropy. 31, 38

CE Cross Entropy. 31

CNN Convolutional Neural Network. 5–7, 58, 61, 64, 68, 69, 78, 82, 84, 86, 96–99

DL Deep Learning. 4, 35, 57, 62, 64, 66–68, 72, 74

DNN Deep Neural Network. 3, 7, 8, 36

ELBO Evidence Lower BOund. 44

GAN Generative adversarial network. 7

GD Gradient Descent. 33

GPU Graphic Processing Unit. 2

HK Hoshen-Kopelman. 15, 25, 49, 50, 65, 97

KL Kullback Leibler. 31, 44

LOTUS Law of The Unconscious Statistician. 45

MIT Anderson Metal-Insulator Transition. 25, 84

ML Machine Learning. 1, 3, 6, 7, 9, 18, 25, 27, 32, 38, 44, 47, 48, 53, 96–98

MSE Mean Squared-Error. 28, 30, 40

NN Neural Network. 2

ReLU Rectified Linear Unit. v, 4, 70

ResNet Residual Network. v, 7, 37, 38, 40, 55, 57, 69, 74, 78, 84–87, 89

SGD Stochastic Gradient Descent. 34, 47

VAE Variational Autoencoder. v, 7, 41–44, 46, 70

Chapter 1

Introduction

1.1 A brief history of machine learning

Throughout history, the concept of an object or machine being able to learn autonomously has always fascinated philosophers and scientists. However, the foundation of modern Machine Learning (ML) as we know it, dates back to the 40s. In 1943, McCulloch and Pitts proposed a mathematical model emulating the behaviour of neurons in the brain: this is the artificial neuron, which is the building block of our modern artificial neural network [3]. With this formalism established, a new field emerged aiming at improving the learning capacity of machines by mimicking the behaviour of neurons in the brain. F. Rosenblatt was the first to successfully implement the model of McCulloch and Pitts and named it perceptron [4]. While this discovery generated a lot of enthusiasm in the field, it was soon quelled by the limitations of the perceptron. Indeed, by 1969 Minsky and Papert proved that this first implementation was only able to distinguish simple patterns contrary to Rosenblatt's early claim [5]. This disillusion and the lack of practical application of the perceptron led to a loss of interest from investors and marked the beginning of what was later called the first Artificial Intelligence (AI) winter [6]. A new interest grew in the 80s helped by two main events, through Hopfield who demonstrated the ability of the simple network to calculate [7] and more importantly the research of D. E. Rumelhart, G. E. Hinton and R. J. Williams [8] which were the first to successfully implement the backpropagation algorithm on an artificial neural network. However, these headways were not sufficient to relaunch sustained interest in the field, which eventually led to a second AI winter by the end of the 80s. In the decades that followed, we observed the publication of several important articles in the field [9, 10] but interest only returned in the early 2000s.

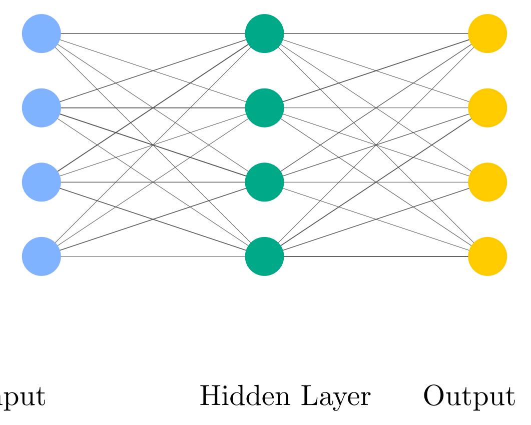


Figure 1.1: Artificial neural network with three layers, an input layer in blue, a hidden layer in green and an output layer in yellow. Each layer comprises four neurons stacked on top of each other. The layers are connected through weighted connections. To emphasise this point, here, we draw connections with different thicknesses. Connections with high weights appear thicker than connections with lighter weights.

In the 2010s we have seen a boom in the field of AI, due to the development of technologies such as Graphic Processing Units (GPUs) and the creation of tools such as CUDA [11, 12].

1.2 Artificial Neural Networks

It is important to emphasise the multidisciplinary effort that contributed to the rise of machine learning. After all, AI was developed with the intent to imitate the human mind. Thus, psychologists, biologists and computer scientists worked hand in hand to further the understanding that scientists had about the functioning of the human brain to replicate it in machines. Among the mist of articles that pave the way to modern machine learning, are the articles of Hubel and Wiesel. Through a series of publications, Hubel and Wiesel [13–15] studied the visual cortex of cats, and made several discoveries that contributed to shaping the functioning of image recognition tools. They found that individual neurons had receptive fields, the whole vision being recovered through overlaps of the receptive fields of several neurons. Nonetheless, a neuron would not always perform the same task as other neurons in the visual cortex. Some neurons specialised in identifying certain patterns such as lines, and different neurons would recognise different line orientations, i.e., horizontal or vertical lines.

Artificial Neural Networks (ANNs) also simply called Neural Networks (NNs)

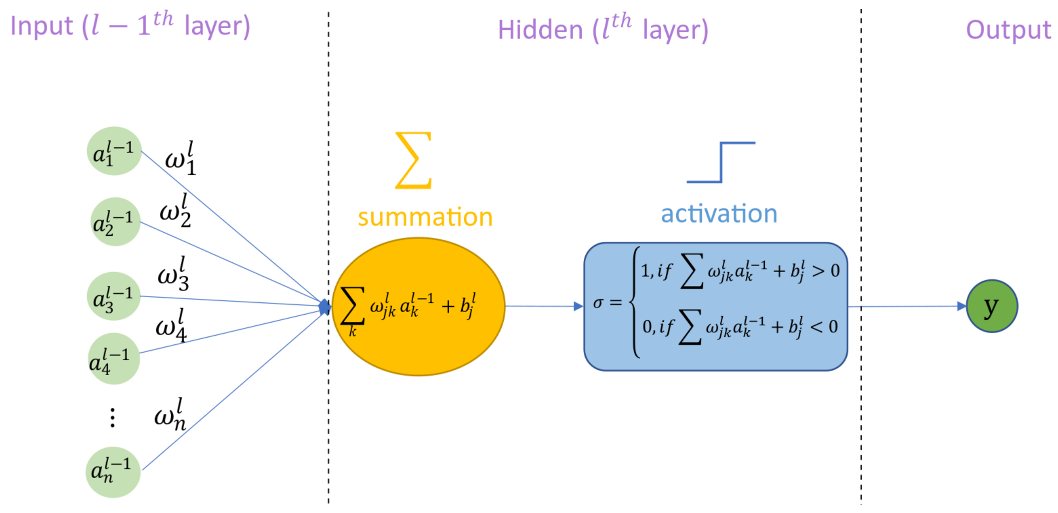


Figure 1.2: Representation of an artificial neuron, on the left we see the layer $l-1$ composed of n neurons. The yellow ellipse displays the operation undergone by the j^{th} neuron of layer l and the blue rectangle the activation function applied after layer l.

are a type of model used during an ML training. They are composed of nodes called neurons, which are stacked on top of one another in layers. Each layer is connected to the next through vertices. In the cases where all the neurons from one layer are connected to all the neurons of the subsequent, as shown in figure 1.1, we say that we have a dense or fully connected layer. The first ANN was the extreme learning machine introduced by Rosenblatt in his book *Perceptron* [4]. It was composed of an input layer, a single hidden layer and an output layer. While this network constituted a revolution in the field, it was later discovered that this type of network was only able to identify simple patterns and was not able to perform complex tasks. Further subsequent studies showed that adding several hidden layers contributed to enhancing the performance of the model [16]. A neural network with more than one hidden layer is then called a Deep Neural Network (DNN).

As we previously mentioned, ANNs are used for performing an ML task. The training occurs after feeding an input to a network. Following this, the information goes from layer to layer until it reaches the output where the performance is evaluated. Let us consider a DNN with $l = 1, 2, \ldots, M$ layers. Each neuron j in a given layer l has a value called activation a_j^l and each of the lines connecting one neuron k from a layer $l-1$ to the neuron j in layer l have a weight ω_{jk}^l. The information passes from one layer $l-1$ to layer l through a linear combination of the activations

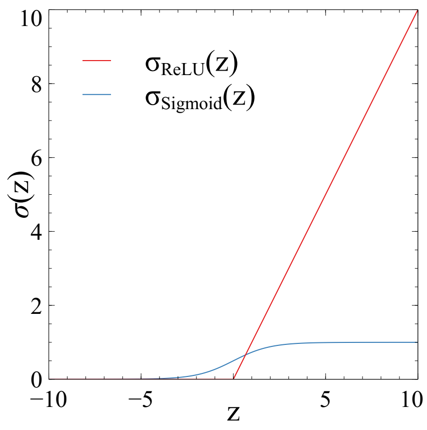


Figure 1.3: In red the $\sigma_{\mathrm{ReLU}}(z)$ activation function. In blue the $\sigma_{\mathrm{sigmoid}}(z)$ function, we remark that this function saturates after $z = 2$ and has a maximum value of 1.

and the weights of the previous layer

$$a_j^l = \sigma\left(\sum_k \omega_{jk}^l a_k^{l-1} + b_j^l\right) = \sigma\left(z_j^l\right), \tag{1.1}$$

where σ is called the activation function and b_j^l is called the bias which essentially allows to shift the activation function from the origin. Similarly to the behaviour of neurons in the brain, all the neurons in a network are not firing at the same time, the role of σ is to find a way to determine which neurons are active. Several different types of activation/non-linearity can be applied to layers, which in turn lead to different training performances [17]. Some of the commonly used activation are Rectified Linear Unit (ReLU) $\sigma_{\mathrm{ReLU}}(z) = \max(0, z)$ or sigmoid $\sigma_{\mathrm{sigmoid}}(z) = \frac{1}{1+e^{-z}}$, shown in figure 1.3. We provide an example of the output of an artificial neuron/perceptron in figure 1.2, here we use a step function, $\sigma_{\mathrm{step}}(z) = \begin{cases} 0, & \text{if } z < 0 \\ 1, & \text{if } z > 1 \end{cases}$ as activation function.

1.3 CNN methods for image recognition

In 1980, we see the first attempt at reproducing the behaviour of natural neurons described by Hubel and Wiesel, this is the neocognitron [18]. The neocognitron had two types of layers that we nowadays identify as a convolutional layer and a downsampling layer. The neocognitron was only using feed-forward techniques but was already able to identify patterns or characters. The first Deep Learning

(DL) Convolutional Neural Network (CNN) was implemented several years later by Yann Lecun et. al [19]. This network was using backpropagation as an optimisation method and was able to identify hand-written ZIP numbers.

1.3.1 The convolutional layer

Now that we have provided a brief background on the timeline of the creation of CNNs, let us look at its architecture in detail. A CNN is composed of two main layers, the convolutional layer and the downsampling layer, also called the pooling layer. The convolutional layer differs from the fully connected layer by introducing the notion of receptive field [17]. Contrary to the dense layer shown in figure 1.1, where neurons in a layer were connected to all neurons of the following layer, here a neuron is only connected to neurons that are in its receptive field. This receptive field, also called kernel, allows to retain local correlation between neighbouring pixels. Similarly to the neurons in the visual cortex of animals, the full image is retrieved through the overlap of the receptive field of several neurons. The training occurs through convolution operation. When the kernel which is a weighted matrix, convolves across the input image. To construct a convolutional layer five parameters need to be defined. We begin by defining the width W, and the height H of the receptive field of our network, which is called a *kernel*. Usually, the height H is chosen as equal to W. Each layer applies a given number of kernels to the input, this number is called the *depth* D. Now that we have defined the size of our kernel/receptive field as $W \times H$, we need to define the *stride* S. This is the number of neurons chosen between two consecutive translations of the kernel. Finally, as the input goes through our network its dimensions might decrease. To remedy this issue we define a parameter P called the *padding*, which adds zero around the input to keep its dimension constant. After going through a convolutional layer with a kernel of size $W \times H$, an input of size $I \times I$ has a new size S_{output} defined as

$$S_{\text{output}} = \frac{I - W + 2P}{S} + 1, \tag{1.2}$$

1.3.2 The pooling layer

The second layer in the CNN architecture is a downsampling layer, also called the pooling layer [17, 20]. This is a coarse-graining layer which retains the spatial structure of the input. A pooling layer does not modify the *depth* of the input, pooling operations are applied to each of the filters taken in input. Similarly to the convolutional layer, the pooling layer is not connected to all the neurons in the subsequent layer, only to neurons in their receptive field. A pooling layer is defined by the size of the kernel $W_{\text{Pool}} \times H_{\text{Pool}}$, and the stride S is usually taken to be equal to W. Several types of pooling operations exist, the most popular being max pooling and average pooling. Let us consider a $W_{\text{Pool}} \times H_{\text{Pool}}$ pooling kernel. In a max pooling layer, the kernel analyses the value of each pixel in the receptive field $W_{\text{Pool}} \times H_{\text{Pool}}$ and replaces it with one value corresponding to the value of the maximum pixel. In an average pooling layer, a $W_{\text{Pool}} \times H_{\text{Pool}}$ replaces the pixel in the receptive field with the average value of the pixels in the kernel window. Pooling operations allow for a decrease in the computational load and in turn, allow to work with deeper networks.

CNN methods are usually used for image recognition, they have been shown to be performing well for image or video recognition, natural language processing, and medical image analysis [21–24].

1.4 Types of learning

ML distinguishes itself from classical programming by the fact that no rule is given to solve a task, instead, we expect the network to find a strategy to achieve the task required according to the dataset given as input. Let us define the dataset $D = (x_i, y_i)$ where x_i is a tensor of independent variables representing the input, and y_i is a tensor of dependent variables representing the label of input x_i. This dataset represents a general input that we would feed to our ML model. In our case x_i could be a percolation lattice ψ_i and y_i the label associated, for instance 'spanning'. Before introducing the different kinds of tasks that could be performed by an ML program we need to introduce the different types of learning. There are three main types of learning, the first one is *supervised learning*. Supervised learning aims at acquiring the optimal strategy to perform a task through the use of a labeled dataset $D = (x_i, y_i)$. Two types of supervised learning tasks can be distinguished, classification and regression. For *classification*, an ML process learns how to separate data in discrete classes. It is equivalent to finding an optimal representation of the dataset that separates samples from each class. A famous example is the classic Cat/Dog classification [25]. In the case of *regression*, the ML algorithms are trained to learn the relationship between the input x_i and the label y_i to make continuous predictions of new y_j according to x_j. Regression can be used to infer the price of a car according to several parameters such as the brand, and year of release. In that sense, regression differs from classification as it allows us to make predictions on samples with labels never seen by the network during the training procedure. Both of these methods will be developed in section 3.10 with examples of applications.

The second type of learning is *unsupervised learning*. In this method unlabeled data is fed to the ML algorithm, i.e. $D = (x_i)$. The network is then expected to find an implicit correlation in the dataset without any other external input. Unsupervised learning can be divided into three different categories, clustering, dimensionality reduction, and association learning. The clustering task intends to find ways to group similar samples in the dataset. Dimensionality reduction methods aim at finding ways to simplify representations of datapoints in a way that preserves the main feature of the dataset, and association learning studies relations between samples in the dataset. Applications of unsupervised learning techniques are wide. It can serve as a preprocessing step to find structure in the dataset before a supervised learning training [26], or can be used to generate new samples with techniques such as Variational Autoencoder (VAE) or Generative adversarial network (GAN).

Finally, the last type of learning is *reinforcement learning*. Contrarily to the two previously presented techniques no dataset is employed during the learning process. The learning occurs through an agent interacting with an environment, if the agent performs the task as expected, it receives a bonus, and if the task is not performed well the agent receives a penalty. Through a set of interactions with the environment, the agent learns how to perform the task as expected.

1.5 Machine learning for phases recognition

As we have seen in the previous section, CNNs are a class of deep, i.e., multi-layered, neural nets (DNNs) in which spatial locality of data values is retained during training in an ML setting. When coupled with a form of residual learning [27], the resulting residual networks (Residual Networks (ResNets)) have been shown to allow astonishing precision when classifying images, e.g., of animals [28] and handwritten characters [29], or when predicting numerical values, e.g., of market prices [30]. Residual networks are a specific type of network retaining memory through the use of skip connections. We will present them in section 3.9. In recent years, we also witnessed the emergence of DNN techniques in several fields of physics as a new tool for data analysis [31–34]. In condensed matter physics in particular, DNN and CNN proved to be efficient in learning the ground state of some many-body system [35–38], and to speed up Monte Carlo sampling [39, 40]. ML has also been shown to be performing well in identifying and classifying the phases of matter or learning order parameters [41–44]. In a specific case, ML proved to be able to reconstruct the phase diagram of site-type quantum percolation after performing

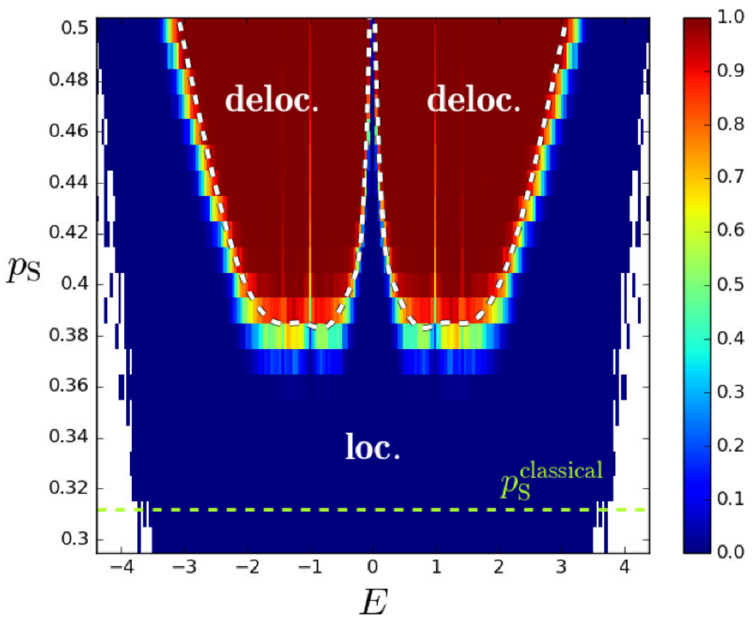


Figure 1.4: Phase diagram of the three-dimension site quantum percolation obtained through classification of the Anderson metal-insulator model reproduced with permission from [45]. On the vertical axis, we show the site probability of occupation p_s and on the horizontal axis, we display the eigenenergies E. The dashed green line denotes the threshold for the classical three-dimensional site percolation $p_s^{classical} = 0.3116 \pm 0.0002$ [46]. The dashed white line is the mobility edge.

a classification task on the two phases of the three-dimensional Anderson metal-insulator model as shown in figure 1.4.

Despite all these studies, the ML process in itself tends to be somewhat of a black box, and it is not yet known what is allowing a DNN to correctly identify a phase. To gain further insight into this issue, we choose a well-known and well-studied classical system that exhibits perhaps the simplest of all second-order phase transitions, the *site-percolation* model in two spatial dimensions [47, 48]. 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 [48]. Several ML studies on the percolation model have been already published, mostly using *supervised* learning to identify the two phases via ML classification [26, 49]. An estimate of the critical exponent, ν, of the percolation transition has also been given [26]. The task of determining p_c was further used to evaluate different ML regression techniques in Ref. [50]. For *unsupervised* and *generative* learning, less work has been done [26,49,51]. While some successes have been reported [51], other works show the complexities involved when trying to predict percolation states [49]. Through our study of the percolation model, we will show that ML methods are able to identify a parameter such as the density and struggle with parameters related to long-range properties such as the spreading of a cluster or the correlation length.

We also introduce a second model, the three-dimensional Anderson model of localisation which is characterised by a localisation of its wavefunction in the strong disorder regime. Similarly to the percolation model, previous ML studies were performed and concluded the success of classification methods to identify the two phases of the system [45, 52, 53]. Here, we will show that supervised methods are able to identify the phases of the Anderson model of localisation. Furthermore, we show that ML methods can make close predictions on the disorder values W.

1.6 Outline of the thesis

In this thesis, we aim to understand the reasons that allow ML techniques to perform so well in phase recognition tasks. Chapter 2 and 4 provides the theory on the models used for our study and the specificity of our datasets. In Chapter 3 we dive into the theory of machine learning and detail the process of training for classification, regression and a variational autoencoder. Finally, in Chapter 5 and 6, respectively, we present the results of our ML analysis for the two-dimensional percolation model and the three-dimensional Anderson model.

Chapter 2

Theory

2.1 The percolation model

The percolation problem is well-known with a rich history across the natural sciences [47, 48, 54–57]. It provides the usual statistical characteristics across a second-order transition such as, e.g., critical exponents, finite-size scaling, renormalisation and universality [48].

Perhaps the simplest non-trivial implementation of percolation is provided by the two-dimensional (2D) site percolation model [48] on a square lattice. In this model a lattice of size $L \times L$ with individual lattice sites $\vec{x} = (x, y)$, $x, y \in [1, L]$, is randomly occupied with an *occupation probability* p such that the state ψ 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 neighbouring sites exists when these are side-to-side nearest-neighbours on the square lattice, while diagonal sites can never be connected. A group of these

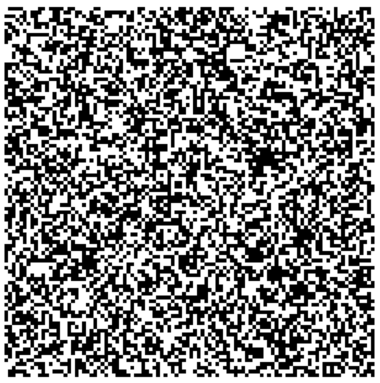


Figure 2.1: Example of percolation lattice of size $L = 100$ at $p = 0.5$. Occupied sites are marked by small black dots while empty sites are left white.

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 Figure 2.1, we show examples of percolation lattice at $p = 0.5$.

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. In the case of an infinite lattice, we observe the emergence of an infinite cluster spanning through the system at $p_c = 0.59274605079210(2)$. This estimate has been determined numerically even more precisely over the preceding decades [58] while no analytical value is yet known [57]. An important quantity is the *percolation strength* which corresponds to the probability of belonging to the infinite cluster $P(p) = \langle s_p(p)/L^2 \rangle$, where $s_p(p)$ gives the size of the percolating cluster for size L and $\langle \cdot \rangle$ denotes an average over many randomly generated realisations. Similarly, we can define the probability of not belonging to the percolating cluster, $Q(p) = \langle (L^2 - s_p(p))/L^2 \rangle$ and $P(p) + Q(p) = 1$.

Given the two very distinct phases of the system (spanning and non-spanning) and the extended literature on this model, the percolation model provides an interesting test case to study ML approaches which intend to predict phases of condensed matter systems.

2.1.1 The cluster structure

In this section, we will define some important parameters related to the cluster structure. As a first step, we will demonstrate these quantities in $1d$. Let us define a chain with occupied and unoccupied sites. We recall that according to the definition, a cluster is defined as a grouping of s nearest neighbours occupied sites. In one dimension this corresponds to having s nearest neighbour sites occupied and at least two empty sites at the extremities of this cluster. From this observation, and assuming that each site is independently occupied, we can define a quantity n_s called the cluster number, corresponding to the probability of a site belonging to a cluster of size s

$$n_s(p) = p^s(1 - p)^2, \tag{2.1}$$

where p is the probability of having an occupied site and $(1 - p)$ the probability to have an empty site. In $1d$, the percolating cluster appears only when all the sites in the chain are occupied, i.e., $p_c = 1$. Using this information and $p^s = \exp(s(\ln(p)),$

eq. (2.1) can then be rewritten as

$$n_s(p) = (1-p)^2 \exp(s\ln(p)) = (p_c - p)^2 \exp\left(-\frac{s}{s_\xi}\right), \tag{2.2}$$

where we define $s_\xi = -\frac{1}{\ln(p}$ as the cutoff length. It corresponds to the size of the largest cluster in the lattice. In order to generalise this formula to higher dimensions we need to introduce the concept of perimeter. We define the perimeter t of a cluster as the number of nearest neighbour empty sites to the cluster. This parameter t is composed of the empty sites delimiting a cluster as well as the holes which might exist in a cluster. In $1d$ the perimeter cluster $t = 2$ corresponds to the two empty sites delimiting a cluster at the left and right border. We now redefine the cluster number as

$$n_s(p) = \sum_t g_{s,t} p^s (1-p)^t, \tag{2.3}$$

where $g_{s,t}$ is the number of possible cluster configurations. The probability of a site to belong to any site in the cluster of size s is defined as $\sum sn_s(p)$

$$\sum_s sn_s(p) = \sum_s sp^s(1-p)^2, \tag{2.4}$$

$$= p(1-p)^2 \sum_s \frac{dp^s}{dp}, \tag{2.5}$$

$$= p(1-p)^2 \frac{d\sum_s p^s}{dp}. \tag{2.6}$$

Using the geometric series formula we obtain

$$\sum_s sn_s(p) = p(1-p)^2 \frac{d(1/1-p)}{dp} = p. \tag{2.7}$$

The result of eq. (2.7) can be easily understood as an occupied site is simply a cluster of size $s = 1$. As we mentioned earlier, in $1d$, the percolation threshold $p_c = 1$. As a consequence, the $1d$ case is particular as only one phase of the transition can be observed $p \leq p_c$.

Another quantity of interest is the probability to have a spanning cluster at a given density p, in the infinite system. It is defined as

$$\Pi(p) = \begin{cases} 0, & \text{if } p < p_c \\ 1, & \text{if } p > p_c. \end{cases} \tag{2.8}$$

We also define a second probability $\kappa(p)$, which corresponds to the probability of having a non-spanning cluster at a given probability p

$$\kappa(p) = \begin{cases} 1, & \text{if } p < p_c \\ 0, & \text{if } p > p_c. \end{cases} \tag{2.9}$$

We remark that in a finite system, the transition from the non-spanning phase to the spanning one is not instantaneous due to finite size effects. This leads to spanning samples with a density $p < p_c$ and non-spanning samples for $p > p_c$. As such, we define the alternative probabilities Π_L and κ_L describing the behaviour of a finite size system of size L.

2.1.2 The correlation function

The correlation function is defined as $g(r) = \langle \psi(\vec{x})\psi(\vec{x}+\vec{r})\rangle_{\vec{x},\vec{x}+\vec{r}\in C, r=|\vec{r}|} - P^2$, where the $\langle\cdot\rangle_{\vec{x},\vec{x}+\vec{r}\in C, r=|\vec{r}|}$ denotes the average over all $\vec{x}$, all distances r and all clusters $\{C\}$ with $\vec{x}$ and $\vec{x}+\vec{r}$ belonging to the same cluster C. This quantity is often used to characterise the formation of clusters in the lattice. It corresponds to the probability of having two occupied sites separated by a distance r which belongs to the same cluster. It is important to note that we exclude the contribution of points in the infinite cluster from $g(r)$ [48]. By definition of the correlation function $g(0) = 1$, as the site is occupied. In $1d$ we find that $g(r) = p^r$, i.e., in order to have a point occupied at a distance r from the origin, every site in between must be occupied. This can be reformulated as

$$g(r) = p^r = \exp(-\frac{r}{\xi}), \tag{2.10}$$

where

$$\xi = -\frac{1}{\ln(p)} = -\frac{1}{\ln[p_c - (p_c - p)]}. \tag{2.11}$$

We recall that in $1d$, the phase transition occurs at $p_c = 1$. Using the expansion $\ln(1-x) \approx -x$, we obtain

$$\xi = -\frac{1}{\ln(p)} \approx \frac{-1}{-(p_c - p)} = (p_c - p)^{-1} \xrightarrow[p\to p_c]{} \infty. \tag{2.12}$$

The quantity ξ is the correlation length, in a higher dimension, we find that ξ diverges at p_c as $|p_c - p|^{-\nu}$ where ν is called the critical exponent. This critical exponent depends on the dimension of the system. In $2d$ it is equal to $\nu = 4/3$ in the $2d$ square lattice [48].

2.1.3 The correlation length

The correlation length provides an estimation of the size of the cluster in the lattice. Beyond the length ξ it is improbable to find two occupied sites belonging to the same cluster, as such ξ can be seen as a cutoff length. The correlation length is defined as

$$\xi = \sqrt{\frac{\sum_r r^2 g(r)}{\sum_r g(r)}}, \tag{2.13}$$

where r is the distance between two occupied sites. For $p < p_c$ the correlation length gives the scale of the largest cluster present in the lattice. Around $p \approx p_c$ we see the emergence of an infinite cluster, $\xi = \infty$. Above p_c, when we see the appearance of the infinite cluster, the contribution of points belonging to this cluster is subtracted from $g(r)$ as shown in the previous section. The correlation ξ can then be interpreted as an indication of the size of the holes in the lattice. Let us now compute the value of the prefactor C

$$C\xi^2 = \frac{\sum_r r^2 g(r)}{\sum_r g(r)} \to \frac{\int r^2\ g(r)}{\int g(r)}. \tag{2.14}$$

As we seen in the previous section $g(r) \propto e^{-\frac{r}{\xi}}$ provides a good estimation of $g(r)$. Let us now compute the numerator

$$\begin{aligned}
\int_0^{2\pi}\int_0^{\infty} r^2\ g(r)\ r\ dr\ d\theta &= 2\pi\left\{\left[-\xi r^3\ e^{-\frac{r}{\xi}}\right]_0^{\infty} + 3\ \xi\int_0^{\infty} r^2 e^{-\frac{r}{\xi}} dr\ \right\}, && (2.15)\\
&= 2\pi\left\{3\ \left[-\xi^2 r^2\ e^{-\frac{r}{\xi}}\right] - 6\int_0^{\infty} \xi^2 r\ e^{-\frac{r}{\xi}} dr\ \right\}, && (2.16)\\
&= 12\pi\xi^4. && (2.17)
\end{aligned}$$

We now look at the denominator

$$\begin{aligned}
\int_0^{2\pi}\int_0^{\infty} g(r)\ r\ dr\ d\theta &= 2\pi\left\{\int_0^{\infty} r\ e^{-\frac{r}{\xi}} dr\ \right\}, && (2.18)\\
&= 2\pi\left\{\left[-\xi r\ e^{-\frac{r}{\xi}}\right]_0^{\infty} - \xi\int_0^{\infty} e^{-\frac{r}{\xi}} dr\ \right\}, && (2.19)\\
&= 2\pi\left[\xi^2 e^{-\frac{r}{\xi}}\right]_0^{\infty}, && (2.20)\\
&= 2\pi\xi^2. && (2.21)
\end{aligned}$$

Combining eqs. (2.17) and (2.21) we find

$$\Rightarrow C\xi^2 = \frac{\int_0^{2\pi}\int_0^{\infty} r^2\ g(r)\ r\ dr\ d\theta}{\int_0^{2\pi}\int_0^{\infty} r\ g(r)\ r\ dr\ d\theta} \propto \frac{12\pi\xi^4}{2\pi\xi^2} \propto 6\xi^2. \tag{2.22}$$

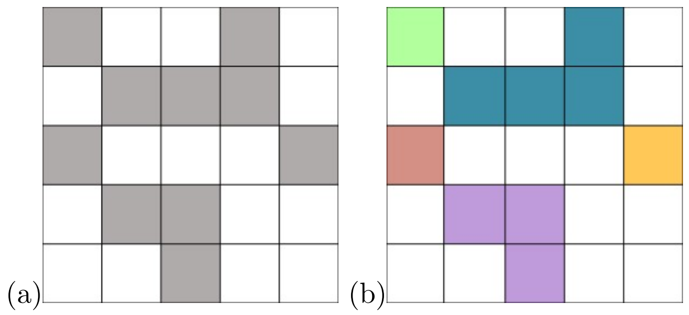


Figure 2.2: (a) Percolation lattice at $p = 0.4$ before the HK labelling, occupied sites shaded in grey and unoccupied sites are coloured in white. (b) Percolation lattice after HK labelling. The five different clusters are coloured with different colours.

From equation eq. (2.22) we note the presence of a pre-factor $C = 6$, that we will need to remove to obtain the normalised value of ξ.

2.1.4 The Hoshen-Kopelman algorithm

Earlier in section, 2.1 we introduced the notion of *clusters* as a group of nearest neighbours occupied site. We now present an algorithm able to identify such clusters, the Hoshen-Kopelman (HK) algorithm. The HK algorithm, based on the union-find algorithm, was first discussed in 1976 by J. Hoshen and R. Kopelman [59]. Given a set of points with possible interconnections, the union-find algorithm provides an effective way to find an equivalence relationship between these points. In percolation, the HK algorithm applies this principle in order to find and identify each cluster in a lattice. Let us define a grid of size $L \times L$, where sites are occupied with a probability p and unoccupied with a probability $1 - p$. The first step of the HK algorithm is to scan the lattice in search of occupied sites, once an occupied site a is identified the second step is to check if this site has occupied neighbouring sites. This is the find portion of the algorithm. In the case where a site a has an occupied neighbour b and this site was previously labelled, the site a joins the equivalence class of site b and is assigned the same label. This corresponds to the union part of the algorithm. Finally, if the site a does not have any occupied neighbours, a new label is assigned to the site a.

In figure 2.2(a) we show a two-dimensional percolation lattice of size 5×5 occupied at density $p = 0.4$. In the rest of this section, we will provide a detailed application of the HK algorithm on this lattice. For the sake of clarity, we denote each site with the notation $s(i, j)$ where i is the row number and j is the column number. We begin the algorithm at the top left corner of the lattice at site $s(0, 0)$. The site is occupied, and does not have any occupied neighbours above or on the left, we assign the label 1 to this site. The two consecutive sites $s(0,1)$ and $s(0,2)$ are unoccupied, we skip them. Site $s(0,3)$ is occupied without any neighbour above or on the left, we relabel this site with label 2. The next occupied site is on the second row, site $s(1,1)$, again, without neighbouring occupied sites on the left or above, we assign it the label 3. The site $s(1,2)$ is occupied and has a nearest neighbour occupied on the left, we relabel it as 3 with the same label as site $s(1,1)$. The next occupied sites, $s(1,3)$ has two occupied neighbours, site $s(1,2)$ labeled as 3 and site $s(0,3)$ with label 2. We join the two cluster and relabel sites $s(1,1)$, $s(1,2)$ and $s(1,3)$ with label 2. We scan the remaining sites in the lattice in the same manner and assign label 3 to site $s(2,0)$, label 4 to site $s(2,4)$ and label 5 to the cluster of sites $s(3,1)$, $s(3,2)$, $s(3,3)$. In figure 2.2(b) we show the relabeled lattice, where each cluster is assigned different colours.

