DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Classification of Ising Phases for the Purpose of Transfer Learning Using Neural Networks

Johanna Franziska Schinabeck

DEPARTMENT OF INFORMATICS

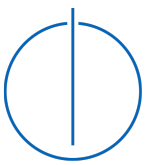
TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Classification of Ising Phases for the Purpose of Transfer Learning Using Neural Networks

Klassifizierung von Ising-Phasen zum Zweck des Transferlernens mithilfe Neuronaler Netze

Author:	Johanna Franziska Schinabeck
Supervisor:	Prof. Dr. Christian Mendl
Advisor:	Irene López Gutiérrez, M.Sc.
Submission Date:	15.08.2021



I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.08.2021 Johanna Franziska Schinabeck

Acknowledgments

At first, and most importantly, I want to thank my advisor, Irene López Gutiérrez, and my supervisor Prof. Dr. Christian Mendl for giving me the opportunity to work on this topic. Particularly I want to thank Irene López Gutiérrez for her excellent supervision and the always friendly atmosphere. Second, I want to thank my family, my dog Lilly, and Lukas, who have endured me during the hard times and always found ways to cheer me up and motivate me. Also, huge thanks to my dearest friends, the Simons, Philip, and Maxi. You brought some light to my daily life, inspired me, and believed in me when I could not. Another person I want to thank is Lukas for giving me physic tutoring and help to understand special notations. Next to that, I want to thank Julia, Miri, and Romy, to name only a few, who listened to me, although they had no idea of what I am talking about. Lastly, thanks to all people, who talked to me, supported me, and got me out of my daily routine. In times of limited social interactions, I am grateful for any social contact, no matter how small.

Abstract

Neural networks are considered as one of the most powerful tools for handling a huge amount of data [1]. Thereby, convolutional neural networks are known for their excellent performance in feature recognition. Those properties solve the problems appearing in many-body quantum systems and establish methods for solving them in numerically inaccessible regimes [4, 22]. Additionally, researches prove that "initializing with transferred features" is able to improve the performance of neural networks [25]. In this work, we want to combine both approaches and analyze the effect of transfer learning on neural-network quantum states. Therefore, we create neural networks for classifying representations of 1D and 2D Transverse-field Ising models into their Ising phases. With these pretrained weights as initial values, the performance of predicting ground state vector entries from their corresponding spin configuration is measured and compared to the random initialized case. It appears, that transfer learning is improving the predictions for the two-dimensional case, but not for 1D Ising models.

Contents

Acknowledgments **iii**

Abstract **iv**

1 Introduction **1**

2 Background **3**
2.1 Transverse-field Ising model 3
2.2 Neural network 5
2.2.1 Fully connected neural network 5
2.2.2 Convolutional neural network 8
2.2.3 Transfer learning 9

3 Neural network for the classification of ground states **11**
3.1 Dataset 11
3.1.1 Distribution of h 11
3.1.2 Number of data-samples 13
3.1.3 Analysis 13
3.2 Neural network architecture 14
3.3 Analysis 16

4 Neural network for the classification of spin configurations **19**
4.1 Dataset 19
4.1.1 Distribution of h 20
4.1.2 Number of samples 21
4.1.3 Equal spin configurations 21
4.1.4 Analysis 22
4.2 2D 25
4.2.1 Neural network architecture 25
4.2.2 Analysis 26
4.3 1D 33
4.3.1 Neural network architecture 33
4.3.2 Analysis 34

5 Transfer learning **36**
5.1 2D 36
5.1.1 Neural network architecture 37
5.1.2 Analysis 37
5.2 1D 44
5.2.1 Neural network architecture 44
5.2.2 Analysis 44

6 Summary **47**
6.1 2D 47
6.2 1D 49

7 Conclusion **50**

List of Figures **51**

List of Tables **53**

Bibliography **54**

1 Introduction

The simulation of strongly interacting many-body quantum systems is limited by the "curse of dimensionality" [5]. Here, it says that the state space grows exponentially with the number of particles. 1D models with 6 particles therefore have $2^6 = 6.87 \cdot 10^{10}$ states and with 7 particles even $2^7 = 5.63 \cdot 10^{14}$ [6], what leads to a general number of 2^N spin configurations, with N denoting the total number of spins in the model. Fortunately, neural networks are known for handling a huge amount of data and have shown "the greatest potential in the study of complicated physical phenomena such as quantum physics" [8, 15]. Therefore, new research directions covering the neural-network quantum states are widely explored and give new insights to many-body quantum systems [18, 26].

In this thesis, we want to determine if transfer learning has a positive impact on such neural networks. In concrete terms, we create a neural network, which predicts entries of the ground state vector for a given spin configuration. Hereby, we consider the Transverse-field Ising model for two dimensions and briefly discuss the one-dimensional case. In order to be able to determine the influence of transferred weights on the Ground State Predictor, two classifiers are created. These should learn features from the representation of the Transverse-field Ising model and accordingly improve the predictions for the ground state. [3, 4, 10, 21]

The structure of the work is as follows. At first, the relevant background is covered in Chapter 2, including the Transverse-field Ising model and neural networks. For the latter, especially convolutional neural networks in Section 2.2.2 and transfer learning in Section 2.2.3 are presented. As mentioned before, the created classifiers should learn special characteristics of the Transverse-field Ising model. We try to archive this by classifying the representations of an Ising model in its phases: ferromagnetic, paramagnetic, and equal. In Chapter 3 we focus on the first, more accurate representation, namely the ground state. In Chapter 4 the second representation with spin configurations is classified. In both Chapters a dataset for the classification task is generated (Section 3.1 and Section 4.1), followed by a convolutional neural network for two-dimensional Ising models (Section 3.2 and Section 4.2.1), which is analyzed in Section 3.3 and Section 4.2.2. For the one-dimensional case, a fully connected neural

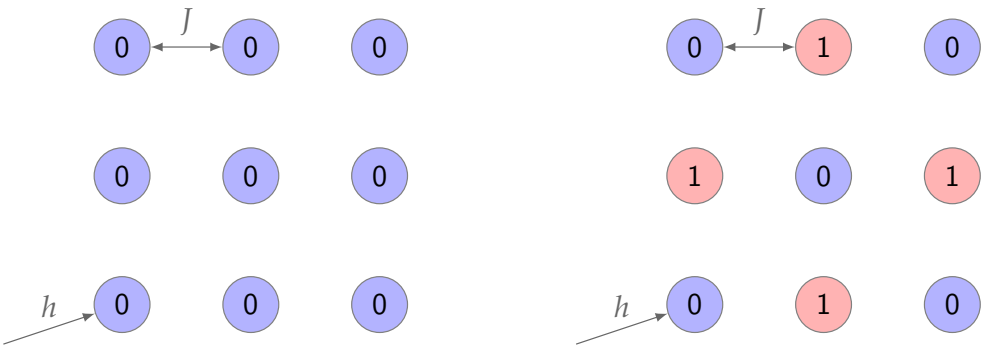
network with spin configurations as input is presented and analyzed in Section 4.3. In the last Chapter 5, the trained weights of the previous Chapters are inserted into the target network. For the 2D Ising model, the convolutional neural network of the target network is explained in Section 5.1.1, and in Section 5.1.2 the performance of the network is analyzed with and without transferred weights. In Section 5.2 the same is done for the 1D Ising model. At the end of the paper, we summarize the results in Chapter 6 and conclude with further work.

The whole work is based on implementations in Python with Tensorflow. All computations are performed on an Intel(R) Core(TM) i5-8250U CPU @ 1.80 GHz with 8,00GB RAM and a 64bit architecture.

2 Background

2.1 Transverse-field Ising model

The following is based on [23].



(a) ferromagnetic ($h << J$)

(b) paramagnetic ($h >> J$)

Figure 2.1: Spin configurations for 2D Ising model with three particles per dimension: Here, h represents the strength of the transverse field and J the spin coupling. (a) visualizes a typical ferromagnetic Ising model, whereas (b) shows a typical paramagnetic Ising model.

The Traverse-field Ising model, in the following also denoted as Ising model, is "described as an ensemble of binary spins with coupling interactions in some lattices" [15]. The spins, or particles, of the Ising model, can point up or down represented by 1 or 0. Two examples for 2D Ising models with nine particles are illustrated in Figure 2.1. Depending on the spin coupling J and the strength of the transverse field h, different spin configurations occur. For $h >> J$, the spins try to align in opposite directions, which is called paramagnetic. For $h << J$, the magnetic field tries to forces the spins in the same direction, which results in a ferromagnetic model. Typical examples for ferromagnetic and paramagnetic models can be seen in Figure 2.1. Because each spin can point up or down, there exist $2^{p^{dim}} = 2^N = 2^9 = 512$ possible spin configurations, with p representing the number of particles per dimension, dim the dimension of the

Ising model and N denoting the total number of spins [5].

Although the Figure represents the typical configurations for the different phases, $h << J$ can also result in a different spin configuration and even the configuration as in Figure 2.1b. On the other hand, there is only a very high probability that a spin configuration as in Figure 2.1a has $h << J$. Therefore, for less typical spin configurations, the correlation between h and J becomes more unclear. [16, 22]

The spin configuration is one possibility to describe an Ising model, but as we could determine, it loses some information of the whole system. Another possibility to describe the system is the ground state of the Hamiltonian. The ground state is obtained by taking the eigenvector with the lowest eigenvalue of the Hamiltonian. For the Traverse-field Ising model, the Hamiltonian is defined as:

$$\mathcal{H} = -J \cdot \sum_{<i,j>} \sigma_i^z \cdot \sigma_j^z - h \cdot \sum_i \sigma_i^x, \tag{2.1}$$

where $\sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ and $\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ are the Pauli-matrices. J represents the spin coupling and is defined as 1. The strength of the transverse field is expressed with h and variable through the whole work [22]. This is possible because, for our purpose, the "absolute value of J and h do not matter, [but] their ratio h/J" [22]. In Equation 2.1 $< i, j >$ denotes the set of neighbored spins. It should be noted that the elements on the borders are also considered neighbors. Consequently, for 1D Ising models we consider the spin configuration as a circle and obtain $< i, j >= \{(1,2),(2,3),(3,1)\}$ for three particles. Accordingly, $\sum_{<i,j>} \sigma_i^z \cdot \sigma_j^z$ represents the impact of the neighbor interaction to the model. Each summation term is calculated with the tensor product of σ_i^z, σ_j^z, and implicit Identity matrices $\mathbb{1} \in \mathbb{R}^{2\times 2} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. In concrete terms, we iterate over the number of particles and check if the current index matches i or j. If this is the case, we multiply σ^z to the current result, otherwise $\mathbb{1}$. The same is repeated for $\sum_i \sigma_i^x$, with the difference that σ^x is used and that we only have one matching index. The result of this summation represents the influence of transverse field strength on the model. If we stick to the example above, the Hamiltonian would evaluate to:

$$\begin{aligned}
\mathcal{H} &= -J \cdot \sum_{\{(1,2),(2,3),(3,1)\}} \sigma_i^z \cdot \sigma_j^z - h \cdot \sum_{\{1,2,3\}} \sigma_i^x \\
&= -J \cdot [\sigma_1^z \cdot \sigma_2^z + \sigma_2^z \cdot \sigma_3^z + \sigma_1^z \cdot \sigma_3^z] - h \cdot [\sigma_1^x + \sigma_2^x + \sigma_3^x] \\
&= -J \cdot [(\sigma_1^z \otimes \sigma_2^z \otimes \mathbb{1}) + (\mathbb{1} \otimes \sigma_2^z \otimes \sigma_3^z) + (\sigma_1^z \otimes \mathbb{1} \otimes \sigma_3^z)] \\
&\quad - h \cdot [(\sigma_1^x \otimes \mathbb{1} \otimes \mathbb{1}) + (\mathbb{1} \otimes \sigma_2^x \otimes \mathbb{1}) + (\mathbb{1} \otimes \mathbb{1} \otimes \sigma_3^x)],
\end{aligned}$$

using the tensor product $\otimes$.

2.2 Neural network

The following Section 2.2 is based on [9, 24].

Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years. It has turned out to be very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, business, and government. The following Section discusses the origin of neural networks, their structure, and special cases like convolutional neural networks or transfer learning. [5, 14, 15]

2.2.1 Fully connected neural network

The following is based on [13, 17].

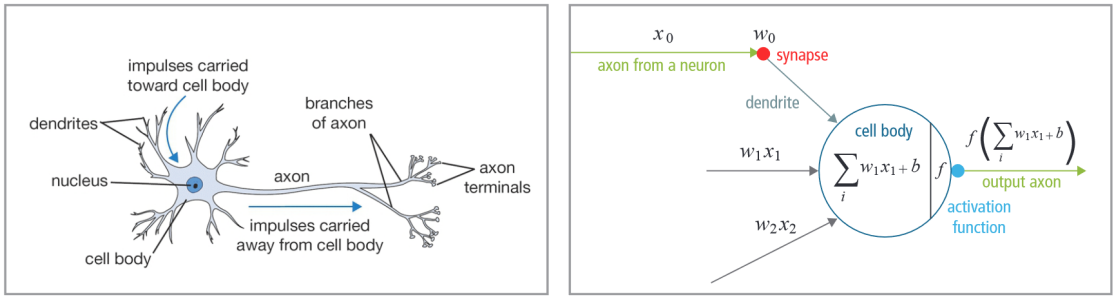


Figure 2.2: Biological neuron (left) and neural network neuron (right) [13]

Neural networks are inspired by the biological neural system, which is able to filter inputs and obtain information. Thereby, the basic computational unit is a neuron, which is interconnected by synapses. The human nervous system consists of approximately 86 billion neurons and 10^{14} to 10^{15} synapses. Figure 2.2 (left) shows the general structure of such a neuron. After receiving input signals of its dendrites, it produces output signals along its axons. Each axon can branch out and "connects via synapses to dendrites of other neurons" [13]. If the summed dendrites are above a certain threshold, the cell body fires signals through the axons. This principle was adopted for neural networks, which is illustrated in Figure 2.2 (right). As in the original neuron, the dendrites ($x_i w_i$) carry the information to the cell body, where they are summed. The dendrite consists

of the signal traveling along the axons (x_i) and the synaptic strength at the synapse (w_i). The idea behind the synaptic strength is that they "are learnable and control the strength of influence [...] of one neuron on another" [13]. After passing the resulting sum to an activation function, it is sent among the axons and input to other neurons. All in all, "each neuron calculates the dot product of inputs and weights, adds the bias, and applies non-linearity [f] as a trigger function" [11] which results in the Equation:

$$f((\sum_i x_i w_i) + b), \tag{2.2}$$

with i denoting the number of dendrites to the neuron.

