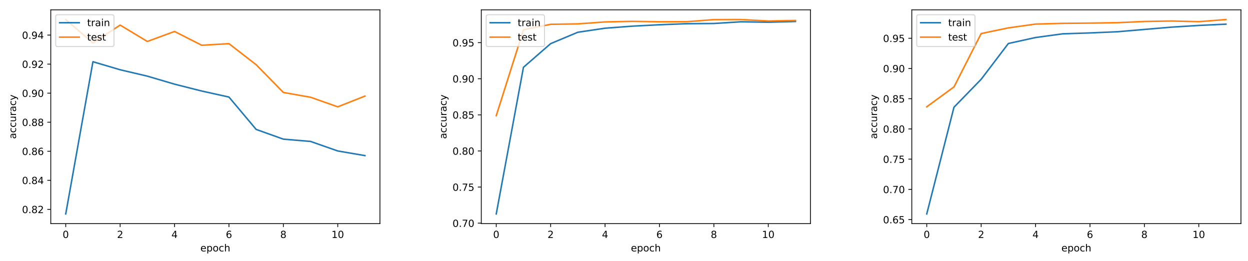
Convolutional Neural Networks
Assignment 2 Part 1

Convolutional Neural Networks (CNN) are a class of deep neural networks suitable for use on image data. CNNs use a layer of convolutional filters to reduce the dimensionality of the image data before feeding it to fully connected layers to classify the images. The weights in these filters aswell as the weights in the fully connected layers are being updated. We look at classifying the MNIST dataset using a CNN. Parts of the code used were adapted from [1]. This code uses 2 convolutional layers, a max pooling layer and 2 fully connected layers. 2 dropout layers are used to reduce over fitting.

The main parameters are the number of epochs, the size of the batches used to update the weights, the size and stride of the kernels used in the convolutional layers and the size of the pooling layers. Using the parameters as used in [1] we achieve an accuracy of 99% on the test set which is a big improvement over the accuracy achieved by the single layer perceptron. In order to study what varying the parameters does the model we look at a simplified network consisting of only 1 convolutional layer, 1 pooling layer and 1 fully connected layer.

When we run this simplified network using the same parameters as the complete network we get an accuracy of 98.1%. In this run the batch size was 128 (figure 1b), when we look at a smaller batch size of 1 or a bigger batch size of 1024 we can look at what happens to the accuracy of the model (figure 1). We can see that for the smaller batch size the accuracy goes down and the convergence becomes more noisy. For the bigger batch size we see the convergence becomes slower. The smaller the batch size the longer the program takes to run.



(a) *Batch size 1.* **(b)** *Batch size 128.* **(c)** *Batch size 1024.*

Figure 1: *The accuracy for the train and test set for each epoch for the simplified CNN network. Batch size as in subcaption.*

References

[1] Towards Datascience: Build your own convolutional neural network in 5 minutes, Rohith Gandi, `https://towardsdatascience.com/build-your-own-convolution-neural-network-in-5-mins-4217c2cf964f`, retrieved on 2019-04-10.

Autoencoders
Assignment 2 Part 1

For many image processing problems we need to reduce the feature space of the images. One way to implement that is with the use of autoencoders. First we use a network to encode the images and then we use a network to decode these images with a lower feature space.

We look at using an autoencoder to reduce the feature size of the MNIST digits (figure 1). The code used for this was adapted from [1]. When we apply this autoencoder the resulting images are sparser than then input images so the feature space of the images is reduced. The cost of this reduction is that we lose some information which is visible in figure 1 in the loss of clarity in the images.

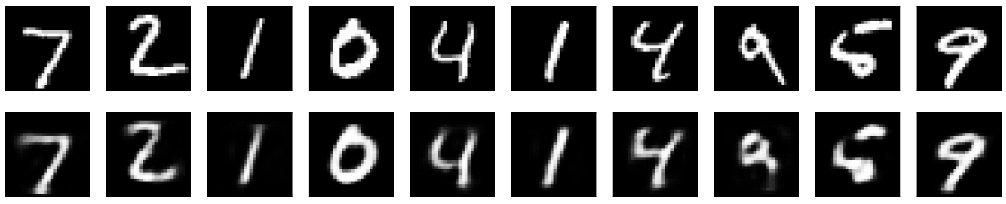


Figure 1: *The input and the output of the autoencoder. The loss of this autoencoder is 0.10 so we lose quite a lot of information.*

References

[1] The Keras Blog: Building Autoencoders in Keras, `https://blog.keras.io/building-autoencoders-in-keras.html`, retrieved on 2019-04-10.