2.1.5 Real space renormalisation

Let us consider a lattice with n sites, this amounts to having 2^n configurations possible as a site can be in two states, either occupied or unoccupied. Thus, it becomes more and more difficult to perform operations on the lattice as the number of sites increases. An important factor is that the physics of the percolation model depends on a long-range behaviour, i.e. the spreading of the cluster. Therefore, the need arises for a method that would neglect short-range correlation while preserving long-range behaviour. Real space renormalisation, also called block spin technique was introduced by Leo Kadanoff in 1965 [60]. This is a coarse-graining process operating over microscopic degrees of freedom. It allows us to observe the large-scale behaviour of the system by overlooking correlation on a scale smaller than a certain length b that is smaller than the characteristic length of the system ξ. Applied to percolation, real space renormalisation relies on the self-similarity of ξ at $p = p_c$, which means that each cluster of size inferior to ξ is all similar. The first step is to divide the lattice in super-site of linear size b, smaller than ξ. Following this, the b^d sites in the new super-site regions are replaced by a unique site according to a predefined rule (for example majority rule on spanning/non-spanning). This leads to an increase in the lattice spacing by a lattice spacing b. After renormalisation, the occupation p' of the new lattice of size can differ from p except at the transition where $p = p' = p_c$. This also applies to ξ, we obtain a new ξ' which can differ from ξ while following the same behaviour

$$\xi' = \frac{\xi}{b}. \tag{2.23}$$

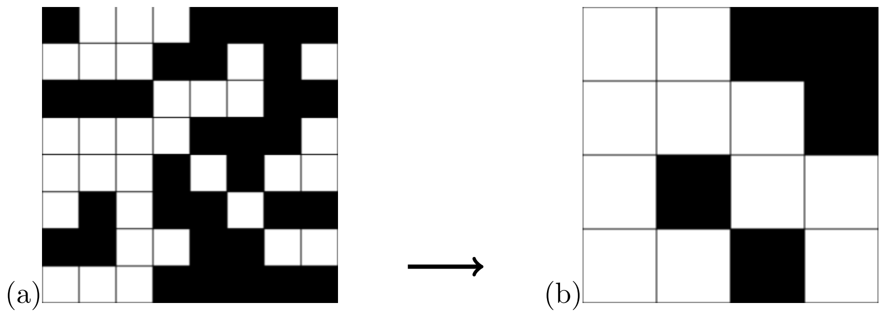


Figure 2.3: Example of renormalisation process on a 8×8 lattice with $b = 2$

We recall that at the transition $\xi \propto |p - p_c|^{-\nu}$,

$$\xi' \propto |p' - p_c|^{-\nu}, \tag{2.24}$$

$$\xi' \propto \frac{|p - p_c|^{-\nu}}{b}. \tag{2.25}$$

We know that an import property of the renormalisation process is that ξ' behaves similarly to ξ at the transition

$$\frac{|p - p_c|^{-\nu}}{b} = |p' - p_c|^{-\nu}, \tag{2.26}$$

$$\nu \ln\left(\left|\frac{p' - p_c}{p - p_c}\right|\right) = \ln(b). \tag{2.27}$$

By reorganising eq. (2.27) we retrieve a new expression of ν,

$$\nu = \frac{\ln(b)}{\ln\left(\left|\frac{dp'}{dp}\right|\right)_{p=p_c}}. \tag{2.28}$$

To illustrate this process, let us define a lattice of size 8×8 at $p = 0.5$, we choose to rescale it with $b = 2$. As an averaging rule, we decide to apply a spanning rule, if sites in the cell of size 2^2 are spanning vertically, then the new super-site is occupied. After renormalisation, we obtain p'

$$p' = p^4 + 4p^3(1 - p) + 2p^2(1 - p)^2. \tag{2.29}$$

We solve the fourth-degree equation to find the fixed points

$$\Rightarrow p^4 + 4p^3(1-p) + 2p^2(1-p)^2 - p = 0, \tag{2.30}$$
$$\Rightarrow p(-p^3 + 2p - 1) = 0, \tag{2.31}$$

$$p^* = \begin{cases} 0 \\ 1 \\ 0.618 \end{cases} \tag{2.32}$$

The solutions $p^* = 0$ and $p^* = 1$ are trivial fixed points, while $p^* = 0.618$ is the non-trivial fixed point. We know from previous studies of the two-dimensional percolation model that $p_c \approx 0.5927$, this means that despite a relatively small system, renormalisation provides a good estimation of p_c. The next step is to compute the value of ν. Using eqs. (2.28) and (2.29) we find

$$\nu = \frac{\ln(b)}{\ln\left(\left|\frac{dp'}{dp}\right|\right)_{p=p^*}}, \tag{2.33}$$
$$= \frac{\ln(2)}{\ln(|-4p^3 + 4p|)_{p=p^*}}, \tag{2.34}$$
$$= 1.5278838. \tag{2.35}$$

From the literature on percolation, we know that in the two-dimensional square lattice $\nu = 4/3 \sim 1.3333$. The result obtained through renormalisation is slightly above the result expected. However, given the small size of the observed system, we accept this result as a good approximation. As such, renormalisation appears as a valid solution to reduce the computation load of large percolation lattices while keeping information about the phase transition.

Through this first section, we presented the percolation model as our first model of interest. Given the extensive studies carried out on this model, it appears as a perfect test candidate for our ML study.

2.2 The Anderson Metal-Insulator model

In the first section, we introduced the percolation model which is the first model that we will use for our ML study. Here, we present our second model of interest, the Anderson model of localisation, and provide some important properties.

An ideal metal is usually represented as a periodic ion lattice immersed in a free electron gas. However, in real life materials are rarely devoid of defects which contribute to hindering the way that electrons move in the system. The first approach to this issue is the Drude model [61, 62]. It introduces disorder in the model and states that the conductivity in a metal depends on the distance between two collisions which is called the mean free path. As such, a high mean free path implies a long distance between successive collisions and therefore a high conductivity. Therefore, when increasing the disorder in the lattice, we observe a slow decrease in conductivity. However, while the Drude model provides an answer to the behaviour of electrons in an imperfect lattice, it does not explicitly account for the high disorder regime. In 1958, Philip W. Anderson in his paper "Absence of Diffusion in certain random lattices" [63] proposed a model to test the validity of the Drude model for a strong disorder. He proved that increasing the disorder indeed contributed to a decrease in conductivity, but furthermore, in three dimensions above a critical disorder, electrons were completely stopped in their motion, this is the Anderson localisation. In the following section, we will develop the main properties of the Anderson model.

2.2.1 Weak localisation

The weak localisation model provides a correction to the model of the ideal metal by introducing disorder in the medium. The conductance G, provides us with an insight into how easily electrons can propagate in a medium, it is defined as the inverse of the resistance R

$$G = \frac{1}{R} = \frac{1}{\rho L^{2-d}} = \sigma L^{d-2}, \tag{2.36}$$

where L is the linear size of the system, ρ is the electrical resistivity, σ is the conductivity of the medium and d is the dimension of the system. In the presence of disorder, the trajectory of electrons is disturbed and they cannot propagate freely. The Drude model states that an electron propagates freely until encountering the first impurity, therefore the conductivity depends on the distance between collisions. This distance between two collisions is called the mean free path and is defined as

$$\ell_{\text{mfp}} = v\tau, \tag{2.37}$$

where v is the average speed of electrons in the medium and τ is the average time between collisions. Consequently, a correlation exists between the disorder in our medium and the conductivity.

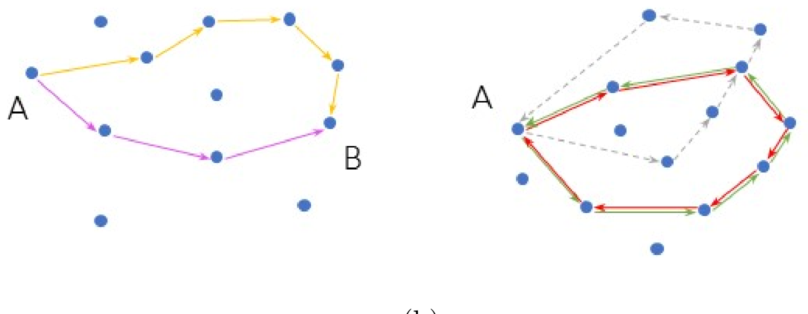


Figure 2.4: (a) Two possible paths taken by electrons to propagate from path A to path B. (b) Several paths displaying the propagation of electrons in the medium. We observe that the red and green paths form a loop travelling in opposite directions, as this doubles the odds of the electron returning to its origin.

A second important length appears due to the quantum nature of electrons, the de Broglie wavelength λ, which dictates the scale at which quantum interference can occur. When $\ell_{\mathrm{mfp}} > \lambda$ we are in the framework of the Drude model and no quantum interference can take place. However, if $\ell_{\mathrm{mfp}} < \lambda$ we enter a new regime where electrons can be strongly localised due to the wave interference effect. As we mentioned before, in the weak localisation regime, a wave propagates through the medium until encountering the first impurity. Following this event part of the wave is scattered. The main wave and the scattered wave keep propagating freely in the medium until encountering the next impurity which will create new scattering waves. Assuming that the scattering events are elastic, the main wave and the scattered waves can interfere. Let us consider the propagation of a particle in a two-dimensional disordered medium. The propagation of the electrons follows a random path pattern, where the electrons undergo diffusive events. We can therefore define the mean square displacement as

$$\Delta x^2 \approx Dt \text{ with } t \gg \tau, \tag{2.38}$$

with $D = L^2/\tau$ the diffusion coefficient for a system of size L.

In this framework, we define a probability p(t,0) as the probability of return. This probability provides the odds of a particle returning to its origin after a series of diffusive events.

Let us now look at two examples of diffusion of electrons. In figure 2.4(a) we see two possible trajectories taken by electrons departing from a point A to reach point B. In the presence of elastic scattering, the main wave and the wavelets maintain similar energy and can be summed. As we can see in figure 2.4(a) several paths exist to connect points A to B, each with different lengths and different phases acquired as a result of diffusive events. In this configuration the addition of the contributions of each path cancel out and the model of Drude still holds. We define the probability of going from a point A to a point B as

$$\begin{aligned} p(B|A) &= |\sum A_i(B|A)|, && (2.39)\\ &= \sum_i |A_i(B|A)| + \sum_{i\neq j} A_i^* A_j, && (2.40)\end{aligned}$$

where A_i designates the amplitude of the paths. The probability of travelling from A to B depends on the coherent and incoherent contributions of the paths. A second kind of path can appear in the system, a closed path which appears when the electron comes back to its point of origin as depicted in 2.4(b). In this case, interference effect emerges as two electrons could follow this loop path in opposite directions. Let us consider such two electrons, with amplitudes A_1, A_2 and phase $\varphi_1 \equiv \varphi_2$. The return probability $p(A|A)$ is then defined as

$$\begin{aligned} p_t(A|A) &= |A_1 + A_2|^2, && (2.41)\\ &= |A_1|^2 + |A_2|^2 + A_1^* A_2 + A_2^* A_1, && (2.42)\\ &= 4|A|^2, && (2.43)\end{aligned}$$

where $A_1^* A_2$ and $A_2^* A_1$ are interference contributions. Without the contribution of the external field, the two electrons travel the same path in opposite directions, implying $A_1 \equiv A_2 \equiv A$ and $\varphi_1 \equiv \varphi_2$. From eq. (2.43) we observe that having electrons in a closed loop increases the probability of backscattering, which in turn reduces the conductivity. This phenomenon is called weak localisation.

2.2.2 The three-dimensional Anderson Model

Now that we have introduced the concept of weak localisation in the previous chapter, let us introduce the strong disorder regime and the Anderson model of localisation. The model proposed by Anderson is a tight binding model with a random onsite disorder $\varepsilon_i \in [-W/2, W/2]$, where the variable W is the amplitude of the disorder. In $1d$ and $2d$, there is always localisation of the wavefunctions. However, in $3d$ it was shown that above a certain disorder $W_c = 16.57$, a phase transition occurs from a metal state, with an extended wavefunction, to an insulated state with a localised wavefunction. The Hamiltonian associated with this model is

$$\begin{aligned}\mathcal{H} &= H_0 + V, &(2.44)\\ &= \sum_i \varepsilon_i |i\rangle\langle i| + \sum_{\langle i,j\rangle} J_{i,j} |i\rangle\langle j|, &(2.45)\end{aligned}$$

where $J_{i,j}$ is a hopping matrix element acting on nearest neighbours, here the ratio $\frac{J}{W}$ gives the probability of hopping. An important quantity to monitor is the return probability density, which provides the odds of an electron departing from a position 0 to return to its origin,

$$p(t) = |\psi(t,0)|^2. \tag{2.46}$$

We will now study this model in the disorder-free regime and in the infinite disorder case. The disorder-free regime corresponds to the model of the ideal metal and $J/W \to \infty$, an electron can move freely and all the onsite energies are identical. The eigenstates of a system of size L with periodic boundary conditions are then given by

$$\psi_k(j) = \frac{e^{ikj}}{\sqrt{L^d}}, \tag{2.47}$$

where $j = 1, 2, \ldots, N$ denotes the number of sites and $k = \frac{2\pi n}{L}$ is the wave vector with $n = 0, \ldots, L-1$. We obtain the energy associated as

$$E = -2J \sum_{a=1}^{d} \cos(k_a). \tag{2.48}$$

This allows us to obtain the time-evolved solution of the Anderson model

$$\psi(j,t) = \frac{1}{\sqrt{L^d}} \sum_k e^{ikj + 2itJ \sum_a cos(k_a)}. \tag{2.49}$$

In the thermodynamic limit, there is a convergence of the solution to

$$\psi(j,t) = \int_{k \in [0,2\pi]^d} \frac{d^d k}{(2\pi)^d} e^{[ikj + 2itJ \sum_a cos(k_a)]} = i^d \prod_{a=1}^{d} \mathcal{J}_{j_a}(2tJ), \tag{2.50}$$

where $\mathcal{J}_n$ corresponds to a Bessel function of the first kind. From this expression, it is possible to retrieve a convergence of the return probability

$$|\psi(0,t)|^2 \sim \frac{1}{t^d}. \tag{2.51}$$

At the $t \to \infty$ we observe a decay of the particle return probability. There is no localisation.

For the infinite disorder case, the hopping probability $J/W \to 0$, the kinetic part of the Hamiltonian is negligible

$$\mathcal{H} = \sum_i \varepsilon_i |i\rangle\langle i|. \tag{2.52}$$

We obtain the trivial solution

$$\psi(j,t) = e^{i\varepsilon t}|0\rangle \tag{2.53}$$

In this configuration, the wave packet is perfectly localised and the probability of return is constant and equal to 1.

2.2.3 Scaling theory of localisation

After studying weak and strong localisation, a need arises to find a bridge between these two regimes. Furthermore, an important question remains, how would a metal or insulator evolve under a change of size? These important issues were studied by the so-called "Gang of Four" in their paper "Scaling theory of localisation" [64]. They theorised that in the case of a system of size L, only one parameter $g(L)$ was needed to describe the behaviour of the system in both phases. Let us consider a d-dimensional block of size L^d.The density of states $\nu(E)$ provides us with all the eigenstates ψ_α of energy E_α and is defined as,

$$\nu(E) = \sum_\alpha \delta(E - E_\alpha). \tag{2.54}$$

We define τ_{esc} as the average time that an electron would take to reach the boundary and escape the system

$$\tau_{\text{esc}} = \frac{L^2}{D}, \tag{2.55}$$

where $D = \sigma/e^2\nu$ is the diffusion coefficient. The Thouless energy is the energy associated with this diffusion event and is defined as

$$E_T = \frac{h}{\tau_{\text{esc}}} = \frac{hD}{L^2}. \tag{2.56}$$

We define the dimensionless conductance g, as the ratio of this energy with the mean level spacing Δ

$$g = \frac{E_T}{\Delta} = \frac{h}{e^2}\sigma L^{d-2} = \frac{h}{e^2}G. \tag{2.57}$$

According to the scaling theory, the change in conductance can be described through a function β defined as

$$\beta(g) = \frac{d\ln g}{d\ln L}. \tag{2.58}$$

In the weak disorder limit, the Thouless energy is larger than the mean level spacing. Several energy levels overlap leading to the presence of extended states

$$g \gg 1 \Rightarrow g \sim \sigma L^{d-2}. \tag{2.59}$$

We now compute the value of $\beta(g)$ associated to this g

$$\beta(g) = \frac{d\ln L^{d-2}}{d\ln L}, \tag{2.60}$$

$$= d - 2. \tag{2.61}$$

We find that $\beta(g)$ is always negative for $d \leq 2$. This result confirms the finding of Anderson, that a phase transition only occurs in the three-dimensional model.

In the strong disorder limit, the mean level spacing is larger than the Thouless energy. Electrons cannot travel easily from one energy level to the next, this leads to a localisation of states. In this setting, the characteristic length of the system is the localisation length λ, and the conductance is expected to decrease exponentially as

$$g \ll 1 \Rightarrow g \sim g_0 \exp(-L/\lambda) \Rightarrow \ln g \sim \ln g_0 - \frac{L}{\lambda}. \tag{2.62}$$

We proceed to a change of variable $\alpha = -1/\lambda$ and we find the new value of β

$$\beta(g) = \frac{d(-\alpha L)}{d\text{ln}(L)}, \tag{2.63}$$

$$= \frac{d(-\alpha e^x)}{dx}, \tag{2.64}$$

$$= -\frac{L}{\lambda}, \tag{2.65}$$

$$= -\alpha e^x, \tag{2.66}$$

using eq. (2.62) we obtain

$$\beta(g) = -\alpha e^x = \ln g - \ln g_0. \tag{2.67}$$

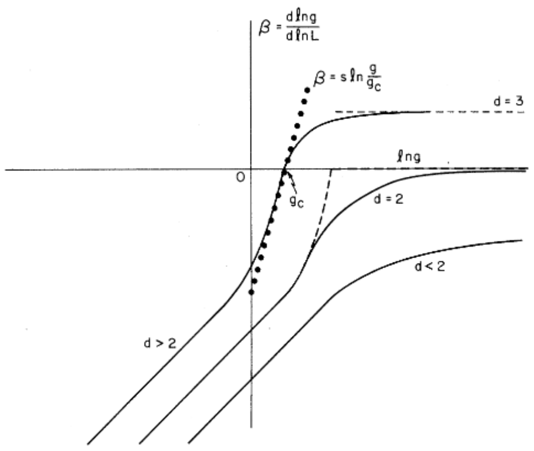


Figure 2.5: Scaling function $\beta(g)$ as a function of $\ln(\beta)$ for the weak and strong disorder regimes in $d = 1$, $d = 2$ and $d = 3$, reproduced with permission from [64]. In $d = 1$ and $d = 2$, $\beta(g)$ is always negative. In $d = 3$, $\beta(g)$ is negative for $\beta(g) < 0$ and positive for $\beta(g) > 0$.

Therefore $\beta(g)$ is always negative in the strong disorder limit. In both regimes, we observe that $\beta(g)$ is independent of the material, and only depends on the dimension d. Now that we established the behaviour of $\beta(g)$ in the two regimes, we can draw $\beta(g)$ by assuming that β is a monotonous smooth function. In figure 2.5 we display the plot of $\beta(g)$ for $d = 1, 2$ and 3. Equation (2.61) tells us that for $d < 2$ the scaling function is always negative in the weak disorder regime. For the three-dimensional case $\beta(g)$ is positive (respectively, negative) for $g > 1$ ($g < 1$). This implies the existence of a critical point at $\beta(g_c) = 0$ corresponding to the metal-insulator phase transition.

2.3 Conclusion

In this chapter, we have presented the two models that will be studied with ML methods in chapter 5 and 6. The classical two-dimensional site percolation constitutes the perfect test case for our study as this model is well-known and studied. Several concepts introduced here will be used in later chapters. For instance, the percolation dataset that we created was labelled by identifying the cluster through the HK algorithm. The quantum Anderson model of localisation was chosen as our second model, due to a previous study displaying the performance of ML methods to identify its phases. In chapter 6 we will train our ML network to identify the Anderson Metal-Insulator Transition (MIT) and W-values.

Chapter 3

Theory of Machine Learning

3.1 Polynomial regression

When dealing with a machine learning task, the first step even before training is to randomly split the dataset in two parts. The first part is the training set and the second is the so-called validation set. Generally, we reserve 90% of the dataset for the training dataset and the 10% remaining for the validation set. A separate test set is also generally created to analyse the performance of the network after training. At the heart of machine learning resides the notion of generalisation. We train a model on a dataset, but we also want this model to make correct predictions on a dataset unseen during the training process. Thus, by having a validation set, we can monitor the capacity of prediction of our model on a test set during the training. It is important to note that while a model can be performing very well on the training set it may fail on the validation and test set. This phenomenon is called *overfitting*. Many factors could be at cause, but this often occurs in the presence of a model that would be too complex for the dataset that it is training on. To illustrate this idea we will use the example of polynomial regression. Let us define a function $y = f(x_i) + \eta$ from which samples are taken, where $f(x_i)$ is an unknown function and η an uncorrelated noise variable i.e $\langle \eta \rangle = 0$ and $\langle \eta_i \eta_j \rangle = \delta_{ij}\sigma^2$, where σ is the noise strength. We define $f_\alpha(x; \theta_\alpha)$ as a family of functions that we use as a model to reproduce the behaviour of the samples in our dataset. Here, the quantity θ_α represents the internal parameters of the model. We train four different polynomial orders $f_1(x; \theta_1)$, $f_3(x; \theta_3)$, $f_{10}(x; \theta_{10})$ and $f_{50}(x; \theta_{50})$, each of them with $\alpha + 1$ parameters. The different orders represent the degree of complexity of the models.

In figure 3.1(a) we show a random sampling of $N = 10$ datapoints with

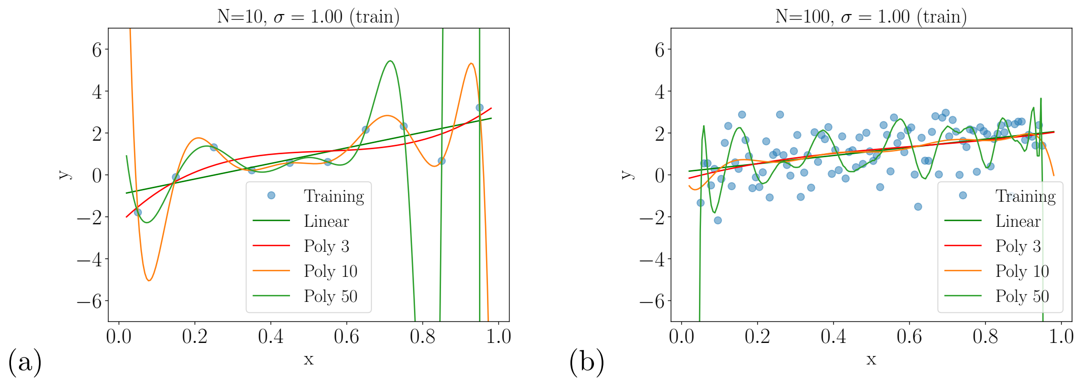


Figure 3.1: (a) Polynomial regression training for $N = 10$ datapoints. We observe that the families of functions $f_{10}(x;\theta_{10})$ and $f_{50}(x;\theta_{50})$ fit perfectly the datapoints. (b) Polynomial regression training for $N = 100$ datapoints. This time, none of the functions perfectly fit the datapoints, but the function $f_{50}(x;\theta_{50})$ seems to follow more the trend of the datapoints.

$\sigma = 1$ and the associated fitting by the four families of functions $f_1(x;\theta_1)$, $f_3(x;\theta_3)$, $f_{10}(x;\theta_{10})$ and $f_{50}(x;\theta_{50})$. Figure 3.1(b) shows a random sampling of $N = 100$ datapoints with $\sigma = 1$. As we can see, higher-order functions are better suited to perfectly fit points in the dataset while the linear function is only able to give the general tendency of the dataset. This is due to the so-called *bias* of the function: a linear function cannot accurately represent a non-linear function as it cannot bend. In this case, we say that the $f_1(x;\theta_1)$ function has a high bias while $f_{10}(x;\theta_{10})$ and $f_{50}(x;\theta_{50})$ have low bias. Therefore, one might conclude that using a high-order function, which here equates to an ML model with high complexity, would be the solution to having a good predictor. However, we now need to evaluate the prediction of such a polynomial function on a test set.

Let us look at the prediction made on a test set, in figure 3.2(a) and (b) we notice that the functions $f_{10}(x;\theta_{10})$ and $f_{50}(x;\theta_{50})$ which previously fitted points in the training dataset are now failing on the test dataset. This example perfectly illustrates the phenomenon of *overfitting* mentioned in the previous section: the function learned the noise in the training set and is now unable to make predictions on the test set. Hence, the selection of the model requires finding a good equilibrium between variance and the bias of the function. This is what is known in the field of ML as the *bias-variance trade-off*.

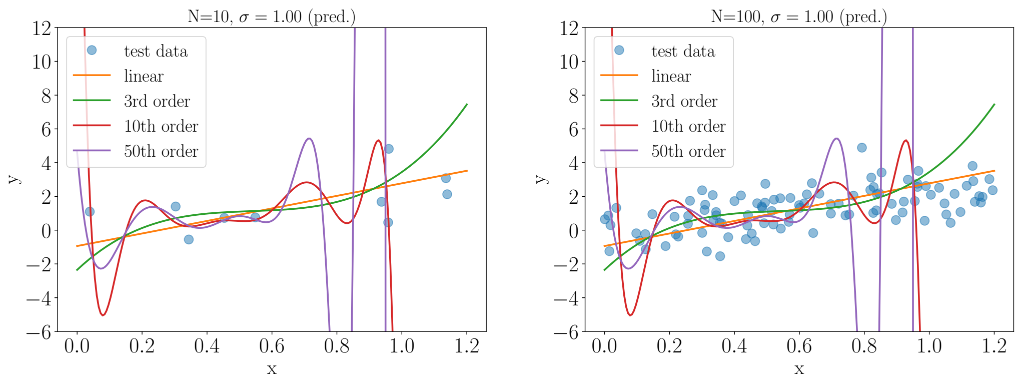


Figure 3.2: (a) Polynomial regression predictions for $N = 10$ datapoints. The families of functions $f_{10}(x;\theta_{10})$, $f_{50}(x;\theta_{50})$, that fit the datapoints so perfectly in figure 3.1(a) are now struggling to make an accurate prediction on the test set. The linear function seems the best suited for the prediction. (b) Polynomial regression predictions for $N = 100$ datapoints. Similarly to (a) the higher order polynomial fails to make a decent prediction on the test set, the lower order polynomial gives the best prediction

3.2 The bias-variance trade off

In this section, we will formalise the concepts of bias-variance briefly introduced in the previous section. Let us again consider a dataset defined as $y = f(x) + \eta$ where η is some Gaussian noise with mean zero and variance σ_η. Similarly to the previous section the aim of our training is to find a family of functions $f(x;\theta_D))$, also called model, capable of fitting the data. We use regression techniques to do so. During the regression process, we monitor the progress of our training through a cost function $C(y, f(x;\theta_D)$. We choose to employ the Mean Squared-Error (MSE) defined as

$$C(y, f(\theta_D)_{MSE}) = \frac{1}{n}\sum_i \left[y_i - f(x;\theta_D)\right]^2 , \tag{3.1}$$

where n is the number of samples in the dataset. Let us look at the expected value of the cost function

$$\mathbb{E}_{D,\eta}[C(y, f(\theta_D))] = \mathbb{E}_{D,\eta}\left[\sum_i (y_i - f(x_i;\theta_D))^2\right], \tag{3.2}$$

$$= \mathbb{E}_{D,\eta}\left[\sum_i (f(x_i) + \eta - f(x_i;\theta_D))^2\right], \tag{3.3}$$

$$= \sum_i \mathbb{E}_{D,\eta}[(f(x_i) - f(x_i;\theta_D))^2] + \mathbb{E}_{D,\eta}[2\eta(f(x_i) - f(x_i;\theta_D))] + \mathbb{E}[\eta^2]. \tag{3.4}$$

Using the independent random variable property $\mathbb{E}[AB] = \mathbb{E}[A]\mathbb{E}[B]$, the first term can be expanded as

$$\mathbb{E}_{D,\eta}[(f(x_i) - f(x_i;\theta_D))^2] = \mathbb{E}[f(x_i)^2] - 2\mathbb{E}[\eta]\mathbb{E}[(f(x_i)]\mathbb{E}[f(x_i;\theta_D)] + \mathbb{E}[f(x_i;\theta_D)^2]. \tag{3.5}$$

Similarly, the second term of eq. (3.4) can be expanded as

$$\mathbb{E}_{D,\eta}[2\eta(f(x_i) - f(x_i;\theta_D)] = 2\mathbb{E}[\eta]\mathbb{E}[f(x_i) - f(x_i;\theta_D)] = 0. \tag{3.6}$$

We recall that the random noise η has a mean of zero, therefore $\mathbb{E}[f(x_i)^2] = f(x_i)^2$. Now by adding and subtracting $\mathbb{E}[f(x_i;\theta_D)]\mathbb{E}[f(x_i;\theta_D)]$ we obtain

$$\mathbb{E}_{D,\eta}[C(y, f(\theta_D))] = \sum_i (\mathbb{E}[(f(x_i;\theta_D)] - f(x_i))^2 + \mathbb{E}[f(x_i;\theta_D)^2] - \mathbb{E}[f(x_i;\theta_D)]\mathbb{E}[f(x_i;\theta_D)] + \mathbb{E}[\eta^2]. \tag{3.7}$$

$$\mathbb{E}_{D,\eta}[C(y, f(\theta_D)) = \sum_i \text{Bias}^2 + \text{Variance} + \mathbb{E}[\eta^2] \tag{3.8}$$

The first term of the equation is the squared bias

$$\text{Bias}^2 = (\mathbb{E}[(f(x_i;\theta_D)] - f(x_i))^2, \tag{3.9}$$

it gives an estimation of the deviation of the prediction from the true value. The second term is the variance

$$\text{Variance} = \mathbb{E}[f(x_i;\theta_D)^2] - \mathbb{E}[f(x_i;\theta_D)]\mathbb{E}[f(x_i;\theta_D)], \tag{3.10}$$

which measures the fluctuation in the prediction due to finite size effect. The average error of our model is a function of the bias, the variance and some noise. By increasing the complexity of the function of the network $f(x_i;\theta_D)$ we reduce its bias. However, this leads to a higher variance as $f(x_i;\theta_D)$ will fit accurately points in the datasets. While a network able to fit every point in the training dataset appears appealing, we saw in the previous section that this leads to *overfitting*, i.e a network learning the noise in the training dataset which is then unable of making correct predictions on a test dataset. Therefore, an optimal network should be complex enough to reduce the bias, and not too complex to avoid overfitting. This is the so-called *bias-variance trade-off*.

3.3 Machine Learning as an optimisation problem

In this part, we are going to describe fundamental machine learning concepts and the steps involved in the training process. In section 1.4 we previously defined the dataset $D = \{(x_i, y_i)\}_{i=1}^N$, where x_i is an input and y_i the label associated to the input x_i. In the case of supervised learning the labelled dataset $D = \{(x_i, y_i)\}_{i=1}^N$ is given as input to our network. Let us define $f(x, \theta_D)$ the function associated with the network that we train, where θ_D is a parameter of our network also called *hyperparameter*. The information flows in the network from one layer to the next through the different activation functions as defined in eq. (1.1). Once the last layer, the output layer, is reached, we evaluate the performance of the prediction of the network and compare it to the label fed in input. This is performed through a cost function $C(y, f(x, \theta_D))$. Several types of cost functions can be used, depending on the task. For instance, regression provides us with continuous predictions while classification gives categorical predictions, as a result, these two methods need two different types of cost functions. Let us look at the case of regression, a popular cost function (also called error or loss functions) is the **MSE** as in eq. (3.1), which provides a direct estimation of the distance between the target and the predictions made by the network. However, as stated above a loss function such as the MSE is not suited for a classification task, we need a loss adapted to categorical predictions. Before introducing this new loss function it is necessary for us to define a concept from information theory, the *Shannon entropy*. Given a multiple outcome event with an associated distribution p, the Shannon entropy is defined as the average optimised number of bits required to communicate information about the outcome [65]

$$H(p) = -\sum_{i}^{n} p_i \log(p_i). \tag{3.11}$$

In case of prior knowledge or prediction of the outcome, the encoding of this information might be optimised to minimise $H(p)$. Thus, we define the Cross Entropy (CE) as the actual number of bits sent per option. The CE is defined as

$$H(p,q) = -\sum_{i}^{n} p_i \log(q_i), \tag{3.12}$$

where p is the distribution of the true event and q is the distribution of the predicted outcomes.

If our prerequisite about the outcome ends up being correct, $H(p) = H(p,q)$ otherwise the CE will differ from the entropy by an amount called the Kullback Leibler (KL) divergence denoted by D_{KL}. The KL divergence $D_{KL}(p||q)$ measures how a probability distribution p differs from a probability distribution q and is defined by

$$\begin{aligned} D_{KL}(p||q) &= H(p,q) - H(p), && (3.13)\\ &= \sum_{i}^{n}\Big(-p_i\log(q_i) + p_i\log(p_i)\Big), && (3.14)\\ &= \sum_{i}^{n} p_i \log\Big(\frac{p_i}{q_i}\Big). && (3.15) \end{aligned}$$

$D_{KL}(p||q)$ is not a metric as it is not symmetric and does not satisfy the triangular inequality, however, we note that the KL is non-negative.

