UTILISING CONVOLUTIONAL NEURAL NETWORKS FOR TEMPERATURE-BASED ANALYSIS OF TWO-DIMENSIONAL ISING MODEL CONFIGURATIONS

A Thesis

by

Onur Kerem Özmen

Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Physics

Özyeğin University
September 2024

Copyright © 2024 by Onur Kerem Özmen

UTILISING CONVOLUTIONAL NEURAL NETWORKS FOR TEMPERATURE-BASED ANALYSIS OF TWO-DIMENSIONAL ISING MODEL CONFIGURATIONS

Approved by:

Prof. Taylan Akdoğan, Advisor
Dept. of Natural & Mathematical Sciences
Özyeğin University

Prof. M. Burçin Ünlü
Dept. of Natural & Mathematical Sciences
Özyeğin University

Prof. Bora Işıldak, Coadvisor
Dept. of Physics
Yıldız Technical University

Dr. Hale Sert
Dept. of Physics
Istanbul University

Date Approved: August 2, 2024

Prof. H. Fatih Uğurdağ
Dept. of Electrical & Electronics Eng.
Özyeğin University

To my friends who are always aiming higher, thank you for your inspiration.

ABSTRACT

This study investigates the application of Convolutional Neural Networks (CNNs) to classify data associated with complex physical systems, specifically focusing on the Two-Dimensional Ising Model and its temperature-dependent ferromagnetic behaviour. The motivation behind this work is to leverage CNNs for distinguishing between phases of the system under different temperature regions: below critical, at critical, and above critical temperatures. Three distinct neural network architectures were implemented, and their performance was evaluated using a dataset of simulated configurations. The results demonstrate that the models achieved high accuracy in predicting the temperature-related phases, indicating the effectiveness of CNNs in capturing physical patterns. These findings contribute to the understanding of CNNs in physical sciences and suggest that machine learning techniques could potentially be applied to other complex systems. Future work might focus on refining the model architectures and extending the analysis to other types of lattice structures.

ÖZETÇE

Bu çalışma, karmaşık fiziksel sistemlerle ilgili verileri sınıflandırmak için Evrişimli Sinir Ağları'nın uygulanmasını araştırmakta olup, özellikle İki Boyutlu Ising Modeli ve bu modelin sıcaklığa bağlı ferromanyetik davranışına odaklanmaktadır. Bu çalışmanın motivasyonu, sistemin farklı sıcaklık bölgelerinde (kritik altı, kritik ve kritik üstü sıcaklıklar) fazları ayırt edebilmek için Evişimli Sinir Ağlarının sınıflandırma becersini test etmektir. Üç farklı sinir ağı mimarisi uygulanmış ve performansları, simüle edilmiş konfigürasyonlardan oluşan bir veri kümesi kullanılarak değerlendirilmiştir. Sonuçlar, modellerin sıcaklıkla ilgili fazları tahmin etmede yüksek doğruluk sağladığını göstererek, Evrişimli Sinir Ağlarının karmaşık fiziksel sistemlerdeki örüntüleri yakalamadaki etkinliğini ortaya koymaktadır. Bu bulgular, Evrişimli Sinir Ağlarının fizik bilimlerindeki anlaşılmasına katkıda bulunmakta ve makine öğrenimi tekniklerinin diğer karmaşık sistemlere de potansiyel olarak uygulanabileceğini önermektedir. Gelecekteki çalışmalar, model mimarilerinin iyileştirilmesine ve diğer örgü yapılarının analizine odaklanabilir.

ACKNOWLEDGEMENTS

I would like to thank my advisor Prof. Bora Işıldak for his support. His enthusiastic approach helped me through this challenging process. I would like to thank my professors at Özyeğin University, Burçin Ünlü and Taylan Akdoğan, for their guidance. Lastly, I would like to thank my close friend and classmate Ömer Fatih Dokumacı for his never ending support.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION TO MACHINE LEARNING

Machine learning can be considered a preeminent area of research, especially in recent years. Using machine learning methods in various research fields has vastly increased our capabilities in understanding complex problems. Since there are numerous complex problems in physics, applying machine learning methods to these problems is undoubtedly a great approach. In this thesis, supervised learning methods will be used for classifying two-dimensional Ising Model configurations according to their temperatures. This task is considered an image classification problem, and because of this, Convolutional Neural Networks, which are well-suited for such tasks, will be employed.

1.1 What is Machine Learning?

Machine learning is a branch of artificial intelligence where the agent can make predictions based on provided data. Unlike traditional coding, machine learning models do not require explicit instructions, allowing for automated analysis of complex data. As its name suggests, these models can learn independently by recognizing patterns. There are numerous learning methodologies, such as supervised learning, unsupervised learning, and reinforcement learning.

1.1.1 Reinforcement learning

Reinforcement learning is a methodology where the agent learns through trial and error. The agent is rewarded for desirable actions and punished for undesirable ones. This type of learning can be used for tasks such as maze traversal, where the agent must navigate obstacles.

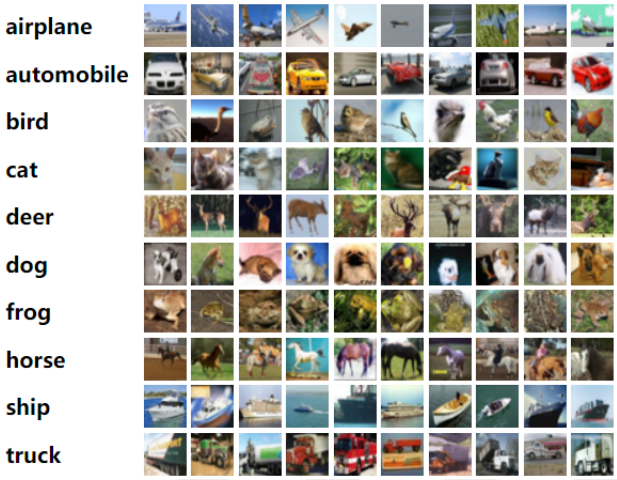


Figure 1: Example dataset for supervised learning - Cifar 10

1.1.2 Unsupervised learning

Unsupervised learning is a method in which the labels of the data are not known. In these tasks, the model groups the data according to their similarities. For example, data collected in particle collisions can be clustered into different groups to identify various particles. This method is also very successful in detecting anomalies and can be used to discover new particles.

1.1.3 Supervised learning

Supervised learning is performed by ”teaching” the model using a labelled dataset. A labelled dataset is a collection of data where each instance includes both input features and the corresponding target labels. For example, if the data consists of images of animals and the goal is to teach the model to recognize the type of animal in each image, the dataset must include both the images and the labels identifying each animal. In this context, the image is represented as a matrix of pixel values, while the label specifies the corresponding animal, such as “dog” for an image of a dog. An example of such a dataset can be seen in Figure 1.

1.1.4 Forward and Backward Propagation

When training machine learning models the data ”moves” through layers that do certain operations leading to the results. This movement is called forward propagation and will be further explained while delving into convolutional neural networks. After forward propagation, the predictions of the model are compared to the actual labels of the data. With this comparison, a loss value is calculated. After calculations, the model adjusts its weights to minimise loss. This process is called backwards propagation and is done several times to maximise learning. The number of these operations is determined by epoch value which is set before the training process. The values that are set beforehand are called hyper-parameters. In the next section general ideas about neural networks will be discussed and the structure of CNNs will be explained.

1.2 Neural Networks

A neural network is a model that is inspired by the structure of the neurons in the brain of a living creature. These so-called neurons contain functions that transform the input according to some functions. After these operations, the data is passed to another neuron. There are layers to these ”neuron chains” and these layers can be seen in Figure 2. The input layer is the layer where the raw data is received and the output layer is where the desired output is obtained. The layers in between are called hidden layers and the amount determines the depth of the model.

Each connection between the neurons has a certain weight and through learning these weights are adjusted to achieve the desired outcomes. This process is done using optimisation algorithms such as gradient descent. There are many types of neural networks such as generative adversarial networks, feed-forward networks, and convolutional neural networks (CNNs). This study focuses on CNNs. The properties of CNNs will be further explained in the next section.