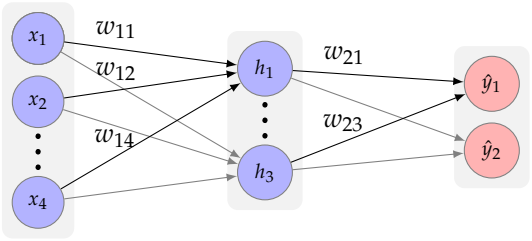


Figure 2.3: Fully connected neural network: w_{ji} is denoting the weights of the network. The bias is implicit for every neuron and not depicted in the Figure.

If one combines those neurons to layers and series those layers, a fully connected neural network, as visualized in Figure 2.3, can be built. Each circle represents a neuron and each arrow to the neuron the dendrites. Inside each neuron, the dendrites are summed, added with the individual bias, and input to the activation function. This output is then input to the neurons of the next layer. In concrete terms, the neurons h_1 and $\hat{y}_1$ evaluate to

$$\begin{aligned} h_1 &= f((\sum_{i=1}^{4} x_i w_{1i}) + b_{11}) \text{ and} \\ \hat{y}_1 &= f((\sum_{i=1}^{3} h_i w_{2i}) + b_{21}), \end{aligned} \tag{2.3}$$

with b_{11}, b_{21} the corresponding bias and f denoting the activation function.

There exist many different activation functions, which can be chosen for f and have different impacts on the final model. Two common choices are ReLU and Softmax, which are used later for the networks. The ReLU or Rectified Linear Unit activation function, which is calculated by $f(x) = max(0, x)$, is shown in Figure 2.4. As one can

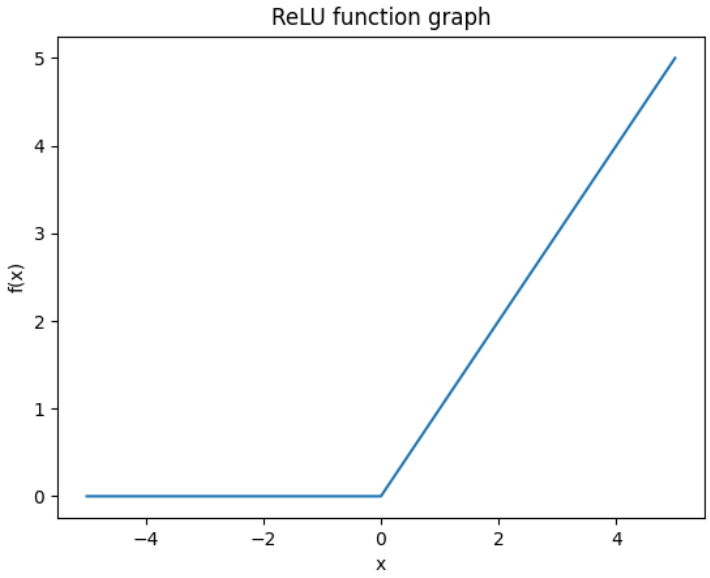


Figure 2.4: ReLU activation function: $f(x) = max(0, x)$

see, the domain of the function is $\mathbb{R}$ and the value set $[0, \infty)$. Therefore, the activation function is applied to each layer output individually. The reason for the success of this activation function is the resulting faster training [11].

The Softmax activation function, on the other hand, is calculated by $f(x) = \frac{e^x}{\sum_i e^{x_i}}$. Its domain is the same as for ReLU, but its output values are in $[0, 1)$ and sum up to 1. For this reason, the output values can be interpreted as a probability distribution, and consequently, Softmax is often used for the output layer of classification networks. The goal of classifications is to divide the input in one of n given classes. Accordingly, the output layer consists of n neurons, each representing a class. If this output layer has Softmax as an activation function, the result of each neuron is a value between 0 and 1, which represents the probability of being in this concrete class. Hence, the input is classified to the class with the highest probability. Alternatively, the network can simply predict a real-valued number, which is then called a regression network. In this case, the last layer does not need an activation function, because the output is already real-valued.

The goal of a network, in general, is to "transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level" to recognize a pattern in the input and predict the right class or value [14].

Thereby, the neurons should extract and learn features. This learning is done during the training process by adjusting the learnable weights w_{ji}. Therefore, a labeled dataset - meaning a dataset with the corresponding class in the case of a classification network or the real value in the case of a regression network - is needed. During training, the network produces an output for each input, which is compared to the label. With an objective function, this error can be measured and is minimized by adjusting the weights with stochastic gradient descent during backpropagation [14]. During this training process, the network is validated with the validation set, to verify the current performance. This procedure of training and validation is continued, till the number of epochs is reached, or in the case of EarlyStopping, the validation loss is getting worse. The latter is an established technique to gain a network, which generalizes the data very well.

In the end, the trained model should respond correctly to the given input. This is measured on unseen data samples from the so-called test set, which "test the generalization ability of the machine - its ability to produce sensible answers on new inputs that it has never seen during training" [14].

2.2.2 Convolutional neural network

The following Section, including its Subsections, is based on [1, 2, 7, 11, 12, 19]
A convolutional neural network, also called CNN, consists of neurons with learnable weights and biases and hence is "a special case of the neural network" described in Section 2.2 [11]. A classical CNN consists of a feature extraction zone and an inference zone. As the name says, the feature extraction zone extracts features from the input. This happens typically in a hierarchical manner, by extracting low-level features at first and combine them by the higher layers. Hereby, the key aspect is that the weights and biases "are learned from data using a general-purpose learning procedure" [14]. This zone often consists of convolutional and pooling layers, which are explained in Section 2.2.2. In the inference zone, the extracted features are usually passed through a fully connected neural network and produce a final result.

Convolution layers

The characteristic operation of convolutional neural networks is convolution. Each convolution has a filter K of size $k \times k$, with k denoting the specified kernel size. During a convolution, the particular filter slides across the width and height of "the input while performing the sum of an element-wise multiplication between the filter values and the corresponding Section of the input" [2].

This procedure is shown in Figure 2.5. In a convolution layer, more than only one kernel can be applied to extract more features of the input. During the backward pass, these kernels are learned and updated. Thereby, the convolution learns to "detect local conjunctions of features from the previous layers" and learns new features [11, 12, 14].

Additionally to the kernel size, one can specify the stride and padding for each convolution. The stride indicates the step size of the filter. In the case of a stride of two, one would jump two entries instead of just one. [12]

For the padding, usually zero-padding, 0 is added on each border before performing the convolution. Through this, not only do the border entries get more included in the calculation but also the output size can be forced to be equal to the input size.

Pooling layers

A convolution layer is usually followed by a pooling layer, which merges similar features and therefore performs feature extraction. This has the positive effect that the features are more robust. There exist two common pooling operations Max pooling and Average pooling. For both, two-dimensional non-overlapping spaces of the input are picked and the max or average values are taken. In Figure 2.6 this operation is illustrated for both pooling operations. [11, 14]

2.2.3 Transfer learning

Generally, neural networks work with the assumption that training and testing data are from the same distribution. This causes the problem that for changing distributions, new models with new training data have to be created. But in many cases collecting an enormous amount of training data is very expensive or not possible. With transfer learning, it is possible to reduce the amount of training data by using pretrained weights from a base class. The intuition behind using these weights is that the learned weights can be relevant for the target task. Therefore, the learned features should be suitable for both the base and the target tasks to gain any improvement. In practice, this means that a base network with a large dataset is trained with randomly initialized weights. Afterwards, these weights are copied to a target network, which usually has a smaller dataset. In the end, the target network is trained with the dataset for the specific task. Due to transfer learning, this target network has usually lower training time and additionally achieves better results than with random initialization. [20, 25, 26]

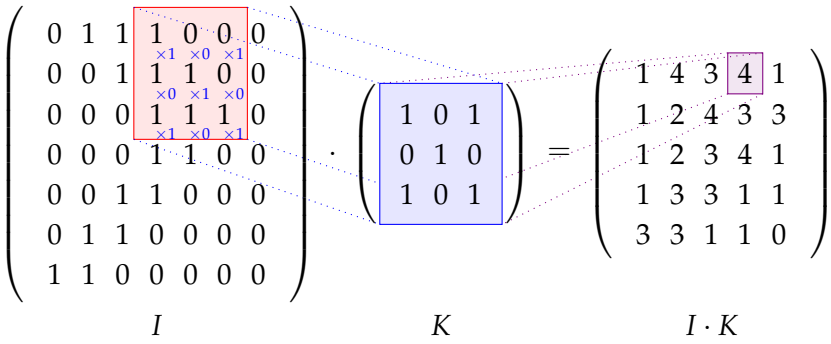


Figure 2.5: Convolution on input I with filter K: Each entry of the output is calculated by summing up the element-wise multiplied values. To compute the whole output matrix, the kernel slides from left to right and top to bottom across the picture.

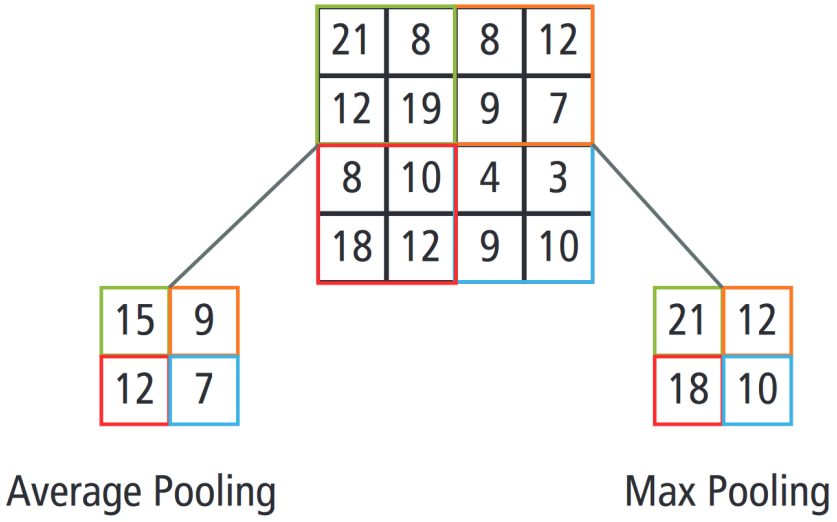


Figure 2.6: Average and Max Pooling [11]: The pooling operation takes the average or the maximum value from a 2×2 section of the input

3 Neural network for the classification of ground states

This Chapter covers the creation and training of our first neural network. It classifies the ground state of an Ising model in one of the three classes: paramagnetic, ferromagnetic, and equal. The first class contains all paramagnetic Ising models and thereby fulfill $h >> J$, so, in our case, $h >> 1$. In the second class, all ferromagnetic models with $h << J = 1$ are contained and for the third, both, J and h, are equal ($h = J = 1$). In the following, these classes are also called labels. For this, we first present in Section 3.1 the details about the created dataset afterward, the model structure is explained in Section 3.2 and analyzed in Section 3.3. Later, in Chapter 5, these trained weights are transferred to the target network and analyzed if they gain any improvement in the ground state prediction.

3.1 Dataset

To create a neural network that classifies the ground state, first, a dataset is created. This dataset depends on the number of particles and the dimension of the Ising model. First of all, we choose a value for h and compute the Hamiltonian with Equation 2.1. From the resulting matrix, the ground state of the Hamiltonian is calculated. This is done by taking the eigenvector of the lowest eigenvalue. The ground state value, as well as h and the corresponding label, are stored in the dataset. We choose HDF5 as data format because it allows very efficient storing and accessing of NumPy arrays.

3.1.1 Distribution of h

The most influential decision for the dataset creation is the distribution of h. By this, the inputs for the network and consequentially the learned features are defined. This results in a realistic h distribution that spreads the h values evenly is desired. Therefore, we choose h such that the paramagnetic and ferromagnetic cases appear equally often. With this property, the classifier is trained on the same amount of data for both labels and should distinguish them only by their ground states. In contrast to these labels, the equal label consists only of one value. Due to that, and the fact that $h = J$ is only a

Fraction	Probability	Label	h value scope
$\frac{16}{33}$	48%	ferromagnetic	$[0, 1)$
$\frac{16}{33}$	48%	paramagnetic	$(1, \infty)$
$\frac{1}{33}$	3%	equal	$\{1\}$

Table 3.1: Distribution of h for the Ground State Dataset: This Table shows the distribution of the h value, as well as the distribution of the labels in the dataset. Inside each scope the h value is evenly distributed. Here, "Fraction" and "Probability" denote what portion of the dataset has an h value from the scope, specified in the last column. "Label" indicates the corresponding label of these datasamples.

Fraction	Probability	Label	h value scope
$\frac{1}{33}$	3%	ferromagnetic	$\{0\}$
$\frac{15}{33}$	45%	ferromagnetic	$(0, 1)$
$\frac{10}{33}$	30%	paramagnetic	$(1, m]$
$\frac{6}{33}$	18%	paramagnetic	(m, ∞)
$\frac{1}{33}$	3%	equal	$\{1\}$

Table 3.2: Detailed distribution of h for the Ground State Dataset: This Table is a specialization of Table 3.1. To get more data in the critical area around 1, the introduced parameter m splits the h values of the paramagnetic case. As before, inside each scope the h value is evenly distributed. Later, for the Spin Configuration Classifier, we refer to this distribution with h1.

special case, we choose fewer data samples with $h = J = 1$. In concrete terms, $\frac{1}{33} \approx 3\%$ of the dataset is labeled with equal. The remaining $\frac{32}{33} \approx 97\%$ of the dataset are split between the ferromagnetic and paramagnetic Ising phase with $\frac{16}{33} \approx 48\%$ each. This distribution is shown in Table 3.1, which also can be seen as the distribution of the corresponding labels. For each label, the probability of having a concrete h value of this scope is evenly distributed. This leads to two problems. First, the probability of having $h = 0$ is very low, and therefore the classifier may never see pure ferromagnetic models. To solve this problem, we split the amount of the ferromagnetic labeled data and assign $h = 0$ to $\frac{1}{33} \approx 3\%$ and $h \in (0, 1)$ for the remaining values. Second, if the h values are evenly distributed in the domain $(1, \infty)$ only a low number of h values are close to 1. But those values are the most critical ones because the transition to the other labels happens in this area. Therefore, we want to have more data in this critical area than close to infinity. To realize this, we introduce a new variable m, which splits the interval for the paramagnetic case in $(1, m]$ and (m, ∞). Now, we assign a value in the

first interval to 30% and a value of the second interval to 18%. For more values close to 1, it is important to choose a small m. We set $m = 50$, which results in the final h distribution as visualized in Table 3.2.