Recurrent Neural Networks

Assignment 2 Part 1

Recurrent Neural Networks are a class of neural networks useful to study sequences of inputs. In order to gain some insight into RNN we look at performing addition using a RNN. The code to perform this task was adapted from [1]. Just feeding a sum with the desired results does not necessarily result in a sequence to study using a RNN. Therefore we feed the RNN the sum symbol by symbol.

First we generate a set of sequences describing our addition problem, for example ['3','2','+','7'] with output ['3','9']. We train the RNN on these sequences and then test its accuracy. After 10 epochs of training we achieve an accuracy of 99.5 %.

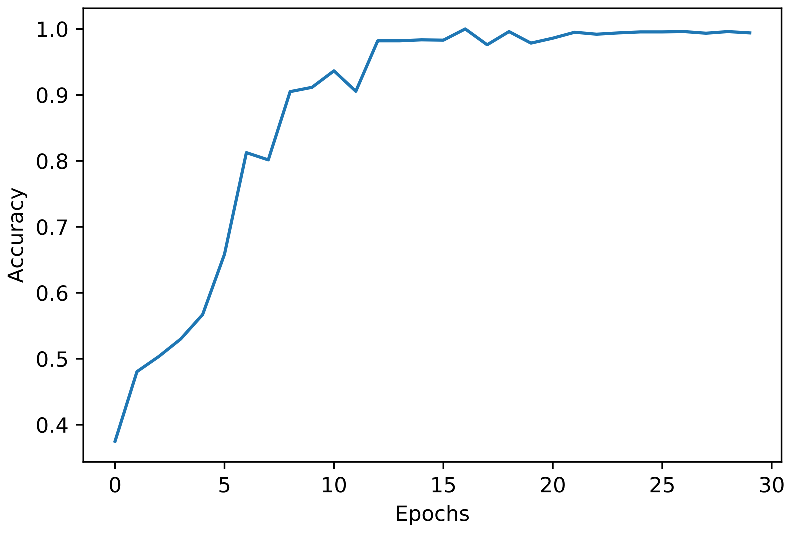


Figure 1: *Accuracy vs. the number of epochs for the addition problem.*

References

[1] Machine Learning Mastery: Learn to Add Numbers with an Encoder-Decoder LSTM Recurrent Neural Network, `https://machinelearningmastery.com/learn-add-numbers-seq2seq-recurrent-neural-networks/`, retrieved on 2019-04-12.

Studying the Phase Transition in the 2D Square Lattice Zero Field Ising Model using a Convolutional Neural Network

Assignment 2 Part 2

Abstract

The 2D Ising model has a phase transition at a finite temperature T_C. The conventional way to study this phase transition is using Markov Chain Monte Carlo (MCMC) to simulate the model at a range of temperatures. In this report we try to use a Convolutional Neural Network to determine the critical temperature by classifying configurations at labelled temperatures generated by MCMC. From the weights of this classification we determined the critical temperature at $k_BT_C/J = 2.23(5)$ which agrees with the analytical results of $k_BT_C/J = \frac{2}{\ln(1+\sqrt{2})} \simeq 2.2691\ldots$ [1].

1 Introduction

In theoretical physics we study simplified models of nature. One of the widely used models in theoretical physics is the Ising model. This model has a phase transition at a finite value of the order parameter. The conventional way to study this phase transition is by simulating the Ising model using Monte Carlo algorithms to evolve the system from the initial values at a given temperature. It might be insightful to reverse this process (ie. determining the temperature given a final state of the system like a kind of thermometer). The Monte Carlo algorithms commonly used to study the Ising model are irreversible so those can not easily be used to study the reverse of the simulation.