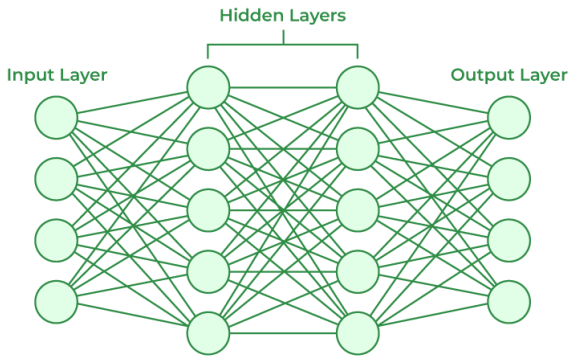


Figure 2: Basic neural network structure

1.2.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks are mainly used to process grid-like data. Because of this CNNs excel at image recognition tasks. Like in all neural networks CNNs have layers that are designed to do certain tasks. When data is going to be processed the input data hits the input layer. As the name suggests the input layer is the layer where the input data is stored. Since we are involved in an image recognition task our input data are the pixel values of the images in question. Depending on the data the size of the matrices and their dimensions can change. If we are dealing with an image recognition task that has coloured images the matrix may contain additional dimensions representing colours in red green and blue format.

After the input layer, the data passes through the convolutional layers hence the name convolutional neural network. In these layers features such as edges and and patterns are detected through the usage of kernels. Kernels, also called filters, are most commonly 2x2, 3x3, or 5x5-sized matrices that contain certain weight values. These kernels move over the input matrices and do certain operations using the weights, aiming to detect the aforementioned features. This movement is called stride and it starts from the upper right corner of the input matrix and ends in the lower right corner. In Figure 3 a representation of a stride operation can be seen The

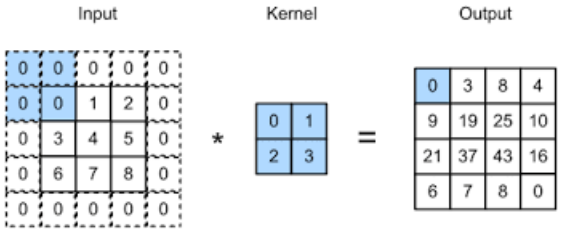


Figure 3: A 2x2 kernel operating over a matrix

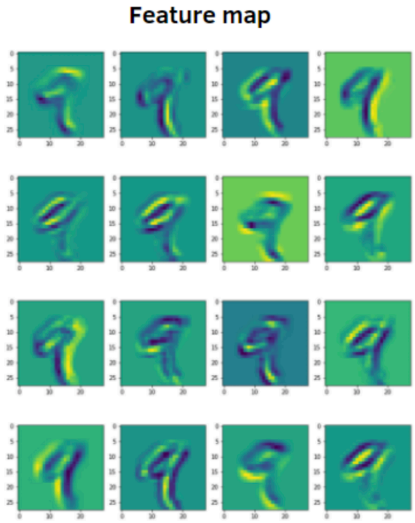


Figure 4: Feature Maps

number of cells for each stride operation is a hyper-parameter, just like the number of epochs, determined by the user. With each stride, the mentioned weights and the corresponding pixel values of the input data undergo a dot product. The resulting matrices formed are called feature maps and can be seen in Figure 4.

There are also activation functions introduced in these layers. These activation functions allow us to introduce non-linearity enabling the network to learn. Without this non-linearity the models would be shallow, rendering them unable to make complex decisions defeating the whole purpose of using such models. Some common activation functions are Rectified Linear Unit (ReLU) and tanh which are abundant in this study. In Figure 5 the graphic representation of these functions can be seen. If a value smaller than zero is introduced to a ReLU activation the output is always zero and if it is bigger it returns the same value. For the tanh activation function, the returned values are always between -1 and 1.

After the convolutional layers, the pooling layers are introduced. In these pooling layers, the main aim is to downsample the data to increase performance and computational efficiency and also to combat over-fitting. Some different types of pooling layers are average pooling and max pooling. In average pooling layers the average of

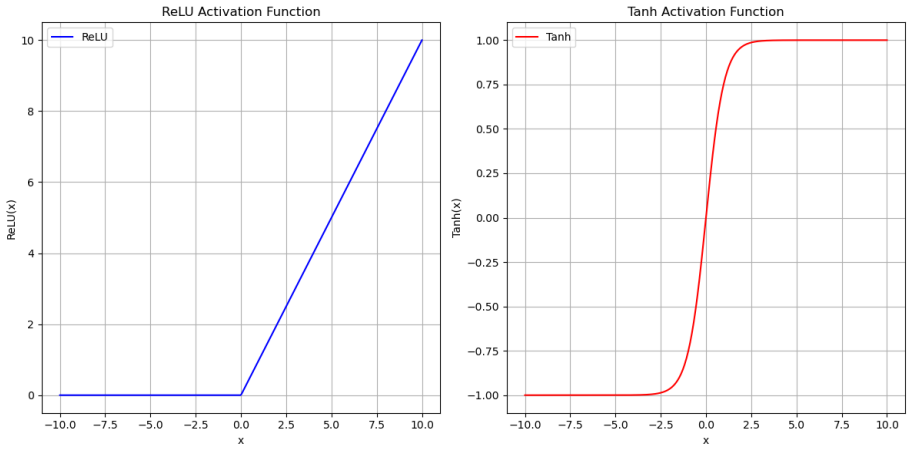


Figure 5: ReLU and Tanh Graphs

the pixel values are taken and for max pooling the highest value is taken as the new pixel value.

The last layers that are included in our models are fully connected layers. These layers are the layers where the final computation for the regression or classification problem. Each connection to this layer has a certain weight and bias learned through the aforementioned processes. With this layer, the data in question is labelled. In the next section, the differences between regression and classification tasks are going to be explained[1].

1.2.2 Regression Problems

A simple linear regression problem can be applied to a dataset that can be fitted to a traditional linear graph. In Figure 6 an example of a dataset that is suitable for linear regression can be seen.

By using specific cost functions and minimising them the model can evaluate its relevancy. Two common cost functions for linear regression problems are Mean Square Error (MSE) and Mean Absolute Error (MAE). Below the mathematical representation of these cost functions can be seen.

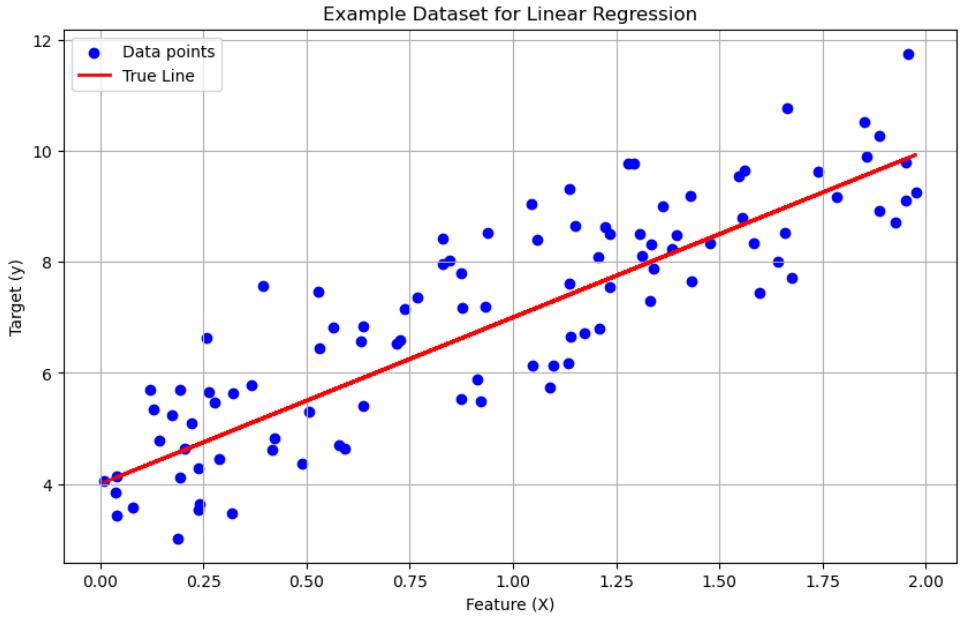


Figure 6: Linear Regression Graph

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2}$$

In MSE the square of the difference between the actual data and the predicted value is calculated. After this calculation, the average of this value is calculated. The ending result is the value that we want to minimise. Using MSE as the cost function allows us to penalise bigger errors more than the smaller ones leading to more generalised data. In contrast o MSE, MAE calculates the average of the absolute difference in the actual data and the prediction. This method is more suitable for data that contains outliers.

1.2.3 Classification Problems

Classification problems are a fundamental type of task in machine learning. These types of problems involve predicting labels for a given input data. Image classification

tasks are well-known examples of these types of problems. In a sense classification problems are specialised regression problems that have distinct decision boundaries. These decision boundaries can be determined by evaluating the results of the prediction. In addition to this, there can be scientific boundaries set by the problem's intrinsic properties, as is the case in our study. This boundary will be further explained in the section on the two-dimensional Ising model.