3.1.2 Number of data-samples

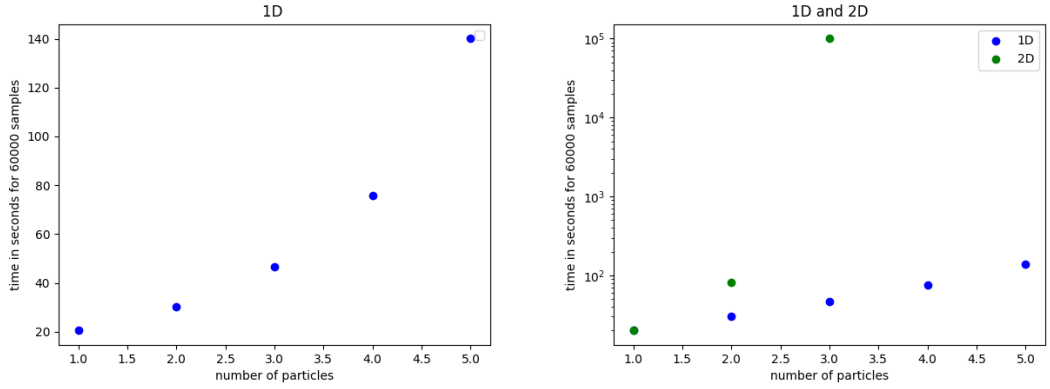
A second important decision is the number of data samples. The more different data the neural networks get, the better it should be able to distinguish between the Ising phases. But because of the high computation time for calculating a Hamiltonian, which is analyzed in Section 3.1.3, too big datasets are not feasible. Therefore, we choose a trade-off between many samples, and an acceptable computation time, which resulted in $60,000$ data samples. It should be noted that for datasets with more than two dimensions and/or more than three particles per dimension, we recommend fewer data samples because of the exponential computation time shown in Figure 3.1. For analyzing the performance of the network, we evaluate it with a separate testing set, which represents 10% of the whole dataset. Hence, $6,000$ samples are used for the testing dataset, and the remaining $54,000$ samples for the training and validation.

3.1.3 Analysis

Dimension	Particles per dimension	Paramagnetic	Ferromagnetic	Equal
1	1	26258	26058	1684
	2	25912	26411	1677
	3	26240	26034	1726
	4	26080	26231	1689
	5	26200	26077	1723
2	1	26229	26113	1658
	2	26219	26150	1631
	3	26250	26133	1617

Table 3.3: Label distribution of the training Ground State Dataset: This Table visualizes the distribution of the labels in the Ground State Dataset depending on the dimension and particles.

To analyze the created datasets and especially its h distribution, Table **??** shows the number of data samples for each label. Just as we expected, the number of paramagnetic and ferromagnetic labeled data is almost evenly distributed, and the equal label exists less often. Additionally, it is a satisfactory observation that the number of samples for



(a) (b)

Figure 3.1: Run time for the creation of the Ground State Dataset: The plots shows the exponential growth of the runtime, depending on the number of dimensions and particles per dimension.

one label is independent of the dimensions and numbers of particles. Those results confirm that the dataset meets the required characteristics.

Next, we analyze the runtime of the dataset creation. As mentioned in Section 3.1.2, the computation time is very high due to the very expensive Hamiltonian calculation. Figure 3.1 illustrates this phenomena for 1D and 2D Ising models. The Figure shows very accurately that with the rising number of particles or dimensions, also the Hamiltonian calculation and consequently, the run time is increasing. This exponential growth is also known as the "curse of dimensionality".

3.2 Neural network architecture

In this Section, we cover the structure of the Ground State Classifier. To learn patterns in the ground state, we create the convolutional neural network illustrated in Figure 3.2. At first, we input a matrix to the convolution with 32 filters and stride 1. It uses the ReLU activation function, as well as zero padding. The kernel size can be chosen concerning the dataset and is specified later. Next, a MaxPool operation is performed, which should extract features learned by the convolution. It follows a dropout layer to add some regularization to the model and prevent overfitting. The rate of the dropout is also chosen later, depending on the dataset. The resulting tensor is flattened and connected to a fully connected layer with 128 neurons and the ReLU activation function.

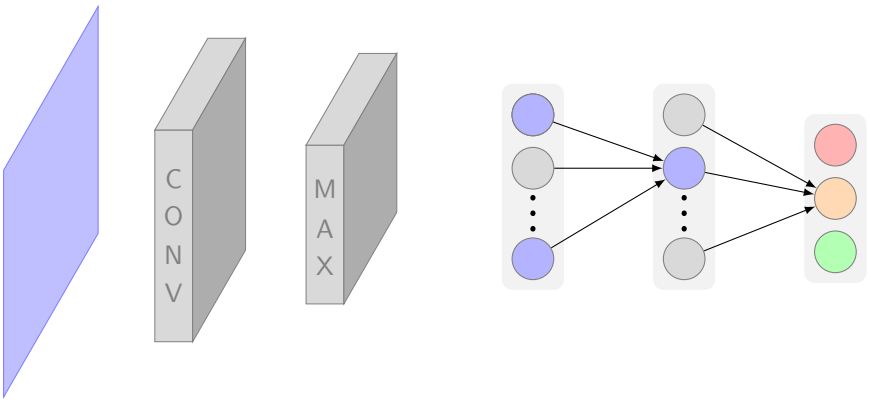


Figure 3.2: Model for the 2D Ground State Classifier: The elements of the convolutional neural network are described from left to right. The first blue rectangle represents the 2D input matrix of reshaped ground states. It it followed by a convolutional layer and a MaxPool layer. The 2D output is flattened, which is represented with the first fully connected layer. The last two layers, also fully connected layers, have 128 and three neurons. The different colors of the last layer should represent the three distinct labels into which the network classifies. The grey neurons should represent the dropout of the model.

After adding another dropout layer with the same rate as before, a fully connected layer with three neurons and the Softmax activation function is added. Here, the classification of the Ising phases takes place. The whole model uses the Categorical Cross-entropy Loss function, the optimizer Adam with a learning rate of 0.0005, and an epoch of 100. To only use the best-trained model and prevent overfitting, we use EarlyStopping and safe the best model. As mentioned in Section 4.1.2, the split for training, validation and testing of the whole dataset is $(80\%, 10\%, 10\%)$.

The format of the ground state, which is a vector, and the expected input for the classifier do not match. Therefore, we reshape the dataset into a matrix while preprocessing the data. It should be noted that the shape of every ground state is a vector, independent of the dimension and number of parameters. This is why this network is useable for any dataset. In our case, we use the dataset of 2D Ising models with three particles per dimension. More particles are not possible in our case because the allocated memory for the Hamiltonian is so high that it leads to a memory error on the used device.

Nr	Layer	Shape	Parameters
1	Conv2D	(None, 16, 32, 32)	544
2	MaxPooling2D	(None, 8, 16, 32)	0
3	Dropout	(None, 8, 16, 32)	0
4	Flatten	(None, 4096)	0
5	Dense	(None, 128)	524416
6	Dropout	(None, 128)	0
7	Dense	(None, 3)	387

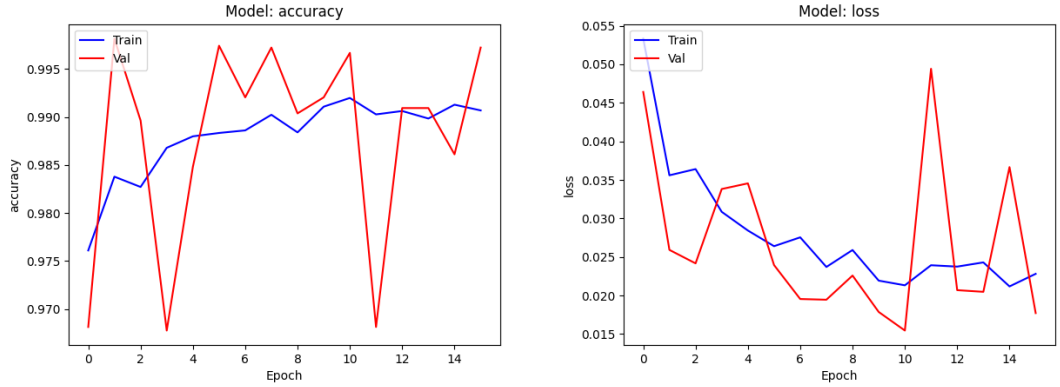
Table 3.4: Detailed Model for 2D Ground State Classifier: This Table is a specialization of Figure 3.2 for three particles per dimension. It lists the concrete layer, their output shape, and learnable parameters.

The concrete dataset with a total of nine particles consists of ground states of length 512 and is reshaped into a matrix of shape $(16, 32)$. With this additional information, the neural network can be further specified, which can be seen in Table 3.4. The Table reflects the explained network structure and additionally shows the output shape after each operation. In total, the network has $525,347$ trainable parameters, where the first 544 parameters are transferred to the target network.

3.3 Analysis

Lastly, the resulting neural network is validated and analyzed. In Section 3.2 the whole network is specified, except for the kernel size and dropout rate. Those parameters can be chosen individually for the concrete dataset. To find the best configuration, we implemented a grid search algorithm to test the combination of different kernel sizes and dropout rates. It appears that there exist a few combinations, which perform almost equal. We decided to choose a kernel size of 3 and a dropout rate of 0 because these parameters achieve accurate results for this classifier and the Spin Configuration Classifier. Using the same parameters enables better comparisons of the model, especially in terms of transfer learning.

After training the model with the set parameters, we receive the accuracy and loss curves shown in Figure 3.3. At first sight, the validation curve is in both plots very fluctuating, whereas the training curve is comparable smooth. If we take a closer look at the scale of the plot, it appears that the interval is in the range of the second decimal digit. Therefore, minor differences look like huge fluctuations. Further, the loss and accuracy during training and validation are very good. This can be validated



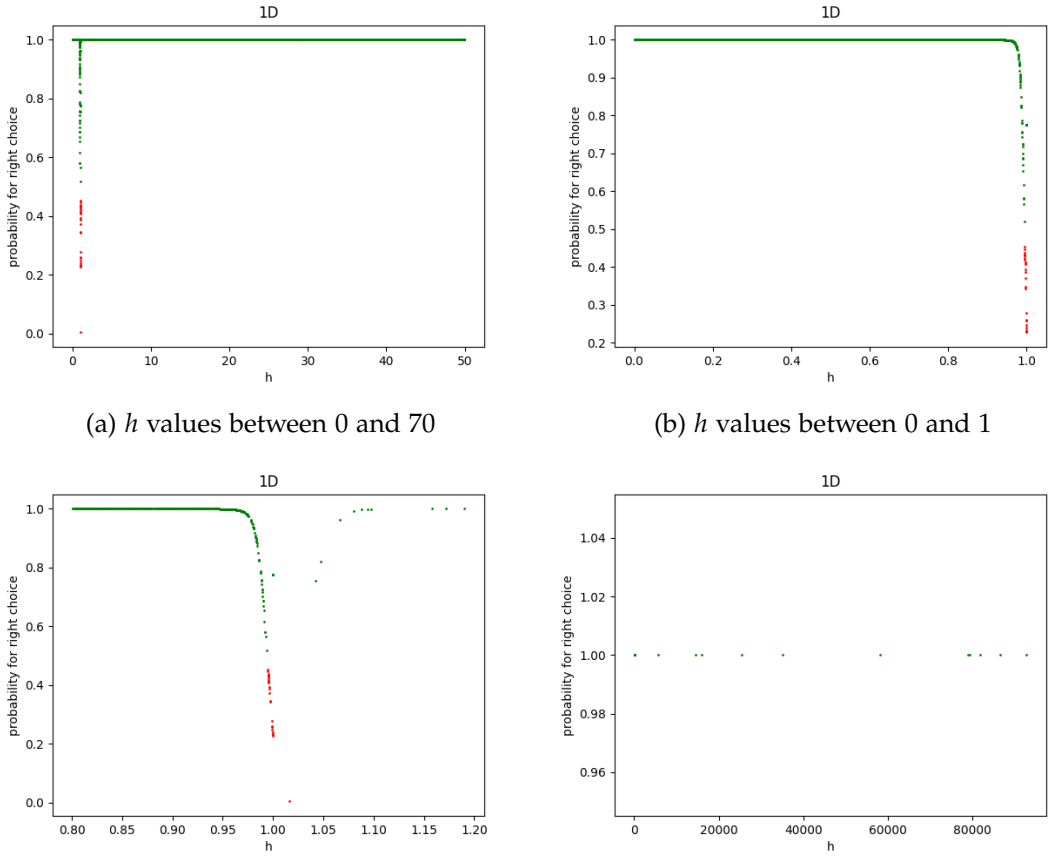
(a) accuracy (b) loss

Figure 3.3: Loss and accuracy curve for 2D Ground State Classifier: The curves are produced during training of the Ground State Classifier with no dropout and kernel size 3. The classifier was trained with a dataset of 2D Ising models with three particles per dimension.

by testing the model with the created testing dataset. With a testing loss of 0.015 and a testing accuracy of 0.996, the classifications are extremely precise, and it seems that the convolution found features in the input.

To get a better feeling for the predictions of the classifier, we plot the probability distribution after the Softmax operation. Therefore, we input ground states of different h values to get the probability of the predicted label and plot the probability of the correct label. If the classifier is predicting the right label, we illustrate this with green, otherwise with a red dot. The related plots can be seen in Figure 3.4 for different h ranges. It is striking that the classifier predicts only in the critical area around 1 wrong labels. Here, the certainty of the prediction is also smaller. In all other cases, the classifier predicts to almost 100% the right label.

To conclude, in this Chapter, we created a dataset with evenly distributed h values for each label. These ground states were fed to the convolutional neural network in Figure 3.2 and resulted in a network with 99% accuracy. This gives hope that the convolution has learned features from the ground state and that this leads to good results during transfer learning.



(a) h values between 0 and 70

(b) h values between 0 and 1

(c) h values between 0.8 and 1.2

(d) h values between 40 and 100000

Figure 3.4: Confidence of h for the trained Ground State Classifier: The Figures show the predicted probability of the correct label depending on the h value. Therefore, the output value of the network on the position of the correct label is plotted in the figure. Here, red dots represent wrong and green right classified samples.

4 Neural network for the classification of spin configurations

As in Chapter 3, this Chapter covers the creation and training for classification neural network. Analog, the classes for the classifier are paramagnetic for $h >> J = 1$, ferromagnetic for $h << J = 1$ and equal for $h = J = 1$. The structure and even the network architecture are strongly based on Chapter 3, which enables a better comparison of the networks, especially concerning transfer learning. The main difference is the dataset of the model, which consists of the concrete spin configuration in contrast to the ground state. Hence, the creation of the new dataset is explained in Section 4.1, followed by the general network structure and the analyzation of concrete networks in Section 4.2.1 and Section 4.2.2. To test transfer learning on 1D Ising models, Section 4.3 creates and analyzes a fully connected neural network for one-dimensional Ising models. The creation of such a network is not able for the Ground State Classifier. More information regarding this can be found in Section 5.2.