2 Theoretical Background

The 2D Ising model consist of spins arranged on a regular 2D lattice. These spins can have a value of -1 or 1. These spins could represent the spin of atoms in a metal. The energy of this system is given by the Hamiltonian

$$H = -J\sum_{\langle i,j\rangle}\sigma_i\sigma_j - \mu h\sum_i \sigma_i \tag{1}$$

where J is the coupling constant between spins, σ_i the spin at lattice site i, μ the coupling constant between the spins and the external field and h the magnitude and the direction of the external magnetic field. The sum $\sum_{\langle i,j\rangle}$ goes over all the nearest neighbour pairs. In order to reduce the complexity of the phase transition we look at the Ising model without external field so we set $h = 0$ which results in the Hamiltonian

$$H = -J\sum_{\langle i,j\rangle}\sigma_i\sigma_j \tag{2}$$

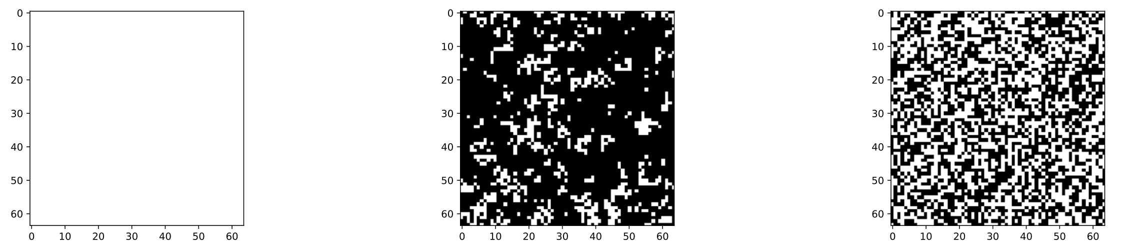
The conventional way to computationally study the 2D Ising model is by using Markov Chain Monte Carlo (MCMC) algorithms. The MCMC algorithm commonly used is the Metropolis algorithm which works by choosing a random lattice site, flip that spin and calculate what that does to the total energy of the system. If the energy decreases accept the flip outright and when the energy increases accept it with the chance given by

$$P = e^{-\beta\Delta H} \tag{3}$$

with $\beta = 1/k_BT$ with k_B the Boltzmann constant and T the temperature. ΔH is the energy change due to this flip. After taking a number of spins the system equilibrates to a final configuration. At low temperatures the chance of flipping a spin become very small so the system goes to either all spins at -1 or 1. When the temperature increases the chance of flipping a spin becomes higher so at high temperatures the final configuration is a random configuration with magnetisation[1] going to zero. At a finite temperature the system crosses over from the low to the high temperature limit. The temperature at which this phase transition occurs is called the critical temperature T_C. Analytically the critical temperature is found to be $k_BT_C/J = \frac{2}{\ln(1+\sqrt{2})} \simeq 2.2691\ldots$[2].

[1]The magnetisation is defined as the total sum over all spins

[2]The derivation of the analytical critical temperature is beyond the scope of this report, see for example Onsager 1944 [1].



(a) $k_B T_C/J = 0.45$. **(b)** $k_B T_C/J = 2.1$. **(c)** $k_B T_C/J = 4.0$.

Figure 1: *2D Ising configurations at three temperatures below, around and above the critical temperature respectively. System size of* 64×64 *spins. Below the critical temperature all the spins are aligned, above the critical temperature the spins are random which results in zero net spin.*

This process is not reversible so it is not a method to study the reverse process, that is, to determine the critical temperature given a labelled[3] set of final configuration.

One of the methods to study the 2D Ising model analytically is by using Renormalization Group (RG) methods. RG methods look at how the model changes when we look at the model at different length scales.