For classification problems, cross-entropy loss can be used as a cost function. Depending on the number of classes present in the problem we can either use binary cross-entropy loss (BCEL) or categorical cross-entropy loss (CCEL). As its name suggests binary cross-entropy loss is used when two classes are present and if there is a multi-class classification problem at hand categorical cross-entropy loss is used. Below the equations for these functions can be seen.

$$\text{BCEL} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right] \tag{3}$$

$$\text{CCEL} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(p_{i,c}) \tag{4}$$

For the BCEL equation y_i represents the true label, p_i represents the predicted probability and N represents the number of samples. In a sense predicted probability is a measurement of confidence. The same goes for CCEL except the calculation is done over multiple classes.

CHAPTER II

INTRODUCTION TO ISING MODEL

Ising model is a mathematical model, named after physicist Ernst Ising, that aims to explain physical phenomena in statistical mechanics. It has many application areas including ferromagnetism, neuroscience and even social dynamics. In this study, we will be focusing on the ferromagnetism aspect of the model and its relation to temperature.

2.1 One Dimensional Ising Model

One does not simply understand the two-dimensional Ising Model without first understanding the one-dimensional model. So before delving into the two-dimensional Ising model, we will explore the one-dimensional version. One dimensional Ising Model also known as Ising Chain is a more simplified version of the Ising Model. In this model, the atoms are assumed to be in a chain-like formation where each atom is neighbouring with the ones directly adjacent to them. Each atom can be in one of two states. These states are spin-up, represented by +1 and spin-down, represented by -1. In Figure 7 a one-dimensional Ising chain can be seen where the arrows represent the spin of each atom.

It is important to note that edge effects are neglected in this discussion; hence, the chain-like structure is assumed to be a loop. This simplification is intended to

Figure 7: 1-D Ising Chain

provide an intuitive understanding of the problem. The Hamiltonian H for the Ising model is given by:

$$H = -J \sum_{i=1}^{N-1} S_i S_{i+1} - h \sum_i S_i \tag{5}$$

however, since we assume there is no external magnetic field (represented by lowercase h to prevent any confusion with the Hamiltonian) the Hamiltonian of the Ising Chain will be as below:

$$H = -J \sum_{i=1}^{N-1} S_i S_{i+1} \tag{6}$$

where J is the coupling constant, S_i is the spin value and N is the number of atoms in the chain.

The coupling constant is a quantity that determines the interaction strength between the neighbouring spins. If the coupling constant is greater than 0 the interaction is ferromagnetic which means the spins want to align in the same direction and if it is smaller than 0 the interaction is paramagnetic.

One other expression that is important is the partition function. The partition function can be used to determine properties such as entropy and internal energy. Below the partition function can be seen.

$$Z = \sum_{\{s_i\}} e^{\beta H} \tag{7}$$

In this function β is the expression that represents the relation of the temperature of the system. In the equation k_B is the Boltzmann constant and T is the temperature of the said system.

$$\beta = \frac{1}{k_B T} \tag{8}$$

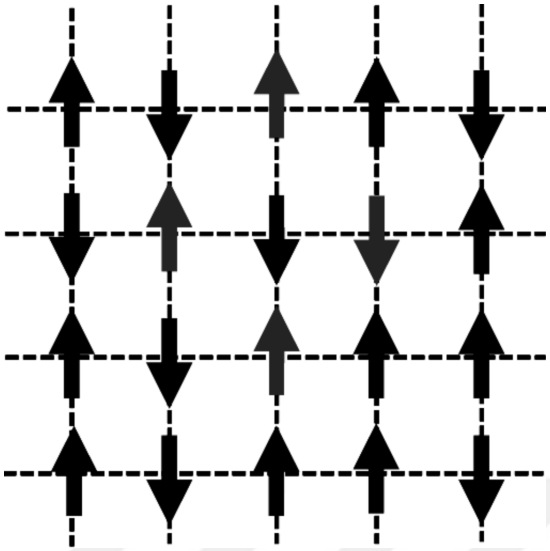


Figure 8: 2-D Square Lattice Structure

The solution of the one-dimensional Ising model reveals that there is no critical temperature at which ferromagnetism undergoes a transition. This is because the Ising chain lacks the complexity of systems such as the two-dimensional Ising model. The exact analytical solution will not be presented here, as it does not contribute to our understanding of the task at hand.

2.2 Two Dimensional Ising Model

The two-dimensional Ising Model is the focus of this study. This model consists of atoms aligned in a square lattice where each atom has four nearest neighbours all having the same distance between them. In Figure 8 a square lattice can be seen.

The Hamiltonian for the system can be seen below. Just like the assumption we made in the Ising Chain, we assume that there are no edge effects and no external magnetic field[2].

$$H = -J \sum_{i,j} (S_{i,j}(S_{i+1,j} + S_{i-1,j} + S_{i,j+1} + S_{i,j-1})) \tag{9}$$

The partition function and the expression for β are seen below.

$$Z = \sum_{\{S\}} e^{-\beta H(\{S\})} \tag{10}$$

$$\beta = \frac{1}{k_B T} \tag{11}$$

Charles Onsager analytically solved this problem and found a critical temperature value denoted with T_c. To find this value the following equation is solved. To reach this state transfer matrix method is used and the below results were achieved[3].

$$\sinh^2\left(\frac{J}{k_B T_c}\right) = 1 + \sqrt{2} \tag{12}$$

$$k_B T_c = \frac{2J}{\ln(1+\sqrt{2})} \tag{13}$$

By assuming normalised units the critical temperature is equal to:

$$T_c \approx \frac{2J}{0.8814 \times 1.380649 \times 10^{-23}} \tag{14}$$

$$T_c \approx 2.27 \tag{15}$$

The critical temperature marks the point at which phase transition occurs. This is the point where the system evolves from ferromagnetic to paramagnetic assuming the temperature is increasing[4]. This study aims to predict these three different temperature phases being; lower than critical, higher than critical and critical using machine learning methods.

CHAPTER III

METHODOLOGY

In this chapter, data generation and machine learning methods that are used in this study will be discussed. The algorithm used in the data generation process will be further explained, the application of convolutioıal neural networks will be shown and technical details will be given.

3.1 Data Generation

3.1.1 Monte Carlo Methods

There are various systems where mathematical solutions are too complex rendering them unsolvable by traditional methods. However, there is a principle that allows us to find precise probabilistic results for these problems. This principle is called the Monte Carlo principle and it will be used to generate the dataset we are going to use.

We can easily explain this principle by looking at a simple coin flip problem. A coin flip can either be heads or tails. Without even making any calculations we can easily say that the probability of getting a head is 1/2 and the same goes for getting tails. In a situation where one would use the Monte Carlo principle multiple events where the coin is flipped will be simulated. Every outcome is recorded and the number of target events is divided by the total number of events. As the number of events increases the aforementioned division will get closer to the theoretical value which is 1/2.

3.1.2 Metropolis Algorithm

In this study, a Monte Carlo algorithm written in Python using the NumPy library is utilised to generate Ising Model configurations in set temperatures[5]. When generating data the Metropolis Algorithm is utilised. The steps for a single event is below:

1. A temperature value is chosen to run the simulation.
2. A random 20x20 matrix is generated with each node having a spin value of either -1 or +1.
3. After the configuration matrix is generated a single node that represents an atom is chosen.
4. A spin flip on this node is proposed.
5. Change in energy is calculated after this proposal.
6. Depending on ΔE one of two scenarios happen.
    - If $\Delta E < 0$:
        - The proposed spin flip is accepted unconditionally.
        - The spin value at the chosen node is flipped.
    - If $\Delta E > 0$:
        - With a probability proportional to $\exp\left(-\frac{\Delta E}{k_B T}\right)$, where k_B is the Boltzmann constant and T is the temperature the spin is either accepted or rejected.
7. If the spin is accepted the new state is saved and the same process is repeated until the set number of moves are accepted.