4.1 Dataset

At first, we have to create a dataset that can be classified in its Ising phases. This time the dataset does not contain the ground state of the Hamiltonian but a concrete spin configuration. Therefore, we have to calculate the ground state for a concrete h value. After taking the absolute values squared, we obtain the probability distribution d. This distribution is a vector, where each value represents the probability of being in the spin configuration identified with their index. Therefore, the number of spin configurations and the size of the vector d are equal and evaluate to $2^{p^{dim}} = 2^N$, as mentioned in Section 2.1. In case of a 1D model with one particle ($p = N = 1$), d could look like: $(0.4, 0, 6)$. This means that to 40% $< 0 >$ and 60% $< 1 >$ is the corresponding spin configuration for the calculated Hamiltonian. From this distribution, the final spin configurations are sampled. For this purpose, we create random numbers r between 0 and 1, which should represent a probability. If r is smaller than the sum of the previous entries, the corresponding spin configuration for this index is chosen. This means, for $r = 0.5$ the first index does not match, because $r = 0.5 \nleq 0.4$. For the second index we have: $r = 0.5 \leq 0.4 + 0.6 = 1$ and therefore we choose the second index with $< 1 >$ as

corresponding spin configuration. In doing so, we sample a few spin configurations for each Hamiltonian and store them with the corresponding h value, the label, and for debugging purposes also the distribution d. As for the Ground State Dataset, we choose HDF5 as the data format.

4.1.1 Distribution of h

Because there is only a fixed number of input spin configurations for a given dimension and number of particles, there exist a few approaches for the distribution of h, which can lead to good results. We try three different approaches, which are explained in the following.

h3

At first, we try to keep the calculation of h as simple as possible. Hence, we choose only very extreme h values. This means, with a probability of 50% a value in $[0, 0.3]$ and to 50% a value in $[m, \infty]$ is taken. Here, we use the same m as introduced in Section 3.1.1, which represents the lower bound for the very high h values. To stay consistent with the other results, we choose $m = 50$.
With this approach, we hope that for every spin configuration, enough samples are produced and that they can be clearly classified. In the following, we refer to this h distribution with h3.

h2

In h3, no samples with an h value in the critical area are chosen. Additionally, there can exist no samples for the equal label. To solve both problems, we choose to have three ranges. Strictly speaking, values in $[0, 0.3]$, $[0.8, 1.2]$ and $[m, \infty]$. The probability of having an h value in one of the ranges is again evenly distributed. This h distribution is referred as h2.

h1

A third approach is to choose h as we did for the Ground State Classifier in Section 3.1.1, which we call h1. This way, h has values distributed over the whole $\mathbf{R}_+ \cup \{0\}$. With h1, the amount of ferromagnetic and paramagnetic data samples is equal, and less equally labeled data samples appear. This approach has the most realistic h distribution,

wherefore we concentrate on the following on that case.

4.1.2 Number of samples

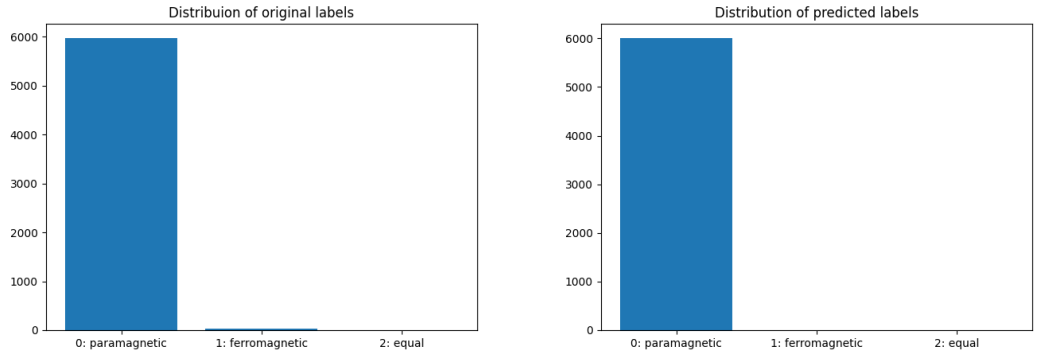
To get comparable results, we create 60.000 samples in total for training, testing and validation set. As we did in Section 3.1.2, this dataset is split into 90% = 54000 training and validation samples and 10% = 6000 testing samples.
As described in Section 4.1, a certain number of samples is derived from one Hamiltonian. Therefore, we have to decide how many samples should be sampled from on Hamiltonian (= num_per_ham) and how many Hamiltonians should be created (= num_ham). The greater num_per_ham, the better the probability distribution d of the Hamiltonian is represented. Consequently, num_ham is very low to fulfill the total number of data samples. But the lower num_ham, the less different h values are represented. Hence, for the training and validation data, we choose num_ham = 60 to have sufficient different h values and num_per_ham = 900 to guarantee that the distribution is well represented. For the testing set, we want to validate our model on many different h values, and as a consequence, we choose a comparatively higher num_ham-value. In this case we choose num_ham = 60 and num_per_ham = 100.

4.1.3 Equal spin configurations

Dimension	Particles per dimension	Paramagnetic	Ferromagnetic	Equal
1	1	30932	21121	1947
	2	34519	17940	1541
	3	39864	12331	1805
	4	40078	11476	2446
	5	43890	7501	2609
2	1	30887	23113	0
	2	43424	10285	291
	3	51245	1193	1562

Table 4.1: Label distribution of the training Equal Spin Configuration Dataset: The Table visualizes the distribution of the labels in the dataset depending on the dimension and particles. In the created dataset, each spin configuration appears equally, whereby we use h1 as h distribution.

Additionally, we try to force the algorithm to generate a dataset where each spin configuration appears equally often. The intention behind this idea is to have the



(a) original

(b) predicted

Figure 4.1: Original and predicted label distribution of the testing Equal Spin Configuration Dataset: (a) visualizes the label distribution for the testing dataset and (b) shows their predicted labels. In the used dataset, each spin configuration appears equally, whereby we use h1 as h distribution.

same number of information for every spin configuration and therefore classify them better. As a calculation method for h, we choose h1 because it is able to represent all h values. After creating the dataset, we analyze the distribution with Table 4.1. It appears that more samples are labeled paramagnetic, especially with increasing particles per dimension. For example, for 2D and three particles per dimension, only 2% of all 54000 data samples are ferromagnetic. This is even less than for the equal label, which should only represent a border case. On the other hand, 95% of the data is labeled paramagnetic. In the one-dimensional case, we have similar results with only 14% ferromagnetic and 81% paramagnetic samples for five particles. With these results, we can assume that the neural network always predicts the paramagnetic label because it appears with an extremely high probability. After creating the network explained in Section 4.2.1 and training it, this hypothesis holds. The input samples contain still a few ferromagnetic labels, as shown in Figure 4.1a. Contrary to this, the predicted label is always paramagnetic, independent of the h value. This can be seen in Figure 4.1b. With this approach, the network is not learning any features related to the spin configuration. Therefore, we do not cover this dataset approach in the following.

4.1.4 Analysis

After building a dataset for each h distribution, the resulting label distribution can be analyzed, which is done in Table 4.2. At first, it is noticeable that only h1 produces spin

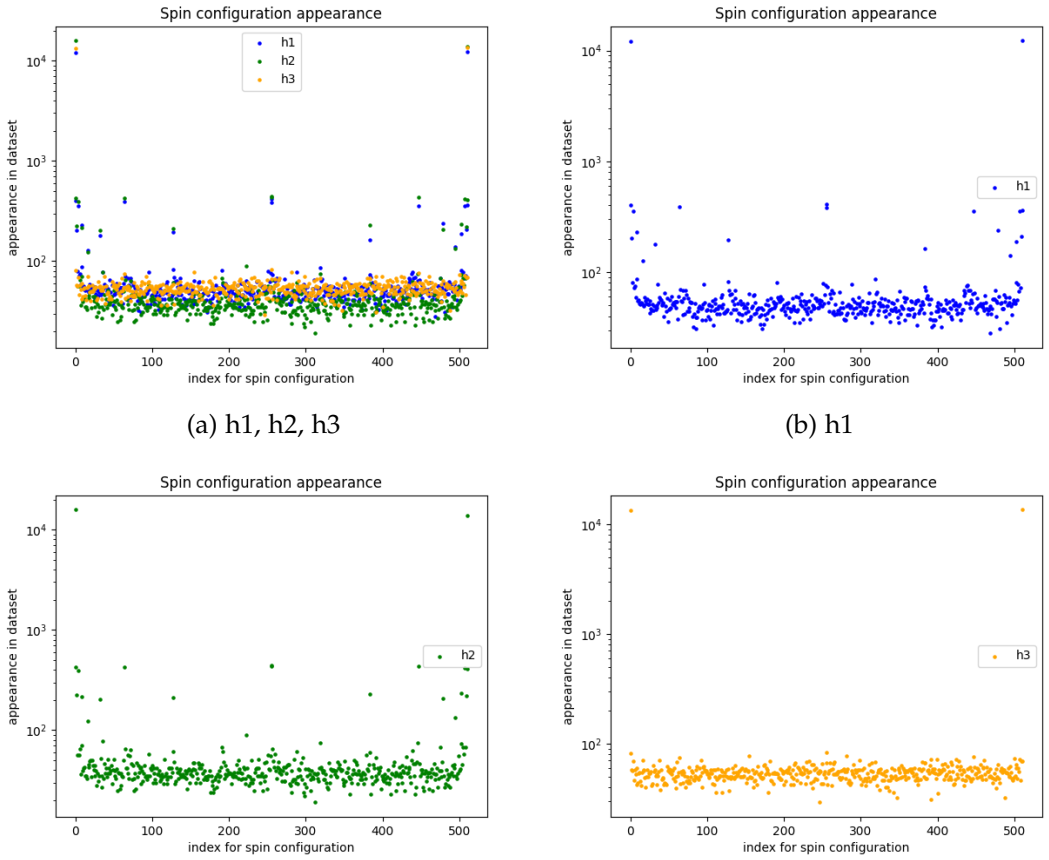
h distribution	Paramagnetic	Ferromagnetic	Equal
h1	25200	27900	900
h2	26100	27900	0
h3	27000	27000	0

Table 4.2: Label distribution of the training Spin Configuration Dataset for h1, h2, and h3: The distribution is for a 2D Ising model with three particles per dimension.

configuration with an equal label. Apparently, the probability for choosing $h = 1$ in h2 is too low that we can assume samples with this label. Besides, every entry distributes the data samples evenly among the paramagnetic and ferromagnetic label, what is beneficial for the classification task.

For a deeper analysis of the resulting datasets, we count the appearance of each spin configuration among the datasets. As one can see in Figure 4.2, each spin configuration exists for each dataset. Therefore, all datasets are trained with all spin configurations and can learn their corresponding Ising phase. Thereby, spin configuration with all spins pointing in one direction is most represented in all datasets. For h3, the remaining spin configurations have all a similar occurrence, as one can see in Figure 4.2d. h1 and h2 result in nearly the same distribution, where all have an equal frequency except for a few samples. Those samples appear at regular intervals and produce a symmetrical pattern. These results validate all approaches for dataset creation. Nevertheless, we are focusing on the dataset h1 because, in contrast to the other datasets, it contains all three labels and additionally enables better comparison to the Ground State Classifier.

Further, we analyze the runtime for the creation of the h1 dataset with Figure 4.3. Because the other h distributions produce comparable runtime results, we do not discuss them here. It should be also mentioned that these values should only be seen as orientation because they differ depending on the concrete machine and implementation. Nevertheless, one can see in Figure 4.3 the exploding runtime for an increasing number of particles for both dimensions. From Figure 4.3b one can also derive that the runtime increases not only for the number of particles but also for increasing dimensions. This characteristic was also noticeable in Figure 3.1. When comparing those two graphs, one can also see that the runtime for higher dimensions and number of particles is lower in Figure 4.3 than in Figure 3.1. This results from the fact, that for the Spin Configuration Dataset only 120 Hamiltonian's are calculated (see Section 4.1.2) compared to the $60,000$ for the Ground State Dataset (see Section 3.1.2). This confirms that the computation of



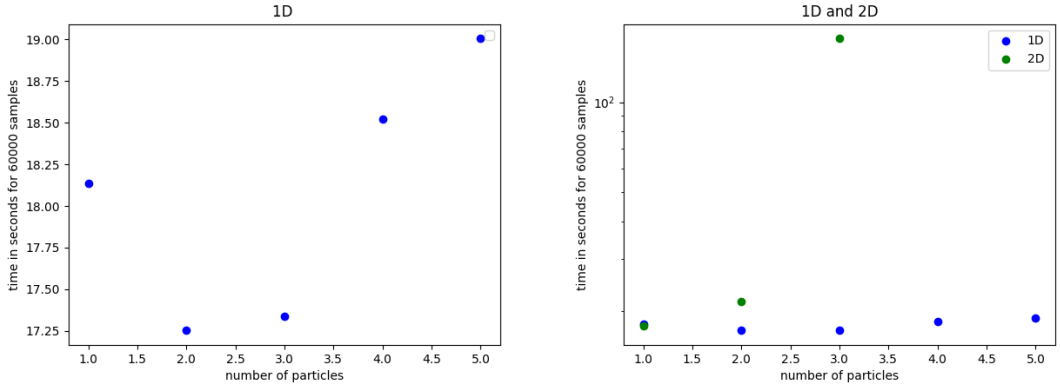
(a) h1, h2, h3

(b) h1

(c) h2

(d) h3

Figure 4.2: Spin configuration appearance in the training Spin Configuration Dataset: The plot illustrates the appearance of each spin configuration for the datasets of Table 4.2. Thereby, the spin configurations on the x-axis are identified with their indices.



(a) (b)

Figure 4.3: Run time for the creation of the Spin Configuration Dataset using h1: The plots show the exponential growth of the runtime, depending on the number of dimensions and particles per dimension.

the Hamiltonian requires most of the runtime.

Summarized, all three distributions produce reasonable datasets, which contain all spin configurations. Because only the h1 dataset contains equal labeled samples, the remaining work focuses on the distribution h1.

4.2 2D

Next, we present the basic structure for the convolutional neural network and analyze it for the previously generated datasets.

4.2.1 Neural network architecture

Because we hope that convolutions can find structures in the given spin configuration, we create a convolutional neural network similar to the one from Section 3.2. As one can see in Figure 4.4, the model contains only one convolution because the model is trained on spin configurations with few spins and therefore less information.