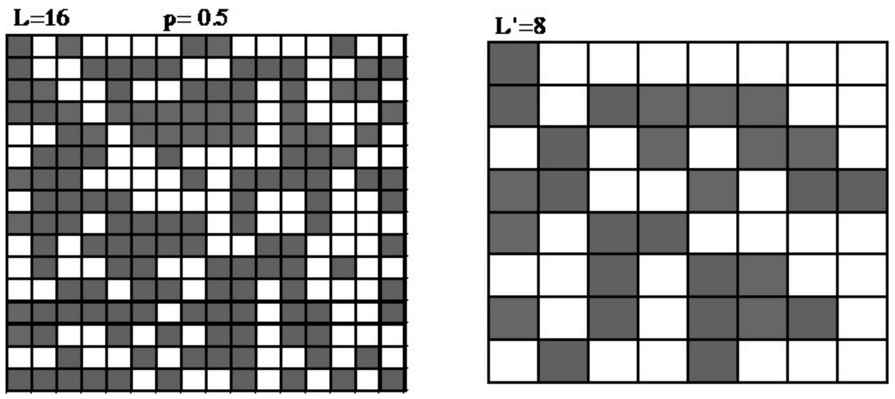


Figure 2: *Example of a transformation used in Renormalization Group theory applied to the 2d square lattice Ising model. The technical details not relevant. The RG transformation resembles the effect a convolutional filter has on an image. Source: [2]*

There are parallels between RG methods and the convolutional filters used in convolutional neural networks (CNN). Therefore it might be possible to get insights in the phase transition by applying a CNN to the labelled set.

In order to make to notation more clear we define a reduced temperature defined by $\hat{T} = k_B T_C/J$ and will from now on drop the hat and use T as the reduced temperature. Likewise we use from now on β to mean the inverse reduced temperature.

3 Methods

In order to train the CNN we first need to generate the dataset. In order to generate this dataset a program was written that uses the Metropolis algorithm to generate final configurations of lattice size L within a range of temperatures. For the dataset used the parameters were

[3]The labels are the temperatures at which these configurations are generated.

Table 1: *Parameters used to generate the dataset using MCMC.*

Parameter:	Value:
T_{min}	0.1
T_{max}	4.5
ΔT	1×10^{-3}
L	100

This generated 4400 test configurations. In order to verify that this dataset has a phase transition within this temperature range we can plot the sum over all spins for each point in the temperature range (figure 3).

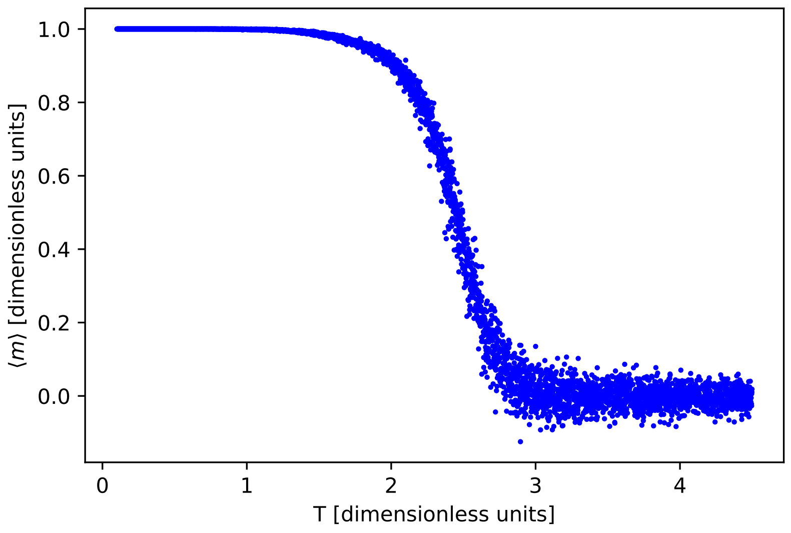


Figure 3: *The average magnetisation per spin for the generated data points generated using the parameters as given in table 1. One can see there is a clear transition happening around the analytical value of* T_C*.*

We want to train a Neural Network to classify these configurations in temperature classes. Therefore we define a discretized temperature, for convenience we choose to use the inverse temperature β to define our classes. The minimal inverse temperature is $\beta_{\text{min}} = 1/T_{\text{max}} = 1/4.5 = 0.22$ and we choose our maximum at $\beta_{\text{max}} = 1$. We discretize this range of inverse temperatures in 100 classes. The final fully connected layer of the network uses these classes.