The cross-entropy allows us to define two important losses for tasks employing categorical labels such as classification. When dealing with multi-class classification, a pre-processing is generally performed to encode the labels, it is called one-hot encoding. Suppose a classification with C-classes denoted by classes index $c = 0, 1, \ldots, C-1$, the label are encoded in a new label χ_{ic} such as $\chi_{ic} = 1$ if $\chi_i = c$ and $\chi_{ic} = 0$ otherwise. The loss function associated to this C-classes classification is the Categorical Cross-Entropy (CCE) , defined as:

$$L_{CCE} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} \chi_{ic} log(f(x_i)) + (1-\chi_{ic}) log(1 - f(x_i)). \tag{3.16}$$

Once an estimation of the error is performed through the appropriate loss function, we need to find an optimal way to decrease it by varying the parameters of the network. As physicists, an intuitive way to see this cost function is to assimilate it to the energy E of a system that we want to minimise.

3.4 Gradient Descent

The essence of an ML training resides in the optimisation of a cost function. For us physicists, it is easy to visualise this cost function as the energy of a system that we wish to minimise. A popular optimisation technique is the gradient-descent. This method allows a decrease in the cost function by following the direction of the gradient. Gradient-based methods are a set of optimisation methods, used to find the optimal solutions for a wide range of problems. Let us define $E(\theta) \equiv C(y, f(x, \theta))$ as the function that we want to minimise. This cost function can be reformulated as

$$C(y, f(x, \theta)) = E(\theta) = \sum_{i=0}^{n-1} e_i(x_i; \theta), \tag{3.17}$$

where e_i is the error on the sample i. At the start of the optimisation process we are usually at a random point θ_0 on the cost function landscape. After computing the gradient of the cost function $\nabla_\theta E(\theta)$, we update the value of θ to move in the direction of the minimum of $E(\theta)$. This is done iteratively by changing the hyperparameter θ of the model,

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta E(\theta), \tag{3.18}$$

where η is a parameter called the *learning rate* which controls the steps taken in the direction of the minimum. We update the value of θ_t as long as the minimum is not reached.

While the gradient descent method provides a nice solution for some optimisation problems, its application for a machine learning scheme is revealed to be quasi-impossible. The main issue remains that gradients are computationally expensive. For a cost function like the mean squared error, the computation of the gradient requires a sum over all the n datapoints in the system. The second issue resides in the non-adaptive nature of η. The cost function landscape is generally rugged, and a constant learning rate would lead to being stuck on a saddle point or completely diverging. A learning rate capable of taking into account the curvature of the landscape would be more adapted. Finally, the gradient descent method inherently depends on the initial conditions and tends to converge to a local minimum without being able to escape it, which implies that good optimisation is highly dependent on the initial point in the cost function landscape.

3.5 Newton's Method

Newton's method operates in a similar way to gradient descent, however, it makes use of the second-order Taylor's expansion

$$E(\theta + \mathtt{v}) \approx E(\theta) + \nabla_\theta E(\theta)\mathtt{v} + \frac{1}{2}H(\theta)\mathtt{v}^2. \tag{3.19}$$

Newton's method aims to find an optimal η to iterate on, to minimise the second-order Taylor expansion. Let us now look at the gradient of the Taylor expansion

$$\nabla_\theta E(\theta) + H(\theta)\mathtt{v}_{opt} = 0, \tag{3.20}$$

by rearranging this expression we obtain

$$\mathtt{v}_t = H^{-1}(\theta_t)\nabla_\theta E(\theta_t), \tag{3.21}$$

$$\theta_{t+1} = \theta_t - \mathtt{v}_t. \tag{3.22}$$

This expression is similar to the eq. (3.18), however, we remark that while η is constant in the gradient method it now depends on the Hessian in Newton methods. This allows an adaptation of the steps taken toward the minimum in real-time according to our position in the cost function landscape. While Newton's method helps us to gain insight into the role of learning rate η and possible improvement, it is most of the time impractical. The main issue resides in the computation of the Hessian matrix that scales badly with an increasing number of parameters. Furthermore, inverting a $n \times n$ Hessian matrix at each iteration of η is computationally expensive.

3.6 Stochastic gradient descent

In section 3.4 we introduced the gradient descent as an optimisation tool. However, as we have seen previously this method suffers from many drawbacks that make it impractical for machine learning. One of the issues resides in the deterministic nature of the Gradient Descent (GD) that makes it impossible to escape local minima. A solution to this issue is the introduction of stochasticity. The dataset is divided into so-called mini-batches and the gradient is computed on each of these subsets. We choose the mini-batches to be smaller than the total dataset. For a dataset composed of n datapoints and a mini-batch of size ω, there are n/ω mini-batches in the dataset. We denote each mini-batch by B_k where $k = 0, \ldots, n/\omega$. The gradient of the cost function can then be reformulated as

$$\nabla_\theta E^{MB}(\theta) = \sum_{i \in B_k} \nabla_\theta e_i(x_i, \theta). \tag{3.23}$$

An iteration of the optimisation process is complete when the gradient of each of the mini-batches has been computed, this is what we call an *epoch*. The eq. (3.18) can be rewritten as

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta E^{MB}(\theta). \tag{3.24}$$

The Stochastic Gradient Descent (SGD) method gives a solution to the expensive computation of the gradient by dividing the task through the mini-batches. It is less costly to compute n/ω times a gradient of ω point than a gradient on the full dataset which comprises n samples. Furthermore, computing the gradient on each of the mini-batches contributes to adding stochasticity and allows to escape local minima in a rugged cost function landscape.

3.7 Back propagation

The optimisation process described in the previous sections relies heavily on the computation of the derivative of the cost function with respect to the parameters of the network θ. Nevertheless, the computation of the derivative can prove to be a tedious task, as it requires to compute as many gradients as parameters at each step of the optimisation process. The backpropagation algorithm [8] takes into account the layered architecture of the network and provides us with a straightforward way to compute the gradient by making use of the chain rule. We recall the activation function a_j^L of neuron j in a layer L

$$a_j^L = \sigma\left(\sum_k \omega_{jk}^L a_k^{L-1} + b_j^L\right) = \sigma(z_j^L), \tag{3.25}$$

where k is the index of a neuron from the previous layer, which has a connection with the neuron j. Therefore at layer $L+1$ we obtain the following activation function

$$a_j^{L+1} = \sigma\left(\sum_k \omega_{jk}^{L+1} a_k^L + b_j^{L+1}\right) = \sigma(z_j^{L+1}). \tag{3.26}$$

As the cost function E depends on the behaviour of all the layers connected to the output, the change in layer L according to the activation z_j^L is defined by

$$\Delta_j^L = \frac{\partial E}{\partial z_j^L}, \tag{3.27}$$

Chain rule coupled with the dependence of E on a_j^L and b^L allow us to obtain

$$\Delta_j^L = \frac{\partial E}{\partial z_j^L} = \frac{\partial E}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial E}{\partial a_j^L}\sigma'(z_j^L), \tag{3.28}$$

$$\Delta_j^L = \frac{\partial E}{\partial z_j^L} = \frac{\partial E}{\partial b^L}\frac{\partial b^L}{\partial z_j^L} = \frac{\partial E}{\partial b^L}. \tag{3.29}$$

One can also find the dependence of E on ω_{kj}^l, by applying chain rule and using the layer $L+1$

$$\Delta_j^L = \frac{\partial E}{\partial z_j^L} = \sum_k \frac{\partial E}{\partial z_k^{L+1}}\frac{\partial z_k^{L+1}}{\partial z_j^L} = \sum_k \Delta_k^{L+1}\frac{\partial z_k^{L+1}}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} \tag{3.30}$$

$$= \sum_k \Delta_k^{L+1}\frac{\partial z_k^{L+1}}{\partial a_j^L}\sigma'(z_j^L) = \sum_k \Delta_k^{L+1}\omega_{kj}^{L+1}\sigma'(z_j^L). \tag{3.31}$$

Finally, we derive E by the weight ω_{jk}^L

$$\frac{\partial E}{\partial \omega_{jk}^L} = \frac{\partial E}{\partial z_j^L}\frac{\partial z_j^L}{\partial \omega_{jk}^L} = \Delta_j^L a_j^{L-1}. \tag{3.32}$$

The eq. (3.31) summarises quite well the backpropagation process. While the feed-forward step described by the activation function a_k^L in eq. (3.25) was computed through the activation of the previous layer a_k^{L-1}, here we notice that the change in layer Δ_k^L depends on the change in layer Δ_k^{L+1}.

3.8 Overfitting, vanishing gradients

Now that we introduced the main steps of the training of an ANN, we need to also explore the different issues that one can encounter during the process. In section 3.1, we briefly introduce the idea of generalisation. We recall that the aim of the training is to obtain a network able to make correct predictions on a dataset never seen before. This ability is called generalisation and is one of the most sought-after capacities of a network in DL. Nonetheless, two main types of issues can arise and hinder this ability, *underfitting* and *overfitting*. Underfitting occurs when a network is not performing enough and overfitting occurs when the network is overperforming and is learning the noise in the training dataset. The main drawback of overfitting is that while the network performs perfectly on the training data, it is unable to make correct predictions on unseen data as depicted in figure 3.2(a). Knowing this, several techniques have been developed to avoid the pitfall of overfitting/underfitting. The first step is to optimise the size of the network according to the task performed. A complex network used for a simple task will lead to overfitting, as the network is too powerful and vice versa. Furthermore, another problem can occur when using a DNN, the so-called *vanishing gradient*. This phenomenon is characterised by a rapid decrease in performance as the depth of the network increases. We recall that the training process involves an optimisation of the parameters of the network performed through backpropagation. The backpropagation step relies heavily on the chain rule derived from the gradient of the cost function as we showed in the eqs. (3.27) to (3.29) and (3.32). However certain types of activation functions such as the sigmoid shown in figure 1.3(b) can contribute to the disappearance of the gradients as $\frac{\partial \sigma_{\text{sigmoid}}}{\partial z} \to 0$ leading to $\Delta_j^L = 0$. Residual learning comes as a solution to this specific issue, it makes use of "skip-connections" which bypass several layers to merge the information with an output further down in the network. By feeding the activation several layers down, we avoid the vanishing of the information in a deep neural network. In figure 3.3 (a) we show a representation of the skip-connection operation. Usually, the number of layers skipped is between two or three, and this unit constitutes what we call a residual block. Let us define an input x and an output $F(x)$ several layers down in the network. This input x merge with $F(x)$ through a skip connection such as,

$$y = x + \mathcal{F}(x). \tag{3.33}$$

While some models preserve the size of the input x, in other networks the input could also see a change in dimension and increase. In this case, the activation x passed several layers down must have the same dimension as the activation $\mathcal{F}(x)$, of the layer that we are trying to merge with. One way to solve this issue is to add a padding of zero around the input x, to increase the dimension before merging with the output $\mathcal{F}(x)$.

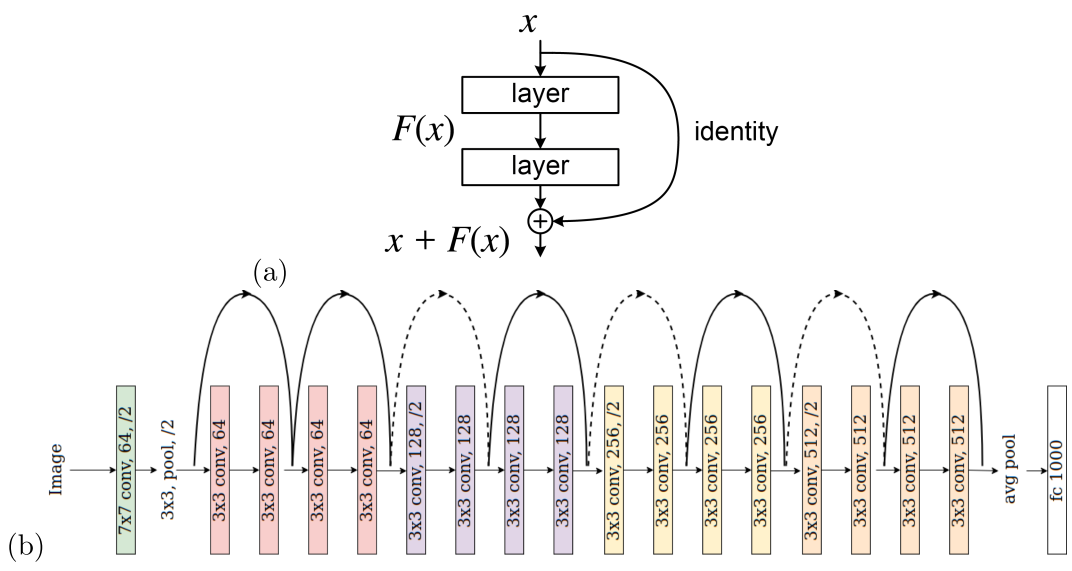


Figure 3.3: (a) Representation of a residual block with two layers (b) Architecture of the ResNet18 [27]. This network is composed of eight residual blocks stacked on top of each other, to which we had one layer in input and a fully connected layer in output. The solid lines show skipped-connection operations where the input x and the output $\mathcal{F}(x)$ have the same dimensions. Meanwhile, the dotted lines show skipped-connection operations where the output $\mathcal{F}(x)$ has a larger dimension than x. In the second case, a padding of zeros is added to the input x before merging with the output $\mathcal{F}(x)$.

3.9 The ResNet18

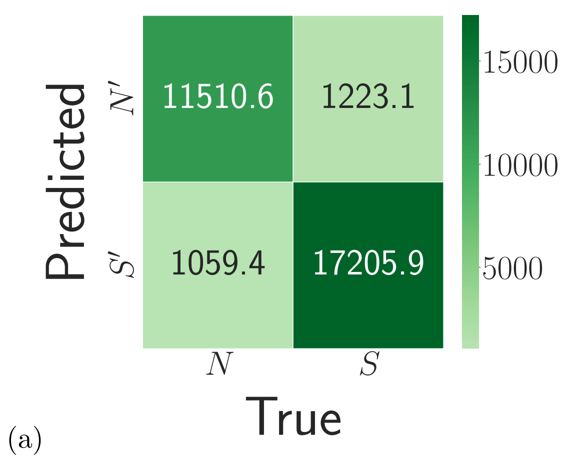
The ResNet [27] is a deep neural network using the skip connection method, it is composed of several residual blocks stacked on top of each other. Several depths of ResNet can be achieved, but we will focus our attention on the ResNet18 shown in figure 3.3 (b). The network is organised in 8 residual blocks of two convolutional layers staked on top of each other, to which we add a convolutional layer as input and a fully connected one as output. This network proved to be performing well in image recognition and won the ImageNet competition in 2015. Furthermore, it is easily adaptable by replacing the last layer with the desired number of outputs. The original network was constructed for two-dimensional images, but in 2018 a three-dimensional version was implemented for training on video [66]. The structure of this ResNet for three-dimensional inputs only differs in the use of three-dimensional convolutional layers instead of the usual two-dimensional ones. Therefore the ResNets architecture appears as a perfect candidate for a deep neural network.

3.10 An example of classification

Now that we have given a brief overview of key concepts and methods to know when training a ML process, we will provide a thorough example of training for phase classification with CNN. This classification aims to obtain a network capable of predicting the two phases of the percolation model. Our dataset is composed of 310000 percolation lattices labelled as spanning or non-spanning, i.e. spanning and non-spanning denoted by S and N, respectively. We randomly split this dataset using a 90%/10% training/validation split. We recall that during the training each of the samples of the training dataset is presented to the network. The performance of the network is checked after each epoch on the validation set. As such, using a validation set provides a way to perform a dynamic test during the training phase. Once the splitting is performed, the labels are encoded using the one-hot encoding method mentioned in section 3.3. For our training, we employ the ResNet18 network, presented in the previous section. We modify the last layer of this network to have two outputs corresponding to the two possible predictions. A softmax activation is added to the last layer and acts as a normalisation function for the predictions of the network. The softmax function is defined as

$$\sigma_{\text{softmax}}(y)_i = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}}, \tag{3.34}$$

where y_i is the output of node i. As we have seen in section 1.4, classification tasks deal with categorical data. The label seen by the network during the training is encoded. We therefore need a loss function adapted to this type of problem. In section 3.3, we introduce the CCE, a cost function tailored for categorical data. This is the cost function that we employ for our example. We recall that a ML training is an optimisation problem. However, as we have seen in sections 3.3, eqs. (3.18) and (3.24), employing gradients-based methods on large datasets is computationally expensive. To reduce the computational load, it is customary to divide the dataset into mini-batches, which in turn allows us to add stochasticity to our optimisation process. We choose to use a mini-batch of size 256. We call an epoch a full passing of the dataset through the network. For this example and the rest of the trainings in this thesis, we launch ten trainings with the same network but different random seeds. As we have seen in section 3.3, in some cases the performances of the network after training might depend on the initial point in the cost function landscape. As such, by repeating the trainings with different initial random seeds, we ensure that the results obtained are not dependent on a specific initial position in the cost



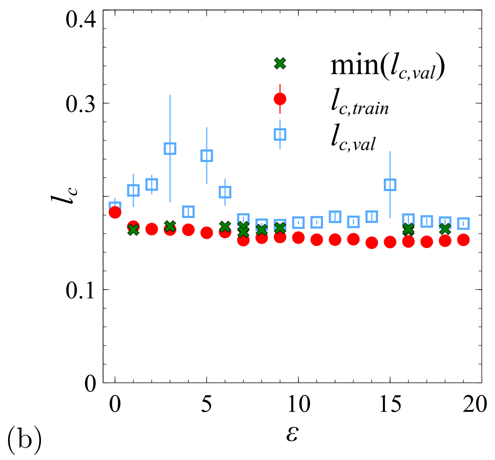


Figure 3.4: (a) Average confusion matrix for *classification* according to spanning/non-spanning. The dataset used is the test data τ and the models used for predictions are those corresponding with a minimal $l_{\mathrm{c,val}}$. The true labels for N and S, are indicated on the horizontal axis while 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 ϵ for classification according to spanning/non-spanning.

function landscape. The implementation is done with the PyTorch [67] library. During the training, we track the performance of the network through the evolution of the validation and training loss functions denoted by $l_{c,\mathrm{train}}$ and $l_{c,val}$, respectively. The curves associated with our example are shown in figure 3.4(b), and correspond to an averaging of the ten differents $l_{c,\mathrm{train}}$ and $l_{c,val}$ obtained.

After training, we evaluate the efficiency of the model on a test set τ. A confusion matrix is an efficient way to summarise the performances of a model after training. The horizontal axis of the matrix corresponds to the true labels while the vertical relates to the labels predicted by the network. As a result, we expect that a well-trained model produces a heavily diagonal confusion matrix and a poorly trained model a confusion matrix with several off-diagonal elements. The confusion matrix obtained after the training of the two phases is shown in figure 3.4(a). This matrix is an average of the ten confusion matrices obtained from the ten different random seed-initiated trainings. We observe that the network is overall able to distinguish the two phases in most cases. However, we note that 7% of the samples were wrongly predicted. This result will be discussed further in the section 5.5.

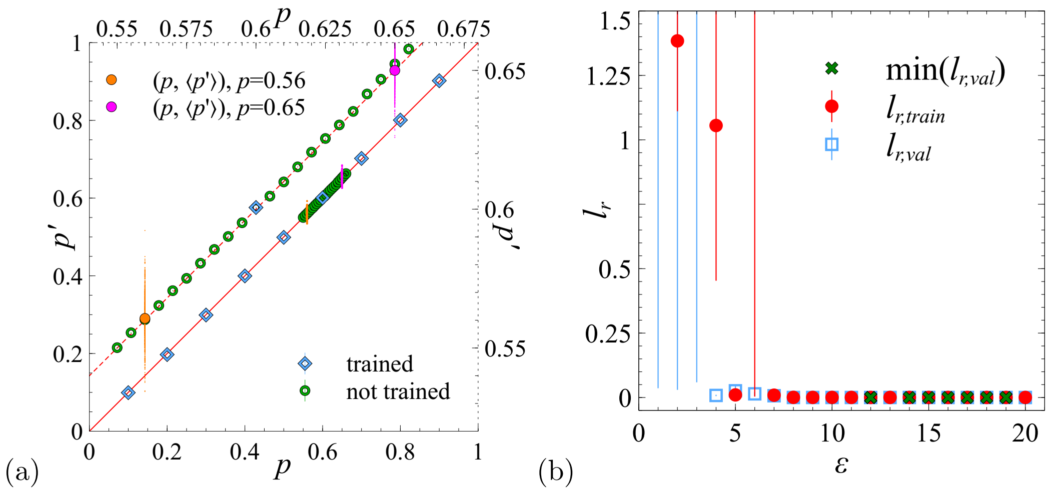


Figure 3.5: (a) Average prediction curve obtained for *regression* according to p at the minimal $l_{\mathrm{r,val}}$. The dataset used is the test data τ and the models used for predictions are those corresponding with a minimal $l_{\mathrm{r,val}}$. The blue open squares denote p-values that have been used during the training and the green open circle shows p-values that were not trained. (b) Dependence of losses $l_{\mathrm{r,train}}$ and $l_{\mathrm{r,val}}$ averaged on the number of epochs ϵ for *regression* according to p. The squares (blue open) denote $l_{\mathrm{r,train}}$ while the circles (red solid) show $l_{\mathrm{r,val}}$. The green crosses show the minimal $l_{\mathrm{r,val}}$ for each of the ten trainings.

3.11 An example of regression

Still using the same percolation dataset, we want to obtain a network capable of making predictions on the density p. This can be achieved by training our network for a regression task [17]. Instead of separating samples into predefined classes, the network then makes continuously ranging predictions. While we previously used the full dataset in the classification example in section 3.10, here we choose to train the network for a subset of the dataset with density $p = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8$ with 10000 samples per classes. Similarly to the previous section, we perform a 90%/10% training/validation split. The key difference between classification training and regression training is the lack of encoding of the label. In the case of the classification training shown in the previous section, an encoding of the label was performed. Here, we recall the existence of a direct correlation between the label y and the inputs x. Therefore, the labels are not encoded. We again use the ResNet18 network and modify the last layer to only include one neuron as output. While in classification the output layer was giving us the probability of belonging to a given class, here the unique output neuron is generating continuous predictions on the density. The cost function chosen for this task is the MSE loss, which allows us to directly monitor the deviation between the predictions of the network and the actual labels of the samples. Again, during the training the quality of the prediction of our network can be tracked through the evolution of validation $l_{r,val}$ and training loss $l_{r,train}$, as shown in figure 3.5(b).

Once the training is done, we challenge the performance of the network on a test set composed of percolation lattices with densities $p = 0.55, 0.555, 0.56, 0.565, \ldots, 0.655, 0.66$. We average the predictions made on the test set and plot the average prediction curve in figure 3.5. The plot obtained follows the red line corresponding to the perfect prediction. This leads us to conclude that the training was well implemented. We will expand more on the result of this training in section 5.2.

3.12 Variational Autoencoder

In the previous sections, we gave two examples of the application of *supervised learning* tasks, classification and regression. While supervised learning methods provide nice applications tasks like classification or regression, *unsupervised learning* can provide useful insight into the dataset. In this section, we give the general background of one specific unsupervised learning task, the VAE.

Before invoking VAEs, it is instructive to introduce the classical autoencoders. Autoencoders are networks used for unsupervised training; their task is to encode an input x_i in a way that allows the reconstruction of the input x_i after decoding [17]. They are composed of two main building blocks, an encoder and a decoder. One might think about the simplicity of this task as it would be equivalent to learning the identity matrix. However, this is made impossible by an intermediate hidden layer. Two main different kinds of autoencoders exist, the *undercomplete* and the *overcomplete*. The *undercomplete* autoencoder is perhaps the most intuitive of the two. In this network, the hidden layer is smaller than the encoder and decoder and acts as a bottleneck. An analogy of this process would be image compression, where the aim is to compress the image in a way that would preserve it after uncompression. The *overcomplete* autoencoder has a larger intermediate layer between the encoder and the decoder. As a result, the intermediate layer contains more neurons than the encoder and decoder layers, which might lead to a network inclined to copy the input to the output. To circumvent this issue, several techniques exist, one of the most popular is the denoising autoencoder (DAE). Let us consider a dataset composed of N independent and identically distributed datapoints $D = \{x_i\}_{i=1}^{N}$. Each datapoint in the dataset follows a distribution $p^*(x)$ and lives on the manifold of the dataset D. The denoising autoencoder operates by displacing these points away

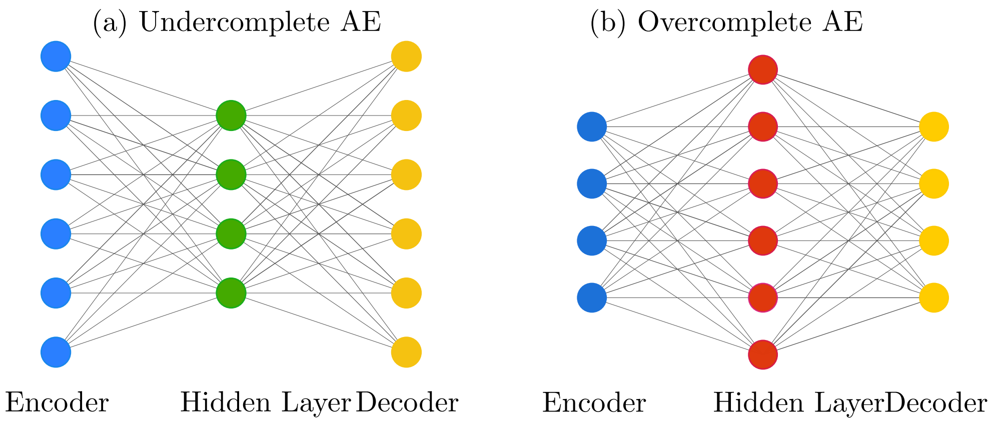


Figure 3.6: (a) Architecture of a fully connected undercomplete autoencoder, the hidden layer contains fewer neurons than the encoder or decoder layers. (b) Architecture of a dense overcomplete autoencoder, this time the hidden layer has more neurons than the encoder or decoder layers.

from the manifold through the addition of noise, the training then aims at moving them back on the manifold. Through this process, the network is forced to learn a vector field mapping each displaced point to the manifold. As such, the trained network knows implicitly the structure of the manifold representing the samples of the dataset, and can reconstruct the input.

Autoencoders are useful tools with many applications such as phase detection [68–70] or anomaly detection in high energy physics [71, 72]. However, we are interested in an even more powerful tool: VAE. While a classical autoencoder aims at reconstructing a compressed representation of the input data, VAEs are generative networks, i.e. they create *new samples* similar to samples of the dataset D through a probabilistic procedure. We wish to have a network able to produce new samples following a distribution $p_\theta(x)$ such that

$$p_\theta(x) \approx p^*(x). \tag{3.35}$$

Here θ denotes the parameters of the network. As we suppose that the samples are independent and identically distributed, the total probability distribution of the model can be written as a product of distribution. It is then possible to define the log-probability of the model as

$$\log p_\theta(D) = \sum_{x \in D} \log p_\theta(x). \tag{3.36}$$

This function acts as a cost function for generative models.

The training of a VAE is performed through the introduction of a latent variable z. A latent or hidden variable is a parameter of the model that is not typically observed. When dealing with marginal distributions in physics, it is customary to integrate out some variables. However, this often contributes to inducing complex correlations between the remaining variables. Here, we apply the reverse idea, by introducing a latent variable z we aim at simplifying correlations between observables in the system. Thus, the distribution function of the model can be rewritten as a marginal distribution over the samples x and the latent variable z

$$p_\theta(x) = \int p_\theta(x,z)dz. \tag{3.37}$$

This distribution is also referred to as the marginal likelihood [73]. According to the general product rule we have,

$$p_\theta(x,z) = p_\theta(x|z)p_\theta(z), \tag{3.38}$$

where $p_\theta(z)$ is the distribution of the latent space, also called prior distribution. Similarly to a classical autoencoder, a VAE is composed of an encoder, where the distribution of the dataset is coded, and a decoder, which this time generates new samples. The hidden block between the encoder and decoder is a probabilistic sampler. The distribution $p_\theta(x|z)$ represents the mapping from the latent space to the new sample that we generate. This distribution is often called *stochastic decoder*. The inverse distribution $p_\theta(z|x)$ maps the distribution of the input sample to the latent space and is often called *stochastic encoder*. However, one problem arises, the marginal likelihood displayed in eq. (3.37) is not tractable as it would require to sum over all the configurations of the latent variables. To solve this issue we introduce an infer distribution $q_\phi(z|x)$ which acts as an approximation of the distribution $p_\theta(z|x)$. Usually, this distribution is chosen as a Gaussian multivariate [17,74]. From this, it is possible to rewrite the log-likelihood as

$$\begin{aligned}
\log\, p_\theta(x) &= \mathbb{E}_{q_\phi(z|x)}[\log\, p_\theta(x)], && (3.39)\\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{p_\theta(x,z)}{p_\theta(z|x)}\right)\right], && (3.40)\\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{p_\theta(x,z)}{q_\phi(z|x)}\frac{q_\phi(z|x)}{p_\theta(z|x)}\right)\right], && (3.41)\\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{p_\theta(x,z)}{q_\phi(z|x)}\right)\right] + \mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{q_\phi(z|x)}{p_\theta(z|x)}\right)\right], && (3.42)\\
&= \mathcal{L}_{\theta,\phi}(x) + D_{KL}(q_\phi(z|x)||p_\theta(z|x)), && (3.43)
\end{aligned}$$

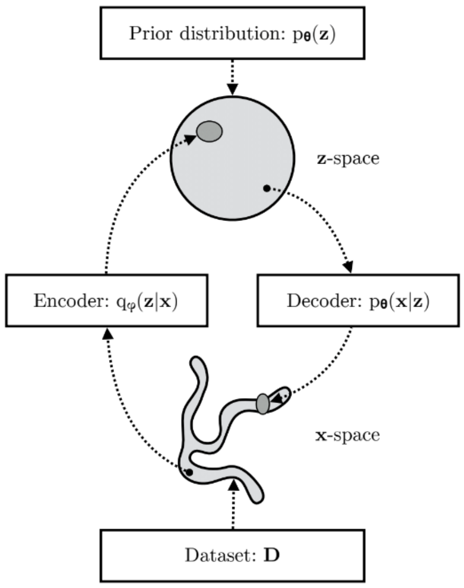

Prior distribution: $p_\theta(\mathbf{z})$
z-space
Encoder: $q_\varphi(\mathbf{z}|\mathbf{x})$
Decoder: $p_\theta(\mathbf{x}|\mathbf{z})$
x-space
Dataset: **D**

Figure 3.7: Diagram of the mappings between the x-space of the dataset D and the z-space of the latent space for a VAE [75]. The distribution $p_\theta(x|z)$ gives the mapping of the z-space to the x-space and is usually refered as *stochastic decoder*. The probability $q_\phi(z|x)$ is called *stochastic encoder*, and is the infer distribution, usually a multivariate Gaussian which approximates the intractable distribution $p_\theta(z|x)$.

where $\mathcal{L}_{\theta,\phi}(x)$ is the reconstruction error and $D_{KL}(q_\phi(z|x)||p_\theta(z|x))$ is the KL divergence. By reorganising the term in 3.43 we obtain,

$$\log\, p_\theta(x_i) - \mathcal{L}_{\theta,\phi}(x_i) = D_{KL}(q_\phi(z|x)||p_\theta(z|x)), \tag{3.44}$$

We recall the non-negativity of the KL divergence defined in section 3.3. As such, $\mathcal{L}_{\theta,\phi}(x)$ is also called the Evidence Lower BOund (ELBO) and corresponds to the lower bound of the log-likelihood. From this, we deduce that maximising the ELBO contributes to maximising the log-likelihood and decreasing the KL divergence.

As we previously mentioned, ML trainings are equivalent to solving optimisation problems, i.e. minimise the cost function. Therefore, we compute the gradient of the reconstruction error with respect to the parameter θ

$$\nabla_\theta \mathcal{L}_{\theta,\phi}(x) = \nabla_\theta \mathbb{E}_{q_\phi(z|x)}[\log\, p_\theta(x,z) - \log\, q_\phi(z|x)], \tag{3.45}$$
$$= \mathbb{E}_{q_\phi(z|x)}[\nabla_\theta(\ \log\, p_\theta(x,z) -\ \log\, q_\phi(z|x))], \tag{3.46}$$
$$= \mathbb{E}_{q_\phi(z|x)}[\nabla_\theta \log\, p_\theta(x,z)]. \tag{3.47}$$

Let us now look at the gradient with respect to ϕ

$$\nabla_\phi \mathcal{L}_{\theta,\phi}(x) = \nabla_\phi \mathbb{E}_{q_\phi(z|x)}[\log\, p_\theta(x,z) - \log\, q_\phi(z|x)]. \tag{3.48}$$

While in eq. (3.47) we could easily move the gradient ∇_θ inside the expected value $\mathbb{E}_{q_\phi(z|x)}$, here we cannot as the expected value $\mathbb{E}_{q_\phi(z|x)}$ depends on the same parameter as our gradient ∇_ϕ. Therefore, the backpropagation method described in section 3.7 cannot be directly applied here. To bypass this issue, we make use of the *reparametrization trick* [75]. We recall that z is sampled from the infer distribution and can be expressed as $z \sim q_\phi(z|x)$. Usually, it is not easy to sample from this distribution. A solution is to define z as a differentiable deterministic function of several parameters

$$z = g(\epsilon, \phi, x), \tag{3.49}$$

where $\epsilon \sim p(\epsilon)$ is a random noise sampled from a simple probability distribution, independent of ϕ and x. Usually, we choose ϵ such as $\epsilon \sim \mathcal{N}(0,1)$, given that $q_\phi(z|x)$ is a multivariate Gaussian distribution, eq. (3.49) can be expressed as

$$z = \mu + \sigma \odot \epsilon. \tag{3.50}$$

According to the Law of The Unconscious Statistician (LOTUS), when dealing with a function $g(X)$ of a random variable X, the expected value $\mathbb{E}_{p(\epsilon)}[g(X)]$ is

$$\mathbb{E}_{p(\epsilon)}[g(X)] = \int g(x)p(\epsilon)d\epsilon \tag{3.51}$$

Using the change of variable in eq. (3.49) and the LOTUS it is then possible to rewrite eq. (3.48)

$$\begin{aligned}
\nabla_\phi \mathcal{L}_{\theta,\phi}(x) &= \nabla_\phi \mathbb{E}_{p(\epsilon)}[\log\, p_\theta(x,z) - \log\, q_\phi(z|x)], && (3.52)\\
&= \mathbb{E}_{p(\epsilon)}[-\nabla_\phi \log\, q_\phi(z|x)], && (3.53)\\
&\approx -\nabla_\phi\, \log\, q_\phi(z|x)], && (3.54)
\end{aligned}$$

The reparametrisation trick provides us with a simple way to optimise the parameters of the network. While backpropagation cannot happen through the random variable z, it can occur through the change of variables $z = g(\phi, x, \epsilon)$.