We begin with a convolution that should extract relations between the different spins. This convolution has 32 filters, a stride of 1, and uses the ReLU activation function, as well as zero padding. The kernel size is set to 3, as in Section 3.3. Both must have

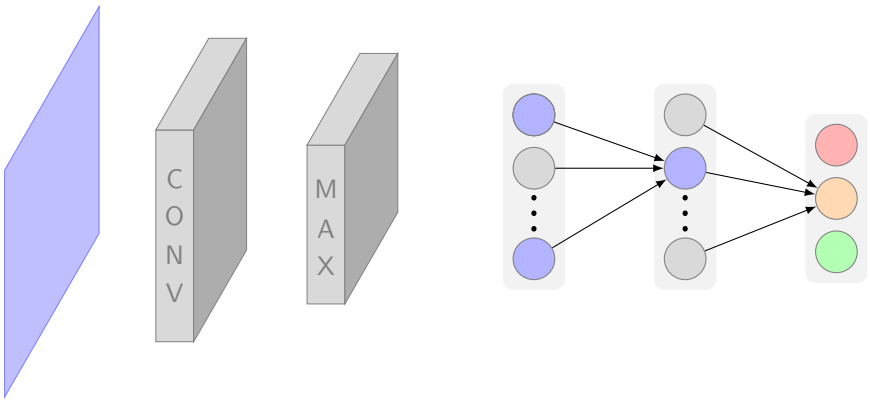


Figure 4.4: Model for the 2D Spin Configuration Classifier: The elements are explained from left to right. The first blue rectangle represents the $(p \times p)$ input matrix of spin configurations. It is followed by a convolutional layer and a MaxPool layer. The 2D output is flattened, which is represented with the first fully connected layer. The last two layers represent fully connected layers, where the last one has exactly three neurons. The different colors of these neurons should represent the three different labels into which the network classifies. The grey neurons of the first two fully connected layers should represent the dropout of the model.

the same kernel size because only then do their filter weights have matching sizes and can be transferred to the same target network. Next, we perform a MaxPool operation, which should extract the learned features. It follows a dropout layer to add some regularization and prevent overfitting. The dropout rate is selected depending on the dataset to find the best customized network. The resulting tensor is flattened and connected to a fully connected layer with 16 neurons and the ReLU activation function. After another dropout layer, we add a fully connected layer with three neurons and a Softmax activation function. This last layer is responsible for the classification in the Ising phases. We choose a learning rate of 0.0002 and keep the remaining settings from the Ground State Classifier in Section 3.2, to enable a good comparison of the resulting models.

4.2.2 Analysis

In this Section, the network of Figure 4.4 is analyzed for the different datasets with a respective dropout rate. Additionally, we limit our observation to Ising models with three parameters. As explained in Section 3.2, more parameters are not possible to compute on the used machine. For fewer parameters, the model would results in zero

or one neuron after the MaxPool operation and thereby prevents reasonable further calculation.

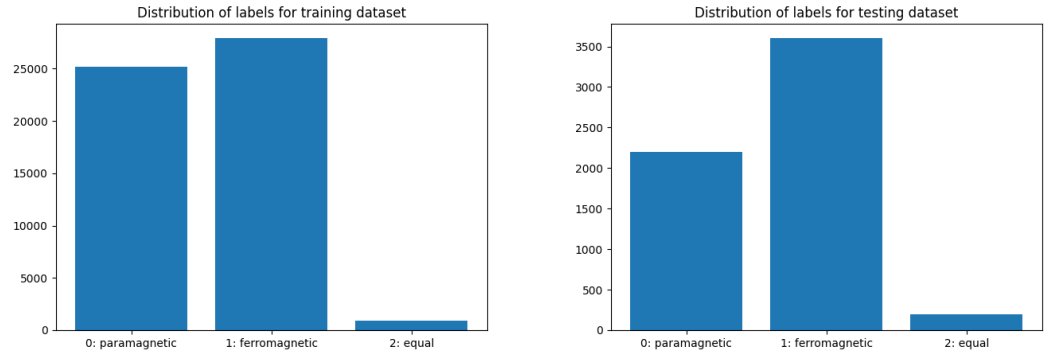
h1

For the first model, we use the dataset created with h1. Figure 4.5 shows the label distribution of the created dataset for the training and testing set. One can see that the distribution between the paramagnetic and ferromagnetic labels is not perfect. Especially for the testing set, the ferromagnetic labels are dominating. This results from the fact that the h values have a stochastic distribution and that the dataset has fewer values than the training dataset. Nevertheless, the values are relatively evenly distributed and should give an accurate representation of the possible spin configurations. The equal labeled data is also represented a few times but, as explained in Section 4.1, not very often.

Nr	Spin	Prediction	Predicted Label
1	[0 0 0 0 0 0 0 0 0 0]	[0.01, 0.97, 0.02]	ferromagnetic
2	[0 0 0 1 0 1 0 1 0]	[1.00, 0.00, 0.00]	paramagnetic
3	[1 0 0 0 0 0 0 0 0 0]	[0.17, 0.79, 0.04]	ferromagnetic
4	[1 0 0 1 0 0 0 0 0]	[0.66, 0.32, 0.02]	paramagnetic

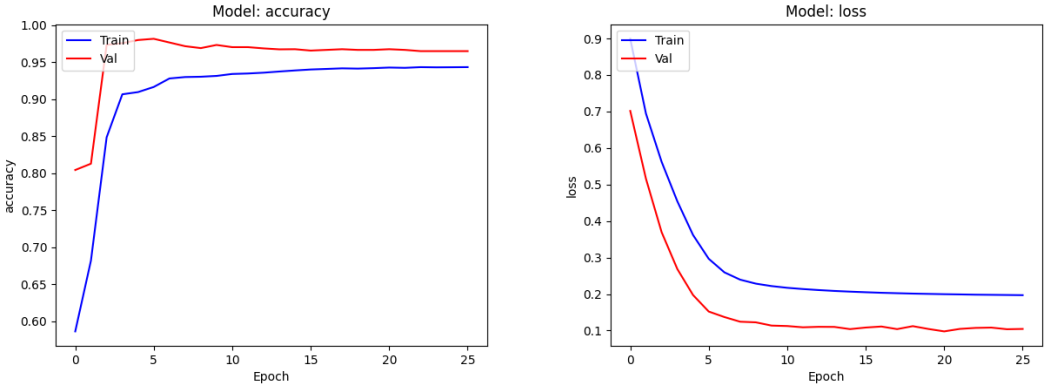
Table 4.3: Exemplary results from the 2D Spin Configuration Classifier (h1): The Table visualizes the output distribution ("Prediction") for given spin configurations.

To find the optimal network for this concrete dataset, we implemented a grid search algorithm, which determines the best dropout rate for the current dataset. In our case, the best configuration is achieved, with a dropout of 0. After train the model with this dropout, we achieve the training and validation curves depicted in Figure 4.6. These curves show a decreasing loss curve as well as very accurate classifications. It is rather untypical for neural networks that the validation curve is better than the training curve in both plots. But it does make sense in our case. We sampled 900 data samples from the same Hamiltonian, and therefore, the validation data is not entirely unseen. Additionally, there exist only a limited number of input data, which plays an important role. It is also visible that the network needed only a few epochs to stop training with precise results. This can also be validated by the testing set, which results in a loss of 0.225 and an accuracy of 0.939. Additionally, we can validate the network with the testing set of the other datasets. For example, the accuracy for the h3 testing set is 0.979 and the loss 0.070, which are even better than for its own testing set. This results from the fact that the h1 dataset has more different h values and therefore classify more



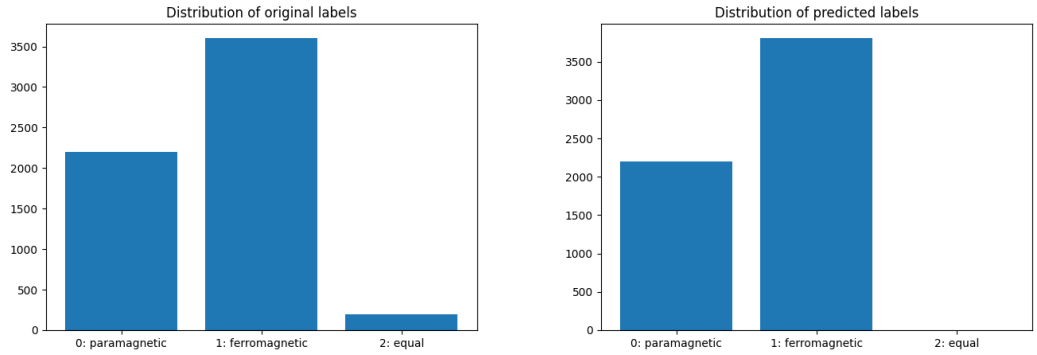
(a) training (b) testing

Figure 4.5: Label distribution of the training and testing Spin Configuration Dataset (h1): The plots illustrate the portion of each label in the dataset. The dataset was generated with the distribution h1.



(a) accuracy (b) loss

Figure 4.6: Loss and accuracy curve for 2D Spin Configuration Classifier using h1: The curves are produced during training of the Spin State Classifier with no dropout and kernel size 3. The classifier was trained with a dataset of 2D Ising models with three particles per dimension and the h1 distribution.



(a) original (b) predicted

Figure 4.7: Original and predicted label distribution of the testing Spin Configuration Dataset using h1: (a) visualizes the label distribution for the testing dataset and (b) shows their predicted labels of the network.

general than h3.

Furthermore, we analyze the network by looking at its classifications. Figure 4.7 shows the distribution of the original and predicted labels of 60,000 data samples. As one can see, the classifier never predicts the equal label. This is plausible because the classifier had less data for this label. On the other hand, $h = J$ is a very special case, which results in the same spin configurations as the paramagnetic and ferromagnetic models. Therefore, it is very hard to distinguish between the three states based on spin configurations. We can also see that the label distributions for the paramagnetic and ferromagnetic samples are rather similar, which sustains the high accuracy.

Lastly, we analyze the network concerning the physical interpretation. Therefore, we have a look at the concrete results of the Spin Configuration Classifier. Because we input spin configurations with a concrete number of spins, the possible inputs are limited. Therefore, we can input each spin configuration and analyze the result from a physical point of view. In Table 4.3 one can see some exemplary results from the classification. Thereby, the second column represents the flattened input spin configuration, the third column the prediction of the model, and the last column interprets the prediction of the model. For the first entry, the spins point all in one direction, and therefore, the model is with a very high probability ferromagnetic. As we could deduce physically, the classifier predicts this label with a very high probability. Analog, for a typical paramagnetic model, the classifier predicts to nearly 100% the correct label. The last

two entries show spin configurations, which could not be assigned clearly to one phase. This is also represented by the probabilities in the prediction. Additionally, the third entry has many spins pointing in the same direction and therefore suggests to be ferromagnetic. This characteristic is also reflected by the classifier. The same holds for the last entry.

To sum it all up, under the circumstances that the classifier can not be perfect, its classification of the spin configurations is accurate. The predictions for the different spin configurations are physically reasonable, and therefore, the model should have learned some features from the spin configuration. In Section 5.1.2 we analyze if these learned weights can produce good results for the target network.

h2

Nr	Spin	Prediction	Predicted Label
1	[0 0 0 0 0 0 0 0 0 0]	[0.20, 0.75, 0.06]	ferromagnetic
2	[0 0 0 1 0 1 0 1 0]	[0.92, 0.07, 0.00]	paramagnetic
3	[1 0 0 0 0 0 0 0 0 0]	[0.72, 0.26, 0.02]	paramagnetic
4	[1 0 0 1 0 0 0 0 0]	[0.83, 0.16, 0.01]	paramagnetic

Table 4.4: Exemplary results from the 2D Spin Configuration Classifier (h2): The Table visualizes the output distribution ("Prediction") for given spin configurations.

For h2, a dropout of 0.2 produces the highest accuracy. The resulting training curves, which are shown in Figure 4.8, are insufficient. On the one hand, the learning curves are worse than for h1. On the other hand, with an accuracy of 0.863 and a loss of 0.381 the network classifies less accurately. This is unexpected because the dataset does not contain any equal labeled data samples and consequently has a better chance to predict the correct label.

Nevertheless, if one looks at the output for concrete spin configuration, which is shown in Table 4.4, the network produces reasonable results, although with a lower certainty as in Table 4.3. As explained in h1 under Section 4.2.2, the third entry is physically classified to the ferromagnetic phase. However, this classifier predicts the paramagnetic label. Except for this, both networks classify correct and to the same class, which is validating both networks.

In conclusion, the network does produce mostly reasonable results but with a lower certainty and also some wrong classifications. All in all, the network is worse than the

classifier trained with the h1 dataset.

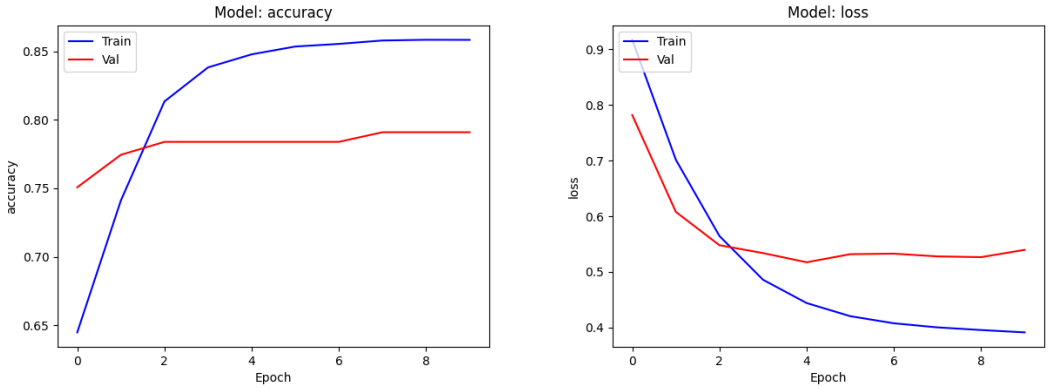
h3

Nr	Spin	Prediction	Predicted Label
1	[0 0 0 0 0 0 0 0 0]	[0.00, 1.00, 0.00]	ferromagnetic
2	[0 0 0 1 0 1 0 1 0]	[1.00, 0.00, 0.00]	paramagnetic
3	[1 0 0 0 0 0 0 0 0]	[0.76, 0.24, 0.00]	paramagnetic
4	[1 0 0 1 0 0 0 0 0]	[1.00, 0.00, 0.00]	paramagnetic

Table 4.5: Exemplary results from the 2D Spin Configuration Classifier (h3): The Table visualizes the output distribution ("Prediction") for given spin configurations.

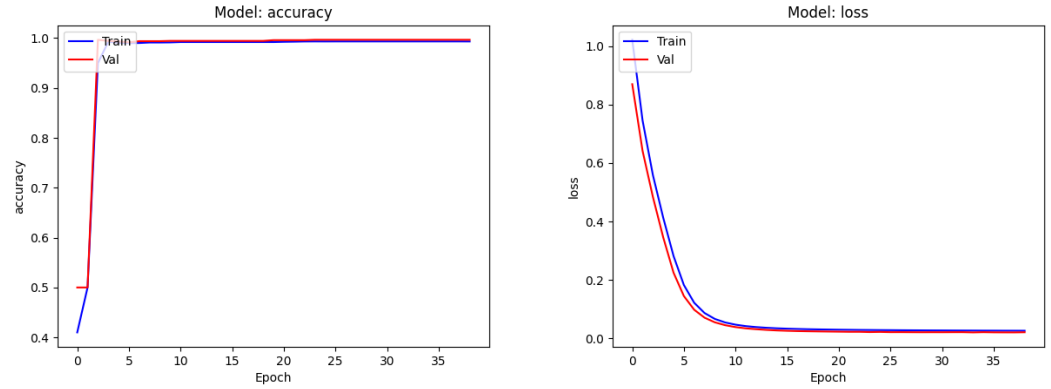
For the third dataset, the optimal dropout rate is again 0. The learning curves, plotted in Figure 4.9, are equal for training and validation, which indicates that the network is quite sure of its decisions. This can be also confirmed with a testing loss of 0.029 and an accuracy of 0.992. The network does not only produce very accurate classification but also physically reasonable distributions. This can be seen in Table 4.5, which visualizes the output for concrete spin configurations. All results are with very high certainty and produce the same label as in Table 4.4. Again, it is surprising that the third entry is evaluated as paramagnetic because most spins point in the same direction. Nevertheless, the results are extremely good and produce even for the h1 testing set accurate results with a loss of 0.507 and an accuracy of 0.900.