Before feeding our data to the fully connected layer we apply a Convolutional layer with the parameters given in table 2. Before feeding the data from the convolutional layer to the fully connected layer we use a dropout layer of 0.5 to reduce over fitting.

Table 2: *Parameters used for the Convolutional layer.*

Parameter:	Value:
Kernel size	25
Stride	13
Activation function	Rectified Linear Unit (ReLU)
# channels	3

When we apply this network to the generated dataset by updating the weights once each epoch we find that it correctly classifies 25% of the data points after 300 epochs. When we study the weight matrix of the last layer (see figure 4) we can see a transition in the weight matrix between two phases.

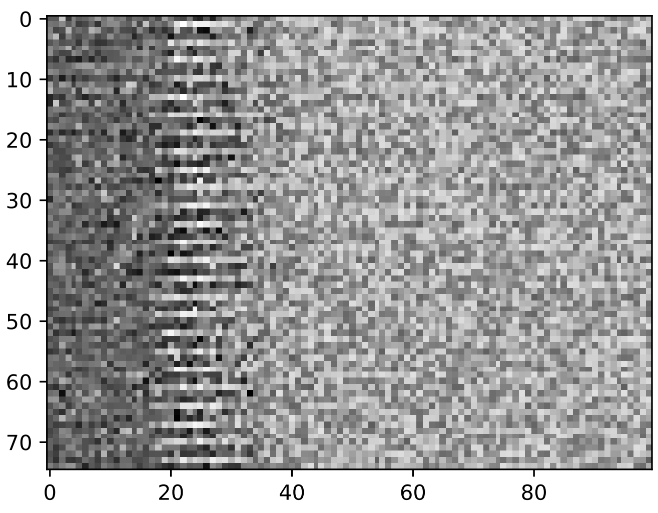


Figure 4: *The weight matrix for the final fully connected layer after 300 epochs. The x axis corresponds with the final classes. Around the 25th class there is a phase transistion from dark to random noise. This 25th class coresponds to* $\beta = \pm 0.45$

In order to verify whether the temperature of this phase transition agrees with the analytical result we need to quantify the transition position. To do this we sum the weights for each class. So when we look at figure 4 we sum over each value on the y axis for each class on the x axis. This applied to the weights in figure 4 is plotted in figure 5. In order to find a value for T_C we need to find where the transition occurs. In order to do that we fit a function to the points. Inspired by Tanaka & Tomiya (2017) [3] we fit to

$$W(\beta) = a \tanh(c(x - \beta_C)) - b$$

and find $\beta_C = 0.449(9)$ which gives a critical temperature of $T_C = 2.23(5)$ which agrees with the analytical value of $T_C = 2.2691\ldots$. The neural network which managed to correctly find the critical temperature only managed to correctly classify the temperature with an accuracy of 25%.

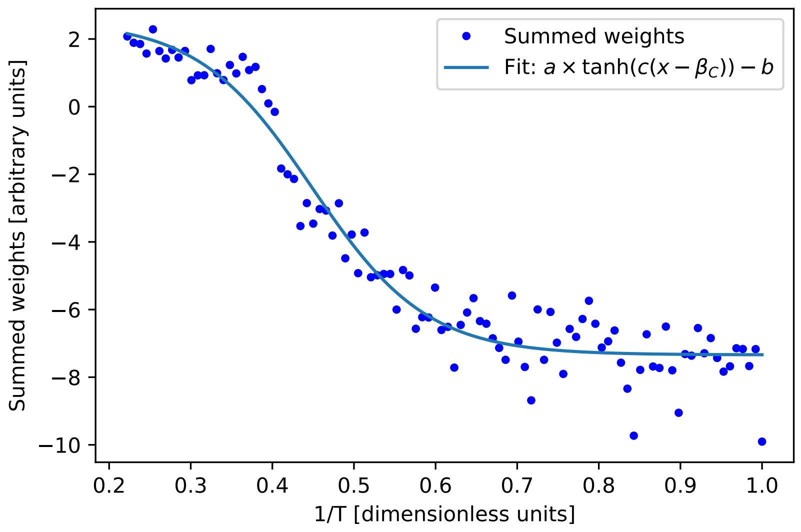


Figure 5: *The sums of the weights after 300 epochs for each class of the inverse temperature for the weights in figure 4 with a fitted* tanh *function to find the critical inverse temperature at* $\beta_C = 0.449(9)$.