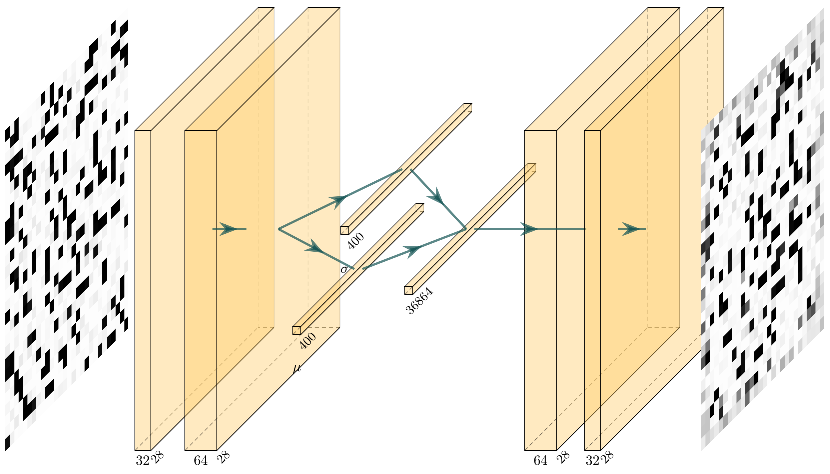


Figure 3.8: Architecture of the VAE used during the training of percolation states.

3.12.1 An example of application of the VAE

We will now provide a detailed example of the training of a VAE. The goal of this training is to obtain a network able to generate new percolation samples. For our study, we use a similar network as the one described in the article of Cheng et al. [76], which is composed of two convolutional layers, a bottleneck with $z = 400$ for the dimension of the latent space and two deconvolutional layers. Our training set is composed of percolation states at densities $p = [0.4, 0.41, 0.42\ldots, 0.79, 0.8]$ with 1000 samples per class. Again, following the strategy employed for the classification and the regression, we split our dataset following a 90%/10% training/split. As this is an unsupervised method we simply feed these configurations to the network without label. For the reconstruction loss, we choose the Binary Cross-Entropy (BCE) defined as

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n} y_i \log[f(x_i)] + (1 - y_i) \log[1 - f(x_i)], \tag{3.55}$$

where $f(x_i)$ is the prediction made by the network on the input x_i.

We set the dimension of our latent space to $z = 400$ and train for 20 epochs. After completing a training cycle we obtain the following outputs In figure 3.9 we display the output of the trained VAE. The network seems to perform the task of reproducing the image, however, the high similarity of the image might lead us to think that the VAE process did not work as it was supposed to. We will discuss the

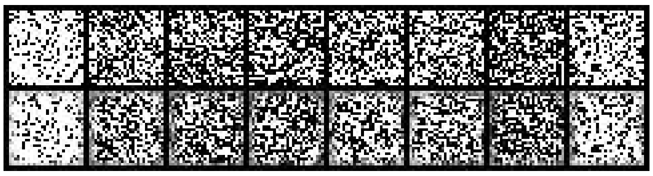


Figure 3.9: On the top row we see the input fed by the network, on the second row, we see the reproduction made by the network after training. As we can see the network is able to reproduce the inputs after training.

result of this training in section 5.10 in more detail.

3.13 Conclusion

In this chapter, we gave a brief introduction to some essential ML concepts. We presented ML trainings as optimisation problems and provided several methods such as Newton's method or SGD to solve it. We introduced the concept of generalisation, at the centre of the ML philosophy. More than being able to perform well on our training set, we aim at performing well on unseen data. Finally, we provided detailed examples of the training processes for *classification*, *regression* and *VAE*.

Chapter 4

The datasets

When training a ML, the most important criterion is the quality of the data fed to our network. A bad dataset can lead to a biased network unable to generalise on a test set. As such, one should worry about the quality of the dataset used for training. In general, when training a ML task, we observe that small inputs tend to be used. In condensed matter and statistical physics, small systems might see a change in the behaviour of some parameters of the systems due to finite size effect. These changes need to be taken into account when proceeding with our ML analysis. For our studies, we generated our own datasets with sizes similar to those observed in the literature. In this part, we will present these two datasets, the percolation dataset and the Anderson dataset.

4.1 The percolation dataset

4.1.1 Training/Validation datasets

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 $\psi_i(p)$. When training, we split this dataset according to a 90%/10% training validation split. In the rest of this thesis, we will use the notation $T \cup \mathcal{V}$ to qualify this dataset.

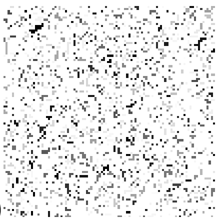
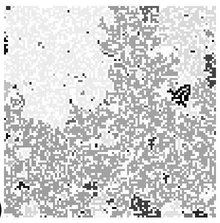
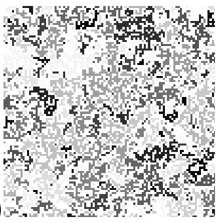
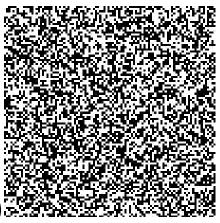
In Figure 4.1 we have shown examples of percolation states generated for various p values. The different grey scales used in Figure 4.1 mark the different (a) (b) (c) (d)

Figure 4.1: Examples of four percolation clusters of size $L^2 = 100^2$, obtained for (a) $p = 0.2 < p_c$, (b) $p = 0.6 > p_c$ and (c) $p = 0.5$, i.e. just below p_c. Occupied sites are marked by small dots while empty sites are left white. Each cluster of connected sites has been identified through the HK algorithm. While individual clusters have been highlighted with different grey scales for the first three images, image (d) 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.

connected clusters. However, for the ML approach below, we shall only use the numerical values 0 and 1 corresponding to the state $\psi_i(p)$. This is visualized as the simple black and white version shown, e.g., for $p = 0.5$ in figure 4.1(d).

We emphasise that in this 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 an exact label for the supervised learning approach. Indeed, using the preexisting binomial function in Python led to the creation of lattices of densities $p \pm \delta p$. This would have resulted in several samples labeled with p which would in reality have slightly different densities which in turn could have affected the quality of the training. We note that with our construction $p = N_{\rm occ}/L^2$ can therefore also be called the percolation *density*. For the ML results discussed below, it will 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. This smaller spacing allows us to have more samples in the transition region. We will later see that this results in some deviations from perfect classification/regression. Last, we have also generated a similar training set with $L = 200$ for 20 p-values $0.1, 0.2, \ldots, 0.5, 0.56, \ldots, 0.66, 0.7, 0.8, 0.9$. We find that our results do not change significantly when using this much larger data set. To prove our point, we will present the result of a classification training to identify the presence of spanning/non-spanning for $L = 200$.

4.1.2 The spanning/non spanning property of the lattices

For our study we choose to focus on three parameters of interest, the density p (i), the correlation function $g(r)$ (ii), and the presence or absence of a spanning cluster (iii).

	N		S	
p	#	κ	#	Π
0.1	10 000	1.0	0.0	0.0
0.2	10 000	1.0	0.0	0.0
0.3	10 000	1.0	0.0	0.0
0.4	10 000	1.0	0.0	0.0
0.5	10 000	1.0	0.0	0.0
0.55	9889	0.9889	111	0.0111
0.56	9778	0.9778	222	0.0222
0.565	9555	0.9555	445	0.0445
0.57	9171	0.9171	829	0.0829
0.575	8541	0.8541	1459	0.1459
0.58	7731	0.7731	2269	0.2269

Table 4.1: Distribution of spanning/non-spanning in the full dataset $\mathcal{T} \cup \mathcal{V}$ for $p \in [0.1, 0.58]$. We notice the presence of several percolating samples from $p = 0.55$, which is below $p_c \approx 0.585$. The inverse trend is observed above p_c, where several non-percolating samples remain.

As a first step to identify $g(r)$ and the presence of spanning clusters, we label each of the clusters in each of $\psi_i(p)$ the states. This labelling is performed through the HK algorithm described in section 2.1.4. After the complete labelling of the dataset, we can give the precise distribution of spanning (non-spanning) samples in the dataset. As a first step to our analysis of our dataset we retrieve the finite size version of the parameters Π and κ defined in section 2.1.1. In figure 4.2(a) we observe the extracted Π_{100}, κ_{100} and $p_c(100)$, the latter obtained through the crossing of Π_{100} and κ_{100}. We note that $\Pi(p)$ behaves qualitatively as expected [48]. Due to finite size effect, close to p_c, we find samples which are already spanning for $p < p_c$ as well as samples for $p > p_c$ which are not spanning. Clearly, $p_c(L = 100) \sim 0.585(5) < p_c$. This latter behaviour 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. For reference, we now have 12 values $p = 0.1, \ldots, 0.58 < p_c(100)$ and 18 values $p = 0.59, \ldots, 0.9 > p_c(100)$. We also note that the training set contains 92.7% of states without a spanning cluster below p_c and 94.8% are spanning above p_c. The detailed analysis of the proportion of spanning and non-spanning samples is given in table 4.1 and 4.2. We observe that the first percolating states appear earlier than $p_c(100) \sim 0.585$. The inverse trend also occurs for $p > p_c(100) \sim 0.585$ where non-percolating samples remain until reaching $p = 0.635$. A training set with $L = 200^2$ was also generated in figure 4.3. Similarly to the dataset with $L = 100$, the HK cluster labelling algorithm was applied to each of the samples. After performing the labelling, we find a new percolation

	N		S	
p	#	κ	#	Π
0.585	6577	0.6577	3423	0.3423
0.59	3841	0.3841	6159	0.6159
0.595	2501	0.2501	7499	0.7499
0.6	1547	0.1547	8453	0.8453
0.605	827	0.827	9173	0.9173
0.61	373	0.0373	9627	0.9627
0.615	162	0.0162	9838	0.9838
0.62	70	0.007	9979	0.9979
0.625	21	0.0021	9993	0.9993
0.63	7	0.0007	9997	0.9997
0.635	3	0.0003	10 000	1.0
0.64	0	0.0	10 000	1.0
0.645	0	0.0	10 000	1.0
0.65	0	0.0	10 000	1.0
0.655	0	0.0	10 000	1.0
0.66	0	0.0	10 000	1.0
0.7	0	0.0	10 000	1.0
0.8	0	0.0	10 000	1.0
0.9	0	0.0	10 000	1.0

Table 4.2: Distribution of spanning/non-spanning in the full dataset $\mathcal{T} \cup \mathcal{V}$ for $p \in [0.585, 0.9]$. We notice the presence of several percolating samples from $p = 0.55$, which is below $p_c \approx 0.585$. The inverse trend is observed above p_c, where several non-percolating samples remain.

threshold $p_c(L = 200) \sim 0.588(6) < p_c$. While $p_c(L = 200)$ remains lower than p_c, we notice that $p_c(L = 100) < p_c(L = 200) < p_c$. As the size L of the lattice increases $p_c(L) \xrightarrow[L\to\infty]{} p_c$, therefore, it appears logical that $p_c(L = 100) < p_c(L = 200)$.

4.1.3 The correlation function and correlation length

The correlation function $g(r)$ was computed for each of the 31000 samples in the dataset with periodic boundary conditions. Let us consider an occupied site i in a lattice $\psi_i(p)$, the computation of $g(r)$ involves visiting each of the other sites in the lattice and checking first if they are occupied and secondly, in the case where they are occupied if they belong to the same cluster as the original site i. For each of the pair of sites i and i' controlled, we keep in memory the distance r separating them, in a list and if they belong to the same cluster or not by appending 1 or 0 to a list. Once all of the sites of the lattice are checked we normalise the numbers obtained by dividing by the number of sites visited. It is useful to note that we subtract

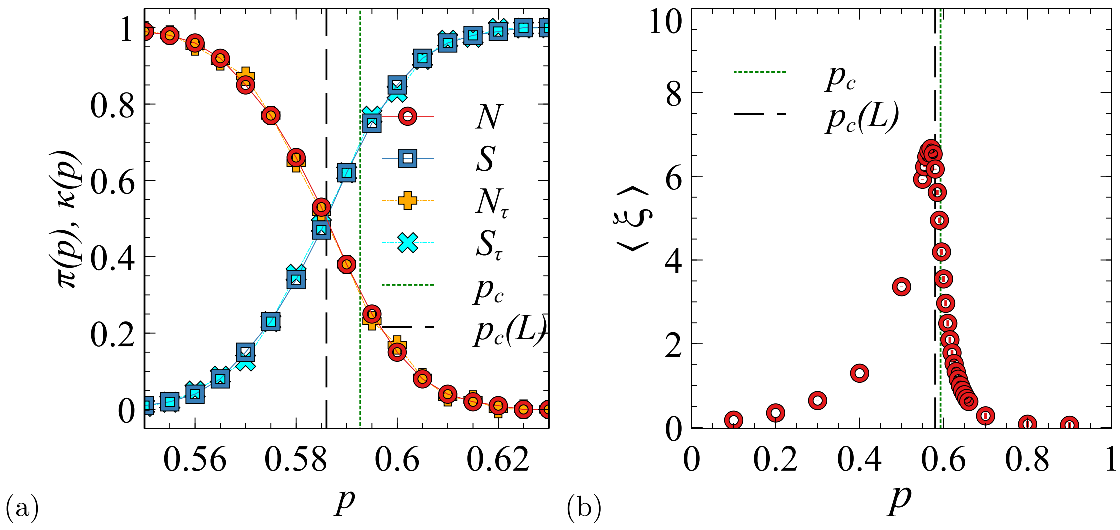


Figure 4.2: (a) Probabilities $\Pi(p)$ and $\kappa(p)$ of having a spanning cluster (S, blue open squares)/ or not (N, red open circles) close to the percolation threshold for dataset $\mathcal{T} \cup \mathcal{V}$, respectively. Similarly, N_τ (orange $+$) and S_τ (cyan $\times$) for probabilities $\Pi(p)$, $\kappa(p)$ obtained for the test data set τ. The vertical lines denote $p_c(L)$ (dashed) and p_c (dotted). (b) Average correlation length $\langle \xi \rangle$ with respect to p. Each of the red open circles designates the average ξ over 10000 samples. The green dotted and black dashed lines indicate respectively $p_c \approx 0.5927$ and the $p_c(100) = 0.585$ associated with our dataset.

the probability of belonging to the largest cluster from the correlation function, the correlation retained is therefore

$$g'(r) = g(r) - P_{\text{largest}}. \tag{4.1}$$

In figure 4.4 we display several correlations associated with random samples in our dataset taken at $p = 0.4 < p_c$, $p = 0.585 \approx p_c$ and at $p = 0.7 > p_c$. On the top row, we show plots of g and g' at these three p-values on a linear scale. On the bottom row, we plot g and g' on a logarithmic scale. In each figure we provide the decay $g \sim \exp(-r/\xi)$, found in section 2.1.2. For every plot, we notice that while $\exp(-r/\xi)$ is close to $g(r)$ and $g'(r)$ the slope does not coincide perfectly. This could be attributed to a normalisation issue. The curves shown in figure 4.4(a) display a common behaviour of g and g'. At low p-values the largest cluster remains small and the probability of belonging to it is negligible. However, when entering the transition region we observe a divergence of the behaviour of g and g'. In figure 4.4(b), we clearly see a convergence of $g' \to 0$ while $g \to 0.1$. The same behaviour is observed in figure 4.4(c), where $g' \to 0$ while $g \to 0.55$. As the value p increases,

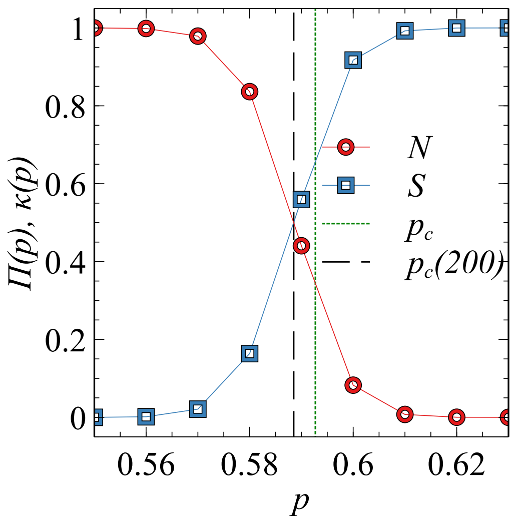


Figure 4.3: (a) Probabilities $\Pi(p)$ and $\kappa(p)$ of having a spanning cluster (S, blue open squares)/ or not (N, red open circles) close to the percolation threshold for dataset $\mathcal{T} \cup \mathcal{V}_{200}$, respectively. Following the same convention as figure 4.2 the vertical lines denote $p_c(L = 200) \sim 0.588(6)$ (dashed) and p_c (dotted).

the probability of a point belonging to the largest cluster increases, leading to a saturation of the correlation function. Therefore, we adopted g' as the correlation function of our system.

From these correlation functions we compute the correlation lengths ξ following eq. (2.13). While theory predicts a diverging ξ at $p \approx p_c$, the correlation length obtained for our dataset does not diverge due to the finite size effect. This finite size effect is also observed through the shift of $p_c(100)$ from p_c.

We recall the presence of the normalisation factor in ξ as seen in eq. (2.22). As such, ξ presented in figure 4.2 is normalised.

4.1.4 The test datasets: τ, $\tau_{\mathbf{sl}}$, $\tau_{\mathbf{rw}}$ and $\tau_{\mathbf{fb}}$

As stated throughout chapter 3, one of the most sought qualities of a ML network is the generalisation. Therefore, to test the ability of prediction of our network we need a test set never seen by the network during the training phase. We generate a test data set, τ, 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 [20].

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

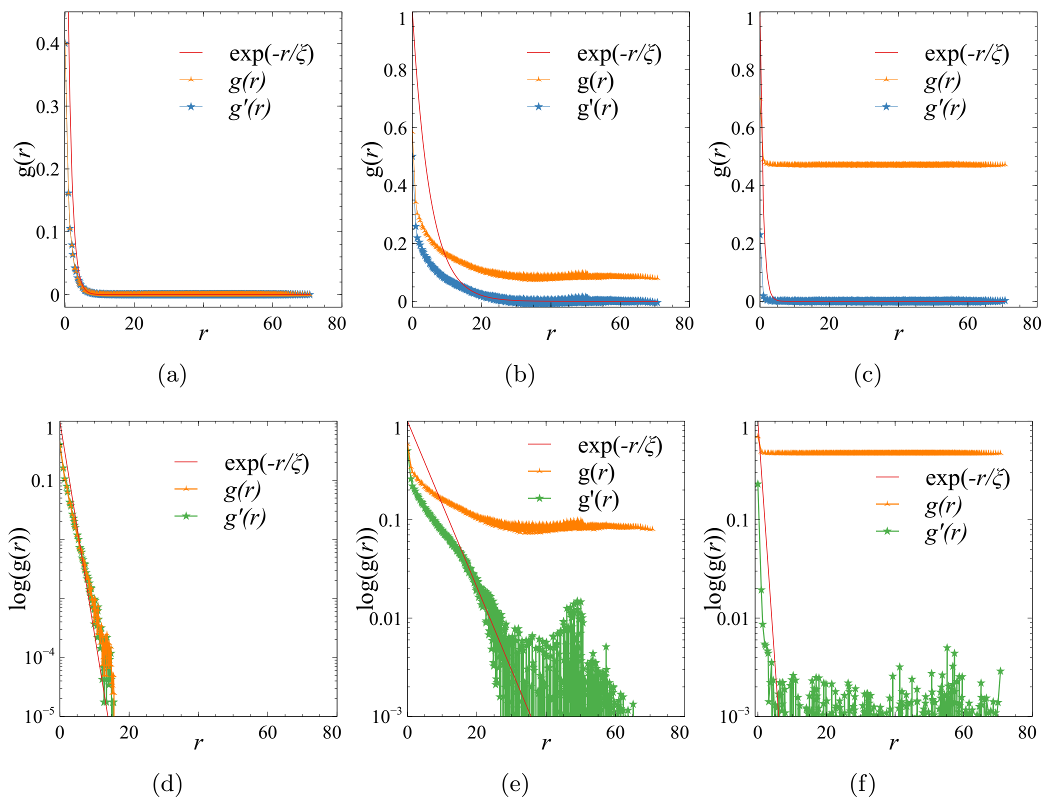


Figure 4.4: On the first row correlation function for (a) $p = 0.4$, (b) $p = 0.585$ and (c) $p = 0.7$ on a decimal scale. On the bottom row correlation function for (d) $p = 0.4$, (e) $p = 0.585$ and (f) $p = 0.7$ on a logarithmic scale. In each plot, the red solid line denotes the decay $\exp(-r/\xi)$. We note that for $p < p_c(100) \approx 0.585$ $g(r) \approx g'(r)$.

first special data set, τ_{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 τ_{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 ψ_i without any connecting path for $p = 0.1$ and 0.2. In the next set, τ_{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 sample in τ_{rw} is spanning. We again add two ψ_i for $p = 0.1$ and 0.2 without the connected random path. Finally, the third special data set, τ_{fb}, again contains 27000 lattices for the same previously mentioned 27 p-values, but in each of the states, we apply random *firebreak* paths, horizontally

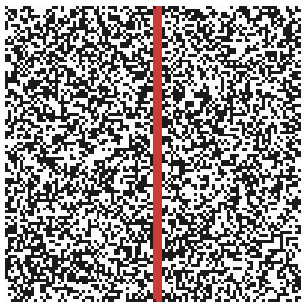
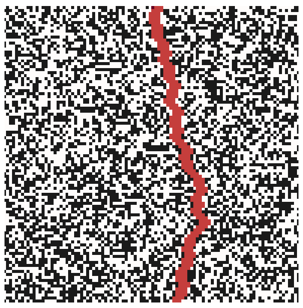
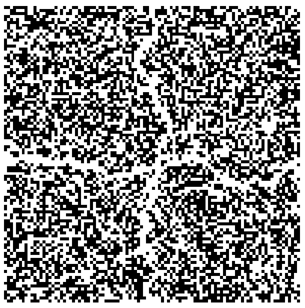
(a)  (b)  (c) 

Figure 4.5: Examples of percolation images from the three special test sets with (a) $\tau_{\rm sl}$, a percolating straight line from top to bottom, (b) $\tau_{\rm rw}$, a percolating random path from top to bottom and (c) $\tau_{\rm fb}$, 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, $p = 0.5$.

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 figure 4.5.

4.2 The Anderson dataset

4.2.1 Training/Validation datasets

For the sake of clarity, we choose to make use of the same notation introduced for the study of the percolation model. As such, we will denote the training/validation sets as $\mathcal{T} \cup \mathcal{V}$ and the test sets as τ. To conduct our study we generated a primary dataset $\mathcal{T} \cup \mathcal{V}_{[15,18]}$ composed of eigenstates for 17 disorders $W = 15, 15.25, \ldots, 17.75, 18$ across the Anderson transition in the centre of the band close to $E = 0$. The computation of the states $\psi = \sum_i \psi_i |i\rangle$ were performed with the JADAMILU library [77–79]. For each disorder, $N = 5000$ independent samples were created for three system sizes, $L = 20^3$, $L = 40^3$ and $L = 100^3$. An associated test set $\tau_{[15,18]}$ was created for these three system sizes with 17 disorders $W = 15, 15.25, \ldots, 17.75, 18$ and $N_{\rm test} = 500$. To compare our results to the previous work of Ohtsuki et al. [45], several disorders $W = 14, 14.25, 14.5, 14.75$ and $W = 18.25, 18.5, 18.75, 19$, also with $N = 5000$ independent samples were added to the original dataset to create the second set $\mathcal{T} \cup \mathcal{V}_{[14,19]}$. The aim is to allow us to observe the difference in performance between the simple network described by Ohtsuki and the ResNet18. Similarly to the first dataset, we also generate an associated test set $\tau_{[14,19]}$ for the three system

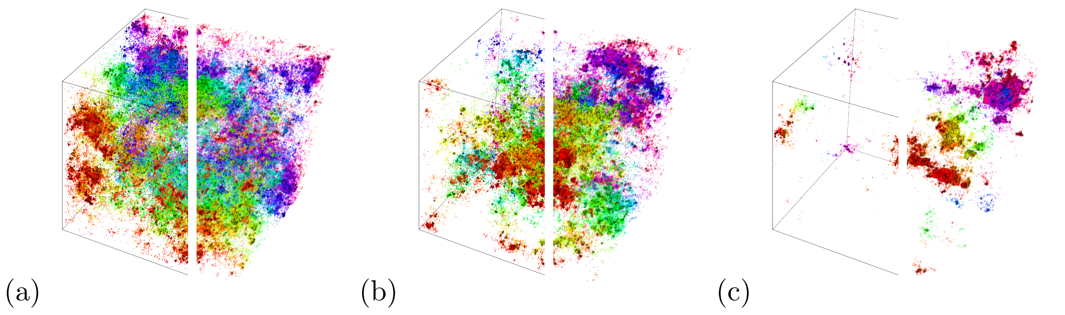


Figure 4.6: (a) Extended, (b) critical and localised (c) wave function probabilities $|\psi(\vec{r})|^2$ for the 3D Anderson model with periodic boundary conditions at $E = 0$ with $N = 100^3$ and $W = 14$, 16.5 and 19, respectively. Every site with probability $|\psi(x, y, z)|^2$ larger than the average $1/N^3$ is shown as a box with volume $N|\psi_{E=0}(x, y, z)|^2$. Boxes with $N|\psi(x, y, z)|^2 > \sqrt{1000}$ are plotted with black edges. The colour scale distinguishes between different slices of the system along the axis into the page. In each panel, the left half is the originally constructed image while the right half shows the image in its converted PNG form with 500×500 pixel resolution. Obviously, upon conversion, the black boxes around the large $|\psi(x, y, z)|^2$ become less prominent and the outside black frames are also removed.

sizes, with $N_{\text{test}} = 500$.

During our study, two types of training were performed. One was performed on images created from the eigenstates and the second one directly on the eigenstates. For each dataset $\mathcal{T} \cup \mathcal{V}_{[15,18]}$, $\mathcal{T} \cup \mathcal{V}_{[14,19]}$ and each system size, for the system size $L = 20^3$, 40^3 and 100^3, two size of images were created, $s = 100^2$, $s = 200^2$. We generated a supplementary size of images $s = 500^2$ for the system $L = 100^3$. The aim is to compare the performance of the trainings realised with images of three-dimensional eigenstates and directly of the eigenstates. To avoid confusion, we will denote the training on images as $\mathcal{T} \cup \mathcal{V}_{i,[t,u]}$, and the training on ψ and $|\psi|^2$ respectively as $\mathcal{T} \cup \mathcal{V}_{\psi,[t,u]}$ and $\mathcal{T} \cup \mathcal{V}_{|\psi|^2,[t,u]}$. Here, t and u represent the range of disorders in the dataset.

Chapter 5

Machine Learning study of the Percolation Problem

5.1 Classification of states labeled with density p

We use the density p-values as labels for the ML task of image recognition with the ResNet-based DL implementation outlined in section 3.10. After ten trainings with all 310000 images for 20 epochs, we find on average a validation loss of $\langle l_{\mathrm{c,val}} \rangle = 0.052 \pm 0.009$ (corresponding to an accuracy of $\langle a_{\mathrm{c,val}} \rangle = 99.323\% \pm 0.003$). This is comparable to the very good image classification results shown on Kaggle [25]. Figure 5.1(a) gives the resulting averaged confusion matrix. The dependence of the training and validation losses, $\langle l_{\mathrm{c,train}} \rangle$ and $\langle l_{\mathrm{c,val}} \rangle$, respectively, on the number of epochs, ϵ, is shown in figure 5.1(b). From the behaviour of the loss functions, we can see that $\langle l_{\mathrm{c,val}} \rangle \geq \langle l_{\mathrm{c,train}} \rangle$ until $\epsilon = 15$ after which both losses remain similar. This suggests that $\epsilon_{\mathrm{max}} = 20$ 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 4.1.

5.2 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 τ and predict their p values. In figure 5.2, we present the results with (a) indicating the fidelity of the predictions for each p-value and (b) showing good convergence of the losses $l_{\mathrm{r,train}}$ and $l_{\mathrm{r,val}}$. Clearly, the regression

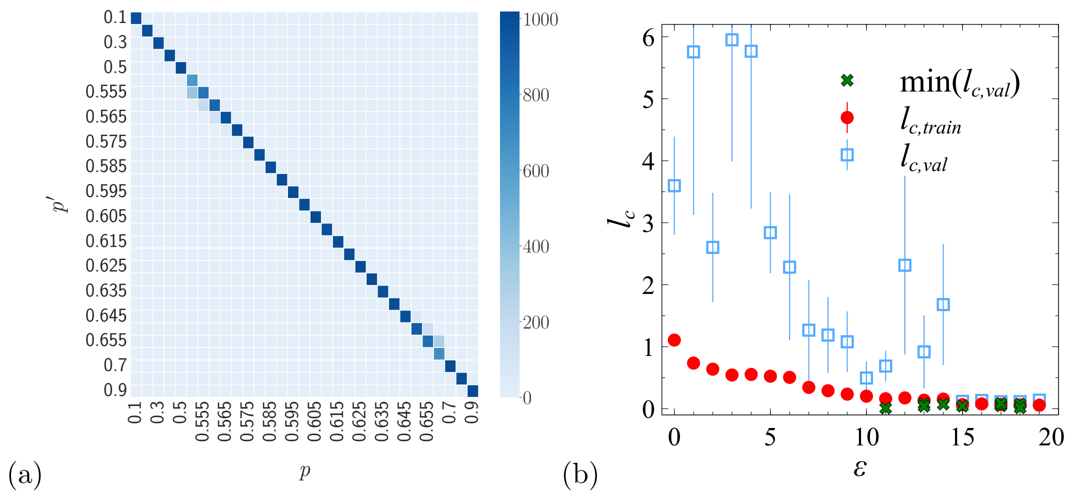


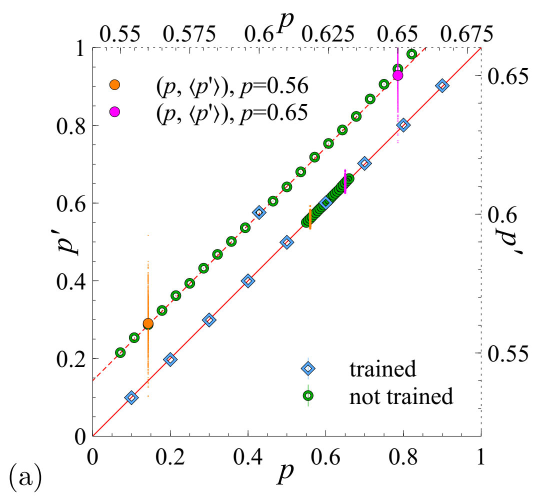
Figure 5.1: (a) Average confusion matrix for *classification* according to p. The dataset used is the test data τ and the models used for predictions are those corresponding to a minimal $l_{\mathrm{c,val}}$. True labels for p are indicated on the horizontal axis while the predicted labels are given on the vertical axis. The colour scale represents the number of samples in each matrix entry. (b) Dependence of losses $l_{\mathrm{c,train}}$ and $l_{\mathrm{c,val}}$ averaged over ten independent training seeds, on the number of epochs ϵ for *classification* according to p. The circles (red solid) denote $l_{\mathrm{c,train}}$ while the squares (blue open) show $l_{\mathrm{c,val}}$. The green crosses indicate the minimal $l_{\mathrm{c,val}}$ for each of the ten trainings.

works very well for the nine trained p-values $p = 0.1, \dots, 0.9$ as well as the untrained values $0.55, 0.555, \dots, 0.0.655, 0.66$ close to $p_c(100)$. After reaching $\epsilon = 20$, we find that $\min_\epsilon[\langle l_{\mathrm{r,train}}\rangle] = 0.0003 \pm 0.0002$ and $\min_\epsilon[\langle l_{\mathrm{r,val}}\rangle] = (6.2 \pm 1.2) \times 10^{-5}$. In other terms, the best model provides predictions on p with a variance $\delta_p = 0.008$.

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

5.3 Classification with correlation length ξ

We now turn our attention to studying image recognition when using the correlation lengths ξ, 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 ξ of the state might be different from the label assigned. Since $\langle \xi(p) \rangle$ can be uniquely identified by p, this strategy in fact should be equivalent to the previous situation and the CNN should give us similar classification results.