Summarized, this dataset produces a very precise classifier with mostly reasonable results. Nevertheless, this network can not outperform the h1 dataset. This can be derived from the lower accuracy for the h1 testing dataset and the remarkably accurate results of the h3 dataset for the network in h1 (compare h1 under Section 4.2.2).



(a) accuracy (b) loss

Figure 4.8: Loss and accuracy curve for 2D Spin Configuration Classifier using h2: The curves are produced during training of the Spin State Classifier with a dropout of 0.2 and kernel size 3. The classifier was trained with a dataset of 2D Ising models with three particles per dimension and the h2 distribution.



(a) accuracy (b) loss

Figure 4.9: Loss and accuracy curve for 2D Spin Configuration Classifier using h3: The curves are produced during training of the Spin State Classifier with no dropout and kernel size 3. The classifier was trained with a dataset of 2D Ising models with three particles per dimension and the h3 distribution.

4.3 1D

In this Chapter, a classifier for one-dimensional Ising models is created, with the goal to test transfer learning for those models. At first, a fully connected neural network is produced, which is trained with the h1 dataset. Afterward, it is analyzed and in Section 5.2 used for transfer learning.

4.3.1 Neural network architecture

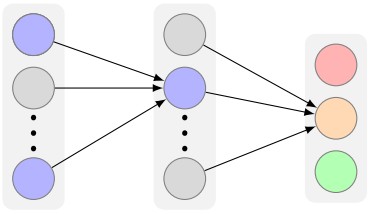


Figure 4.10: Model for the 1D Spin Configuration Classifier: The whole model consists of fully connected layers. The first layer, also called the input layer, receives spin configurations with p particles as input, wherefore the number of neurons is p. The second layer represents a hidden layer with five neurons and the last one the output layer with three neurons. The different colors of those neurons should represent the three different labels into which the network classifies. The grey neurons should represent the dropout of the model.

For the one-dimensional case, the input is only a vector of spins, pointing up or down. Hence, it is not possible to use the classical 2D convolution directly. We could reshape the vector to a matrix to use convolutions, but depending on the number of particles, this is not always possible. Another approach would be to use 1D convolutions, but this functionality is not supported in the target network. Therefore, we decided to use a neural network, which is only using fully connected layers. Because the input size, which is equal to the number of particles, is usually smaller than 10, we select a small model with three layers. The first layer has p neurons, whereby p denotes the length of the input vector. After this layer, a dropout and a hidden layer with five neurons follow. Lastly, we add a second dropout and the output layer with three neurons. As we did in the other models, the dropout rate is customized for each dataset. For the first two fully connected layers, we use ReLU as an activation function. The last layer has three neurons and uses Softmax as an activation function to classify into the corresponding classes. The remaining settings are equal to the two-dimensional model in Section 4.2.1.

4.3.2 Analysis

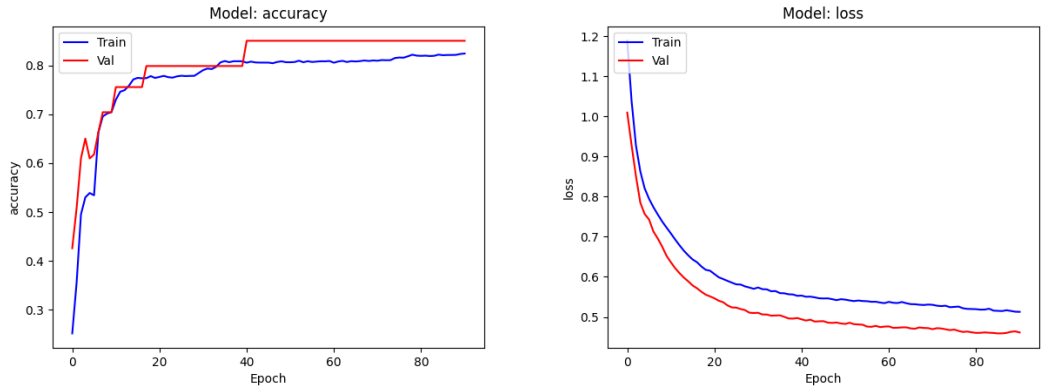


Figure 4.11: Loss and accuracy curve for 1D Spin Configuration Classifier using h1: The curves are produced during training of the Spin State Classifier with a dropout of 0.1 and kernel size 3. The classifier was trained with a dataset of 1D Ising models with four particles per dimension and the h1 distribution.

The best dropout rate for our dataset is 0.1, which results in the learning curves shown in Figure 4.11. Both curves show continuous improvement and result in an accuracy of 0.847 and a loss of 0.519. These are not very good results, but under the circumstance that it is only a fully connected neural network, the results are acceptable. Table 4.6 shows the outputs for all possible spin configurations. As one can see, for all configurations except [0000] and [1111], the network predicts the paramagnetic phase. In the case of only 4 particles, this makes sense from a physical standpoint. It is nice to observe that for classical paramagnetic configurations, as in entry 10, the probability of being in this state is higher than for non-classical configurations. Therefore, the results are reasonable from a physical point of view.

All in all, the model is good but not very precise. It delivers comprehensible results and has acceptable accuracy.

Nr	Spin	Prediction	Predicted Label
1	[0 0 0 0]	[0.15, 0.82, 0.02]	ferromagnetic
2	[0 0 0 1]	[0.69, 0.29, 0.02]	paramagnetic
3	[0 0 1 0]	[0.88, 0.11, 0.01]	paramagnetic
4	[0 0 1 1]	[0.76, 0.22, 0.02]	paramagnetic
5	[0 1 0 0]	[0.81, 0.17, 0.02]	paramagnetic
6	[0 1 0 1]	[0.85, 0.13, 0.01]	paramagnetic
7	[0 1 1 0]	[0.90, 0.09, 0.01]	paramagnetic
8	[0 1 1 1]	[0.75, 0.23, 0.02]	paramagnetic
9	[1 0 0 0]	[0.72, 0.26, 0.02]	paramagnetic
10	[1 0 0 1]	[0.92, 0.08, 0.01]	paramagnetic
11	[1 0 1 0]	[0.88, 0.11, 0.01]	paramagnetic
12	[1 0 1 1]	[0.84, 0.14, 0.01]	paramagnetic
13	[1 1 0 0]	[0.74, 0.24, 0.02]	paramagnetic
14	[1 1 0 1]	[0.88, 0.11, 0.01]	paramagnetic
15	[1 1 1 0]	[0.66, 0.32, 0.03]	paramagnetic
16	[1 1 1 1]	[0.15, 0.82, 0.02]	ferromagnetic

Table 4.6: Results from the 1D Spin Configuration Classifier (h1): The Table visualizes the output distribution ("Prediction") for given spin configurations.

5 Transfer learning

In this Chapter, we cover the target network, which returns the corresponding entry of the ground state vector for a given spin configuration. For this, the network needs a specified h value, a dimension, and a number of particles per dimension. For example, for a 2D Ising model with one particle, the ground state vector would have two entries. For the first spin configuration $< 0 >$, the network should predict the first entry of the ground state, for $< 1 >$ the second one. In case the network predicts 0.96 for $< 0 >$ and 0.28 for $< 1 >$, the predicted ground state of the network would evaluate to $[0.96, 0, 29]$. Therefore, the network is able to predict the whole ground state of the system by inputting and merging all spin configurations.

Because the input of the network is a spin configuration, we have to distinguish between 1D and 2D Ising models. At first, in Section 5.1 the more interesting 2D Ising models are covered and analyzed for pretrained and random weights. The same is done for the 1D Ising model in Section 5.2. For the analysis of the networks, we use the deviation of the exact energy $E0$ and the energy E. E is calculated by $< psi|H|psi >$, with H denoting the Hamiltonian and psi denoting the predicted ground state by the network. So it depends on the predicted ground state entries and is, therefore, a good indicator for the performance of the network.

The used code for the ground state optimization is based on a previous implementation from Irene López Gutiérrez and Prof. Dr. Christian Mendl.

5.1 2D

At first, the model and results for 2D Ising models are discussed. Therefore, the general structure of the neural network is explained in Section 5.1.1. Next, the network is trained with and without transferred weights, and the results are compared and analyzed in Section 5.1.2. We limit the observation to the ground state weights and the spin configuration weights with the dataset h1. The other distributions for h are not discussed here because they produce worse results. Additionally, the comparison of the ground state and spin configuration weights is more reasonable because they share the same h distribution.

5.1.1 Neural network architecture

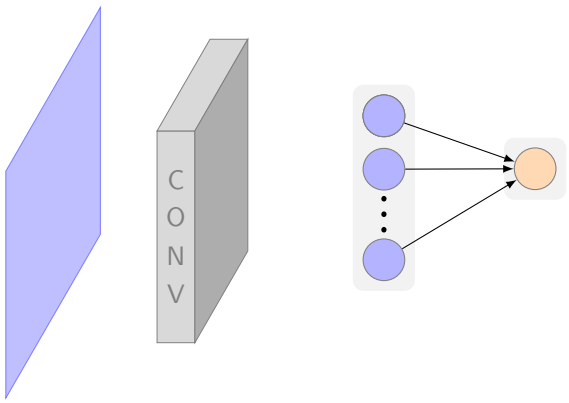


Figure 5.1: Model for the 2D Ground State Calculator: The elements are explained from left to right. The blue rectangle represents the $(p \times p)$ input matrix of a spin configuration. It is followed by a convolution layer and a fully connected layer with 20 neurons. The last layer consists of one neuron, which outputs the corresponding ground state entry of the Ising model.

The concrete model of the target network is illustrated in Figure 5.1. It consists of only one convolution layer and two fully connected layers with 20 and one neuron. The kernel and bias weights of the convolution are randomly selected if we do not use transfer learning. Otherwise, those weights are reused from the pretrained models. Therefore, the kernel size, stride, and the number of filters are fixed to 3, 1, and 32. The fully connected layers with 20 and one neuron use random initialization. To stay consistent with the pretrained models, the activation function is ReLU for all layers. The networks use 512 number of samples, a learning rate of $1e-3$, and a batch size of 100. In the following, this network is trained and analyzed for 2D Ising models with three particles per dimension.

5.1.2 Analysis

To compare the performance of the network with and without transferred weights, this Section analyses the results for three different h values among the network in Figure 5.1. At first, a very high h value of $1e5$ is chosen with an exact ground state energy of $-900,000$, denoted as $E0$. The Figures 5.2, 5.3 and 5.4 show the plots for the objective function, the probability distribution d (see Chapter 4.1), and the recorded ground state. In each case, the plots are generated for random weights and transferred weights from the Ground State Classifier and the Spin Configuration Classifier with the h1 dataset.

Weights	Deviation	Relative error	Overlap error
random	0.053	5.856e-08	3.140e-08
ground	2.466e-05	2.740e-11	2.899e-11
spin	4.500e-05	5.000e-11	5.625e-11

(a) $h = 1e5$

Weights	Deviation	Relative error	Overlap error
random	6.907	0.384	0.825
ground	8.000	0.444	1.000
spin	1.125e-10	6.251e-12	0.197

(b) $h = 1e - 5$

Weights	Deviation	Relative error	Overlap error
random	1.131	0.059	0.342
ground	0.344	0.018	0.152
spin	1.066	0.056	0.323

(c) $h = 1$

Table 5.1: Evaluation of transfer learning for 2D Ising models: The Table shows the evaluation of the predicted result with and without transferred weights. The column "Weights" specifies which initial weights are used for the network. Hereby, "random" represents the network without transferred weights and "spin" or "ground" the transferred weights of the Spin Configuration Classifier or the Ground State Classifier. "Deviation" means the deviation of the prediction to the exact energy, so $E - E0$, with E being the energy calculated from the network and $E0$ the exact energy. The relative error between E and $E0$ is calculated with $|(E - E0)/E0|$, and the overlap error with $1 - | < gs|found > |$. Here, gs denotes the exact ground state and $found$ the predicted ground state of the network. Hence, $found$ is the concatenation of the network results for all possible spin configurations. With the whole formula, we can estimate how similar the ground state vectors are. For an easier interpretation, the cell color represents the quality of the result.

As one can see in Figure 5.2a, for random weights the ground state energy starts at a value around $-850,000$ and reaches after a low number of iterations $E0$. For the pretrained weights of the ground state model, the initial value of about $-899,700$ is already very close to $E0$, which can be derived from Figure 5.2b. In Figure 5.2c one can see, that for the spin configuration weights, the initial value is already matching $E0$. The fact that both transferred weights cause a closer initial value to the exact energy

suggests that the pretrained weights improve the network.

Figure 5.3 shows the probability distribution for the different spin configurations. The network without trained weights has a very fluctuating graph and, therefore, very different probabilities. This can be seen in Figure 5.3a. Figure 5.3b with the ground state weights has already less fluctuations and a pattern is recognizable, and Figure 5.3c with the spin configuration weights has no fluctuation at all and has constant probabilities.

Finally, in Figure 5.4 the predicted and exact ground state are compared. Here, the exact ground state entries for the corresponding spin configurations are plotted in blue. They are compared to *psi*, the ground state values, which are painted in orange. Ideally, the blue and orange curves are matching, which is not the case for any of the three plots. Figure 5.4a, which has no trained weights, has very high fluctuation in the *psi* curve and a large deviation between the curves. Therefore, the prediction of the ground state is not very accurate. The second plot, Figure 5.4b, which has weights of the Ground State Classifier, has some matching intervals. Besides, the *psi* function has nearly no fluctuation and is close to the blue curve, which can be derived from the y-coordinates. Similar applies to Figure 5.4c, which uses weights of the Spin Configuration Classifier. The deviation to the blue curve is very small and additionally very similar to Figure 5.4b. Although the two curves have only one matching point, the predictions are very close to the exact ground state. Therefore, both transferred weights obtain a more accurate representation of the ground state than the randomly initialized weights.

Summarized, with transfer learning, not only the performance at the beginning of the training is better, but also the predicted ground state is more accurate. Therefore, the Figures indicate that transfer learning improves the performance of the network.