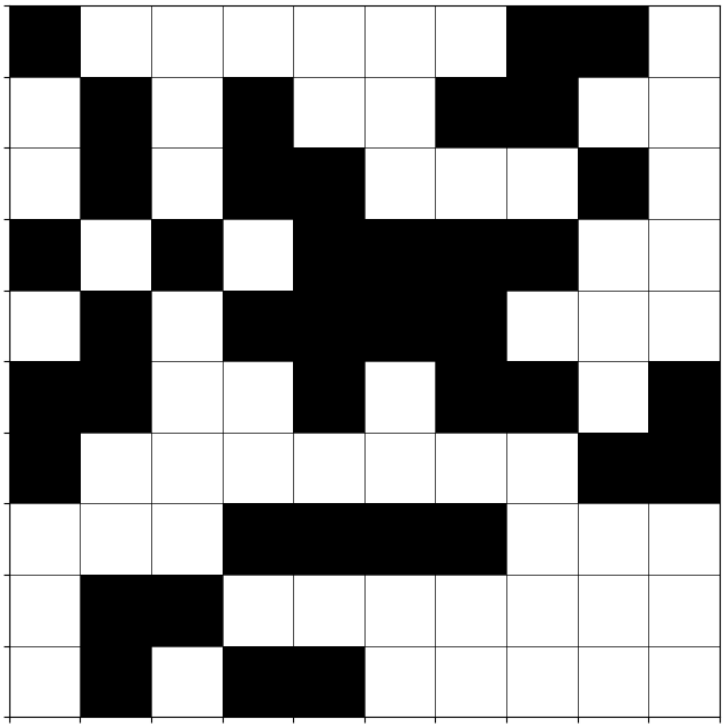


Figure 9: Ising Configuration

In Figure 9 a 20x20 configuration can be seen. To visualise these matrices easily -1 spins are coloured white and +1 spins are coloured black.

To ensure the physical relevance of the simulation magnetisation is calculated for each configuration. In the graph below (Figure 10) magnetisation changes exactly at the critical temperature. Since this is consistent with the Onsager solution we can say that our simulation is working correctly. To calculate magnetisation the following formula is used[6].

$$M = \sum_{i=1}^{N} S_i \tag{16}$$

After each run the last configuration is saved as a csv file in order to use as inputs for training data. Certain temperature points are chosen to add variety to data[7]. The chosen temperature points are going to be further discussed in the training section.

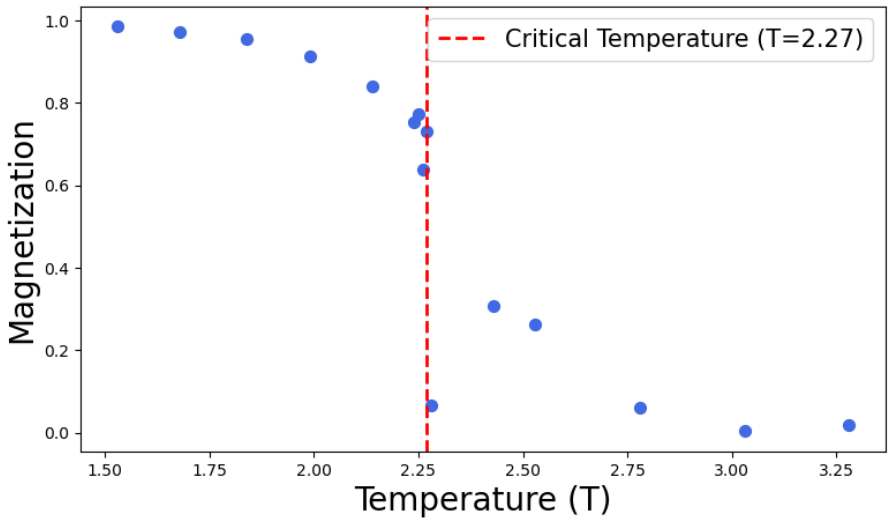


Figure 10: Magnetisation Graph for Monte Carlo simulations

3.2 Training The Neural Network

3.2.1 Problem Type: Regression or Classification?

To decide on what type of approach we are going to use we must first determine the advantages and disadvantages of the types of approaches we might use. There are two approaches suitable for this problem either regression or classification. The main ideas were discussed in the section about machine learning methods. In this section, only the advantages and disadvantages of each approach will be discussed, along with the reasoning behind favouring one approach over the other.

If we were to prepare the dataset for a supervised learning problem with the regression approach we would label each configuration with their corresponding temperature value. This is a very simple task since we are generating data using simulations. One of the biggest advantages of a regression approach is that it gives precise predictions. Rather than having classes for results, the model gives exact temperature values and linear results. For the classification approach to our problem, we would get outcome labels such as lower than critical temperature, critical temperature and higher than critical temperature. Deciding on where to set the boundaries is also a challenge that should be tackled.

It may seem like the regression approach is significantly better however our problem at hand eliminates most of the disadvantages presented. Getting labels instead of exact temperature values might look less tempting however getting labels allows us to use this training in different types of lattice structures such as triangle lattices. This allows the usage of current training data for further research. The decision boundary problem is also easily resolved because the problem at hand has clear theoretical solutions and involves a physical system. It is straightforward to define three classes: one for the critical temperature and the others for temperatures below and above the critical value. The critical temperature value, which is already determined by Onsager, is used for labelling configurations. So the type of approach we are going to use is classification [8][5].

3.2.2 Data Preparation

To feed the data into a convolutional neural network we must first prepare the data for upcoming processes. To ensure that the temperature labels are true, the configuration files were named in such a way that includes the temperature values in the name of the CSV file. After each file is matched with its corresponding temperature, temperature classes are set. For temperature values below 2.24, the label is set to be 1 which means lower than critical temperature. For temperatures higher than 2.28, the label is set to be 2 which means higher than critical temperature. Lastly, for temperature values between 2.24 and 2.28, the label is set to be 0 which indicates that these configurations are at critical temperature. When deciding these boundaries previous work done is considered[5]. The interval for lower and higher temperatures is significantly greater than the critical region so if we were to divide the whole temperature region equally there would be imbalances in the data where the configurations under critical temperature would be sparse in contrast to the other ones. To ensure that there are no imbalances in the data 5 different temperature points were chosen for

	Configuration	Temperature	Label
0	[[1.0, 1.0, 1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1....	2.26	0
1	[[1.0, -1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, ...	2.53	2
2	[[-1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1...	2.14	1
3	[[-1.0, -1.0, -1.0, 1.0, -1.0, -1.0, -1.0, -1....	2.14	1
4	[[-1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 1.0, 1.0...	2.24	0
...	...	...	...
29995	[[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,...	1.99	1
29996	[[-1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1...	2.14	1
29997	[[-1.0, -1.0, -1.0, -1.0, -1.0, 1.0, -1.0, -1....	2.43	2
29998	[[1.0, 1.0, -1.0, -1.0, 1.0, -1.0, 1.0, 1.0, 1...	3.03	2
29999	[[1.0, 1.0, 1.0, 1.0, -1.0, -1.0, 1.0, 1.0, 1....	3.28	2

30000 rows × 3 columns

Figure 11: Configurations and Temperature Labels Data Frame

each interval. With this, we will have equal amounts of configurations for each class at the end of data generation. In Figure 10 a section of the data frame after labeling is done can be seen.

3.2.3 Training and Architectures

In order to train and evaluate the data we should split the data into training and testing parts. The testing parts should not be seen by the networks so we ensure that the model can generalise and not ”memorise” the given data. The split ratio is 80 % training and 20 % testing. A separate validation dataset is not used due to the small dataset size. Splitting further would reduce the training data, impacting learning.[9] Instead, model performance is monitored throughout training, and testing data is reserved for final evaluation.

There are three architectures used in this study. Each of these architectures is designed for image classification tasks which is suitable for our study. The architectures used are ”LeNet5”, ”AlexNet” and ”ResNet18”. Below the brief explanations and summaries for each architecture can be seen. To implement these architectures Keras Deep Learning Library was used.

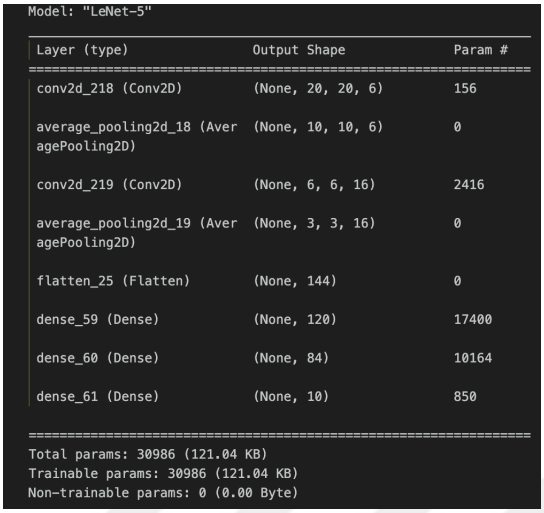


Figure 12: LeNet5 - Summary

3.2.3.1 LeNet5

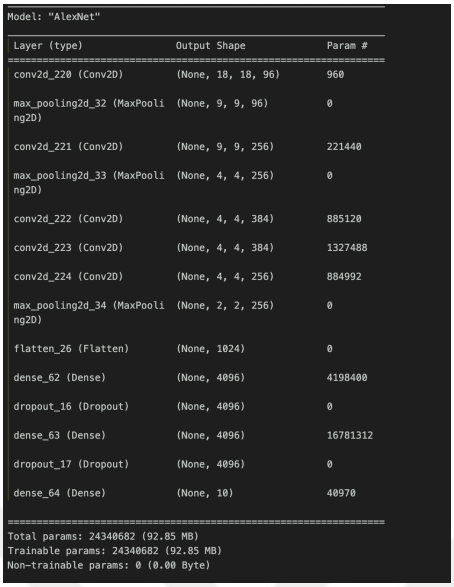
LeNet5 is an architecture that was first created for the MNIST dataset which consists of handwritten digits. The neural network consists of two convolutional layers followed by sub-sampling layers, and three fully connected layers. It uses tanh activation functions and average pooling. In Figure 12 the summary of the model can be seen.[10]