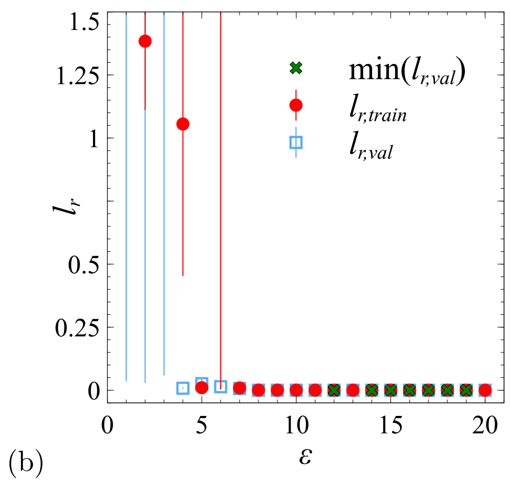
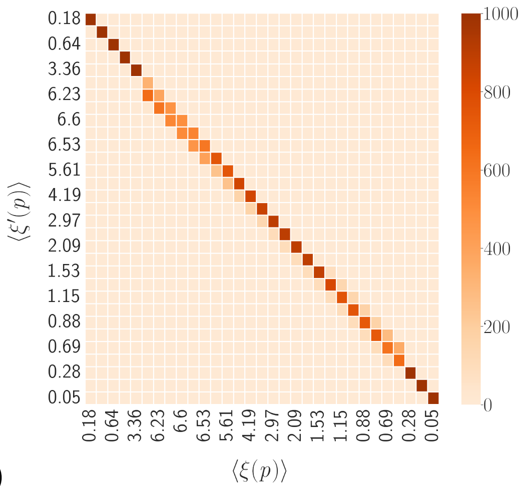


Figure 5.2: (a) Average prediction $\langle p' \rangle$ obtained for *regression* according to p. The dataset used is the test data τ and the models used for predictions are those corresponding to a minimal $l_{\mathrm{r,val}}$. The $\langle p' \rangle$ on the solid $p = p'$ line corresponds to the solid bottom and left axes, while the same $\langle p' \rangle$ data lying in the dashed $p = p'$ line is associated with the dashed top and right axes. The latter corresponds to a detailed representation of the region around $p_c(L)$. For both data representations, the blue open diamonds denote p-values that have been used during the training and the green open circles show p-values that were not trained. The vertical cloud of small orange dots shows the spread of (p, p') values for 1000 individual samples at $p = 0.56$ with average $(p, \langle p' \rangle)$ indicated by an orange circle. In the same fashion, the magenta dots give (p, p') for $p = 0.65$ with the average denoted by the magenta circle. (b) Dependence of losses $l_{\mathrm{r,train}}$ and $l_{\mathrm{r,val}}$ averaged as in Fig. 5.1 on the number of epochs ϵ for *regression* according to p. The circles (red solid) denote $l_{\mathrm{r,train}}$ while the squares (blue open) show $l_{\mathrm{r,val}}$. The green crosses show the minimal $l_{\mathrm{r,val}}$ for each of the ten trainings. Both plots were used previously in figure 3.5 as an example of regression training.

The results of such a classification are shown in figure 5.3 where similarly to figure 5.1 we present in (a) the average confusion matrix for the 31 $\langle \xi(p) \rangle$ values (cf. also figure 4.2(b) 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 ξ's which



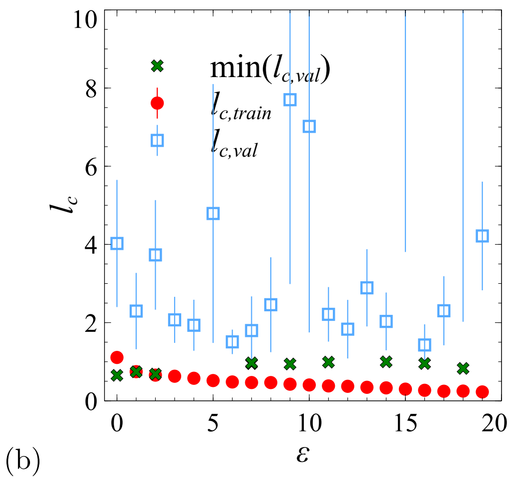


Figure 5.3: (a) Average confusion matrix for *classification* according to $\langle \xi \rangle$. The dataset used is the test data τ and the models used for predictions are those corresponding to a minimal $l_{\mathrm{c,val}}$. (b) Dependence of losses $l_{\mathrm{c,train}}$ and $l_{\mathrm{c,val}}$ on the number of epochs ϵ for classification according to $\langle \xi \rangle$. We follow the same convention as in figure 5.1 and figure 5.2.

is independent of the p's. In figure 5.4 we show the distribution Ξ of the ξ's in $\mathcal{T} \cup \mathcal{V}$. Clearly, the number of small ξ values is larger than the number of ξ values close to the maximal value of $\max[\xi] = 15.771$ (cp. figure 4.1(c)). Hence simply using each ξ as label for the corresponding ψ_i would result in a biased dataset. We therefore reorganise the $\mathcal{T} \cup \mathcal{V}$ data set. This can be done in two ways. For the first reorganisation, we create bins with a constant *number* of 10000 samples in each bin. We call this dataset Ξ_n. This results in a varying bin width. The second way to reorganise the data set is to keep the bin *width* constant while restricting the number of samples in each bin. We shall denote this reorganisation as Ξ_w. Since $\xi(p)$ is non-monotonic in p, we split the reorganisation into the case (i) $p < p_c$ with and (ii) $p > p_c$. We emphasise that the reorganised 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 Ξ_n and Ξ_w. Furthermore, there is now no longer any direct connection of the new labels to the original p densities.

In figure 5.5 and 5.6, we plot the resulting confusion matrices and losses. We see that the classification for Ξ_n and Ξ_w only results in large diagonal entries in the confusion matrices for small correlation lengths labels ξ. Overall, the classification for Ξ_w is somewhat better than for Ξ_n when away from $p_c(L)$. We attribute this to the uneven spread of ξ values for the Ξ_n.

Still, with overall 52.4% and 59.6% of states misclassified for Ξ_w and Ξ_n,

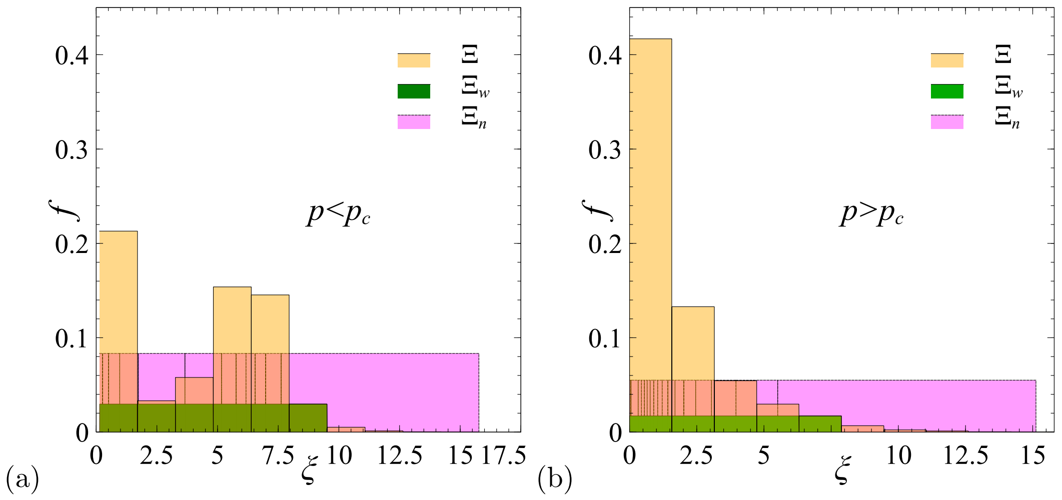


Figure 5.4: Probability distributions for correlation lengths ξ when (a) $p < p_c$ (with 12 p-values) and $p > p_c$ (18 p-values) with unbalanced Ξ and the balanced counterparts Ξ_n and Ξ_w denoted by yellow, magenta and green, respectively. In each case, the distributions are normalised relative to the total number of ξ's in each set, i.e. for (a) 120000 in Ξ and Ξ_n and $6 \times 3560 = 21360$ in Ξ_w while for (b) there are 180000 in Ξ and Ξ_n and $5 \times 3077 = 15385$ in Ξ_w.

respectively, it seems clear that classification for correlation lengths must be considered unsatisfactory.

5.4 Regression with correlation length ξ

For the regression task with ξ, we proceed analogously to section 5.2. Again, we train the CNN for the individual correlation length $\xi_i(p)$ corresponding to each $\psi_i \in \mathcal{T}$ for the nine densities $p = 0.1, \ldots, 0.9$. We then compute the predictions of $\xi_i(p)$ for all 31 densities in τ. The results are shown in figure 5.7. We find that the network architecture which previously predicted the density quite accurately is now struggling to correctly predict ξ.

A structure seems to exist in the predictions. By looking closely we notice that the network makes use of the density for its predictions: although the individual true $\xi_i(p)$ values have a large spread for the untrained p-values, the regression reassigns them all a predicted $\langle \xi' \rangle$ value similar to the average of the ξ' from the two neighbouring p values. Furthermore, by plotting the correlation length $\langle \xi' \rangle(p)$ for $p = 0.1, \ldots, 0.9$ we retrieve the plot of $\xi(p)$ as seen in figure 5.8. We have detailed the case $p = 0.56$ in Fig. 5.7(a). We find that while the spread of ξ values is rather large, reflecting the fluctuations inherent when calculating ξ close to p_c,

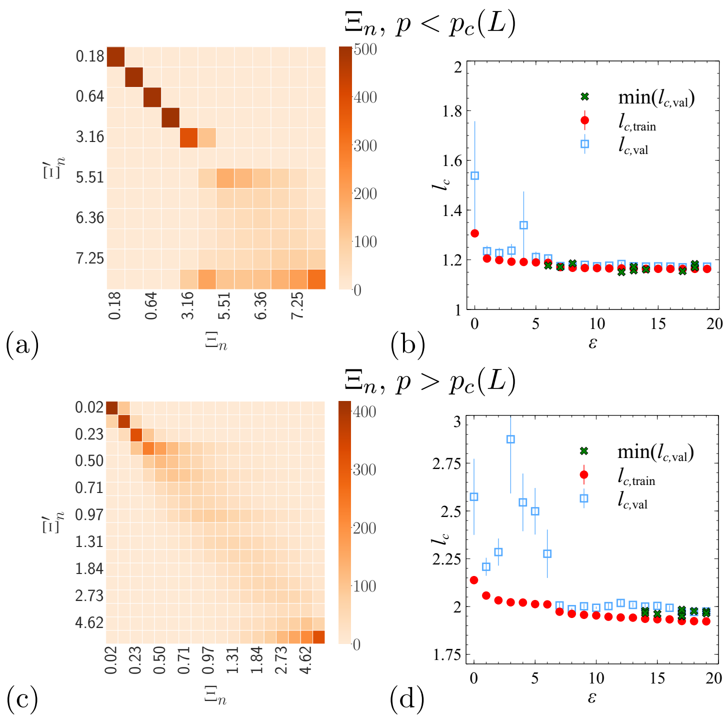


Figure 5.5:
(a+c) Confusion matrices on τ set for the *classification* when using the correlation-function-relabeled Ξ_n data sets for $p < p_c(L)$ and $p > p_c(L)$, respectively. (b+d) Dependence of losses $l_{\text{c,train}}$ and $l_{\text{c,val}}$ on the number of epochs ϵ for classification according to Ξ_n for $p < p_c(L)$ and $p > p_c(L)$, respectively. The model used for prediction in (a) was trained with 12 classes and the model (b) was trained with 18 classes. For (a) and (b) we used 10000 samples per class.

the spread of ξ' is much less, and $\langle \xi'(0.56) \rangle = 3.48$ is close to the average of the trained $\langle \xi'(0.5) \rangle$ and $\langle \xi'(0.6) \rangle$, namely 3.46. Similarly, the prediction at $p = 0.65$ is 2.14 and hence close to the average of $\langle \xi'(0.6) \rangle$ and $\langle \xi'(0.7) \rangle$, namely 1.91. We note that in obtaining these results, we have taken care to avoid unintentional feature leakage [80, 81] by removing information about p in the τ data set.

5.5 Classification with the spanning or non-spanning properties with $L = 100$

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 [48]. In the previous section, our DL approach has classified according to p or ξ values without testing whether spanning clusters actually exist. We now want to check

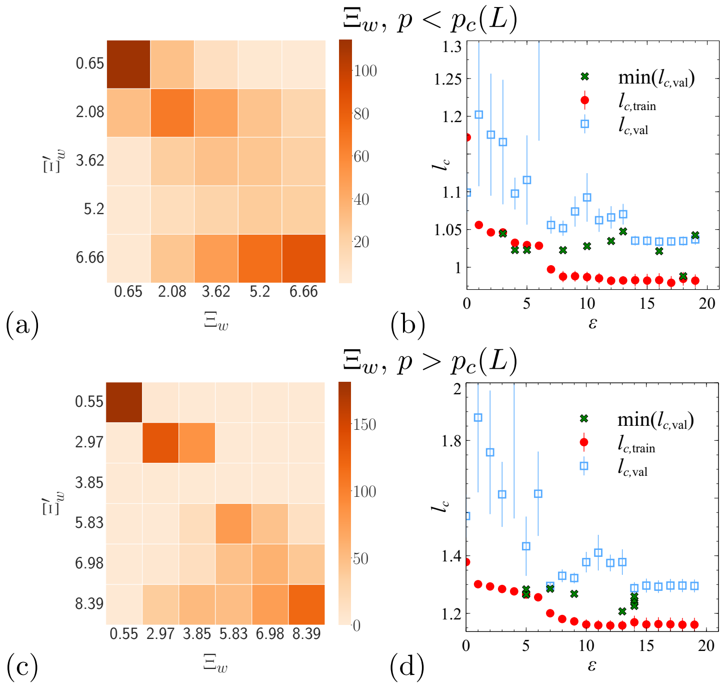
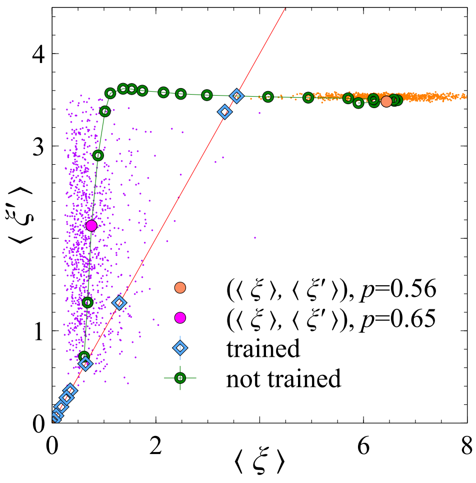


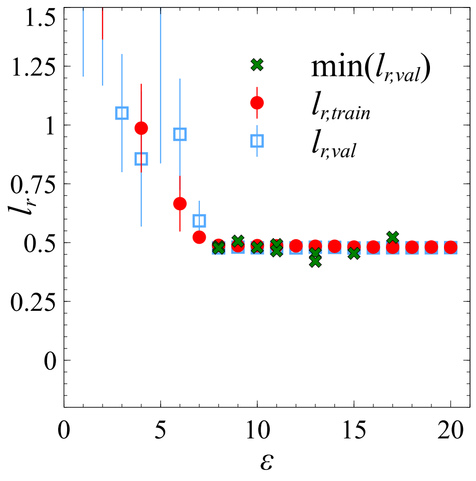
Figure 5.6:
(a+c) Confusion matrices on τ set for the *classification* when using the correlation-function-relabeled Ξ_w data sets for $p < p_c(L)$ and $p > p_c(L)$, respectively. (b+d) Dependence of losses $l_{\mathrm{c,train}}$ and $l_{\mathrm{c,val}}$ on the number of epochs ϵ for classification according to Ξ_w for $p < p_c(L)$ and $p > p_c(L)$, respectively. The model used for prediction in (a) was trained with 5 classes and 3560 states per class. For (b), 6 classes were trained with 3077 states per class.

this and label all states according to whether they are spanning or non-spanning. From figure 4.2(a), it is immediately clear that for finite-sized systems considered here, there are a non-negligible number of states which 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 $p_c(100) \sim 0.585$ as the appropriate value to distinguish the two phases.

Figure 5.9 shows the average results after $\epsilon = 20$ with a 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 τ, 11510.6 have been correctly classified as non-spanning (i.e., $N \to N'$), and 17206.9 as spanning ($S \to S'$) while only 1223.1 are wrongly labeled as non-spanning ($S \to N'$) and 1059.41 as spanning



(a)



(b)

Figure 5.7: (a) Average predictions for *regression* according to ξ. The dataset used is the test data τ and the models used for predictions are those corresponding to a minimal $l_{\mathrm{r,val}}$. The cloud of small purple dots shows the distribution of the individual samples predictions for the purple circle, in the same fashion, the orange cloud shows the distribution of predictions for the orange circle. (b) Dependence of losses $l_{\mathrm{r,train}}$ and $l_{\mathrm{r,val}}$ on the number of epochs ϵ for regression according to ξ. We follow the same convention as in figure 5.2.

$(N \to S')$ [1]. 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 τ, we can compute that there are 11127 states (92.7%) without a spanning cluster below $p_c(L)$ while 873 states (7.3%) already contain a spanning cluster. Similarly, for $p > p_c(L)$, 94.9% of states, equivalent to 17075 states, are spanning and 5.1% are not corresponding to 925 states. At $p_c(L) = 0.585$, we furthermore have 482 spanning and 518 non-spanning states. Hence in total, we expect 2280 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 recognise. Let us rephrase for clarity: it seems that the DL CNN, when trained in whether a cluster is spanning or non-spanning, completely disregards this information in its classification outputs.

[1] We note that these numbers are not integers since they are computed as averages over the 10 independent training runs as mentioned in section 3.10

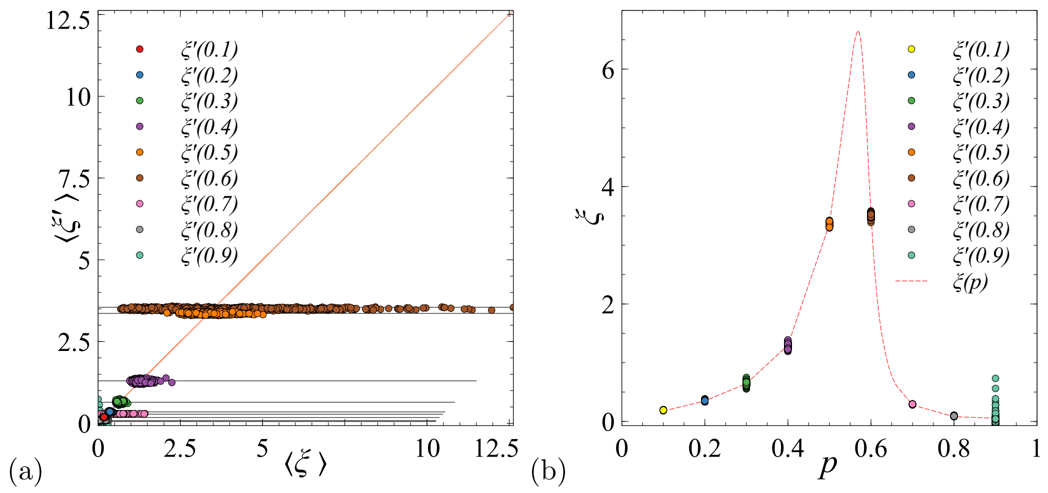


Figure 5.8: (a) Average predictions for *regression* according to ξ. For each of the trained $\xi(p)$ we give the distribution for 1000 datapoints. The horizontal line denotes $\langle\xi(p)\rangle$ for each p. (b) Plot of the prediction of the trained $\xi(p)$ according to p. We observe that the predictions allow us to retrieve the $\xi(p)$ (cf. figure 4.2(b)) from the $\mathcal{T} \cup \mathcal{V}$ dataset.

5.6 Classification with the spanning or non-spanning properties with $L = 200$

In the previous section, we presented the result of the classification of percolation lattice according to the presence of a spanning cluster. We raised the hypothesis that our network failed to identify spanning samples below the percolation threshold. To see if this trend is confirmed in larger system sizes we propose to study one classification training for spanning/non-spanning with $L = 200$. In figure 5.10 we display the result of one training after 20 epochs.

5.7 Density-resolved study of spanning/non-spanning close to $p_c(L)$

In order to understand the behaviour observed in the last section, we now reexamine the result of figure 5.9 by analysing the ML-predicted probabilities, $\Pi_{\mathrm{ML}}(p)$. In figure 5.11, we show both $\Pi_{\mathrm{ML}}(p)$ as well as $\Pi(p)$; the latter having been obtained by the HK algorithm, cf. figure 4.1(a). While the $\Pi(p)$ and $\Pi_{\mathrm{ML}}(p)$ curves — and of course also the corresponding $\kappa(p)$ and $\kappa_{\mathrm{ML}}(p)$ — appear qualitatively similar, they are nevertheless not identical and the slopes of $\Pi_{\mathrm{ML}}(p)$, $\kappa_{\mathrm{ML}}(p)$ are different. We emphasise that the slopes are important for determining the universality class

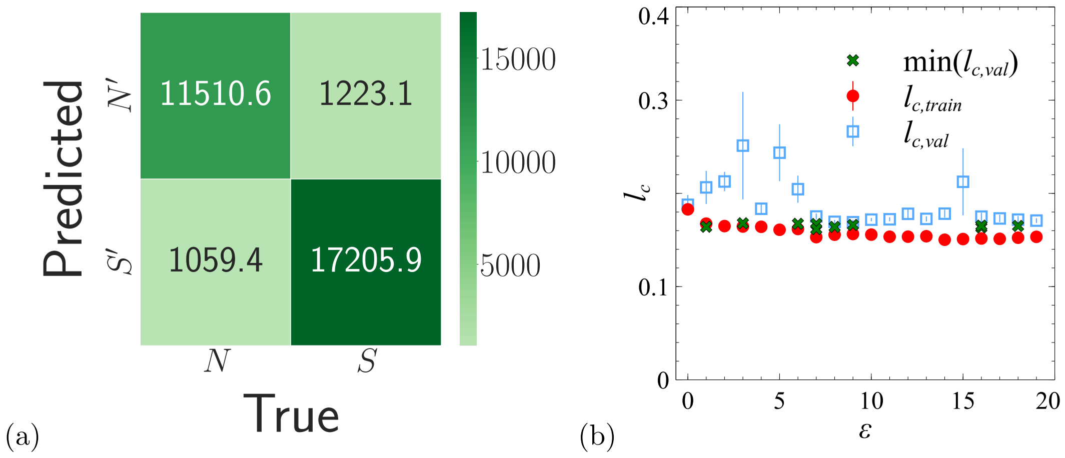
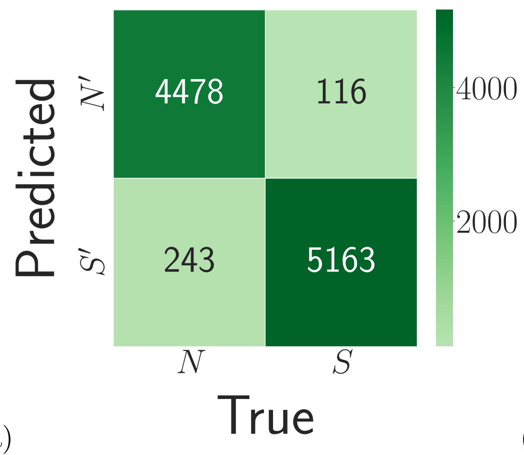


Figure 5.9: (a) Average confusion matrix for *classification* according to spanning/non-spanning. The dataset used is the test data τ and the models used for predictions are those corresponding to a minimal $l_{\text{c,val}}$. The true labels for N and S, are indicated on the horizontal axis while the predicted labels are given on the vertical axis. (b) Dependence of losses $l_{\text{c,train}}$ and $l_{\text{c,val}}$ on the number of epochs ϵ for classification according to spanning/non-spanning. Both figures were used previously in figure 3.4 as an example of classification training. Again, we follow the same convention as for figures 5.1, 5.3, 5.5 and 5.6

of a second-order phase transition via finite-size scaling [82]. 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 $\zeta_{\text{ML}}(\psi_i(p))$ for the prediction by the DL network. Then $\text{cov}(\zeta, \zeta_{\text{ML}})(p)$ measures the covariance of states being found to span by percolation *and* by ML for given p. In figure 5.11(b) we show the normalised result, i.e., the Pearson coefficient $r_{\zeta,\zeta_{\text{ML}}}(p) = \text{cov}(\zeta, \zeta_{\text{ML}})(p)/[\sigma_\zeta(p)\sigma_{\zeta_{\text{ML}}}(p)]$, where σ_ζ and $\sigma_{\zeta_{\text{ML}}}$ are the standard deviations of the percolation results and the ML predictions. We see that in the transition region, $r_{\zeta,\zeta_{\text{ML}}} \lesssim 0.12$ which is very far from the maximally possible value 1. This suggests that while the ML predictions are not simply random, they are also not very much better than random. Furthermore, we explain the null value $r_{\zeta,\zeta_{\text{ML}}}(p)$ for $p < 0.57$ and $p > 0.6$ by the trivial character of the phases in these two regions. Indeed, for $p < 0.57$, we are far away from the transition region and most of the samples are labeled and predicted as non-spanning. The same could be said for $p > 0.6$, most of the samples are labeled and predicted as spanning. As such, $\text{cov}(\zeta, \zeta_{\text{ML}})(p) \sim 0$ in both of these regions and $r_{\zeta,\zeta_{\text{ML}}}(p)$ is only non-negative in the region where true decisions happen, i.e the transition region.



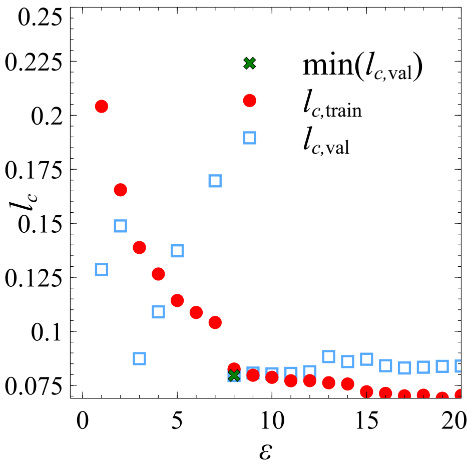


Figure 5.10: (a) Confusion matrix for *classification* according to spanning/non-spanning and $L = 200$. The validation set $\mathcal{V}$ was used for predictions and the models are those corresponding to a minimal $l_{\text{c,val}}$. The true labels for N and S, are indicated on the horizontal axis while the predicted labels are given on the vertical axis. (b) Dependence of losses $l_{\text{c,train}}$ and $l_{\text{c,val}}$ on the number of epochs ϵ for classification according to spanning/non-spanning.

Let us now study the classification into spanning/non-spanning states in detail for each p. Figure 5.12 and Table 5.1 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 $(= 48 + 87 + 126.5 + 223.5)$ of 492 $(= 48 + 87 + 127 + 230)$ samples, which are already spanning, have been misclassified as non-spanning. Similarly, for $p = 0.59, 0.595, 0.6, 0.605 > p_c(L)$, 745.9 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 figure 5.9, we now find that the seemingly few misclassified states of figure 5.9 are indeed precisely those which represent the correct physics. Saying it differently, the ML process seems to have led to a DL network that 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.

5.8 Testing the accuracy of the DL network

The difficulties that the trained DL network has with recognising whether a state contains a percolating cluster or not can be made more explicit. In section 4.1.4, we generated three test sets for this purpose. Namely, percolating states even for

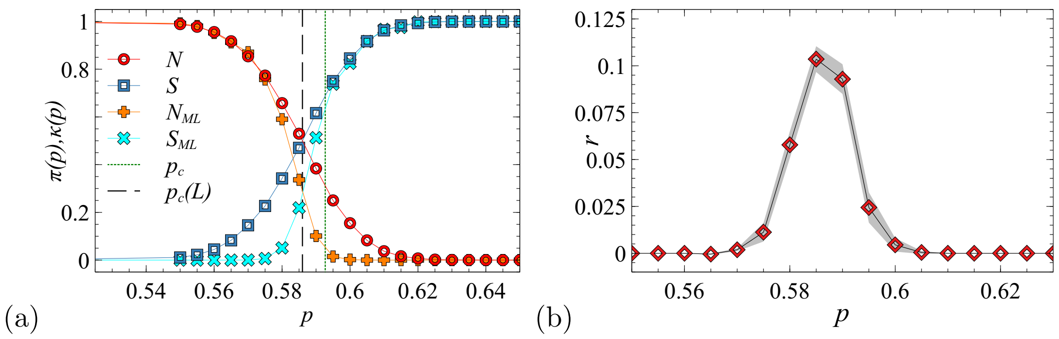


Figure 5.11: (a) The blue curve (red curve) shows the probability of having a spanning (non-spanning) sample in the training dataset. The cyan (orange) curve gives us the prediction of the probability of having a spanning (non-spanning) sample, according to the trained network. (b) Dependence of the Pearson correlation coefficient r on the density p for *classification* according to spanning/non-spanning. The confidence interval is indicated in grey. In both (a) and (b), The lines connecting the symbols are only a guide to the eye.

$p < p_c(L)$ by adding (i) a straight line, τ_{sl}, and (ii) a random walk, τ_{rw}, of connecting sites as well as for $p > p_c(L)$ (iii) the firebreak states, τ_{fb}, of percolation-prohibiting random unoccupied sites. We now use these sets and feed them independently as test sets to the DL network. Figure 5.13 shows the three confusion matrices obtained when classifying for spanning vs. non-spanning. In figure 5.13(a+b), we see that the network completely misclassifies the spanning datasets τ_{sl} and τ_{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 figure 5.13(c), we see that this time the network cannot correctly identify the non-spanning samples in τ_{fb}. Again, the two samples correctly identified are the ones without the firebreak.

5.9 Training of the Spanning/Non-spanning at $p = 0.585$

In the previous sections, we found out that CNN methods were struggling to understand the concept of connectivity in the lattice and were instead relying on the density as a proxy. However, one could think about improving the quality of the training by removing the involuntary intake of the density. For this purpose, we performed a final training on a dataset composed of 30000 lattices labelled as spanning/non-spanning at $p = 0.585$, close to the percolation threshold. Following the same protocol as the previous sections, we performed ten independent runs with

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

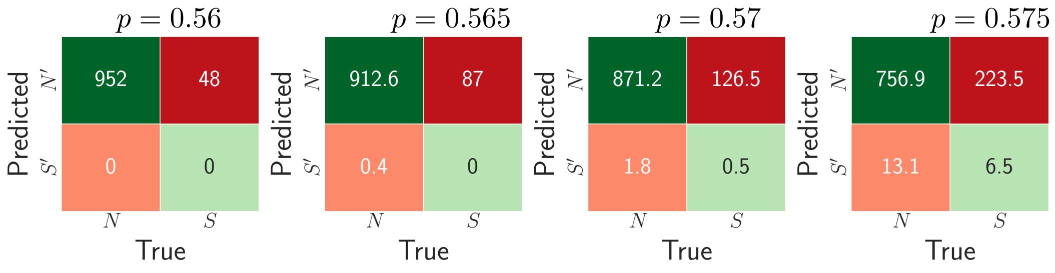
Table 5.1: Predictions of the trained network on the test data set τ with $n = 1000$ for $p \in [0.56, 0.605]$. N and S denote, respectively, the number of non-spanning and the number of spanning samples in τ. The four following columns $S \to S'$, $S \to N'$, $N \to S'$, and $N \to N'$ give the averaged results of 10 independent prediction runs.

a pre-trained ResNet18 and tested the performance on a test set $\tau_{0.585}$ composed of 3000 samples. In figure 5.14(a) we give the confusion matrix obtained after ten independent trainings. We clearly observe that the network fails to systematically identify the two phases and produces 47% of misclassification. Additionally, we notice that the network struggles more to identify the spanning samples with 78.7% of the misclassification being spanning samples misclassified as non-spanning. Looking at 5.14(b) we obtain a $\min_\epsilon[\langle l_{c,val}\rangle] = 0.6864285 \pm 0.000686$ corresponding to a maximal validation accuracy $\max_\epsilon[\langle l_{c,val}\rangle] = 54.3\% \pm 0.004031$. These results confirm the shortcomings of classical CNN techniques in understanding the importance of connectivity in our percolation samples. Despite removing the input of density the network was not able to understand on its own the global property in our dataset. We emphasise that this training was also performed for several different learning rates and another optimizer, Adam. In all such combinations, we were not able to obtain a network able to predict the phases with an accuracy higher than 55%.

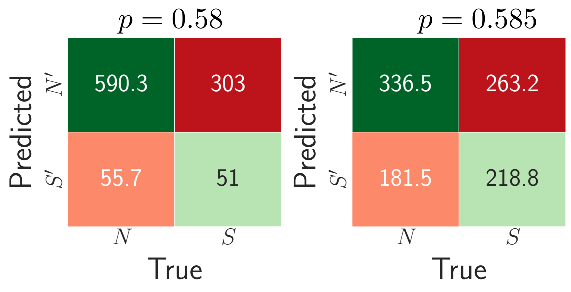
5.10 VAE for percolation

Until this point, our study of the percolation model was performed through the lens of supervised learning. However, unsupervised learning could give us some insight into special structures in the dataset which would give an indication of the phase. As such we wish to test how unsupervised learning deals with a model such as percolation. This study was motivated by previous results, claiming the ability of unsupervised learning and in particular VAE to reconstruct percolation states [76]. In this section, we aim at reproducing and analyse the performance of the model described in the article. We construct a Convolution VAE with two convolution layers in the encoder, the first layer comprised of 32 filters and the second with 64 filters. At the output of each convolutional layer, we apply a ReLU activation. The decoder follows the same parameters but uses transposed convolutional layers. For the hidden layer, we choose a latent space dimension $z = 400$. The training process follows the procedure described in section 3.12. After training we obtain percolation states highly similar to the one fed in input. Subtracting the input from the output allows us to notice the high similarity of the two samples. Observing this leads us to hypothesise that the output presented might be a close copy of the input. The copying could be explained by the large dimension of the latent space, we recall that our dataset is composed of percolation states of size $L = 28^2$. Therefore a latent space $z = 400$ cannot act as a bottleneck for our dataset, too much information passes through the hidden block. In figure 5.15(a) we present samples generated by our VAE after training. While the new samples on the bottom, are not fidele copies of the input on top, we notice the presence of common cluster architectures in the input and the output.