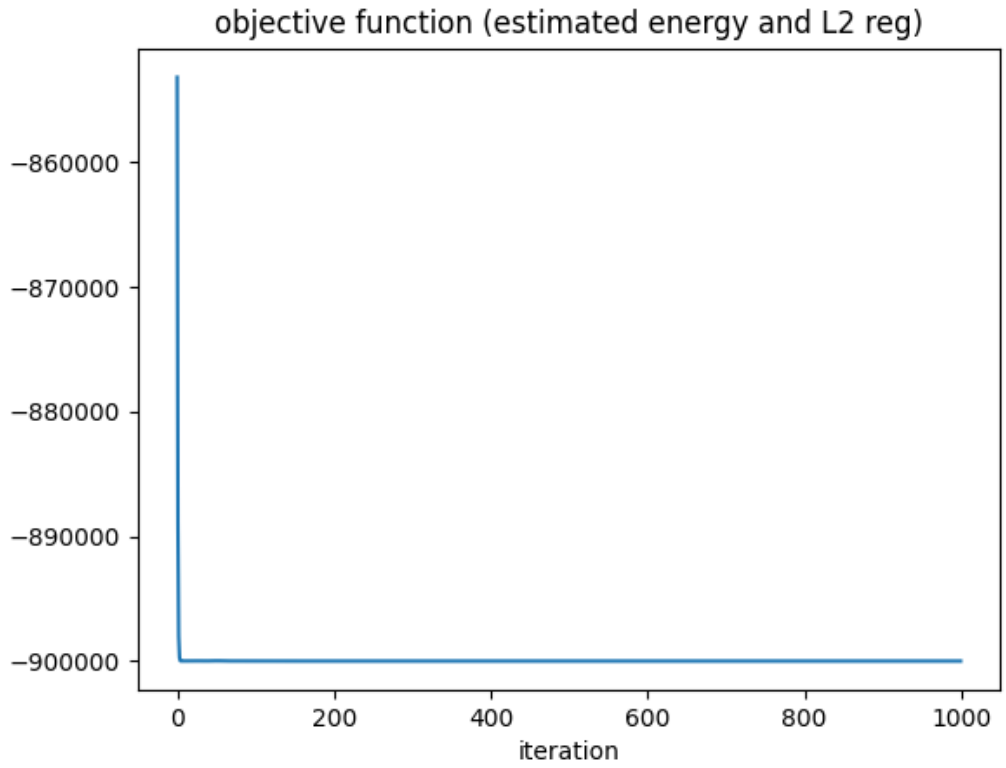
To validate these observations, the results of the network are evaluated in Table 5.1a using the energy and the ground state of the system. The deviation of the energy is for both transferred weights minimal. Additionally, they achieve similar deviations and improve the random weights. The same holds for the relative error. Therefore, the energy of the model can be predicted very accurate. For the overlap error, we achieve similar results. The transferred weights are able to reduce the error by about two decimal digits. Thereby, the ground state produces slightly better ground states, which is also visible in Figure 5.4. But all in all, the accuracy of the ground state prediction is very high for all weights.

In the following, we validate the transferred weights for lower h values, for example

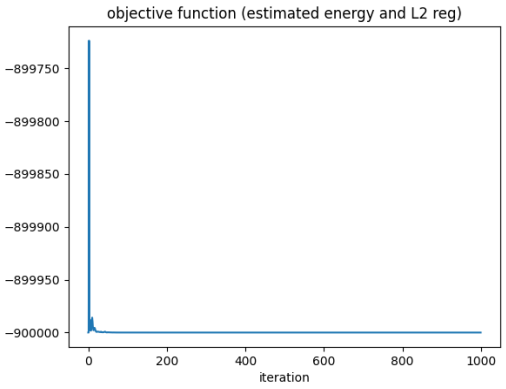
for $h = 1e-5$ as shown in Table 5.1b. For random and ground state weights, the results are very imprecise. Additionally, there is deterioration due to the ground state weights. Consequently, both the predicted energy and the predicted ground state are inaccurate for those two entries. This is in total contrast to the "Deviation" and "Relative error" for the spin configuration weights, which are remarkably good and even outperform the results of Table 5.1a. The "Overlap error" is still very high but better than for the other weights. This means that with the weights of the Spin Configuration Classifier, the network is able to predict E very accurately, there are still deviations in the ground state predictions. Nevertheless, these weights achieve an improvement in contrast to the untrained weights.

Lastly, we check the performance for $h = 1$. In this case, all results are similar and not very good. Nevertheless, they are better than in Table 5.1b for random or ground state weights. Here, exactly the opposite than in Table 5.1b happens. The ground state weights show better results, whereas the spin configuration weights have only a very low impact.

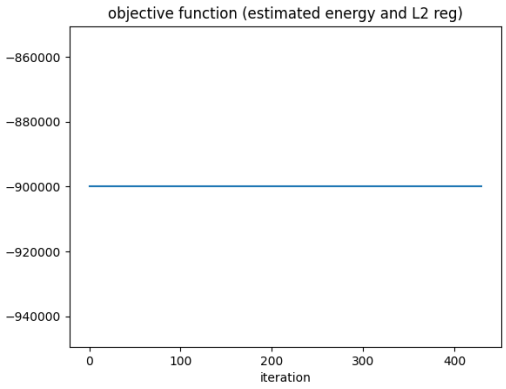
All in all, the transferred weights show, in most cases, improved results in comparison to the untrained network. Only with very high h values perform both transferred weights comparably and significantly better than with random weights. For lower h values, the impact of the transferred weights is diminishing and differs among the two transferred weights. Especially for h close to 0, the results are different, where the spin configuration weights show excellent results for the energy. Further analysis of the impact due to transfer learning is covered in Chapter 6.



(a) random weights

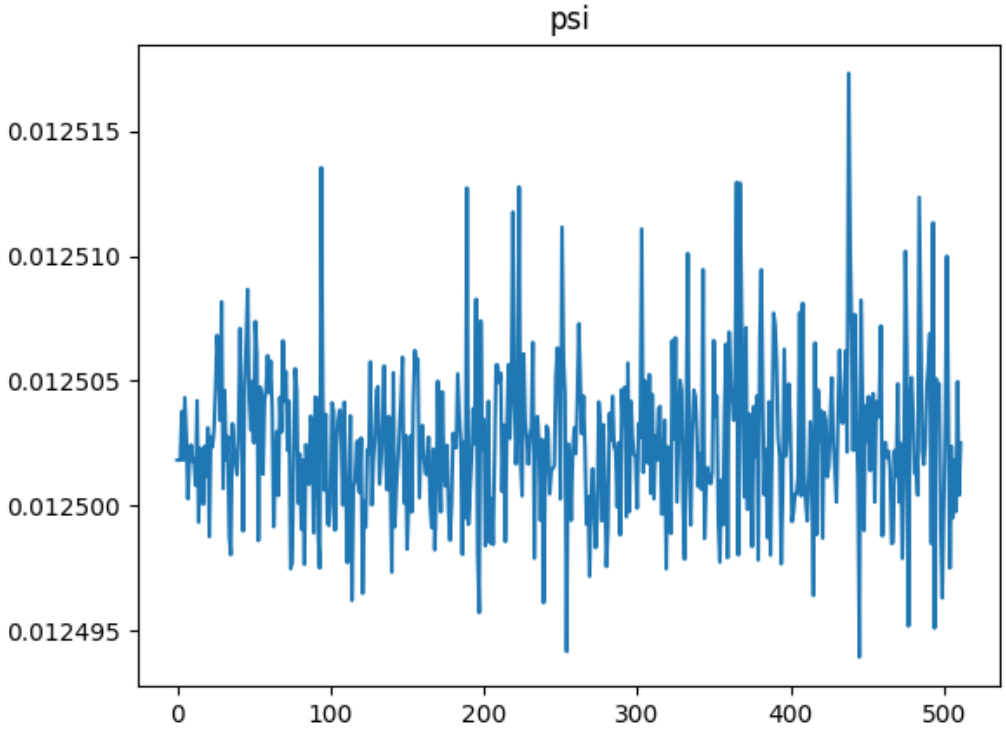


(b) Ground State Classifier weights

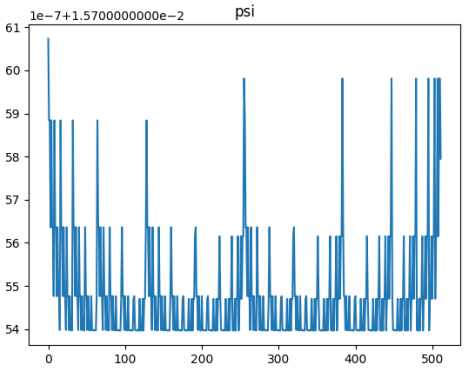


(c) Spin Configuration Classifier weights

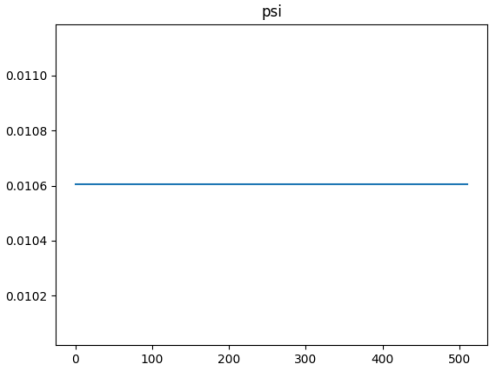
Figure 5.2: Predicted energy per iteration for 2D model with and without pretrained weights for $h = 1e5$



(a) random weights

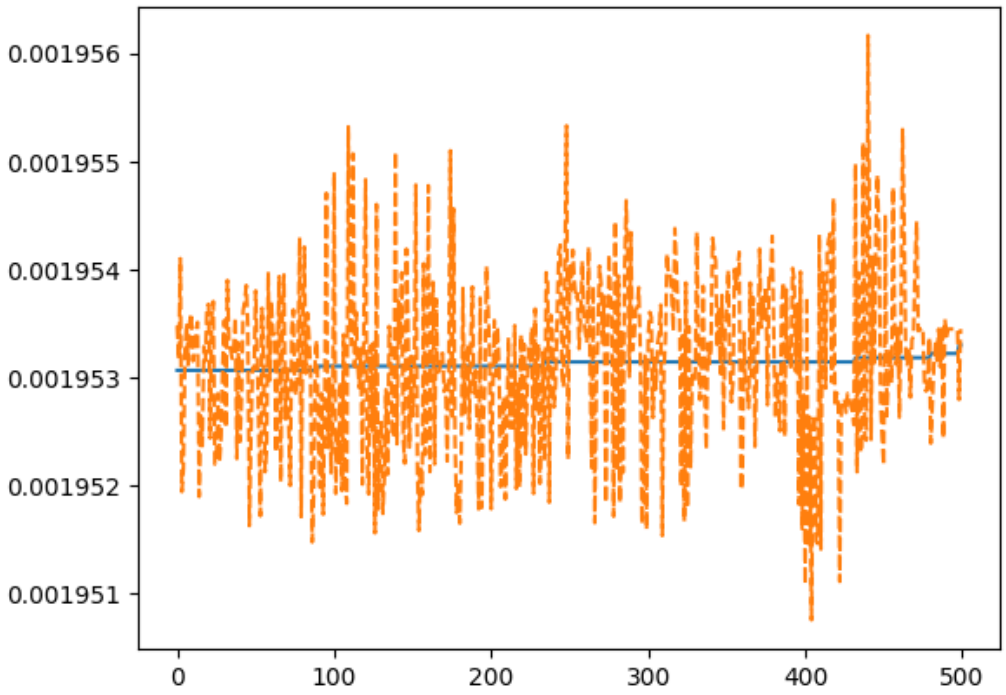


(b) Ground State Classifier weights

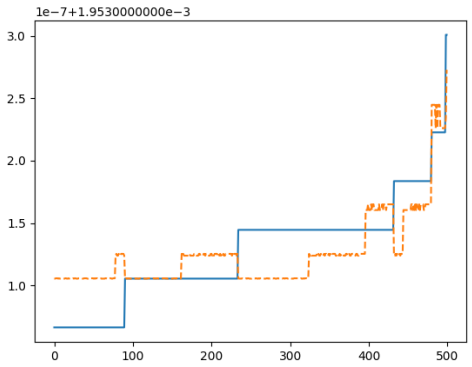


(c) Spin Configuration Classifier weights

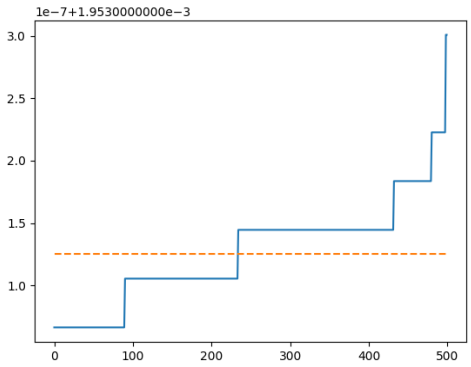
Figure 5.3: Probability distribution d ($|psi|^2$) for 2D model with and without pretrained weights for $h = 1e5$



(a) random weights



(b) Ground State Classifier weights



(c) Spin Configuration Classifier weights

Figure 5.4: Recorded ground state for 2D model with and without pretrained weights for $h = 1e5$: The blue line represents the exact ground state values for the corresponding spin configuration and the orange line the predicted ground state *psi*.

5.2 1D

In the following, we have a look at transfer learning for 1D Ising models with a fully connected neural network. Because the input of the neural network are spin configurations, and the weights of the first fully connected layer are transferred, only the spin configuration weights can be used. For the ground state, the input vector has $2^p \neq p$ neurons, and therefore, the size of the weights and biases does not match with the target network. As already defined in in Section 4.1, p denotes the number of particles per dimension and is in our case set to $p = 4$.

5.2.1 Neural network architecture

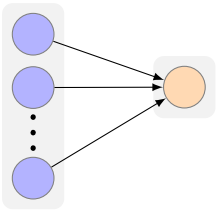


Figure 5.5: Model for the 1D Ground State Calculator: The whole model consists of fully connected layers. The inputs are spin configurations with p particles per dimension, wherefore the input layer consists of p neurons. The last layer consists of a single neuron, which outputs the corresponding ground state entry.

The structure of the network is very simple and consists only of two fully connected layers. The first layer consists of p neurons and is directly connected to the output neuron. This neuron outputs the value of the ground state vector corresponding to the input spin configuration. The weights and biases of the first layer are transferred from the Spin Configuration Classifier or chosen randomly. For the last neuron, these values are chosen randomly for all cases. We use the highest possible number of samples, which is $2^{p \cdot dim}$ and evaluates to 16, and ReLU as activation function. The remaining parameters stay the same as in the 2D case in Section 5.1.1.