3.2.3.2 AlexNet

AlexNet was the winner of the ImageNet competition in 2012. It consists of five convolutional layers followed by three fully connected layers, utilizing ReLU activations and max pooling. In Figure 13 the summary of the model can be seen.[11]

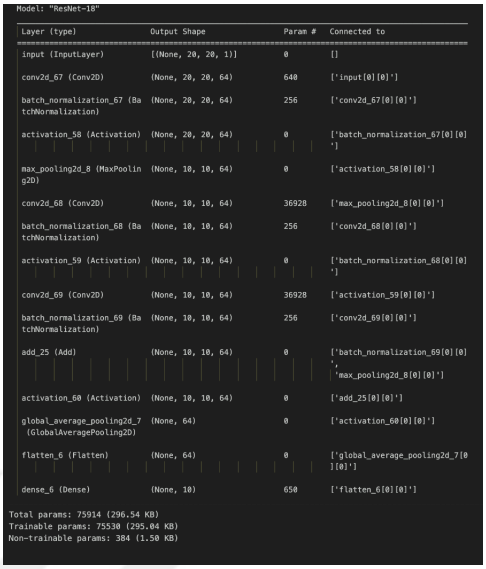
3.2.3.3 ResNet18

ResNet18, short for Residual Network, is a deep neural network that aims to solve the vanishing gradient problem in networks with many layers. It consists of residual blocks and various convolutional and pooling layers. In Figure 14 the summary of the model can be seen.[12]



```
Model: "AlexNet"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_220 (Conv2D)         (None, 18, 18, 96)        960

 max_pooling2d_32 (MaxPooli  (None, 9, 9, 96)          0
 ng2D)

 conv2d_221 (Conv2D)         (None, 9, 9, 256)         221440

 max_pooling2d_33 (MaxPooli  (None, 4, 4, 256)         0
 ng2D)

 conv2d_222 (Conv2D)         (None, 4, 4, 384)         885120

 conv2d_223 (Conv2D)         (None, 4, 4, 384)         1327488

 conv2d_224 (Conv2D)         (None, 4, 4, 256)         884992

 max_pooling2d_34 (MaxPooli  (None, 2, 2, 256)         0
 ng2D)

 flatten_26 (Flatten)        (None, 1024)              0

 dense_62 (Dense)            (None, 4096)              4198400

 dropout_16 (Dropout)        (None, 4096)              0

 dense_63 (Dense)            (None, 4096)              16781312

 dropout_17 (Dropout)        (None, 4096)              0

 dense_64 (Dense)            (None, 10)                40970

=================================================================
Total params: 24340682 (92.85 MB)
Trainable params: 24340682 (92.85 MB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 13: AlexNet - Summary



```
Model: "ResNet-18"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input (InputLayer)          [(None, 20, 20, 1)]       0         []

 conv2d_67 (Conv2D)          (None, 20, 20, 64)        640       ['input[0][0]']

 batch_normalization_67 (Ba  (None, 20, 20, 64)        256       ['conv2d_67[0][0]']
 tchNormalization)

 activation_58 (Activation)  (None, 20, 20, 64)        0         ['batch_normalization_67[0][0
                                                                 ']

 max_pooling2d_8 (MaxPoolin  (None, 10, 10, 64)        0         ['activation_58[0][0]']
 g2D)

 conv2d_68 (Conv2D)          (None, 10, 10, 64)        36928     ['max_pooling2d_8[0][0]']

 batch_normalization_68 (Ba  (None, 10, 10, 64)        256       ['conv2d_68[0][0]']
 tchNormalization)

 activation_59 (Activation)  (None, 10, 10, 64)        0         ['batch_normalization_68[0][0
                                                                 ']

 conv2d_69 (Conv2D)          (None, 10, 10, 64)        36928     ['activation_59[0][0]']

 batch_normalization_69 (Ba  (None, 10, 10, 64)        256       ['conv2d_69[0][0]']
 tchNormalization)

 add_25 (Add)                (None, 10, 10, 64)        0         ['batch_normalization_69[0][0
                                                                 ]',
                                                                  'max_pooling2d_8[0][0]']

 activation_60 (Activation)  (None, 10, 10, 64)        0         ['add_25[0][0]']

 global_average_pooling2d_7  (None, 64)                0         ['activation_60[0][0]']
  (GlobalAveragePooling2D)

 flatten_6 (Flatten)         (None, 64)                0         ['global_average_pooling2d_7[0
                                                                 ][0]']

 dense_6 (Dense)             (None, 10)                650       ['flatten_6[0][0]']