(a) $p < p_c$



(b) $p \sim p_c$



(c) $p > p_c$

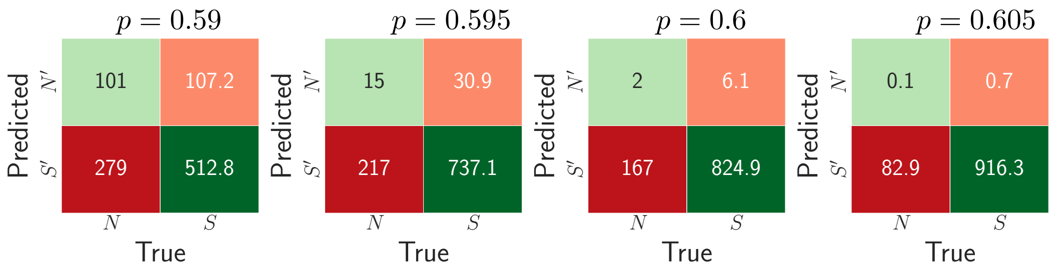


Figure 5.12: Confusion matrices showing the predictions of the trained network figure 5.9 in a region $p = [0.56, 0.605]$ comprising p_c, with (a) for predictions made before the percolation threshold, (b) in the threshold region and (c) after the percolation threshold. Each confusion matrix is an average of the predictions made by the 10 trained models shown in figure 5.9(b).

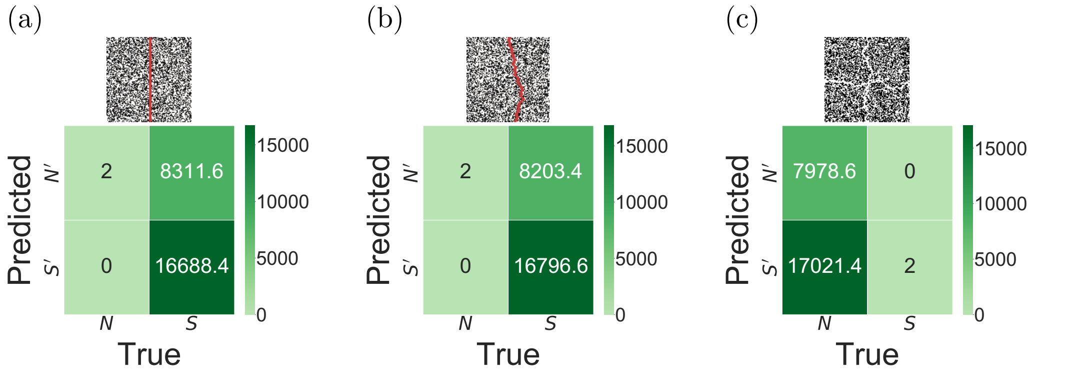
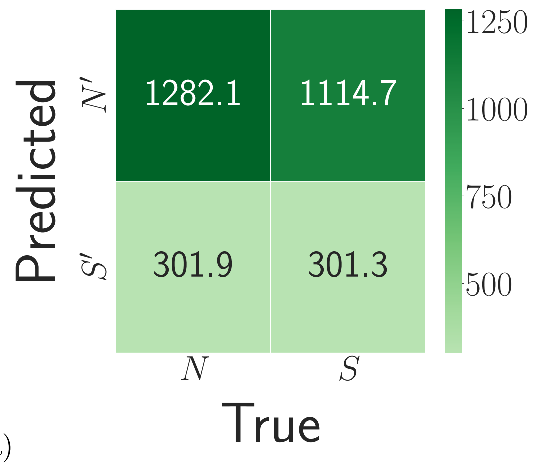
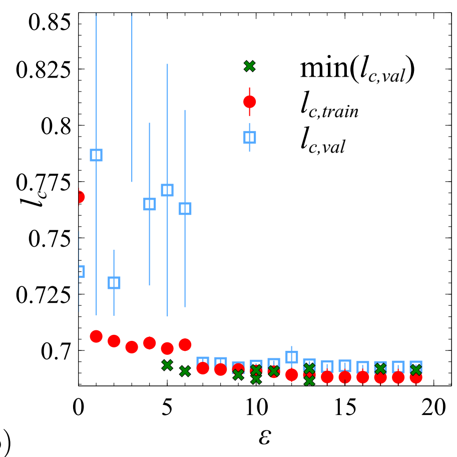


Figure 5.13: Sample states for the three special test sets (a) τ_{sl} with added straight spanning lines, (b) τ_{rw} with spanning random walks and (c) τ_{fb} with the firebreaks. In each case, the bottom plots gives the confusions matrices obtained from the DL model previously trained in a spanning vs. non-spanning classification. In all cases, the density is strictly $p = 0.5$. The states shown above each confusion matrix are taken from figure 4.5.



(a)



(b)

Figure 5.14: (a) Average confusion matrix for *classification* according to spanning/non-spanning at $p = 0.585$. The dataset used is the test data $\tau_{0.585}$ and the models used for predictions are those corresponding to a minimal $l_{\text{c,val}}$. 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 $l_{\text{c,train}}$ and $l_{\text{c,val}}$ on the number of epochs ϵ for classification according to spanning/non-spanning. Again, we follow the same convention as for figures 5.1, 5.3 and 5.5

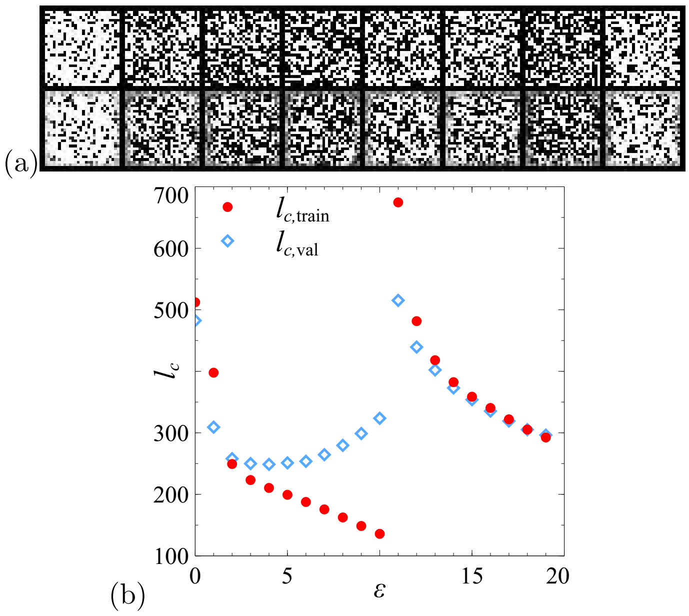


Figure 5.15: (a) Output of the VAE training, on the top row we see the input fed by the network, on the second row, we see the reproduction made by the network after training. (b) Dependence of losses $l_{\mathrm{v,train}}$ and $l_{\mathrm{v,val}}$ on the number of epochs ϵ for one VAE training. The circles (red solid) denote $l_{\mathrm{v,train}}$ while the squares (blue open) show $l_{\mathrm{v,val}}$. Figure (a) was used previously in figure 3.9 as an example of training of a VAE.

Chapter 6

ML study of the Anderson model of localisation

In this section, we will study our second model. Contrary to the study of the percolation model where we worked directly with the arrays, here, we will present the result of our study on three different types of input. We will begin by realising several classifications on image representation of $|\psi|^2$ states for phases and disorder classifications. Following this, we will reproduce similar training on $|\psi|^2$ and ψ states. The aim of the study is to evaluate how different types of input can influence the quality of our training. Initial scoping work was performed by a student of our group, Quangminh Bui-Le, as part of his master's project.

6.1 Image classification of phases from $|\psi|^2$ at $E = 0$

In the first part of this study, we intend to discover if standard ML classification methods applied to images of $|\psi|^2$ states can accurately predict the two phases of the three-dimensional Anderson model. The training was performed on the dataset $\mathcal{T} \cup \mathcal{V}_{i,[15,18]}$, with images of $|\psi|^2$ at two disorder values, $W = 15 \ll W_c$ for the extended phase and $W = 18 \gg W_c$ for the localised one. We choose to use 5000 samples per class. In order to observe how different parameters such as the size of the system or the size of the images could influence the performance of our network, several trainings were implemented. The phase classification was done for the sizes $L = 20^3$, $L = 40^3$, $L = 100^3$, and a fixed image size of $s = 100^2$. For each of these systems, the dataset $\mathcal{T} \cup \mathcal{V}_{i,[15,18]}$ was split according to a 90%/10% training/validation split and we train the ResNet18 DL architecture previously described in section 3.9 for 50 epochs with 10 different seeds. After completing a full training

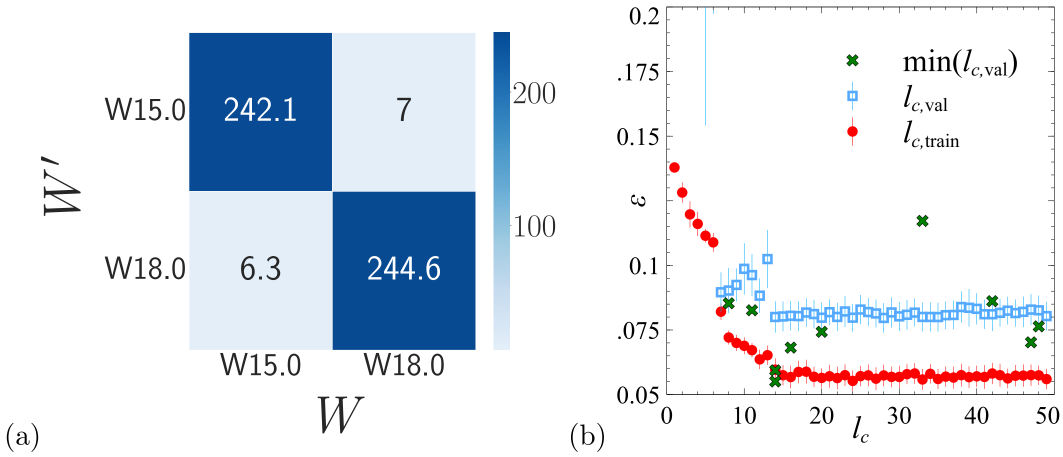


Figure 6.1: (a) Average confusion matrix for image *classification* according to extended/localised on system size $L = 20^3$ and $s = 100^2$. The dataset used is the test data $\tau_{i,[15,18]}$ and the predictions are made with models corresponding to a minimal $l_{\mathrm{c,val}}$. The true labels for $W = 15.0$ and $W = 18.0$, are indicated on the horizontal axis while the predicted labels are given on the vertical axis. In (b) dependence of losses $l_{\mathrm{c,train}}$ and $l_{\mathrm{c,val}}$ on the number of epochs ϵ for this training.

cycle, the generalisation capacity of the ten models was tested on the dataset $\tau_{i,[15,18]}$ and summarised in average confusion matrices.

The first training was performed for the system size $L = 20^3$. After a full training cycle we achieved a minimal validation loss $\min_\epsilon[\langle l_{c,\mathrm{val}}\rangle] = 0.06 \pm 0.01$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle l_{c,\mathrm{val}}\rangle] = 98.3\% \pm 0.22$). Looking at the confusion matrix in figure 6.1(a), we notice the heavy diagonal character of the matrix with a misclassification rate of 2.7%. This training shows us that even for a relatively small system size such as $L = 20^3$, ML image recognition tools are able to identify reasonably well the localised and extended phases of the Anderson MIT model.

We are now interested in knowing if an increase in the size of the system, while keeping the size of the image constant, could contribute to an increase in the performance of our model. We reproduce the same training process with now $L = 40^3$ and $s = 100^2$. Following the training we obtain $\min_\epsilon[\langle l_{c,\mathrm{val}}\rangle] = (5.2 \pm 3.4) \times 10^{-4}$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\mathrm{val}}\rangle] = 100\% \pm 0.02$). In figure 6.2(a) we display the average confusion matrix of the predictions on the dataset $\tau_{i,[15,18]}$.

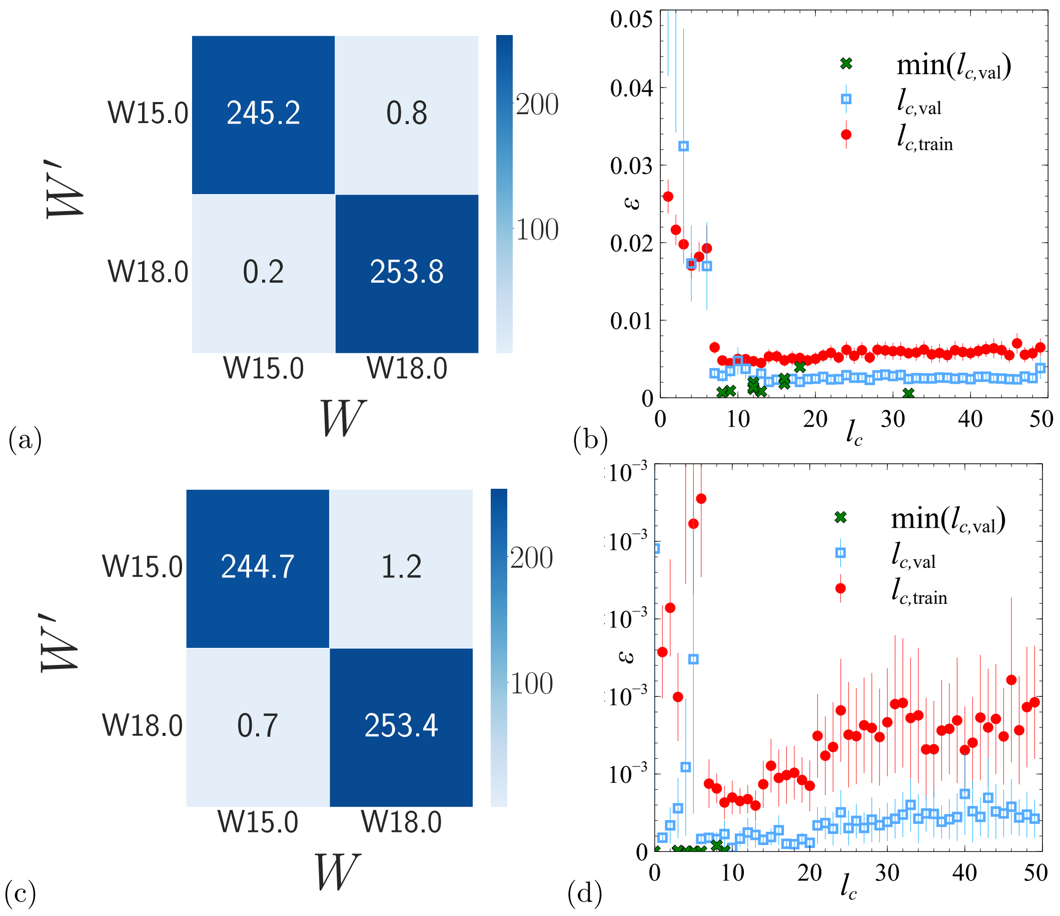


Figure 6.2: (a+c) Average confusion matrices for image *classification* according to extended/localised on system size $L = 40^3$,$s = 100^2$ and $L = 100^3$, $s = 100^2$ respectively. The dataset used is the test data $\tau_{i,[15,18]}$ and the models used for predictions are those corresponding to a minimal $l_{\mathrm{c,val}}$. The true labels for $W = 15.0$ and $W = 18.0$, are indicated on the horizontal axis while the predicted labels are given on the vertical axis. In (b+d) dependence of losses $l_{\mathrm{c,train}}$ and $l_{\mathrm{c,val}}$ on the number of epochs ϵ for classification according to extended/localised on $L = 20^3$, $L = 40^3$ and $s = 100^2$, respectively.

Similarly to 6.1(a) we obtain a highly diagonal matrix. Concerning the misclassification, only 0.2% of the samples are incorrectly predicted which is a drop of more than 10% compared to the misclassification previously obtained for $L = 20^3$. This suggests that an increase in L contributes to increasing the classification performance of the network. The last system that we study is $L = 100^3$, again we perform ten independent iterations and obtain the confusion matrix shown in figure 6.2(c). We rapidly notice that this matrix is fully diagonal, implying a perfect prediction on the test set. This result is confirmed by the minimal validation loss $\min_\epsilon[\langle l_{c,\mathrm{val}}\rangle] = (1 \pm 8) \times 10^{-6}$ which here corresponds to a maximal validation accuracy $\max_\epsilon[\langle l_{c,\mathrm{val}}\rangle] = 100\% \pm 0.0$. The misclassification rate is 0.4%, which is slightly above the one obtained for $L = 40^3$ while remaining a good training. These three trainings clearly confirm the idea that while conserving the same image sizes, an enhancement of the prediction of our model can be achieved through an increase in the size of the system L.

6.2 Size dependent image classification of phases from $|\psi|^2$ at $E = 0$

Now that we tested the impact of the size of the system on the performance of our network, one might like to know how the size of the image input could influence the performance. To do so, we train the network for the system of size $L = 100^3$ at two additional image sizes, $s = 200^2$ and $s = 500^2$. We begin by training on the $L = 100^3$ with image size $s = 200^2$. After 50 epochs we obtain $\min_\epsilon[\langle l_{c,\mathrm{val}}\rangle] = (2 \pm 3) \times 10^{-6}$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\mathrm{val}}\rangle] = 100\% \pm 0.0$) which is comparable to the results obtained for the $L = 100^3$ with image size $s = 100^2$. As we can see from the averaged confusion matrix in figure 6.3(a), 0.14% of the test samples are misclassified. This is more than the misclassification obtained with image size $s = 100^2$ but it is not significant. Finally, we perform the training with images of size $s = 500^2$. The training produce a minimal validation loss $\min_\epsilon[\langle l_{c,\mathrm{val}}\rangle] = (4 \pm 2) \times 10^{-7}$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\mathrm{val}}\rangle] = 100\% \pm 0.0$). Again, similarly to the previous training with $L = 100^3$ and image size $s = 200^2$, we observe in figure 6.3(c) that the performances of the network do not increase compared to the performance of $L = 100^3$ with image size $s = 100^2$. Furthermore, by looking at the confusion matrix we notice that the percentage of misclassification is slightly higher than the previous one at 0.16%. However, it is important to note that training a system with $s = 500^2$ induces a higher training time. Therefore, the training of $L = 100^3$ and $s = 100^2$ appears as the better-suited option, which provides good performance while keeping a reasonable training time.

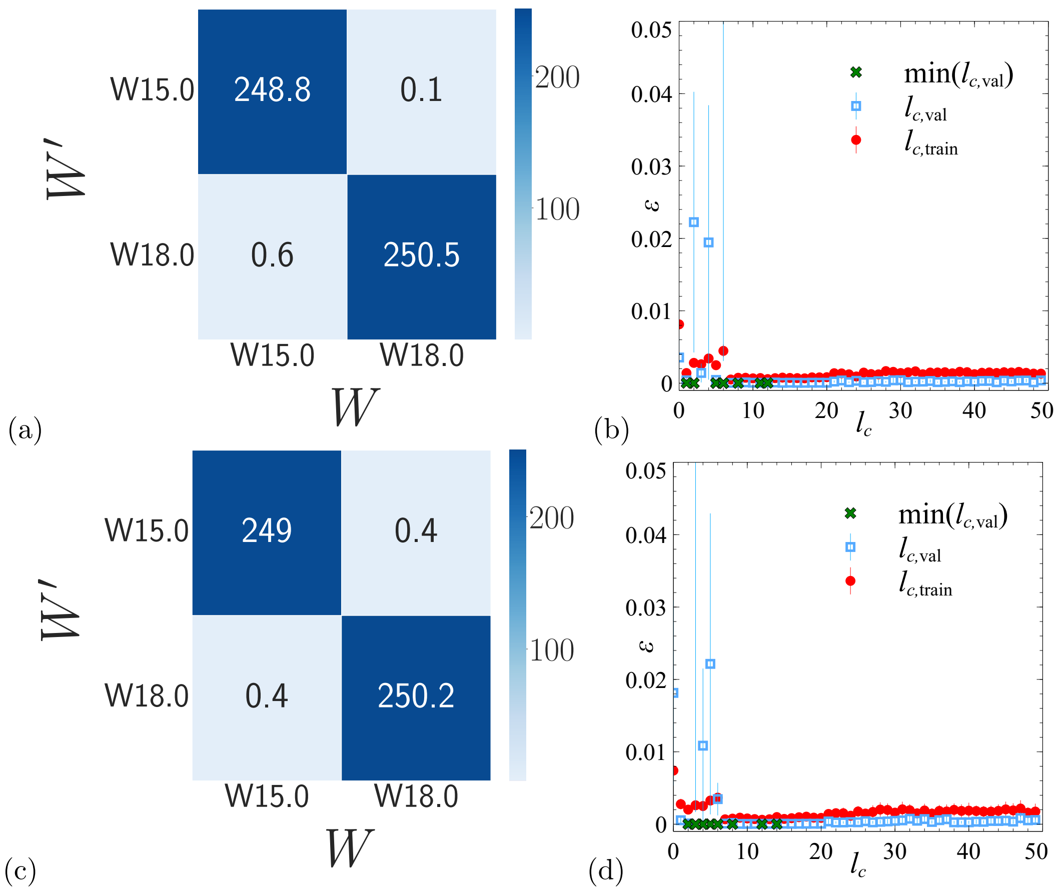


Figure 6.3: In (a+c) confusion matrices for images*classification* according to extended/localised for $L = 100^3$, $s = 200^2$ and $L = 100^3$, $s = 500^2$, respectively. The dataset used is the test data $\tau_{i,[15,18]}]$ and the models are the ones corresponding to minimal $l_{\mathrm{c,val}}$ for $L = 100^3$ and respectively image size $s = 200^2$ and $s = 500^2$. The true labels for $W = 15.0$ and $W = 18.0$, are indicated on the horizontal axis while the predicted labels are given on the vertical axis. In (b+d) dependence of losses $l_{\mathrm{c,train}}$ and $l_{\mathrm{c,val}}$ on the number of epochs ϵ for the associated training.

6.3 Image classification of disorders from $|\psi|^2$

Now that we have established the capacity of CNN methods to identify phases of the Anderson model from images of $|\psi|^2$, we would like to determine if these same CNN methods are able to identify $|\psi|^2$ at different (more than two) disorder values. For that purpose we train the ResNet18 on the dataset $\mathcal{T} \cup \mathcal{V}_{i,[15,18]}$ with images of $|\psi|^2$ at 17 disorder values $W = 15.0, 15.25, \ldots, 17.25, 17.5, 17.75, 18$ and $s = 100^2$. As for all the previous trainings, each training process is repeated ten times with

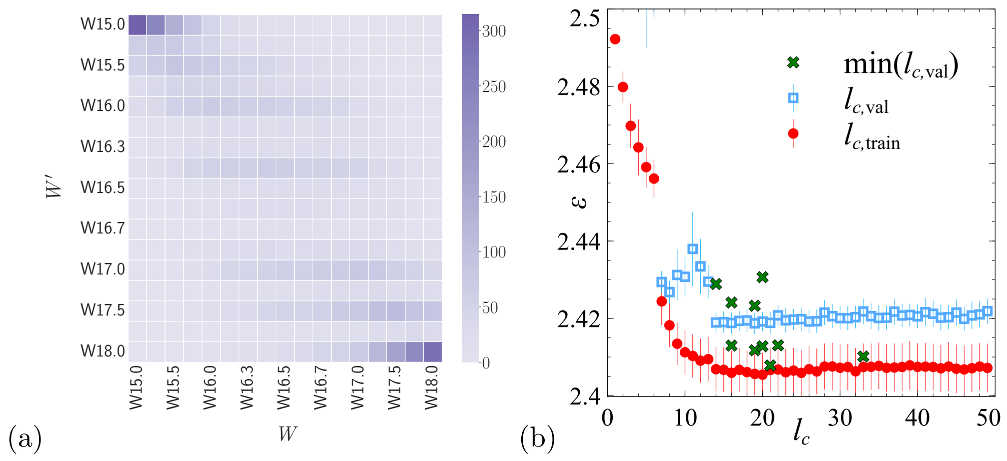


Figure 6.4: In (a) average confusion matrix for image *classification* according to 17 disorders on $L = 20^3$ and $s = 100^2$. The dataset used is the test data $\tau_{i,[15,18]}$ and the models used for predictions are those corresponding to a minimal $l_{\mathrm{c,val}}$. The true labels for $W = 15.0, 15.25, \ldots, 17.5, 18$ are indicated on the horizontal axis while the predicted labels are given on the vertical axis. In (b) dependence of losses $l_{\mathrm{c,train}}$ and $l_{\mathrm{c,val}}$ on the number of epochs ϵ for the associated training.

ten independent seeds. Three system sizes were trained, $L = 20^3$, $L = 40^3$ and $L = 100^3$.

The first system that we train is $L = 20^3$. After 50 epochs we obtain a minimal validation loss $\min_\epsilon[\langle l_{c,\mathrm{val}}\rangle] = 2.408 \pm 0.003$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle l_{c,\mathrm{val}}\rangle] = 15.9\% \pm 0.13$) which is relatively high compared to loss values obtained in the previous section. As we can see from the averaged confusion matrix in fig 6.4 (a), we can guess the hint of the beginning of a diagonal matrix. Nevertheless, correct predictions remain rare and seem to concern disorder values at the edge of the disorder spectrum such as $W = 15$ and $W = 18$. Following the logic of section 6.1 we hope to increase the number of correct predictions of the network by training for a larger system size.

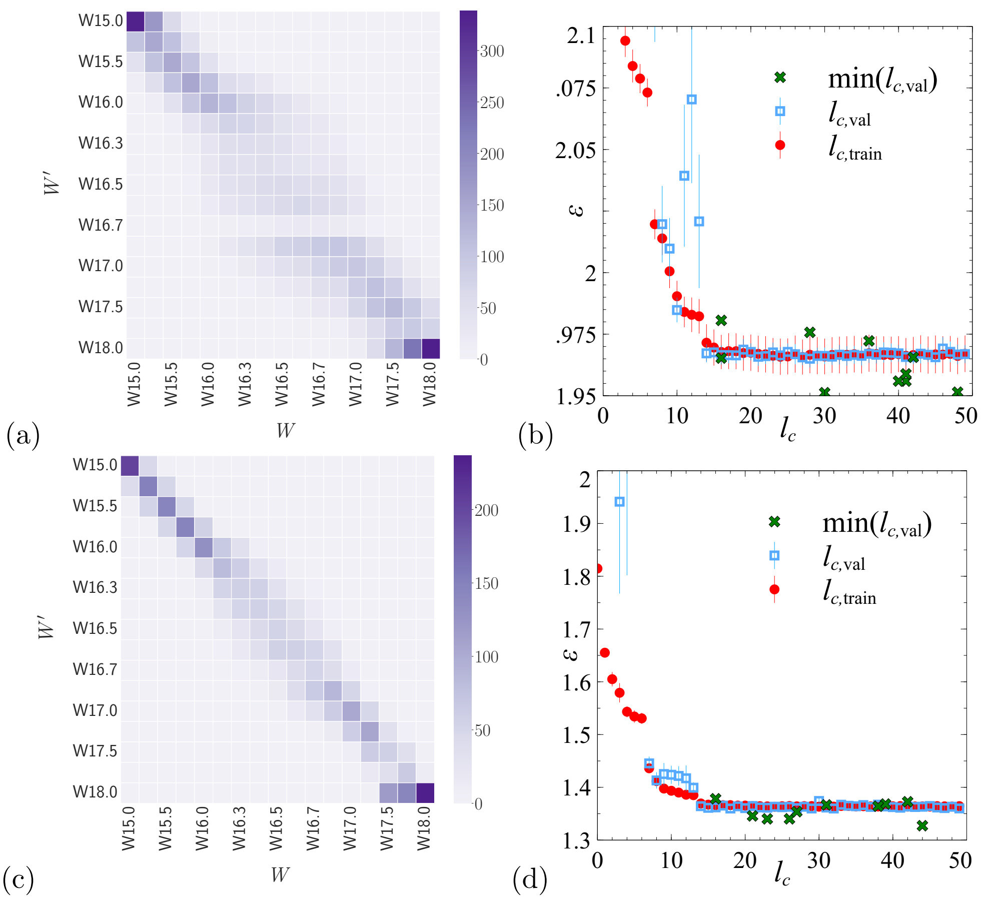


Figure 6.5: In (a+c) average confusion matrices for image *classification* according to 17 disorders for $L = 40^3$, $s = 100^2$ and $L = 100^3$ $s = 100^2$, respectively. The dataset used is the test data $\tau_{i,[15,18]}$ and the models used for predictions are those corresponding to a minimal $l_{\mathrm{c,val}}$. The true labels for $W = 15.0, 15.25, \ldots, 17.5, 18$ are indicated on the horizontal axis while the predicted labels are given on the vertical axis. In (b+d) dependence of losses $l_{\mathrm{c,train}}$ and $l_{\mathrm{c,val}}$ on the number of epochs ϵ for classification according to 17 disorders for the corresponding trainings.

The second training is conducted on $L = 40^3$. Once the training completed we obtain a minimal validation accuracy $\min_\epsilon[\langle l_{c,\mathrm{val}}\rangle] = 1.951 \pm 0.004$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\mathrm{val}}\rangle] = 25.7\% \pm 0.19$). As expected, we observe an increase in the accuracy of the network, as shown by the confusion matrix in figure 6.5(a). We perform a final training for $L = 100^3$ and $s = 100^2$. The minimal accuracy obtained is $\min_\epsilon[\langle l_{c,\mathrm{val}}\rangle] = 1.327 \pm 0.006$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\mathrm{val}}\rangle] = 43.9\% \pm 0.22$), which is an improvement compared to $\max_\epsilon[\langle a_{c,\mathrm{val}}\rangle] = 25.7\% \pm 0.19$) obtained for the training with $L = 40^3$.

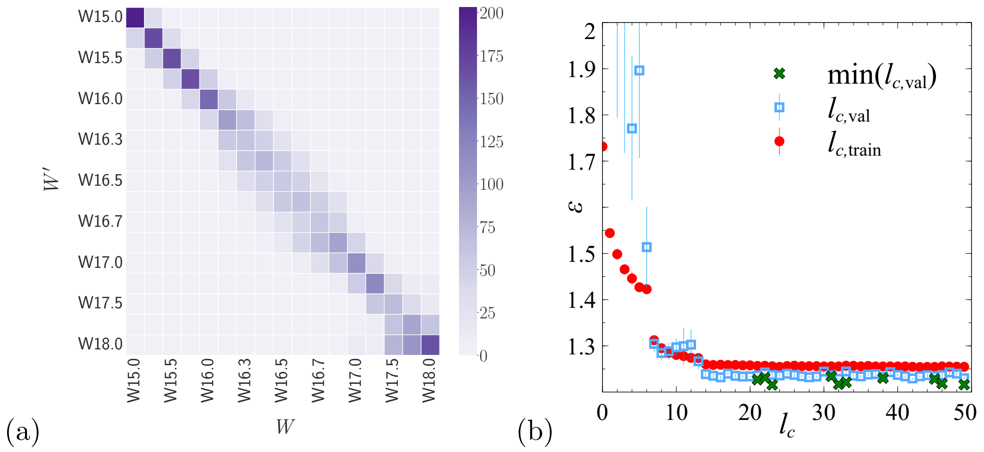


Figure 6.6: In (a) average confusion matrix for image *classification* according to 17 disorders for $L = 100^3$, and $s = 200^2$. The dataset used is the test data τ and the models used for predictions are those corresponding to a minimal. The true labels for $W = 15.0, 15.25, \dots, 17.5, 18$ are indicated on the horizontal axis while the predicted labels are given on the vertical axis. In (b) dependence of losses $l_{\text{c,train}}$ and $l_{\text{c,val}}$ on the number of epochs ϵ for the associated training.

By observing the confusion matrix displayed in figure 6.5(c) we notice that the predictions appear to follow closely the diagonal. Thus, while the accuracy of this trained network remains below 50%, we conclude here that the network performs reasonably well. Indeed, in our setting classes are not totally independent from one another. Therefore, a network identifying disorder values close to the true label is an improvement.

6.4 Size dependent image classification of disorders from $|\psi|^2$ at $E = 0$

The last step is to attempt to enhance the performance of the network by increasing the resolution of the images. We trained the system $L = 100$ at size $s = 200^2$. After training, we obtain the average confusion matrix shown in figure 6.6(a). This confusion matrix is highly similar to the one obtained in figure 6.5(c), the increase in the size of the input did not seem to affect the prediction significantly. This is confirmed by the metrics, we obtain a $\min_\epsilon[\langle l_{c,\text{val}}\rangle] = 1.22 \pm 0.003$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\text{val}}\rangle] = 47.0\% \pm 0.22$).

Overall, we conclude that working with a large system size such as $L = 100^3$ increase drastically the quality of the predictions. In figure 6.7(a) we present a comparison of the losses according to the system size. We notice that $L = 100^3$ has

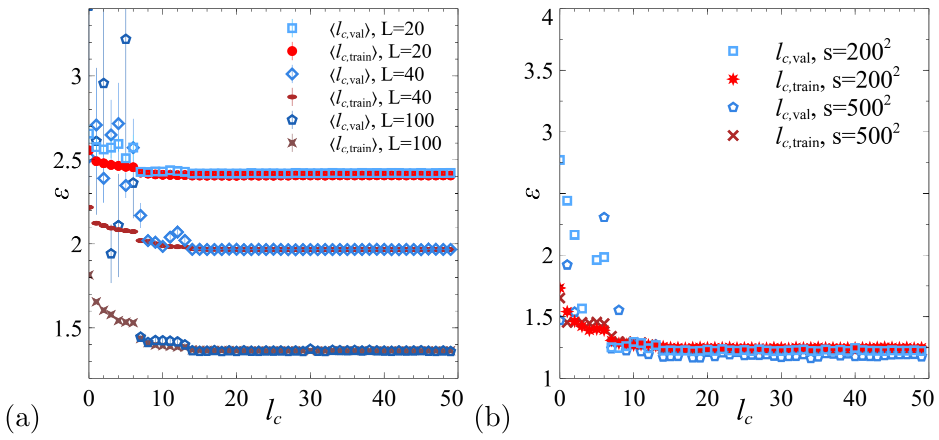


Figure 6.7: In (a) dependence of losses $l_{\text{c,train}}$ and $l_{\text{c,val}}$ on the number of epochs ϵ for *classification* according to 17 disorder values for $L = 20^3$, 40^3 and 100^3 and $s = 100^2$. In (b) dependence of losses $l_{\text{c,train}}$ and $l_{\text{c,val}}$ on the number of epochs ϵ for classification according to 17 disorder values for $L = 100^3$ for image size $s = 100^2$ and $s = 200^2$.

much lower metrics than $L = 20^3$ and $L = 40^3$. Nonetheless, looking at the figure 6.7(b), we do not observe important differences between the losses of the training $s = 200^2$ and $s = 500^3$. Therefore, good trainings could be achieved with a large system size and a reasonable image size such as $s = 100^2$.