5.2.2 Analysis

As for the two-dimensional case in Section 5.1.2, we analyze the performance of the network with the deviation of the energies, the relative error, and the overlap error, which are listed in Table 5.2. For very high and low h values, the deviation and the relative error are not only very low but also nearly equal for both weights. Hence, the model is

Weights	Deviation	Relative error	Overlap error
random	7.452e-06	1.863e-11	1.029e-11
spin	7.143e-06	1.786e-11	8.164e-12

(a) $h = 1e5$

Weights	Deviation	Relative error	Overlap error
random	1.000e-10	2.500e-11	0.324
spin	1.000e-10	2.500e-11	0.763

(b) $h = 1e-5$

Weights	Deviation	Relative error	Overlap error
random	0.152	0.029	0.175
spin	0.398	0.076	0.332

(c) $h = 1$

Table 5.2: Evaluation of transfer learning for 1D Ising models: The Table shows the evaluation of the predicted result with and without transferred weights. The column "Weights" specifies which initial weights are used for the network. Hereby, "random" stands for the network without transferred weights and "spin" for the transferred weights of the Spin Configuration Classifier. "Deviation" means the deviation of the prediction to the exact energy, so $E - E0$, with E being the energy calculated from the network and $E0$ the exact energy. The relative error between E and $E0$ is calculated with $|(E - E0)/E0|$, and the overlap error with $1 - | < gs|found > |$. Here, gs denotes the exact ground state and $found$ the predicted ground state of the network. Hence, $found$ is the concatenation of the network results for all possible spin configurations. With the whole formula, we can estimate how similar the ground state vectors are. For an easier interpretation, the cell color represents the quality of the result.

predicting the energy very accurately. For $h = 1$, the spin configuration weights have even worsened the model. However, these minor deviations among the weights could also have stochastic reasons. Accordingly, any improvement or deterioration should not be interpreted as a result of transfer learning. If one takes a look at the overlap error, only the results of $h = 1e5$ are very accurate and can outperform the random weights. For the other two h values, the transferred weights increase the overlap error and consequently predict the ground state more inaccurate.

All in all, the neural network produces precise results, except for $h = 1$. Thereby,

the quality of the results does not depend on the initial weights. From that, we can conclude that the neural network optimizes so well that it can not benefit from the transferred weights.

6 Summary

In this Chapter, we summarize all results and give an overview of what did and did not improve the Ground State Predictor.

6.1 2D

Weights	h	Energy	Ground state
ground	$1e5$	high improvement	high improvement
	1	high improvement	high improvement
	$1e-5$	high deterioration	small deterioration
spin	$1e5$	high improvement	high improvement
	1	small improvement	small improvement
	$1e-5$	very high improvement	very high improvement

Table 6.1: Improvement due to transfer learning (2D): This Table shows the interpretation of Table 5.1. This is done, by comparing the results of the transferred weights for "Deviation" and "Overlap error" with the randomly initialized weights. The Table entry "Weights" denotes the used initialization weights. Hence, for "ground", the transferred weights of the Ground State Classifier are used. The cell colors highlight the improvement due to transferred weights.

For the two-dimensional case, the impact of transfer learning to the predicted energy and ground state can be derived from Table 6.1. These interpreted results are combined with the results from Table 5.1 to get a final conclusion. All observations apply to both the energy and the ground state. For the weights of the Ground State Classifier, the target network is improved for $h \geq 1$. For $h = 1e5$, the transferred weights are able to further improve the already very good results of the randomly initialized network for both the energy and the ground state. In the case of $h = 1$, the improvement due to the transferred weights leads also to precise results, which stands in total contrast to $h = 1e-5$. Here, the transferred weights are not able to achieve any improvement and even deteriorate the results. When looking at the concrete results in Table 5.1b, one can observe that the predictions with and without the transferred ground state

weights are very inaccurate. Therefore, transferring the weights of the Ground State Classifier and using it for very low h values is not recommendable. With h values greater than 1, the predictions of the network are precise, especially for high h values. All in all, the ground state weights are only suitable for $h \geq 1$, where they achieve large improvements.

For the spin configuration weights, we obtain improvements for all h values. Just as for the ground state weights, the improvement of $h = 1e5$ is remarkably good and results in very accurate predictions. Lower h values, such as $h = 1$, perform worse than the ground state weights. Nevertheless, a minor improvement to the random initialization is identifiable, which results in a network with acceptable accuracy. With $h = 1e - 5$ we were able to observe the greatest improvement of all covered scenarios. Here, the predictions of the randomly initialized network are very inaccurate, in total contrast to the network, with the transferred weights. These results are extremely good and even outperform all other values in terms of energy. For the ground state, the prediction is still accurate but significantly worse than for $h = 1e5$. This great improvement is very surprising, especially considering the deterioration of the ground state weights. However, it holds the advantage of using these weights for h values close to 0. All in all, the weights of the spin configuration show improvements over the whole range. Therefore, the learned weights from the Spin Configuration Classifier have learned features, which were helpful for the prediction of the ground state.

In conclusion, for each considered h, the results of the randomly initialized weights could be outperformed with one of the transferred weights. Thereby, depending on the concrete h value, either the ground state weights or the spin configuration weights achieve the best performance. We can conclude that transfer learning is able to significantly improve the target network.

6.2 1D

h	Energy	Ground state
$1e5$	small improvement	small improvement
1	small deterioration	small deterioration
$1e-5$	no improvement	small deterioration

Table 6.2: Improvement due to transfer learning (1D): This Table shows the interpretation of Table 5.2. This is done, by comparing the results of the spin configuration weights for "Deviation" and "Overlap error" with the random initialized weights. The cell colors highlight the improvement due to transferred weights.

As for the two-dimensional case, we analyze the results of the target network from 1D Ising models. Hereby, Table 6.2 interprets the results of Table 5.2. In the following, both are used to obtain a final estimate of the influence of transferred weights. Except for $h = 1e5$, the pretrained weights could not obtain any improvement in the network. But even for this high h value, the transfer learning could not significantly improve the predictions. As we can derive from Table 5.2a, the results of both networks are extremely precise. The same holds for the energy of $h = 1e-5$, where the pretrained weights show no difference to the random weights. It is interesting to see that the results of the Ground State Calculator are worse due to the transferred weights and also predict very imprecise. In the case of $h = 1$, the spin configuration weights only worsened the predictions of energy and ground state. Their final results are acceptable but not as good as for the other h values.

Summarized, the pretrained weights show mainly deterioration or no significant improvement. Thereby, the results of trained and untrained weights have a very small difference. This leads to the assumption that some improvements and deteriorations have stochastic reasons and can vary. Consequently, it is not recommendable to use pretrained weights for the 1D case. Nevertheless, most results of both models are precise, and therefore, the models are predicting very well.

7 Conclusion

This work covers the creation of two neural networks classifying the Transverse-field Ising model into their Ising phases. At first, the two-dimensional case with convolutional neural networks and three particles per dimension is considered. Here, we created a few networks and analyzed their performance. The first network, which classifies the ground state, achieves very precise results with an accuracy of 99%. Similarly, the second network, which classifies the concrete spin configurations, achieves an accuracy of 94%. We transferred these learned weights to the target network, which is predicting the ground state entry for a corresponding spin configuration. After analyzing the results for different h values, it appears that the transferred weights almost always outperform the randomly initialized network. Only for the ground state weights and a h close to 0, the performance is worse due to pretrained weights. For the case of very high h values, the network predicted the ground state very accurately and with a huge improvement due to the transferred weights.
Second, we discover the influence of transfer learning for the 1D case with four particles. We limit the observation to the spin configuration weights created by a fully connected neural network. With an accuracy of 85%, the classification is less accurate than for the 2D classifiers. During transfer learning, we determined that we could achieve almost no improvement due to the transferred weights. Additionally, the difference between the trained and untrained weights is so low that the reasons could also be stochastic. Consequently, the transfer learning was not able to improve the Ground State Predictor and is not recommended for that case.
All in all, this work shows that it is possible to improve the performance of the Ground State Predictor with transferred weights in the two-dimensional case. For the one-dimensional case, we could gain no significant improvement. Consequently, we assume that only for high dimensions meaningful features can be extracted, which results in improvements. This presumption can be covered in future work. Also, this work covers only the influence of transfer learning, trained on a low number of particles. For future work, it would be interesting how the influence changes for Ising models with more particles.

List of Figures

2.1 Spin configurations for 2D Ising model with three particles per dimension 3
2.2 Biological neuron (left) and neural network neuron (right) [13] 5
2.3 Fully connected neural network 6
2.4 ReLU activation function 7
2.5 Convolution on input I with filter K 10
2.6 Average and Max Pooling [11] 10

3.1 Run time for the creation of the Ground State Dataset 14
3.2 Model for the 2D Ground State Classifier 15
3.3 Loss and accuracy curve for 2D Ground State Classifier 17
3.4 Confidence of h for the trained Ground State Classifier 18

4.1 Original and predicted label distribution of the testing Equal Spin Configuration Dataset 22
4.2 Spin configuration appearance in the training Spin Configuration Dataset 24
4.3 Run time for the creation of the Spin Configuration Dataset using h1 . . 25
4.4 Model for the 2D Spin Configuration Classifier 26
4.5 Label distribution of the training and testing Spin Configuration Dataset (h1) 28
4.6 Loss and accuracy curve for 2D Spin Configuration Classifier using h1 . 28
4.7 Original and predicted label distribution of the testing Spin Configuration Dataset using h1 29
4.8 Loss and accuracy curve for 2D Spin Configuration Classifier using h2 . 32
4.9 Loss and accuracy curve for 2D Spin Configuration Classifier using h3 . 32
4.10 Model for the 1D Spin Configuration Classifier 33
4.11 Loss and accuracy curve for 1D Spin Configuration Classifier using h1 . 34

5.1 Model for the 2D Ground State Calculator 37
5.2 Predicted energy per iteration for 2D model with and without pretrained weights for $h = 1e5$ 41
5.3 Probability distribution d ($|psi|^2$) for 2D model with and without pretrained weights for $h = 1e5$ 42

5.4 Recorded ground state for 2D model with and without pretrained weights for $h = 1e5$ 43
5.5 Model for the 1D Ground State Calculator 44

List of Tables

3.1 Distribution of h for the Ground State Dataset 12
3.2 Detailed distribution of h for the Ground State Dataset 12
3.3 Label distribution of the training Ground State Dataset 13
3.4 Detailed Model for 2D Ground State Classifier 16

4.1 Label distribution of the training Equal Spin Configuration Dataset . . . 21
4.2 Label distribution of the training Spin Configuration Dataset for h1, h2, and h3 23
4.3 Exemplary results from the 2D Spin Configuration Classifier (h1) 27
4.4 Exemplary results from the 2D Spin Configuration Classifier (h2) 30
4.5 Exemplary results from the 2D Spin Configuration Classifier (h3) 31
4.6 Results from the 1D Spin Configuration Classifier (h1) 35

5.1 Evaluation of transfer learning for 2D Ising models 38
5.2 Evaluation of transfer learning for 1D Ising models 45

6.1 Improvement due to transfer learning (2D) 47
6.2 Improvement due to transfer learning (1D) 49

Bibliography

[1] S. Albawi, T. A. Mohammed, and S. Al-Zawi. "Understanding of a convolutional neural network." In: *2017 International Conference on Engineering and Technology (ICET)*. Ieee. 2017, pp. 1–6.

[2] S. Avalos and J. M. Ortiz. "Convolutional neural networks architecture: A tutorial." In: (2019), pp. 159–168.

[3] A. Borin and D. A. Abanin. "Approximating power of machine-learning ansatz for quantum many-body states." In: *Physical Review B* 101.19 (2020).

[4] G. Carleo and M. Troyer. "Solving the quantum many-body problem with artificial neural networks." In: *Science* 355.6325 (Feb. 2017), pp. 602–606.

[5] J. Carrasquilla and R. G. Melko. "Machine learning phases of matter." In: *Nature Physics* 13.5 (Feb. 2017), pp. 431–434.

[6] J. Côté. *The Curse of Dimensionality, From Combinatorics to many-body quantum systems*. Accessed: 2021-08-06. Oct. 2020.

[7] V. Dumoulin and F. Visin. "A guide to convolution arithmetic for deep learning." In: *arXiv preprint arXiv:1603.07285* (2016).

[8] X. Gao and L.-M. Duan. "Efficient representation of quantum many-body states with deep neural networks." In: *Nature communications* 8.1 (2017), pp. 1–6.

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[10] I. L. Gutiérrez and C. B. Mendl. *Real time evolution with neural-network quantum states*. 2021.

[11] S. Hijazi, R. Kumar, C. Rowen, et al. "Using convolutional neural networks for image recognition." In: *Cadence Design Systems Inc.: San Jose, CA, USA* (2015), pp. 1–12.

[12] A. Karpathy. *Convolutional Neural Networks: Architectures, Convolution / Pooling Layers*. Accessed: 2021-08-03. Aug. 2015.

[13] A. Karpathy. *Neural Networks Part 1: Setting Up the Architecture*. Accessed: 2021-08-03. Aug. 2015.

[14] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning." In: *Nature Cell Biology* 521.7553 (May 2015), pp. 436–444.

[15] Z. Liu, S. P. Rodrigues, and W. Cai. *Simulating the Ising Model with a Deep Convolutional Generative Adversarial Network*. 2017.

[16] A. Morningstar and R. G. Melko. *Deep Learning the Ising Model Near Criticality*. 2017.

[17] M. Nielsen. *Using neural nets to recognize handwritten digits, Chapter 1*. Accessed: 2021-08-03. 2019.

[18] Y. Nomura, A. S. Darmawan, Y. Yamaji, and M. Imada. "Restricted Boltzmann machine learning for solving strongly correlated quantum systems." In: *Phys. Rev. B* 96 (20 Nov. 2017).

[19] K. O'Shea and R. Nash. "An introduction to convolutional neural networks." In: *arXiv preprint arXiv:1511.08458* (2015).

[20] S. J. Pan and Q. Yang. "A Survey on Transfer Learning." In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010).

[21] J. A. Reyes, D. C. Marinescu, and E. R. Mucciolo. "Simulation of quantum many-body systems on Amazon cloud." In: *Computer Physics Communications* 261 (2021), p. 107750.

[22] H.-Q. Shi, X.-Y. Sun, and D.-F. Zeng. "Neural-Network Quantum State of Transverse-Field Ising Model." In: *Communications in Theoretical Physics* 71.11 (2019), p. 1379.

[23] R. Stinchcombe. "Ising model in a transverse field. i. basic theory." In: *Journal of Physics C: Solid State Physics* 6.15 (1973), p. 2459.

[24] J. Wu. "Introduction to convolutional neural networks." In: *National Key Lab for Novel Software Technology. Nanjing University. China* 5.23 (2017), p. 495.

[25] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. *How transferable are features in deep neural networks?* 2014.

[26] R. Zen, L. My, R. Tan, F. Hébert, M. Gattobigio, C. Miniatura, D. Poletti, and S. Bressan. "Transfer learning for scalability of neural-network quantum states." In: *Physical Review E* 101.5 (May 2020).