Total params: 75914 (296.54 KB)
Trainable params: 75530 (295.04 KB)
Non-trainable params: 384 (1.50 KB)
```

Figure 14: ResNet18 - Summary

3.2.4 Hyper-parameter Tuning

In order to train these neural networks with maximum efficiency and success some hyper-parameters should be determined. These hyper-parameters are learning rate, batch size and number of epochs. To tune these parameters a Python library called Optuna is used. The range of hyper-parameters is; between 1×10^{-5} and 1×10^{-2} for learning rates 16, 32 or 64 for batch size and between 15 and 100 for the number of epochs.

3.2.4.1 Learning Rate

Learning rate is the parameter that determines the size of steps taken that aim to minimise the loss function. Higher learning rates will cause the model to train faster however, this can cause the model to settle on a lesser solution. On the other hand, a very low learning rate might cause very long training times and may cause it to get stuck in a local minima so a balance should be found which is done by Optuna in this case.

3.2.4.2 Batch Size

Batch size is the parameter that determines the number of training samples processed before the model updates itself. Bigger batch sizes require more batch sizes and are faster however they can again lead to sub-optimal results.

3.2.4.3 Epochs

Number of epochs is the number of times the data is processed through the network. As the number of epochs increases over-fitting may occur so it is best to optimise it just like the other parameters.

CHAPTER IV

RESULTS

4.1 Performance Metrics

To measure the success of each model certain parameters are used. These parameters are accuracy, F1 score and confusion matrices. The properties of these parameters are explained in the section below.

4.1.1 Accuracy

Accuracy is one of the most straightforward metrics and it gives us the ratio of true predicted labels to the total number of configurations. Below the equation for accuracy can be seen.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Configurations}} \tag{17}$$

4.1.2 F1 Score

F1 score is a combination of two other metrics and is used widely in classification tasks as an evaluation metric. The two metrics it combines are precision and recall.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positives} + \text{False Positives}} \tag{18}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{19}$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{20}$$

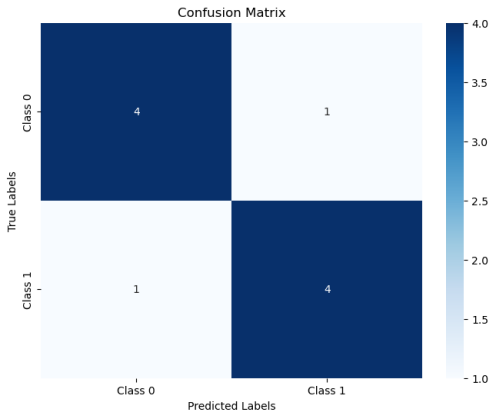


Figure 15: Example Confusion Matrix

For multi-class classification problems, one could calculate the F1 score for each class individually which is the approach used in this study.

4.1.3 Confusion Matrix

A confusion matrix is a visual metric that allows us to measure how many instances of data are predicted correctly and incorrectly. Ideally, the diagonal from the upper left to the lower right of the matrix, representing correct classifications, should be filled with higher values, indicating accurate predictions. This means that the predictions for these classes are correct. In Figure 15 an example confusion matrix can be seen.

4.2 Evaluation of Models

Each model was trained using a dataset comprising 30,000 Ising Model configurations, with an equal number of configurations at each temperature group to prevent imbalances. The model hyper-parameters were optimised using the Python library Optuna.

4.2.1 LeNet5

As a basic model for more complex tasks, LeNet5 could be expected to perform poorly. However, since the images that we provide are only black and white images with sparse pixels LeNet5 performed well enough. Also, the relatively small size of