6.5 State classification of phases from $|\psi|^2$ with a six layers CNN

In the two previous sections, we showed that image classification of $|\psi|^2$ was successful in predicting the localised and extended phase of the Anderson. However, we saw the shortcomings of image classification when trying to predict several disorder values. In this section, we explore the idea of directly using the $|\psi|^2$ states to improve the accuracy of the CNN method. Again, we train the three sizes of systems used in the previous sections, $L = 20^3$, 40^3, and 100^3 to identify the two phases. To compare our results to previously published work [45], we use the dataset $\mathcal{T} \cup \mathcal{V}_{[|\psi|^2,[14,19]}$ and choose 5000 $|\psi|^2$-values with $W \in [14, 16]$ labeled as *extended* and 5000 $|\psi|^2$-values with $W \in [17, 19]$ labeled as *localised*. As a first step, we choose to use the same network architecture as the one described in a previous study of the same model, i.e., three blocks of two convolutional layers separated by max-pooling layers followed by two fully connected layers [45]. For the sake of clarity, we choose to call this network the Ohtsuki network.

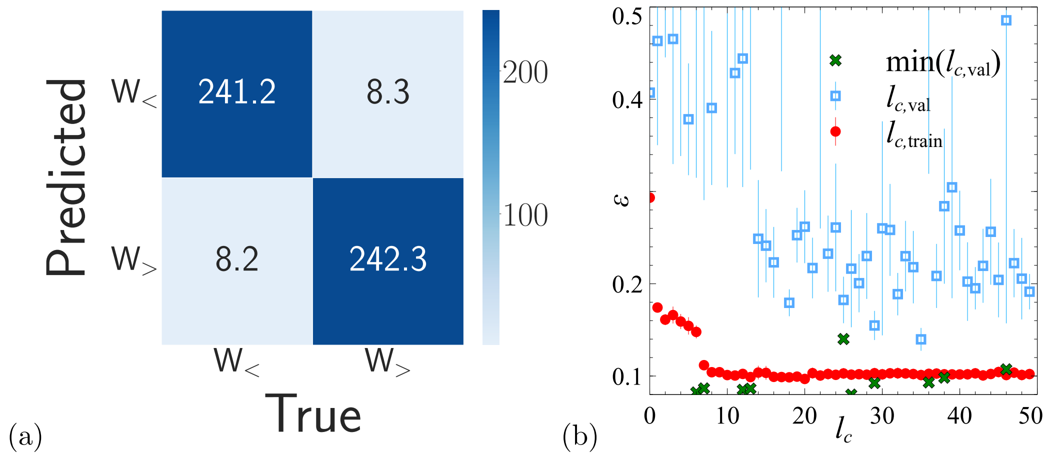
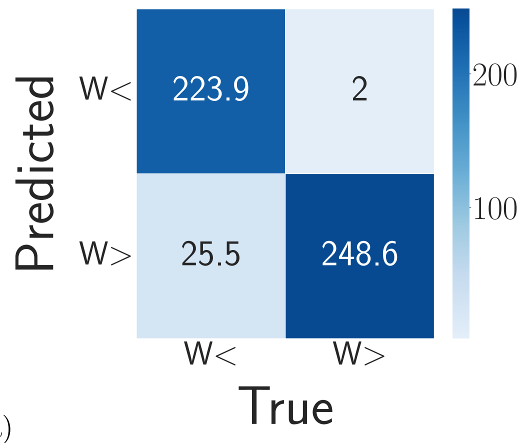


Figure 6.8: (a) Average confusion matrices for state *classification* of $|\psi|^2$ according to extended/localised for the Ohtsuki network. The dataset used is the test data $\mathcal{T}_{[|\psi|^2,[14,19]}$ and the models used for predictions are those corresponding to a minimal $l_{\text{c,val}}$ for $L = 20^3$. The true labels for $W_< \in [14.0, 16.0]$ and $W_> \in [17.0, 19.0]$, are indicated on the horizontal axis while the predicted labels are given on the vertical axis. (b) Dependence of losses $l_{\text{c,train}}$ and $l_{\text{c,val}}$ on the number of epochs ϵ for classification according to extended/localised for $L = 20^3$.

Following the training of system size $L = 20^3$ we obtain the confusion matrix shown in figure 6.8(a). We observe that 3.3% of the samples are misclassified. Through the training of $|\psi|^2$ we manage to achieve a minimal validation accuracy $\min_\epsilon[\langle l_{c,\text{val}}\rangle] = 0.080 \pm 0.006$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\text{val}}\rangle] = 97.3\% \pm 0.21$). We notice a decrease in accuracy compared to the image classification training implemented in section 6.1 but the present result still constitutes a good training. Our second training is done for the size $L = 40^3$. The confusion matrix shown in figure 6.9(a) allows us to see a 5.4% misclassification rate. Through a study of the ten independent trainings, we notice the existence of several trainings which appear to get stuck in local minimal on the cost function landscape. This is confirmed by the plot in figure 6.9(b) displaying an almost constant confidence interval of $\langle l_{c,\text{val}}\rangle$ and $\langle l_{c,\text{train}}\rangle$. This training allowed us to reach a minimal validation accuracy $\min_\epsilon[\langle l_{c,\text{val}}\rangle] = 0.035 \pm 0.066$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\text{val}}\rangle] = 99.0\% \pm 4.2$). We deduce that while eight of our trainings seem to provide low accuracy the network is not complex enough to avoid falling into a local minima for two of our trainings.



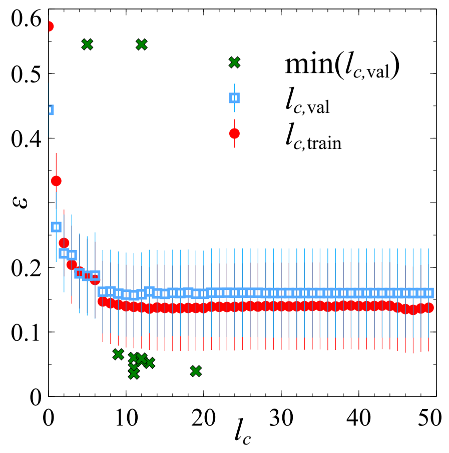
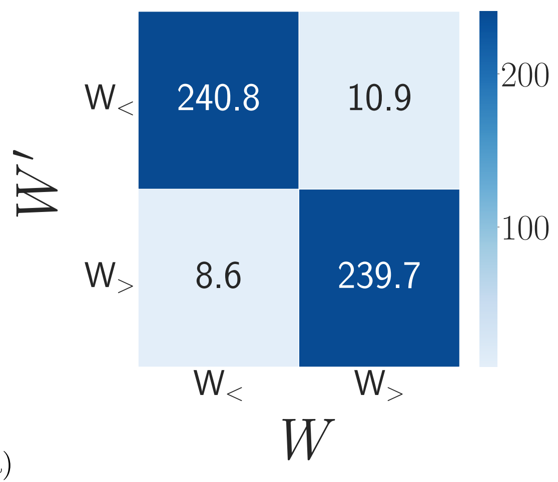


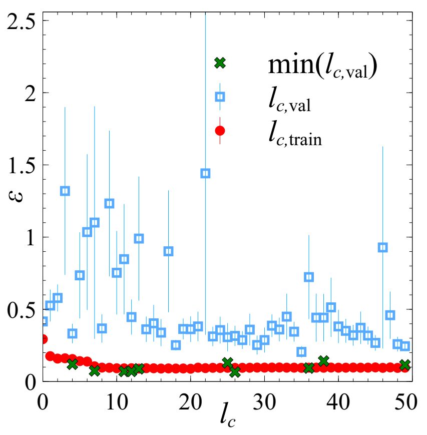
Figure 6.9: In (a) average confusion matrices for state *classification* of $|\psi|^2$ according to extended/localised for the Ohtsuki network. The dataset used is the test data $\mathcal{T}_{[|\psi|^2,[14,19]}$ and the models used for predictions are those corresponding to a minimal $l_{\mathrm{c,val}}$ for $L = 40^3$. The true labels for $W_< \in [14.0, 16.0]$ and $W_> \in [17.0, 19.0]$, are indicated on the horizontal axis while the predicted labels are given on the vertical axis. In (b) dependence of losses $l_{\mathrm{c,train}}$ and $l_{\mathrm{c,val}}$ on the number of epochs ϵ for classification according to extended/localised for $L = 40^3$.

6.6 State classification of phases from $|\psi|^2$ with a ResNet

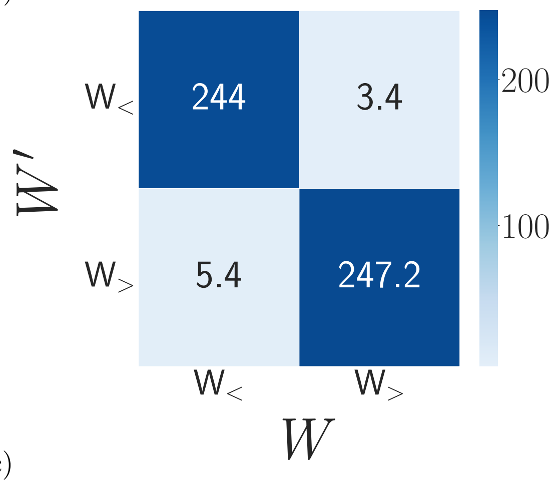
In the previous section, we confirmed the phase prediction ability of a simple network composed of six convolutional layers, for the MIT while highlighting the possible shortcomings of this network [45]. We now would like to challenge CNN methods to identify $|\psi|^2$ at several different disorder values. Given the complex nature of this task, we decide to employ a deeper network, i.e. the 3D ResNet18 architecture [66]. As a first step in our study, we aim to emulate the result obtained in the previous section for phase classification with the Ohtsuki network. We trained $|\psi|^2$ states for system size $L = 20^3$, 40^3 and 100^3. Following ten independent training on system size $L = 20^3$ we achieve minimal validation accuracy $\min_\epsilon[\langle l_{c,\mathrm{val}}\rangle] = 0.066 \pm 0.009$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\mathrm{val}}\rangle] = 97.9\% \pm 0.437$). The average confusion matrix computed after training is shown in figure 6.10(a). We observe that 3.9% of the samples are misclassified. The validation accuracy is slightly worse than the one previously obtained in the case of image classification for $L = 20^3$. However, one could argue about the difference in the task, a classification of $|\psi|^2$ states appears as much more complex than the classification of their associated image representation.



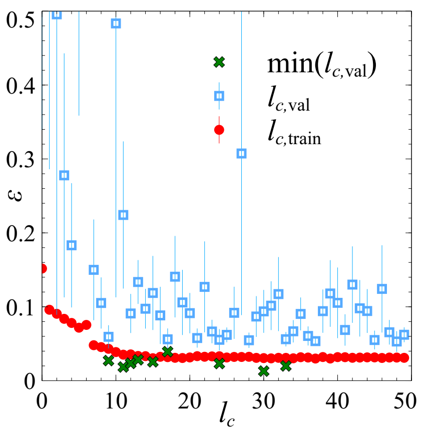
(a)



(b)



(c)



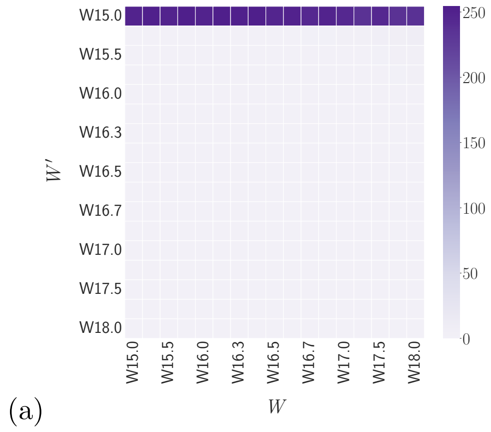
(d)

Figure 6.10: (a+c) Average confusion matrices for state *classification* of $|\psi|^2$ according to extended/localised with the ResNet18 network and $L = 20^3$, $L = 40^3$, respectively. The dataset used is the test data $\tau_{|\psi|^2,[14,19]}$ and the models used for predictions are those corresponding to a minimal $l_{\mathrm{c,val}}$ for $L = 20^3$ and $L = 40^3$, respectively. The true labels for $W_< \in [14.0, 16.0]$ and $W_> \in [17.0, 19.0]$, are indicated on the horizontal axis while the predicted labels are given on the vertical axis. (b+d) Dependence of losses $l_{\mathrm{c,train}}$ and $l_{\mathrm{c,val}}$ on the number of epochs ϵ for classification according to extended/localised for $L = 20^3$ and $L = 40^3$, respectively.

Overall, the quality of the predictions remains satisfactory. We increase the system size to $L = 40^3$ and perform the same training. After 50 epochs we reach a minimal validation loss $\min_\epsilon[\langle l_{c,\mathrm{val}}\rangle] = 0.0123 \pm 0.003$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\mathrm{val}}\rangle] = 99.7\% \pm 0.088$). Looking at the confusion matrix we observe a drop in misclassification compared to the training on system $L = 20^3$, only 1.8% of the samples end up being misclassified. These results lead us to conclude about the similar performance of the deep ResNet18 and a simple six layers CNN architecture in the case of phase prediction.

6.7 State classification of disorders from $|\psi|^2$ with a ResNet

We now train our network directly with the $|\psi|^2$ eigenstates to predict 17 disorder values. Given the time taken to train $L = 20^3$, we will only be able to show the result obtained for one system size. Furthermore, we note that the result displayed here corresponds to a reduced number of epochs, $\varepsilon = 30$. Looking at the figure 6.11(b), we see that the network quickly overfits and $l_{c,\text{val}}$ diverges from $l_{c,\text{train}}$. After training for $\varepsilon = 30$ epochs we barely rich $\min_\epsilon[\langle l_{c,\text{val}}\rangle] = 3.414 \pm 1.436$ (corresponding to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\text{val}}\rangle] = 15.75\% \pm 0.967$). This result is further confirmed by the confusion matrix shown in figure 6.11(a). The network completely fails to make any meaningful predictions and predicts every disorder as $W = 15.0$. It appears that our network is unable to identify disorders from $|\psi|^2$ states.



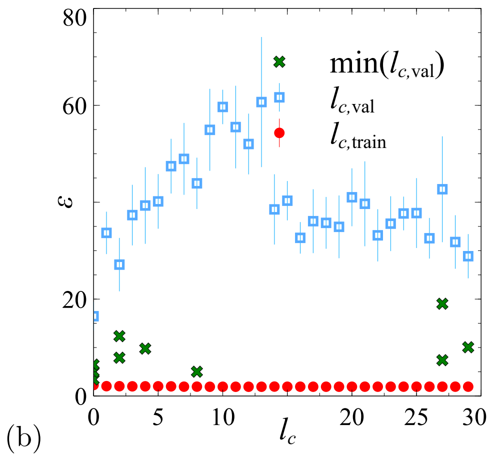


Figure 6.11: (a) Average confusion matrices for state *classification* of $|\psi|^2$ according to 17 disorder values for the ResNet18 network. The dataset used is the test data $\mathcal{T}_{|\psi|^2,[15,18]}$ and the models used for predictions are those corresponding to a minimal $l_{\text{c,val}}$ for $L = 20^3$. The true labels for $W = 15.0, 15.25, \ldots, 18.0$, are indicated on the horizontal axis while the predicted labels are given on the vertical axis. (b) Dependence of losses $l_{\text{c,train}}$ and $l_{\text{c,val}}$ on the number of epochs ϵ for classification according to extended/localised for $L = 20^3$.

6.8 State classification of phases from ψ with a ResNet

Now that we have compared the performance of the ResNet for images and state recognition with $|\psi|^2$, we would like to test the possibility of simply feeding the ψ states as input to our ML process.

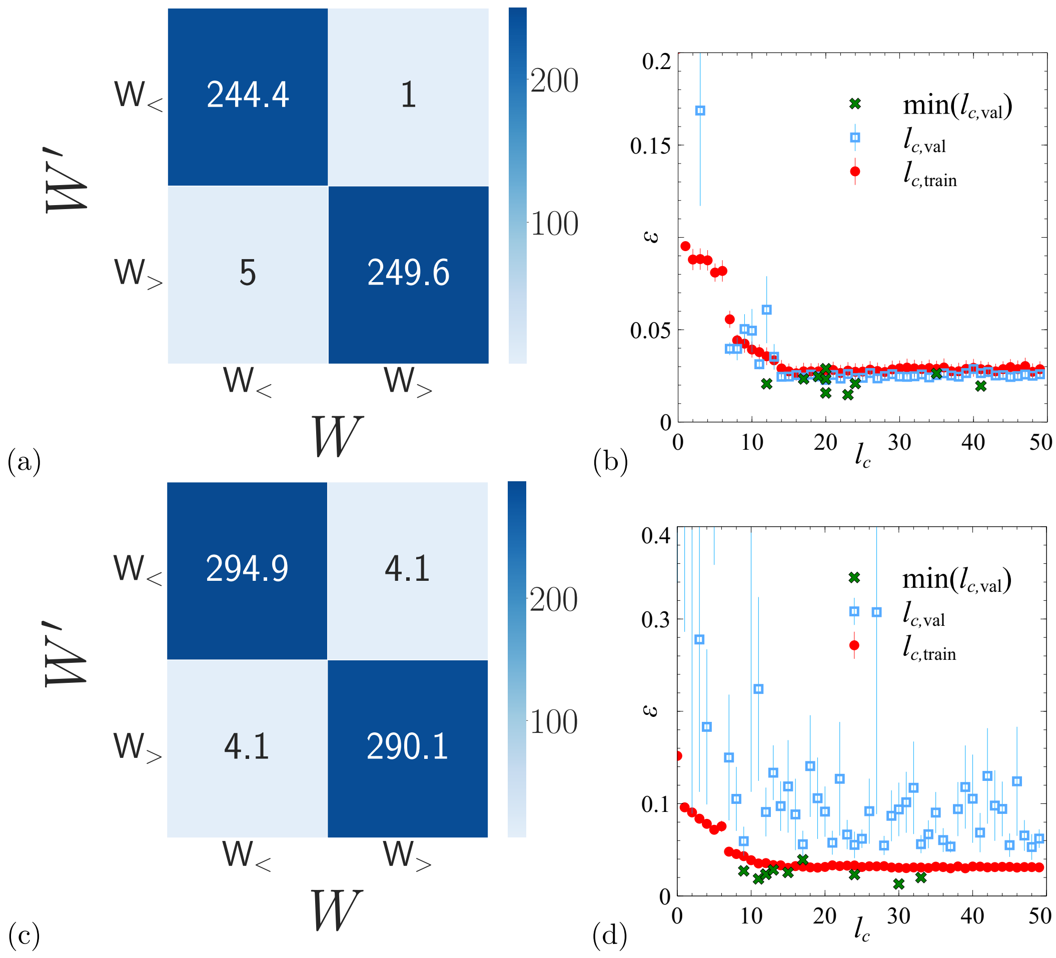


Figure 6.12: In (a+c) average confusion matrices for state *classification* of ψ according to extended/localised. The dataset used is the test data $\tau_{|\psi|,[14,19]}$ and the models used for predictions are those corresponding to a minimal $l_{\mathrm{c,val}}$ for $L = 20^3$ and $L = 40^3$, respectively. The true labels for $W_< \in [14.0, 16.0]$ and $W_> \in [17.0, 19.0]$, are indicated on the horizontal axis while the predicted labels are given on the vertical axis. In (b+d) dependence of losses $l_{\mathrm{c,train}}$ and $l_{\mathrm{c,val}}$ on the number of epochs ϵ for classification according to extended/localised for $L = 20^3$ and $L = 40^3$, respectively.

If the classification with ψ provides satisfying results, this could contribute to reducing the preprocessing steps (squaring of the ψ states) taken before the ML training. For the phase recognition we train the same classes as the one mentioned in section 6.6 with 5000 $|\psi|^2$-values $\in [14, 16]$ label as $W_<$ and 5000 $|\psi|^2$-values $\in [17, 19]$ that we label as $W_>$. The training $L = 20^3$ results in a $\min_\epsilon[\langle l_{c,\text{val}}\rangle] = 0.015 \pm 0.002$ which corresponds to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\text{val}}\rangle] = 99.8\% \pm 0.092)$. Thus, by training on ψ we gain almost 1% in terms of the maximal validation accuracy compared to the training performed on the same system with $|\psi|^2$. This result is confirmed by the confusion matrix given in figure 6.12(a), which shows that only 1.2% of the sample ends up misclassified.

We once again increase the system to size $L = 40^3$ and train the network. This time we obtain $\min_\epsilon[\langle l_{c,\text{val}}\rangle] = 0.002 \pm 0.001$ which corresponds to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\text{val}}\rangle] = 100\% \pm 0.0154)$. This result might appear counter-intuitive by looking at the confusion matrix obtained in figure 6.12(c), as the misclassification appears higher than the one obtained in figure 6.12(a). However, we recall that each confusion matrix corresponds to an average over ten prediction runs. To conclude, our ResNet appears to be able to identify the two phases of the Anderson model by training on ψ.

6.9 State classification of disorders from ψ with a ResNet

We again aim at predicting 17 disorder values. Here, we choose ψ as input. Similarly to section 6.7 we only present $L = 20^3$ due to the long running time of the training. Surprisingly, we see that the training which previously failed to give meaningful results in section 6.7, now provides good results. Looking at $l_{c,\text{val}}$ and lc,train in figure 6.13(b), we immediately notice drastic change from 6.11(b). After 45 epochs we reach a $\min_\epsilon[\langle l_{c,\text{val}}\rangle] = 0.907 \pm 0.014$ which corresponds to a maximal validation accuracy $\max_\epsilon[\langle a_{c,\text{val}}\rangle] = 64.1\% \pm 0.497)$. This low validation accuracy might lead us to deduce a complete failure of our training. However, looking at 6.13(a), we observe that we obtain a heavily diagonal confusion matrix with misclassification remaining close to the true disorder value. We hypothesise that the use of the larger system might contribute to refining the result obtained. An explanation for the success of the training with ψ compared to the failed training of $|\psi|^2$ might reside in the nature of these two inputs. Indeed, ψ-states are complex while $|\psi|^2$-states are real. Thus, we hypothesise that our network finds meaningful information in ψ-states that are not in $|\psi|^2$-states.

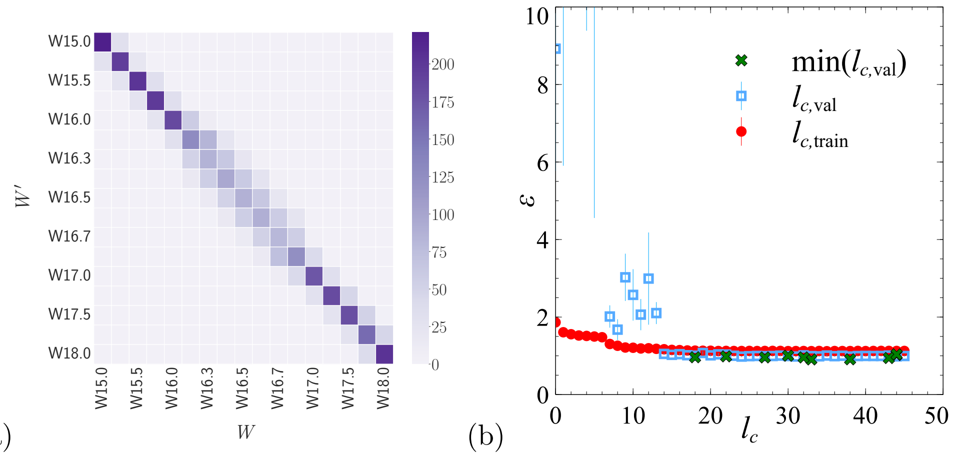


Figure 6.13: (a) Average confusion matrices for ψ *classification* according to 17 disorder values using the ResNet18 network. The dataset used is the test data $\tau_{\psi,[15,18]}$ and the models used for predictions are those corresponding to a minimal $l_{\text{c,val}}$ for $L = 20^3$. The true labels for $W = 15.0, 15.25, \ldots, 18.0$, are indicated on the horizontal axis while the predicted labels are given on the vertical axis. (b) Dependence of losses $l_{\text{c,train}}$ and $l_{\text{c,val}}$ on the number of epochs ϵ for classification according to extended/localised for $L = 20^3$.

6.10 Regression with disorders $|\psi|^2$ with a ResNet

Now that we have explored classification methods to identify the phases and states of the three-dimensional Anderson model, we would like to train a network to make continuous predictions on the disorders from $|\psi|^2$ states. This is done through the use of the regression method. We use the 3D ResNet network architecture employed in section 6.5-6.9 and modify the last layer to have one output neuron making continuous predictions. Following the scheme of the previous sections we train two different system sizes $L = 20^3$ and $L = 40^3$. For each of these systems, we train two sets of $|\psi|^2$. The first series of training is done at five different disorder values $W = 15.0, 15.75, 16.5, 17.25, 18.0$ and the second one for 4 disorder values $W = 14.0, 16.0, 17.75, 19.0$. In each case, ten independent networks are trained for 50 epochs. Once the trainings are performed, we evaluate their prediction abilities on 24 disorder value $W = [14.0, 14.25, \ldots, 18.25, 18.5, 18.75, 19.0]$.

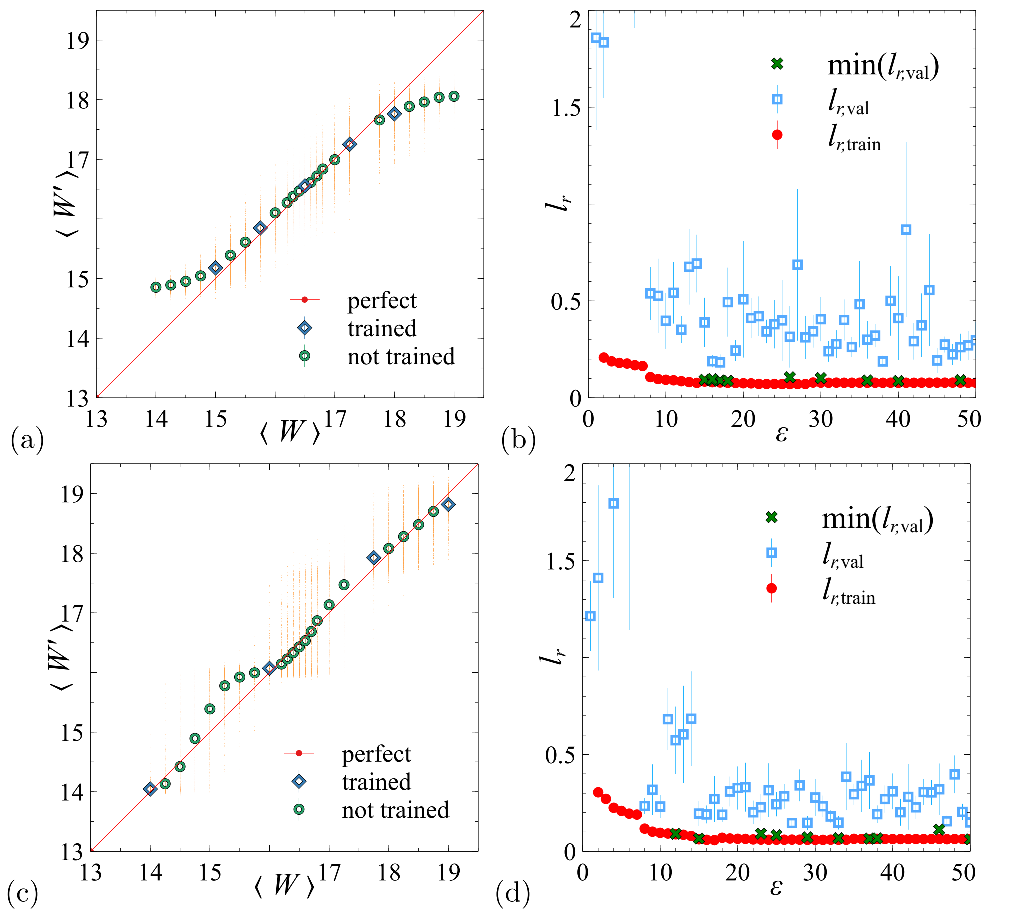


Figure 6.14: (a) Average prediction curve obtained for *regression* according to $|\psi|^2$, $L = 20^3$ and $W = 15.0, 15.75, 16.5, 17.25, 18.0$ at the minimal $l_{r,\text{val}}$. (b) Dependence of losses $l_{r,\text{train}}$ and $l_{r,\text{val}}$ averaged on the number of epochs ϵ for associated to training (a). (c) Average prediction curve obtained for regression according to $|\psi|^2$ trained for $W = 14.0, 16.0, 17.75, 19.0$ at the minimal $l_{r,\text{val}}$. (d) Dependence of losses $l_{r,\text{train}}$ and $l_{r,\text{val}}$ averaged on the number of epochs ϵ for regression according to W. For both training, the dataset used is the test data $\tau_{[|\psi|^2,[14,19]}$ and the models used for predictions are those corresponding to a minimal $l_{r,\text{val}}$.

In figure 6.14(a) we show the prediction curve obtained after training the system $L = 20^3$. After 50 epochs we obtain a minimal validation loss of $\min_\epsilon[\langle l_{r,\text{val}}\rangle] = 0.087 \pm 0.002$. Overall, the network seems to understand the task at hand. Despite a spread in prediction, the mean associated with samples with disorder $W \in]15, 18[$ follows the line of perfect predictions. We however notice that the network struggles to correctly identify samples with disorder values out of the range of the training. For the training with 4 disorder values, we reach a minimal $\min_\epsilon[\langle l_{r,\text{val}}\rangle] = 0.062 \pm 0.006$.

In figure 6.14(c) we display the result of the training for $|\psi|^2$-states trained for four disorder values, $W \in [14.0, 16.0, 17.75, 19.0]$. The general trend of the predictions seems to follow the curve of perfect prediction despite the small number of W values trained. We repeat the same process for the $L = 40^3$ system. For the five disorders, we observe a similar result as the one obtained for $L = 20^3$, with the out-of-range value deviating from the line of perfect prediction. This training allows us to reach a minimal validation loss of $\min_\epsilon[\langle l_{r,\text{val}}\rangle] = 0.022 \pm 0.024$. Finally, we train the $L = 40^3$ for four disorder values. In figure 6.15(c) we show the results of the predictions. We observe a clear improvement of the prediction compared to the 4-disorders training of the $L = 20^3$. This training allows us to reach a minimal $\min_\epsilon[\langle l_{r,\text{val}}\rangle] = 0.032 \pm 0.033$. We notice this time that the confidence interval of this training is of the same order of magnitude as the minimal loss. For $L = 20^3$ and $L = 40^3$, we conclude that the better performances of the models trained with only four datapoints result from a larger range in the disorders trained. Indeed, in the case of five disorders value trained, $W = 15.0, 15.75, 16.5, 17.25, 18.0$, the network has difficulty in identifying disorder values beyond the trained disorders $W = 15.0$ and $W = 18.0$.

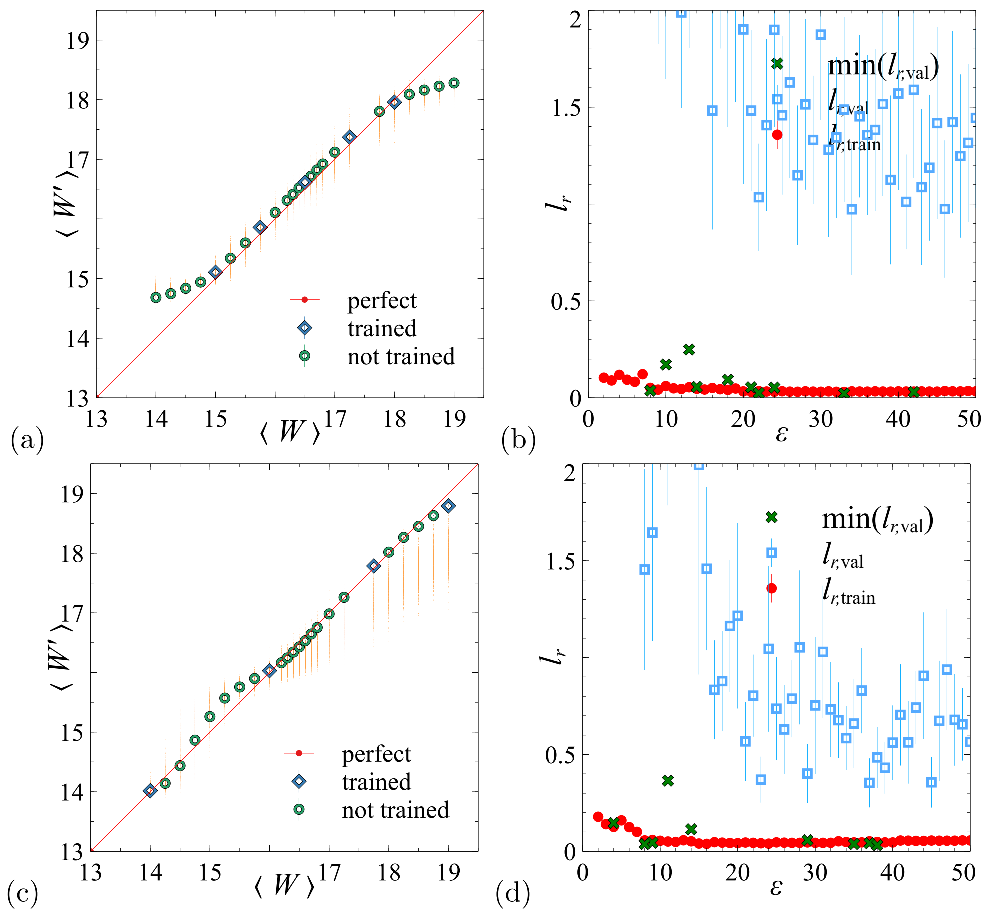


Figure 6.15: (a) Average prediction curve obtained for *regression* according to $|\psi|^2$, $L = 40^3$ and $W = 15.0, 15.75, 16.5, 17.25, 18.0$ at the minimal $l_{r,\mathrm{val}}$. (b) Dependence of losses $l_{r,\mathrm{train}}$ and $l_{r,\mathrm{val}}$ averaged on the number of epochs ϵ for regression according to five disorder values. (c) Average prediction curve obtained for *regression* according to $|\psi|^2$ trained for $W = 14.0, 16.0, 17.75, 19.0$ at the minimal $l_{r,\mathrm{val}}$. (d) Dependence of losses $l_{r,\mathrm{train}}$ and $l_{r,\mathrm{val}}$ averaged on the number of epochs ϵ for regression according to four disorder values.