the model allowed the training process to be much quicker. The model was trained with 32 batch size, approximately 0.0012 learning rate and, 65 epochs. By looking at the confusion matrix in Figure 19 the model can be considered successful. Even though the model is fairly successful among all classes it struggled to differentiate below critical temperature configurations. This could be because of the more ordered nature of the spins at low temperatures. The more ordered spin structure may cause the model to have fewer features such as edges to detect. By looking at the loss and accuracy graphs in Figures 16 and 17 respectively, it is clear that the model learned the training dataset as well as it could and performed according to its capacity. In Figure 18 the F1 score of each class is seen and as mentioned LeNet5 struggled to differentiate between below critical class and the others. Overall it can be said that the model was fairly successful since the data is clustered around the diagonal of the confusion matrix.

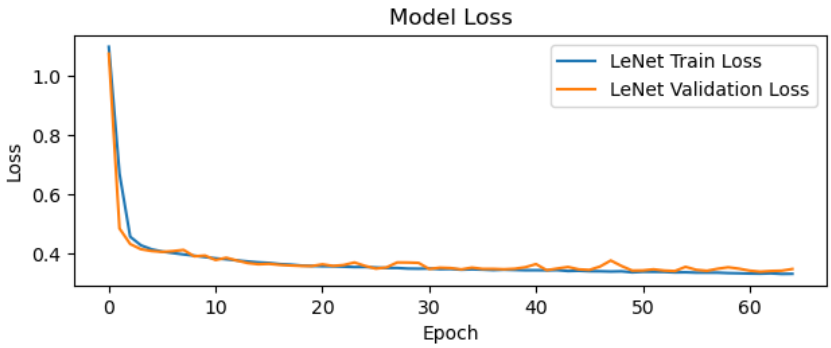


Figure 16: LeNet5-Loss

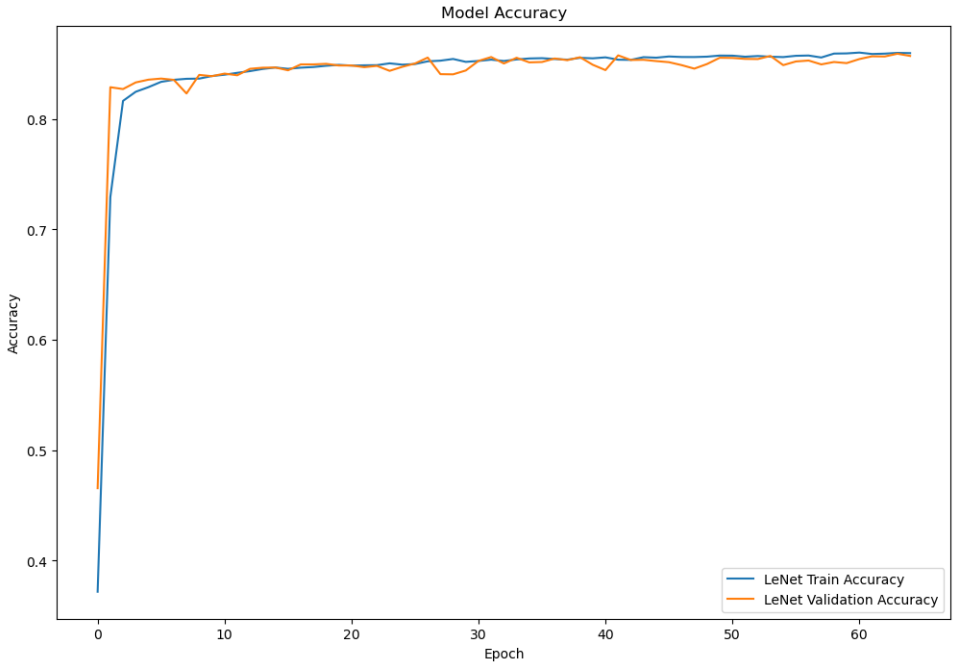


Figure 17: LeNet5 - Accuracy

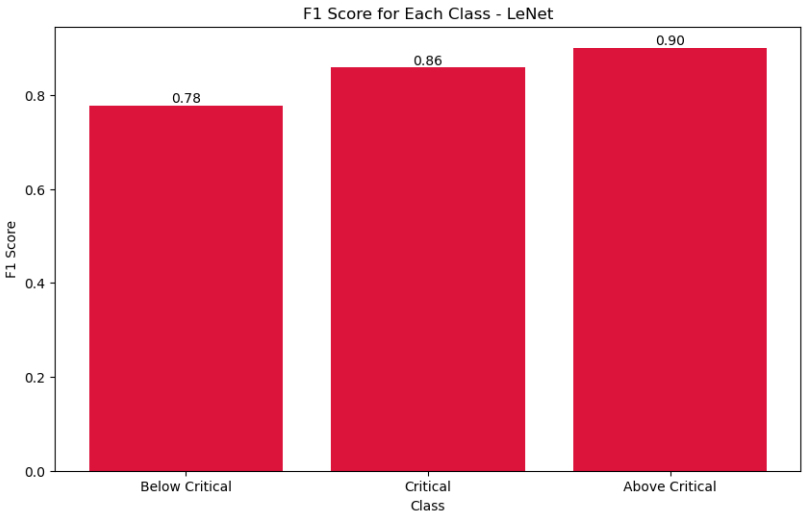


Figure 18: LeNet5-F1 Score

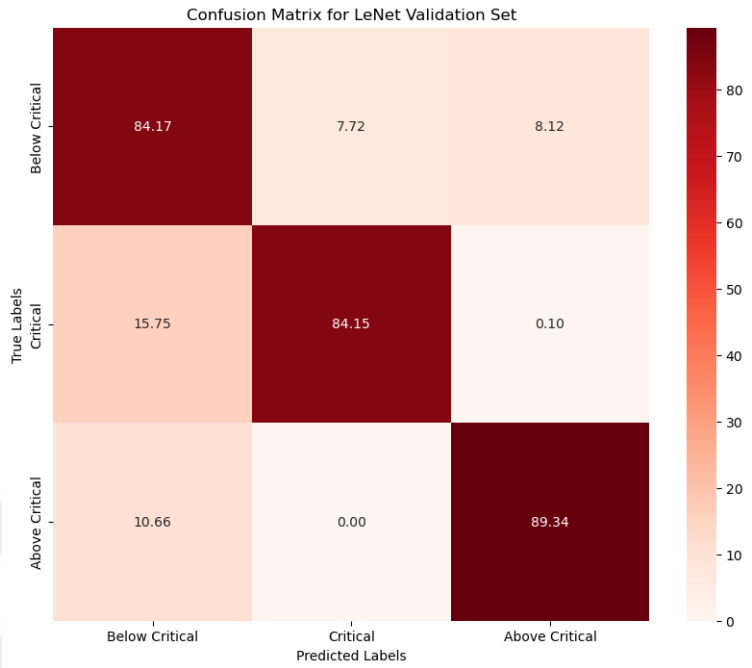


Figure 19: LeNet5 - Confusion Matrix

4.2.2 AlexNet

A more complex model, AlexNet, was also trained. Given its increased complexity compared to LeNet5, one would typically expect AlexNet to perform better. However, the model exhibited over-fitting and failed to generalize well to the test data. This over-fitting may be due to several factors. While hyper-parameter tuning was performed using Optuna, making this an unlikely cause, the issue may stem from an insufficient dataset size. The training procedure involved a batch size of 16, a learning rate of approximately 0.00010, and 95 epochs. The model's complexity, combined with a limited amount of data, might lead to it learning the training data too well and not generalising. As it is seen in the loss graph and accuracy graph in Figure 20 and 21 respectively, the model is over-fitting and failing to generalise in the test set. This is due to the small size of the images and the few number of training images.

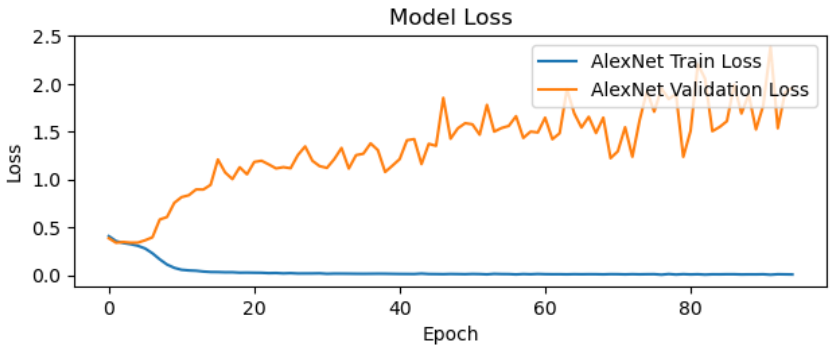


Figure 20: AlexNet -Loss

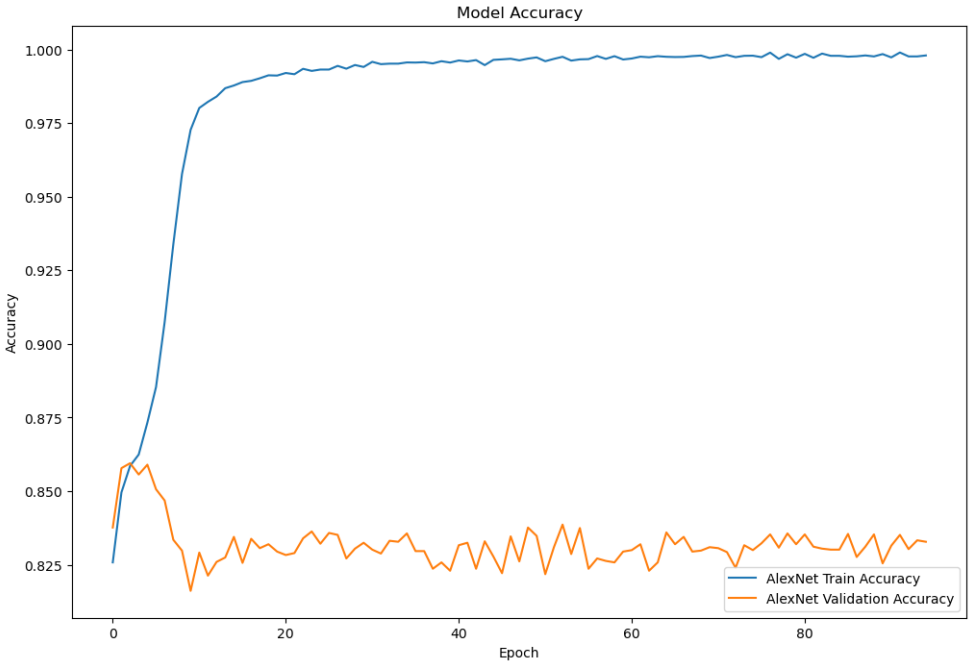


Figure 21: AlexNet - Accuracy

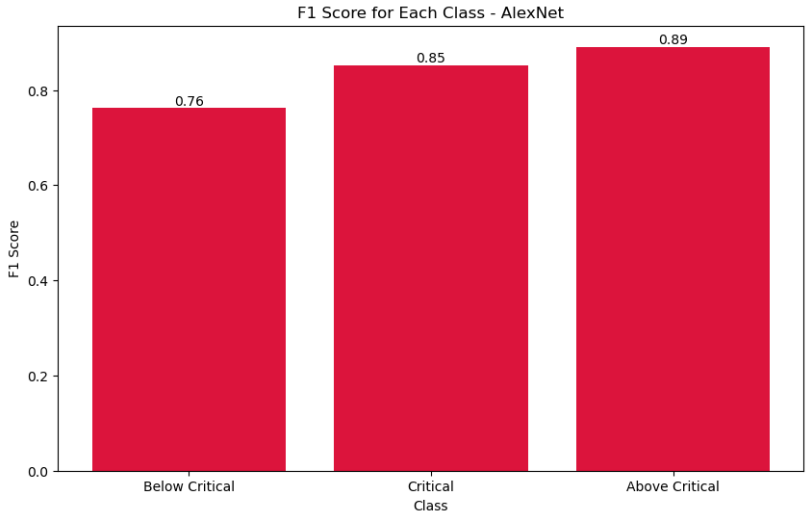


Figure 22: AlexNet -F1 Score

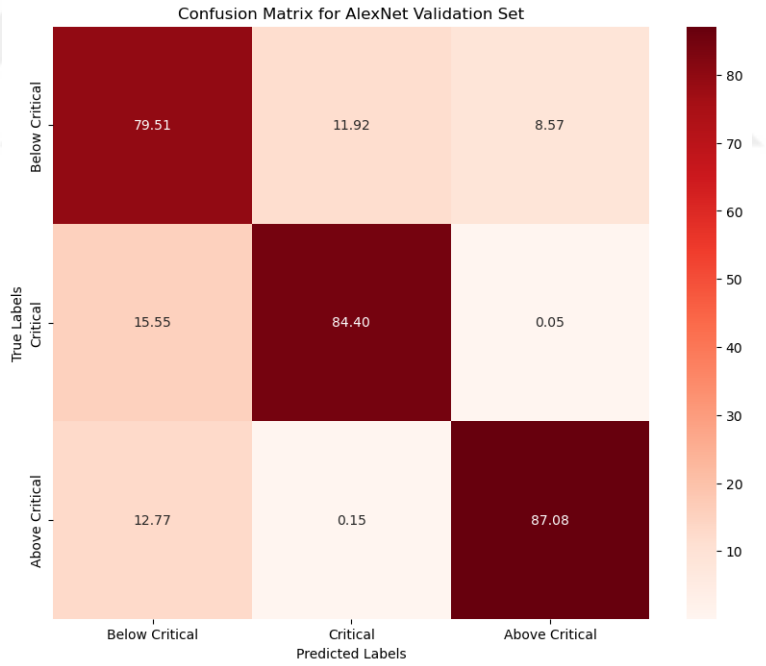


Figure 23: AlexNet - Confusion Matrix

4.2.3 ResNet18

Thirdly, the most complex model, ResNet18, was trained. Due to its residual blocks and overall complexity, better performance was expected compared to the previous

models. The training parameters included a batch size of 64, a learning rate of approximately 0.0033, and 37 epochs. However, similar to AlexNet, ResNet18 did not achieve the anticipated success, particularly in the "below critical" temperature region. This under performance is likely for the same reasons as AlexNet: either insufficient hyper-parameter tuning or a lack of sufficient data, leading the model to over-fit as it can be seen in the loss and accuracy graphs in Figure 24 and 25 respectively.

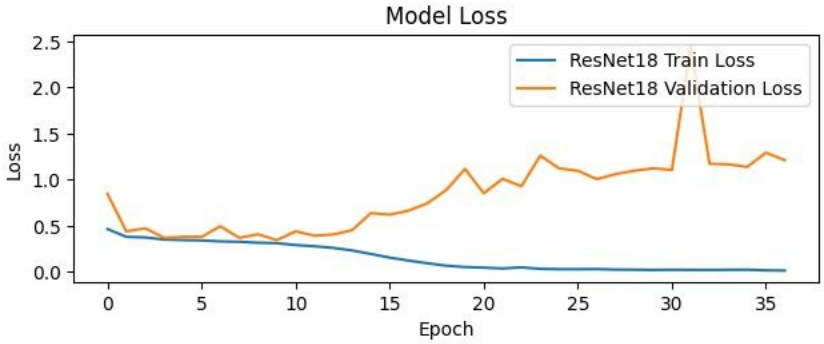


Figure 24: ResNet18 - Loss

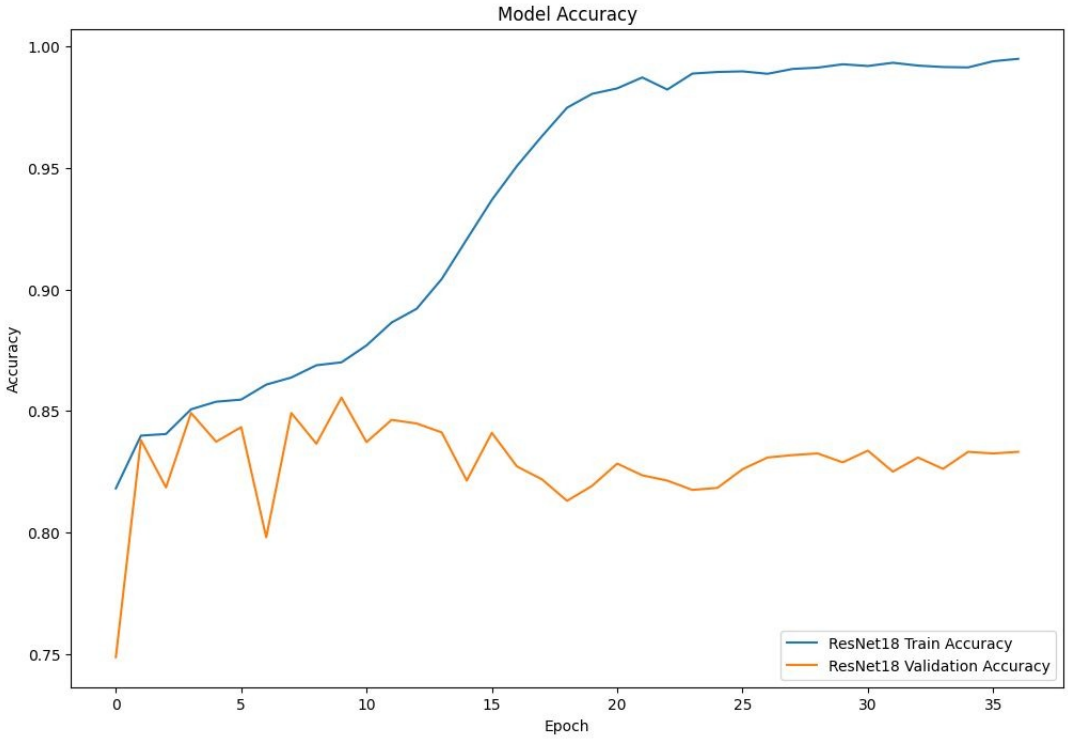


Figure 25: ResNet18 - Accuracy

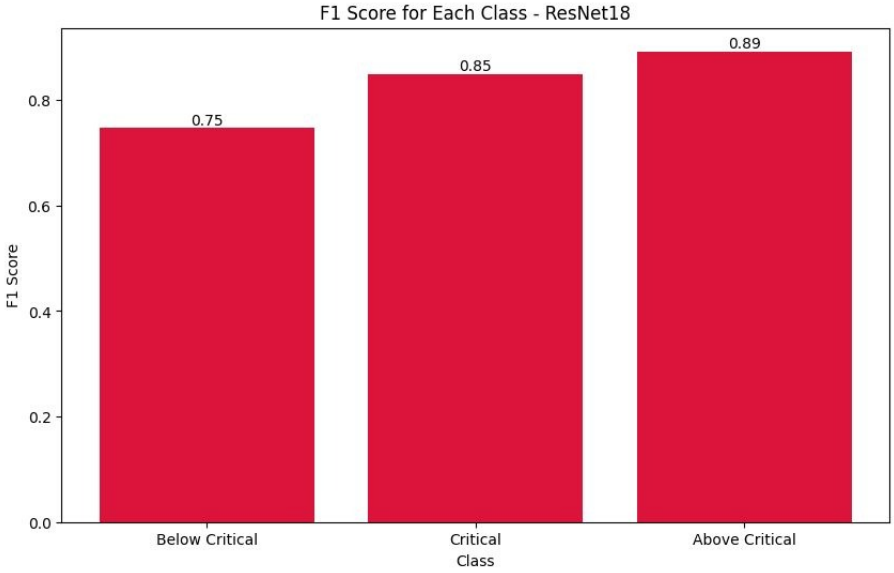


Figure 26: ResNet18 - F1 Score

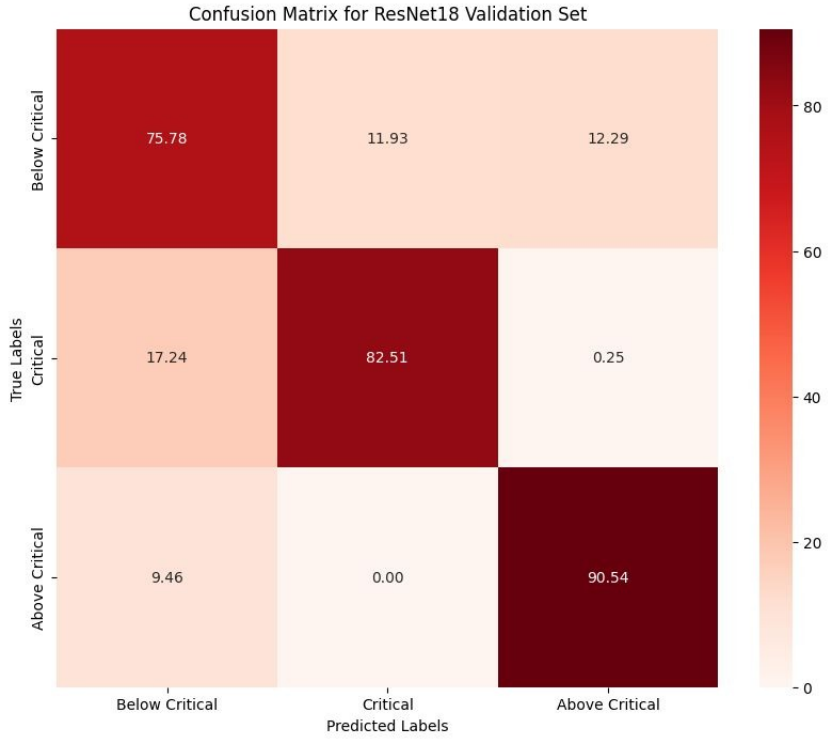


Figure 27: ResNet18 - Confusion Matrix