6.11 Regression with disorders ψ with a ResNet

While training for the classification of states according to W in section 6.9, we showed an increase in performance when training on ψ value compared to $|\psi|^2$. We therefore decide to implement a regression study similar to the one done in the previous section but using ψ states as input. Following the protocol of the previous section, we train two different system sizes $L = 20^3$ and $L = 40^3$ at five and four disorder values.

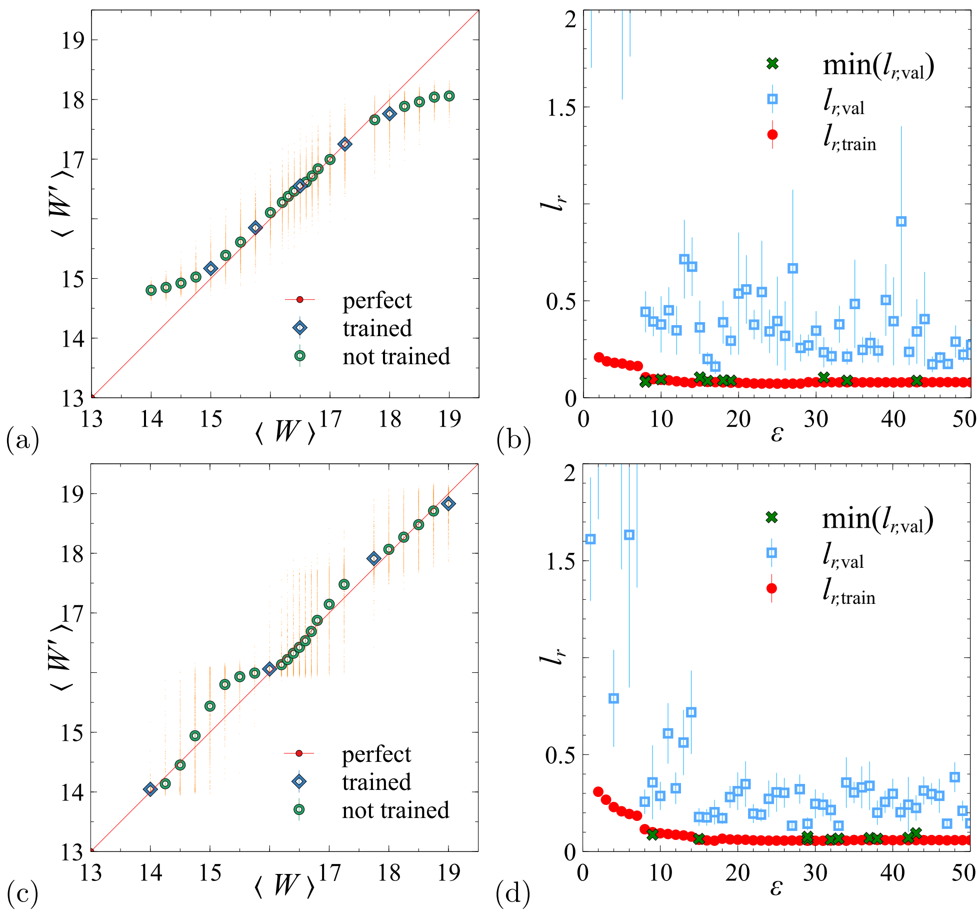
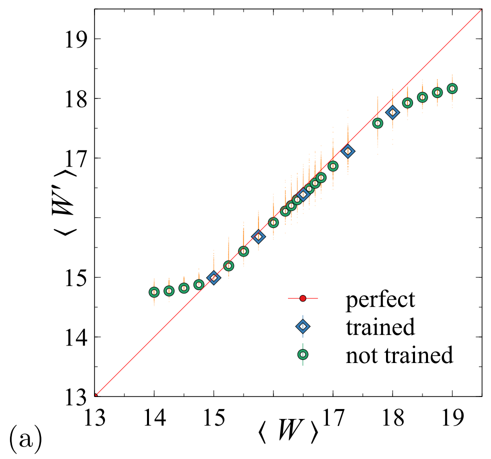
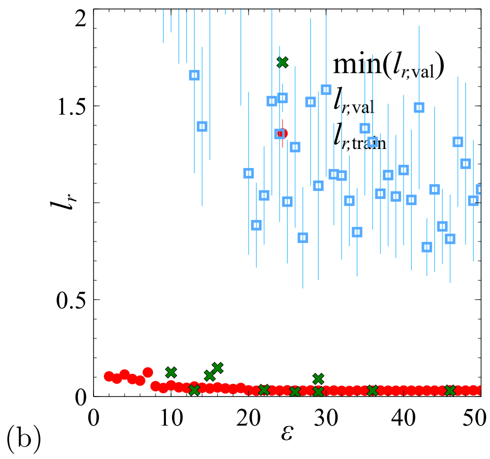
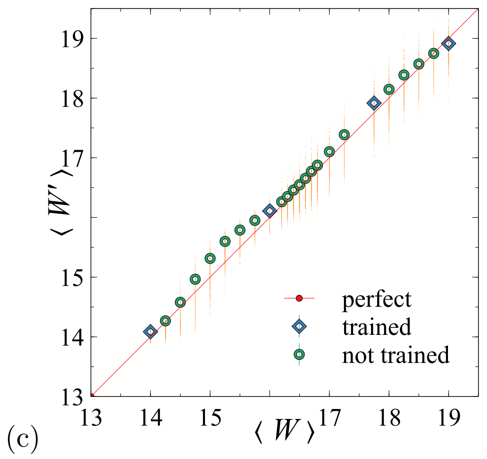


Figure 6.16: (a) Average prediction curve obtained for *regression* according to ψ, $L = 20^3$ and trained at $W = 15.0, 15.75, 16.5, 17.25, 18.0$ at the minimal $l_{r,\text{val}}$. (b) Dependence of losses $l_{r,\text{train}}$ and $l_{r,\text{val}}$ averaged on the number of epochs ϵ for regression according to W. The dataset used is the test data $\tau_{[\psi,[14,19]}$ and the models used for predictions are those corresponding to a minimal $l_{r,\text{val}}$. (c) Average prediction curve obtained for *regression* according to ψ, for $L = 20^3$ and trained at $W = 14.0, 16.0, 17.75, 19.0$ at the minimal $l_{r,\text{val}}$. (d) Dependence of losses $l_{r,\text{train}}$ and $l_{r,\text{val}}$ averaged on the number of epochs ϵ for regression according to W.

We begin by training the $L = 20^3$ system for $W = 15.0, 15.75, 16.5, 17.25, 18.0$ and 50 epochs. After training, we obtain a minimal validation loss $\min_\epsilon[\langle l_{r,\text{val}}\rangle] = 0.082 \pm 0.003$. The results of the prediction of the trained network are displayed in figure 6.16(a). The curve obtained is similar to the plot in figure 6.14(a), with the W-values out of the range of training diverging from the curve of perfect predictions. Training for four disorder values allows us to obtain the curve seen in figure 6.14(c) with an associated minimal validation loss of $\min_\epsilon[\langle l_{r,\text{val}}\rangle] = 0.062 \pm 0.004$. Again, the curve of prediction is similar to the one obtained in figure 6.14(c), with the average of the prediction remaining close to the curve of perfect prediction. The next step in our study is to increase the size of the system and reproduce the same training on the system of size $L = 40^3$. After training for 50 epochs at five disorder values we reach a minimal validation of $\min_\epsilon[\langle l_{r,\mathrm{val}}\rangle] = 0.023 \pm 0.016$. The curve of prediction displayed in figure 6.17(a) is similar to the one obtained in figure 6.16(a), however, we note a smaller spread of the distribution for the system $L = 40^3$. Unsurprisingly, training on a larger system appears to improve and reduce the spreading in the predictions. We perform one last training on the $L = 40^3$ system for the four disorder values $W = 14.0, 16.0, 17.75, 19.0$ and reach a validation loss $\min_\epsilon[\langle l_{r,\mathrm{val}}\rangle] = 0.024 \pm 0.028$. As shown by figure 6.17(c), the predictions made by the network follow the curve of perfect prediction. We do not observe significant differences in the results of the predictions made for $|\psi|^2$ and ψ. Nonetheless, in both cases, we observe an increase in the quality of the predictions made by the network between $L = 20^3$ and $L = 40^3$.







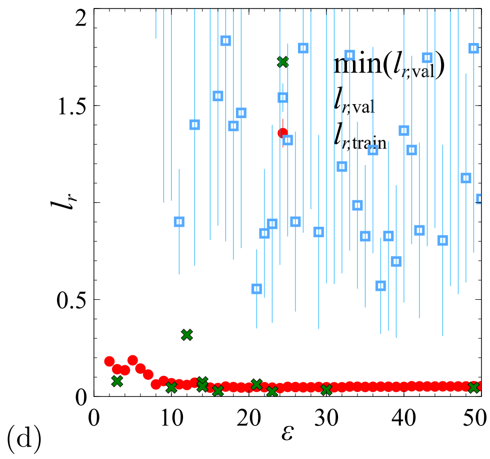


Figure 6.17: (a) Average prediction curve obtained for *regression* according to ψ, $L = 40^3$ and trained at $W = 15.0, 15.75, 16.5, 17.25, 18.0$ at the minimal $l_{r,\mathrm{val}}$. (b) Dependence of losses $l_{r,\mathrm{train}}$ and $l_{r,\mathrm{val}}$ averaged on the number of epochs ϵ for regression according to W. The dataset used is the test data $\tau_{[\psi,[14,19]}$ and the models used for predictions are those corresponding to a minimal $l_{r,\mathrm{val}}$. (c) Average prediction curve obtained for *regression* according to ψ, for $L = 40^3$ and trained at $W = 14.0, 16.0, 17.75, 19.0$ at the minimal $l_{r,\mathrm{val}}$. (d) Dependence of losses $l_{r,\mathrm{train}}$ and $l_{r,\mathrm{val}}$ averaged on the number of epochs ϵ for regression according to W.

Chapter 7

Summary and outlook

Throughout this thesis we proposed ML studies for two different models, the classical two-dimensional site percolation model and the quantum Anderson model of localisation. By using standard image recognition tools such as CNN coupled with supervised and unsupervised methods we were able to display the ability of ML to correctly identify phases and the pitfalls that one might want to avoid.

In Chapter 2 we presented the well-known two-dimensional percolation model, a model where sites are randomly occupied with a probability p and left empty with a probability $1-p$. We gave several important quantities associated with the model. In the second part of this chapter, we introduced our second model of interest, the three-dimensional Anderson model, which is characterised by a transition from an extended to a localised regime in the presence of high disorder. Chapter 3 allowed us to provide a general introduction to machine learning and the main tasks, supervised learning and unsupervised learning. We presented the two datasets used during our study in chapter 4. Through several analyses of the parameters of our percolation dataset, we showed the effect of finite size of the samples on parameters such as p_c or ξ. In the case of the Anderson model, we displayed a representation of $|\psi|^2$ samples in the metal, insulator and transition regime.

In chapter 5 we displayed the results of our study for the two-dimensional site percolation model. Three topics were studied, the density p, the correlation length ξ and the spanning/non-spanning property. In the case of density p, we demonstrated that supervised and unsupervised learning methods permit us to extract the density in our lattices. Furthermore, the use of regression proved a deep understanding of the task performed, as we show that the network was able to correctly predict p-values of samples with resolution smaller than the ones trained for. However, when looking at parameters related to the connectivity in the lattices, these CNN methods fall short. We began by relabeling the 31 classes previously used for image classification according to the average correlation length of the set $\langle\xi\rangle$. After classification, we obtained the confusion matrix shown in figure 5.3. We note that similarly to the confusion matrix obtained for the classification training by density in figure 5.11 this matrix is heavily diagonal. Nonetheless, it is essential to point out the spread of ξ-values in each $\langle\xi\rangle$ class. Following this observation we proposed two reorganisations of the classes, taking into account the distribution of ξ in the dataset $\mathcal{T} \cup \mathcal{V}$. These new trainings highlighted the lack of recognition of ξ by our network, which was further confirmed by a regression study of ξ where the network was unable to provide a correct prediction. Through the classification training of the spanning/ non-spanning, we cemented this hypothesis. Instead of searching for connectivity, the network seemed to utilise the proxy of density to identify the phases. From this observation, we attempted to remove the dependence on the density by performing a training for the spanning/non-spanning property at $p = 0.585 \approx p_c$. Once again, the network struggled to identify the phases of the model. The last step of our study was to reproduce an unsupervised study. Our main finding is that the inefficiency of our ML method is due to the type of model studied. The physics of percolation relies on the formation of large clusters which is a long-range property of the system, while CNN are known to retain information about short-range properties [17, 20]. Some might argue that part of CNN methods could be assimilated into a renormalisation process. As we saw in section 2.1.5, renormalisation provides an efficient way to reduce short-range correlation in our system while preserving long-range behaviour. However, the success of renormalisation can only be attributed to the prior knowledge of the form of a spanning cluster. Both methods are coarse-graining but while renormalisation neglects short correlation interaction and preserves long-range behaviour, CNN methods do the opposite. As such, the CNN method cannot identify the spanning property in the samples. We could argue about the possibility of improving the training by providing the cluster information through HK similar method, as it was performed in a recent study [83]. However, one might think about the validity of this process as this would require a two-step process being the pre-processing of the lattice coupled with an ML training (through the application of the HK algorithm). Furthermore, following labelling with the HK method, the phase transition can be easily identified and a ML training would prove to be redundant. Although our study of the 2D site percolation model provides a cautionary tale of the possible pitfalls of ML in the study of phase transitions, it does not intend to negate the performance displayed in previously published papers.

Finally, in chapter 6 we presented a study of the Anderson model of localisation. We began by performing classification trainings on images and discovered that at the same resolution, it was possible to increase the performance of the network by increasing the size of the system presented. This result was valid in the cases of phase and disorder classification. However, we saw the limitation of image classification for disorders classification which led us to focus $|\psi|^2$ and ψ states. While training for phase classification of $|\psi|^2$, we successfully reproduced previous results [45] with a simple six convolutional network as well as with the ResNet18. Nonetheless, despite the success of the simple six layers CNN, we noticed that such architecture could lead to remaining trapped in local minima and therefore concluded to the better performance of the ResNet. While training for disorders classification, we identify a weird behaviour of the network which seemingly provides better performance when using ψ instead of $|\psi|^2$ as input. The last phase of our study of the Anderson model was performed through regression to predict disorders from $|\psi|^2$ states. Similarly to classification, we noted an increase in the performance as the size of the system increased. Despite the low number of disorder values trained by the network, our model provides good predictions of untrained disorder values.

The main question remains, why is the ML study of the Anderson model of localisation so successful while the percolation model fails? After all, for both models, one observes the spreading of a cluster and wavefunctions. Again, this relies on the very technique used to study these models. CNNs retains neighbouring correlation in the receptive field. Therefore, we hypothesise that while a subset of a percolation lattice cannot provide us with some knowledge on the spreading of potential spanning clusters, a subset of an Anderson state can inform us of the localisation of the wavefunction. In the case of the classification through images of the Anderson model, a localised state would present a zone of high density with several regions around, of low densities.

However, while we note an overall success of CNN tools in the phase classification of the Anderson model with images and $|\psi|^2$ state, one might still wonder about the failure of the disorder classification with $|\psi|^2$. To get some insight into the reasons behind this peculiar result, further work needs to be performed on this model. A possible lead to fix this issue might reside in looking at the feature maps of the network, or even relaunching a hyperparameter search. Another important point highlighted by our training of the Anderson model is the inability of classical ML metrics such as the accuracy to convey a training that we could visually judge reasonably successful, such as 6.13(a). Indeed, classification tasks were originally used to identify very distinct classes of objects such as cars or planes. However, in our case, a misclassification between two neighbouring disorders already constitutes a success. As such we need a metric conveying this result. Studying all these leads would allow us to provide a solid review of the performance of supervised CNN methods applied to the three-dimensional Anderson model.

Bibliography

[1] D. Bayo, A. Honecker, and R. A. Römer, "Machine learning the 2D percolation model," *Journal of Physics: Conference Series*, vol. 2207, no. 1, p. 012057, 3 2022. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/2207/1/012057

[2] D. Bayo, A. Honecker, and R. A. Römer, "The percolating cluster is invisible to image recognition with deep learning," *New Journal of Physics*, vol. 25, no. 11, p. 113041, Novem 2023. [Online]. Available: https://dx.doi.org/10.1088/1367-2630/ad0525

[3] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, Dec 1943. [Online]. Available: https://doi.org/10.1007/BF02478259

[4] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, vol. 65 6, pp. 386–408, 1958. [Online]. Available: https://doi.org/10.1037/h0042519

[5] M. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*. The MIT Press, 09 2017. [Online]. Available: https://doi.org/10.7551/mitpress/11301.001.0001

[6] D. Crevier, *AI: The Tumultuous History of the Search for Artificial Intelligence*. Cambridge University Press, 1997, vol. 30.

[7] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc Natl Acad Sci USA*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982.

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct 1986. [Online]. Available: https://doi.org/10.1038/323533a0

[9] T. K. Ho, “Random decision forests,” in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, 1995, pp. 278–282 vol.1.

[10] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 11 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[11] S. Raschka, J. Patterson, and C. Nolet, “Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence,” *Information*, vol. 11, no. 4, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/4/193

[12] NVIDIA, P. Vingelmann, and F. H. Fitzek, “Cuda, release: 10.2.89,” 2020. [Online]. Available: https://developer.nvidia.com/cuda-toolkit

[13] D. H. Hubel, “Single unit activity in striate cortex of unrestrained cats.” *J Physiol.*, vol. 147, pp. 226–238., Sep 1959. [Online]. Available: https://doi.org/10.1113/jphysiol.1959.sp006238.

[14] D. H. Hubel and T. N. Wiesel, “Receptive fields of single neurones in the cat’s striate cortex,” *J Physiol.*, vol. 148, p. 574–591., Oct 1959. [Online]. Available: https://doi.org/10.1113/jphysiol.1959.sp006308

[15] ——, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex.” *J Physiol.*, vol. 160, pp. 106–154., Jan 1962. [Online]. Available: https://doi.org/10.1113/jphysiol.1962.sp006837

[16] A. Ivakhnenko and V. Lapa, *Cybernetics and Forecasting Techniques*, ser. Modern analytic and computational methods in science and mathematics. American Elsevier Publishing Company, 1967.

[17] P. Mehta, M. Bukov, C. H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, “A high-bias, low-variance introduction to Machine Learning for physicists,” *Phys. Rep.*, vol. 810, pp. 1–124, 3 2019. [Online]. Available: https://doi.org/10.1016/j.physrep.2019.03.001

[18] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[19] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation Applied to Handwritten Zip Code

Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 12 1989. [Online]. Available: https://doi.org/10.1162/neco.1989.1.4.541

[20] A. Dawid, J. Arnold, B. Requena, A. Gresch, M. Płodzień, K. Donatella, K. A. Nicoli, P. Stornati, R. Koch, M. Büttner, R. Okuła, G. Muñoz-Gil, R. A. Vargas-Hernández, A. Cervera-Lierta, J. Carrasquilla, V. Dunjko, M. Gabrié, P. Huembeli, E. van Nieuwenburg, F. Vicentini, L. Wang, S. J. Wetzel, G. Carleo, E. Greplová, R. Krems, F. Marquardt, M. Tomza, M. Lewenstein, and A. Dauphin, "Modern applications of machine learning in quantum sciences," 2023, (*Preprint* 2204.04198).

[21] G. Yao, T. Lei, and J. Zhong, "A review of convolutional-neural-network-based action recognition," *Pattern Recognition Letters*, vol. 118, pp. 14–22, 2019, cooperative and Social Robots: Understanding Human Activities and Intentions. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865518302058

[22] S. Soni, S. S. Chouhan, and S. S. Rathore, "Textconvonet: a convolutional neural network based architecture for text classification," *Applied Intelligence*, vol. 53, no. 11, pp. 14 249–14 268, Jun 2023. [Online]. Available: https://doi.org/10.1007/s10489-022-04221-9

[23] H. Wang, Y. Li, S. A. Khan, and Y. Luo, "Prediction of breast cancer distant recurrence using natural language processing and knowledge-guided convolutional neural network," *Artificial Intelligence in Medicine*, vol. 110, p. 101977, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0933365720312422

[24] S. M. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, and M. K. Khan, "Medical image analysis using convolutional neural networks: A review," *Journal of Medical Systems*, vol. 42, no. 11, p. 226, Oct 2018. [Online]. Available: https://doi.org/10.1007/s10916-018-1088-1

[25] "Kaggle competition "Dogs vs. Cats": Create an algorithm to distinguish dogs from cats," 2012. [Online]. Available: https://www.kaggle.com/competitions/dogs-vs-cats/leaderboard

[26] W. Zhang, J. Liu, and T.-C. Wei, "Machine learning of phase transitions in the percolation and *XY* models," *Phys. Rev. E*, vol. 99, p. 032142, Mar 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.99.032142

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016.

[28] 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, "Machine learning to classify animal species in camera trap images: Applications in ecology," *Methods in Ecology and Evolution*, vol. 10, no. 4, pp. 585–590, 4 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/2041-210X.13120

[29] R. Zhang, Q. Wang, and Y. Lu, "Combination of ResNet and Center Loss Based Metric Learning for Handwritten Chinese Character Recognition," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR).* IEEE, 2017, pp. 25–29. [Online]. Available: http://ieeexplore.ieee.org/document/8270270/

[30] Y. Zhao and M. Khushi, "Wavelet Denoised-ResNet CNN and LightGBM Method to Predict Forex Rate of Change," *IEEE International Conference on Data Mining Workshops, ICDMW*, vol. 2020-Novem, pp. 385–391, 2020.

[31] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada, "Restricted Boltzmann machine learning for solving strongly correlated quantum systems," *Phys. Rev. B*, vol. 96, p. 205152, Novem 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.96.205152

[32] W.-J. Rao, "Machine learning the many-body localization transition in random spin systems," *J. Phys.: Condens. Matter*, vol. 30, no. 39, p. 395902, Sep 2018. [Online]. Available: https://dx.doi.org/10.1088/1361-648X/aaddc6

[33] Y. Zhang and E.-A. Kim, "Quantum loop topography for machine learning," *Phys. Rev. Lett.*, vol. 118, p. 216401, May 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.118.216401

[34] M. Stoudenmire, "Learning with quantum-inspired tensor networks," Aug 2016. [Online]. Available: https://pirsa.org/16080007

[35] H. Saito, "Solving the Bose–Hubbard Model with Machine Learning," *Journal of the Physical Society of Japan*, vol. 86, no. 9, p. 093001, 2017. [Online]. Available: https://doi.org/10.7566/JPSJ.86.093001

[36] ——, "Method to solve quantum few-body problems with artificial neural networks," *Journal of the Physical Society of Japan*, vol. 87, no. 7, p. 074002, 2018. [Online]. Available: https://doi.org/10.7566/JPSJ.87.074002

[37] Z. Cai and J. Liu, "Approximating quantum many-body wave functions using artificial neural networks," *Phys. Rev. B*, vol. 97, p. 035116, Jan 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.97.035116

[38] G. Carleo and M. Troyer, "Solving the quantum many-body problem with artificial neural networks," *Science*, vol. 355, no. 6325, pp. 602–606, 2 2017. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/28183973

[39] L. Huang and L. Wang, "Accelerated Monte Carlo simulations with restricted Boltzmann machines," *Phys. Rev. B*, vol. 95, p. 035105, Jan 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.95.035105

[40] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, "Self-learning Monte Carlo method," *Phys. Rev. B*, vol. 95, p. 041101, Jan 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.95.041101

[41] J. Carrasquilla and R. G. Melko, "Machine learning phases of matter," *Nature Physics*, vol. 13, no. 5, pp. 431–434, 2017. [Online]. Available: https://www.nature.com/articles/nphys4035.pdf

[42] E. P. L. van Nieuwenburg, Y.-H. Liu, and S. D. Huber, "Learning phase transitions by confusion," *Nature Physics*, vol. 13, pp. 435 – 439, 2016.

[43] J. Venderley, V. Khemani, and E.-A. Kim, "Machine learning out-of-equilibrium phases of matter," *Phys. Rev. Lett.*, vol. 120, p. 257204, Jun 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.120.257204

[44] K. Ch'ng, J. Carrasquilla, R. G. Melko, and E. Khatami, "Machine learning phases of strongly correlated fermions," *Phys. Rev. X*, vol. 7, p. 031038, Aug 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.7.031038

[45] T. Ohtsuki and T. Mano, "Drawing Phase Diagrams of Random Quantum Systems by Deep Learning the Wave Functions," *J. Phys. Soc. Jpn.*, vol. 89, no. 2, p. 022001, 2 2020. [Online]. Available: https://doi.org/10.7566/JPSJ.89.022001

[46] L. Ujfalusi and I. Varga, “Quantum percolation transition in three dimensions: Density of states, finite-size scaling, and multifractality,” *Phys. Rev. B*, vol. 90, p. 174203, Nov 2014. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.90.174203

[47] S. R. Broadbent and J. M. Hammersley, “Percolation processes: I. Crystals and mazes,” *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 53, no. 3, pp. 629–641, 1957. [Online]. Available: https://doi.org/10.1017/S0305004100032680

[48] D. Stauffer and A. Aharony, *Introduction to Percolation Theory*, 2nd ed. Taylor & Francis Group, 1991.

[49] J. Shen, W. Li, S. Deng, and T. Zhang, “Supervised and unsupervised learning of directed percolation,” *Phys. Rev. E*, vol. 103, p. 052140, May 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.103.052140

[50] 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, (*Preprint* 2212.14694).

[51] W. Yu and P. Lyu, “Unsupervised machine learning of phase transition in percolation,” *Physica A: Statistical Mechanics and its Applications*, vol. 559, p. 125065, 2020. [Online]. Available: https://doi.org/10.1016/j.physa.2020.125065

[52] T. Mano and T. Ohtsuki, “Phase diagrams of three-dimensional anderson and quantum percolation models using deep three-dimensional convolutional neural network,” *Journal of the Physical Society of Japan*, vol. 86, no. 11, p. 113704, 2017. [Online]. Available: https://doi.org/10.7566/JPSJ.86.113704

[53] T. Čadež, B. Dietz, D. Rosa, A. Andreanov, K. Slevin, and T. Ohtsuki, “Machine learning wave functions to identify fractal phases,” *Phys. Rev. B*, vol. 108, p. 184202, Nov 2023. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.108.184202

[54] R. J. Elliott, B. R. Heap, D. J. Morgan, and G. S. Rushbrooke, “Equivalence of the Critical Concentrations in the Ising and Heisenberg Models of Ferromagnetism,” *Phys. Rev. Lett.*, vol. 5, pp. 366–367, Oct 1960. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.5.366

[55] P. J. Flory, *Principles of Polymer Chemistry.* Cornell University Press, 1953. [Online]. Available: https://books.google.com.mx/books?id=CQ0EbEkT5R0C

[56] B. Derrida and D. Stauffer, "Corrections To Scaling and Phenomenological Renormalization for 2-Dimensional Percolation and Lattice Animal Problems." *Journal de physique Paris*, vol. 46, no. 10, pp. 1623–1630, 1985. [Online]. Available: https://doi.org/10.1051/jphys:0198500460100162300

[57] G. Grimmett, *Percolation.* Berlin: Springer Verlag, 1989. [Online]. Available: https://link.springer.com/book/10.1007/978-3-662-03981-6

[58] J. L. Jacobsen, "High-precision percolation thresholds and Potts-model critical manifolds from graph polynomials," *J. Phys. A: Math. Theor.*, vol. 47, no. 13, p. 135001, Mar 2014. [Online]. Available: https://dx.doi.org/10.1088/1751-8113/47/13/135001

[59] J. Hoshen and R. Kopelman, "Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm," *Phys. Rev. B*, vol. 14, no. 8, pp. 3438–3445, 1976. [Online]. Available: https://doi.org/10.1103/PhysRevB.14.3438

[60] L. P. Kadanoff, "Scaling laws for Ising models near T_c," *Physics Physique Fizika*, vol. 2, pp. 263–272, Jun 1966. [Online]. Available: https://link.aps.org/doi/10.1103/PhysicsPhysiqueFizika.2.263

[61] P. Drude, "Zur Elektronentheorie der Metalle," *Annalen der Physik*, vol. 306, no. 3, pp. 566–613, 1900. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.19003060312

[62] ——, "Zur Elektronentheorie der Metalle; II. Teil. Galvanomagnetische und thermomagnetische Effecte," *Annalen der Physik*, vol. 308, no. 11, pp. 369–402, 1900. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.19003081102

[63] P. W. Anderson, "Absence of diffusion in certain random lattices," *Phys. Rev.*, vol. 109, pp. 1492–1505, Mar 1958. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.109.1492

[64] E. Abrahams, P. W. Anderson, D. C. Licciardello, and T. V. Ramakrishnan, "Scaling theory of localization: Absence of quantum diffusion in two dimensions," *Phys. Rev. Lett.*, vol. 42, pp. 673–676, Mar 1979. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.42.673

[65] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. O'Reilly Media, Inc., 2019.

[66] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," 2018, (*Preprint* 1711.11248).

[67] 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, "PyTorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, no. NeurIPS, 2019, (*Preprint* 1912.01703).

[68] K. Kottmann, P. Huembeli, M. Lewenstein, and A. Acín, "Unsupervised phase discovery with deep anomaly detection," *Phys. Rev. Lett.*, vol. 125, p. 170603, Oct 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.125.170603

[69] C. Alexandrou, A. Athenodorou, C. Chrysostomou, and S. Paul, "The critical temperature of the 2D-Ising model through deep learning autoencoders," *Eur. Phys. J. B*, vol. 93, no. 12, p. 226, Dec 2020. [Online]. Available: https://doi.org/10.1140/epjb/e2020-100506-5

[70] W. Hu, R. R. P. Singh, and R. T. Scalettar, "Discovering phases, phase transitions, and crossovers through unsupervised machine learning: A critical examination," *Phys. Rev. E*, vol. 95, p. 062122, Jun 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.95.062122

[71] M. Farina, Y. Nakai, and D. Shih, "Searching for new physics with deep autoencoders," *Phys. Rev. D*, vol. 101, p. 075021, Apr 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevD.101.075021

[72] T. Finke, M. Krämer, A. Morandini, A. Mück, and I. Oleksiyuk, "Autoencoders for unsupervised anomaly detection in high energy physics," *Journal of High Energy Physics*, vol. 2021, no. 6, p. 161, Jun 2021. [Online]. Available: https://doi.org/10.1007/JHEP06(2021)161

[73] V. Šmídl and A. Quinn, *Bayesian Theory.* Springer Berlin Heidelberg, 2006. [Online]. Available: https://doi.org/10.1007/3-540-28820-1_2

[74] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022, (*Preprint* 1312.6114).

[75] D. P. Kingma, "Variational inference & deep learning: A new synthesis," Ph.D. dissertation, University of Amsterdam, 2017.

[76] S. Cheng, F. He, H. Zhang, K.-D. Zhu, and Y. Shi, "Machine learning percolation model," 2021, (*Preprint* 2101.08928).

[77] O. Schenk, M. Bollhöfer, and R. A. Römer, "On Large-Scale Diagonalization Techniques for the Anderson Model of Localization," *SIAM Journal on Scientific Computing*, vol. 28, no. 3, pp. 963–983, 2006. [Online]. Available: https://doi.org/10.1137/050637649

[78] M. Bollhöfer and Y. Notay, "JADAMILU: a software code for computing selected eigenvalues of large sparse symmetric matrices," *Computer Physics Communications*, vol. 177, no. 12, pp. 951–964, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010465507003700

[79] L. J. Vasquez, A. Rodriguez, and R. A. Römer, "Multifractal analysis of the metal-insulator transition in the three-dimensional Anderson model. I. Symmetry relation under typical averaging," *Phys. Rev. B*, vol. 78, p. 195106, Novem 2008. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.78.195106

[80] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman, "Leakage in data mining: Formulation, detection, and avoidance," *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 4, 2012. [Online]. Available: https://doi.org/10.1145/2382577.2382579

[81] S. Kapoor and A. Narayanan, "Leakage and the reproducibility crisis in machine-learning-based science," *Patterns*, p. 100804, 8 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2666389923001599

[82] T. R. Kirkpatrick and D. Belitz, "A new universality class for the metal-insulator transition problem," *J. Phys.: Condens. Matter*, vol. 4, no. 1, p. L37, Jan 1992. [Online]. Available: https://dx.doi.org/10.1088/0953-8984/4/1/009

[83] S. M. Oh, K. Choi, and B. Kahng, "Machine learning approach to percolation transitions: global information," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2023, no. 8, p. 083210, Aug 2023. [Online]. Available: https://dx.doi.org/10.1088/1742-5468/aceef1