In summary, while LeNet5, AlexNet, and ResNet18 each had distinct advantages, they did not consistently outperform one another. LeNet5 performed adequately across all classes despite its simpler architecture, indicating that model simplicity can be effective with appropriate data. Also the simplicity of the model allowed for easier training with less computational resources. In contrast, both AlexNet and ResNet18, despite their complexity, struggled particularly in identifying configurations below the critical temperature, likely due to over fitting caused by a limited dataset. This suggests that increasing model complexity does not necessarily yield better results and highlights the importance of balancing model complexity with data quantity and quality.

CHAPTER V

CONCLUSIONS

5.1 Overview

In this study, machine learning methods were used in temperature-based analysis of a complex physical system known as the Two-Dimensional Ising Model. A dataset that consists of two dimensional Ising model configurations was generated using Python. The dataset is ensured to be physically accurate by the use of the Metropolis Algorithm. Also, the Onsager solution is used to theoretically check the relevance of the dataset. For the machine learning section of the study, three convolutional neural networks were used to tackle the classification task. Using these networks the temperature groups that the configurations belong to were predicted. The results of these predictions were evaluated using metrics that are suitable for multi-class classification problems. This multi-disciplinary approach to this complex physical problem allows us to have a better understanding of the ferromagnetic properties of matter under different temperature conditions.

5.2 Improvements

As in all studies, there are many different improvements to this one. To better understand the behaviour of the system larger lattice sizes could be used or even different types of lattice structures, such as triangle lattice, can be observed. Different architectures and their performance on the dataset can also be evaluated. The number of classes could be increased leading to accurate results. This problem could also be handled as a regression problem allowing us to get exact temperature results. To - make the dataset more physically accurate secondary neighbour interactions can be

used.

5.3 Final Remarks

While working on this study I have learned about a completely different area of research. As a physics major, working with machine learning methods broadened my horizons. Also working on the Ising Model which has been an interesting concept for me since my bachelor years was an exciting experience. I hope the skills I have learned here will help allow me to approach complex physical problems more creatively.

REFERENCES

[1] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015.

[2] M. Kardar, *Statistical Physics of Particles*. Cambridge University Press, 2007.

[3] C. Kittel and H. Kroemer, *Thermal Physics*. W.H. Freeman and Company, 2nd ed., 1980.

[4] D. Tong, *Statistical Mechanics*. Cambridge University Press, 1st ed., 2018.

[5] B. Civitcioğlu, R. A. Römer, and A. Honecker, "Machine learning the square-lattice ising model," *Journal of Physics: Conference Series*, vol. 2207, p. 012058, mar 2022.

[6] B. A. Cipra, "An introduction to the ising model," *The American Mathematical Monthly*, vol. 94, no. 10, pp. 937–959, 1987.

[7] X. Li, R. Guo, Y. Zhou, K. Liu, J. Zhao, F. Long, Y. Wu, and Z. Li, "Machine learning phase transitions of the three-dimensional ising universality class," 2021.

[8] N. Walker, K.-M. Tam, and M. Jarrell, "Deep learning on the 2-dimensional ising model to extract the crossover region with a variational autoencoder," *Scientific Reports*, vol. 10, August 2020.

[9] R. Zhang, B. Wei, D. Zhang, J.-J. Zhu, and K. Chang, "Few-shot machine learning in the three-dimensional ising model," *Phys. Rev. B*, vol. 99, p. 094427, Mar 2019.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.