The model that manages to correctly determine the critical temperature only correctly classifies with an accuracy of 25%. This accuracy was achieved after 300 epochs. When we train the network for more epochs the accuracy increases but the phase transition in the weight matrix of the last layer becomes less defined and therefore it becomes more difficult to determine the critical temperature. In figure 6 we can see the summed weights after 1000 epochs were we can see that the phase transition becomes less well defined and the fitted function does not manage to properly fit the data. The accuracy after 1000 epochs does increase to 37%.

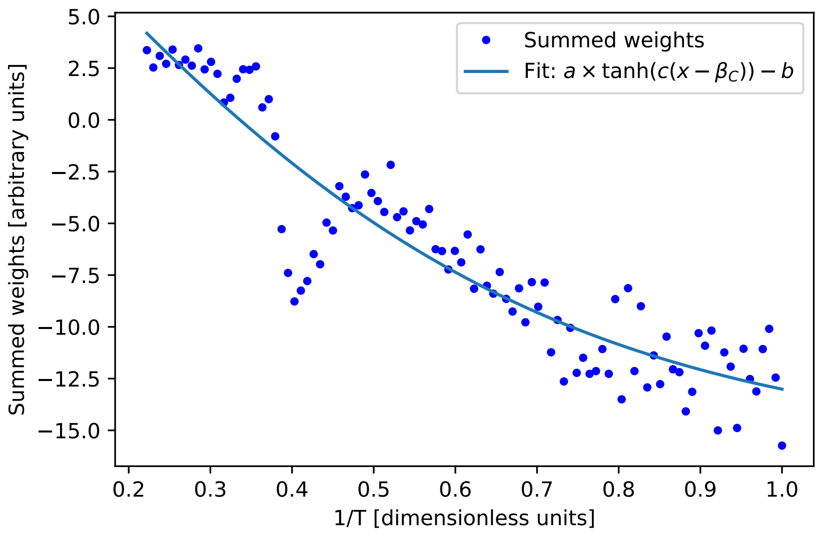


Figure 6: *The sums of the weights after 1000 epochs for each class of the inverse temperature for the weights in figure 7 with a fitted* tanh *function where we can see that the fit does not fit the data properly.*

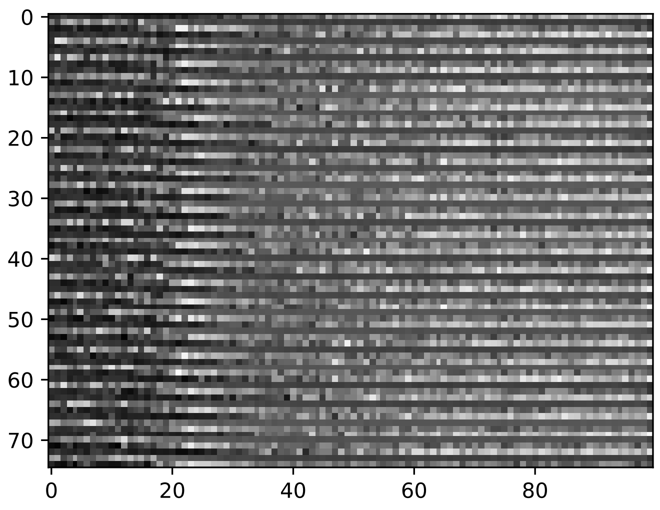


Figure 7: *The weight matrix for the final fully connected layer after 1000 epochs. The x axis corresponds with the final classes. The transistion that was visible in figure 4 after 300 epochs has become less clear.*

When we calculate the estimate for the inverse critical temperature every 10 epochs (figure 8) we see that it slowly increases towards the analytical value (indicated by the black line) until after about 500 epochs the phase transition becomes less clear and the estimate for the inverse critical temperature goes to zero. This might be due to over fitting, this might be reduced by adding extra convolutional layers to the model.

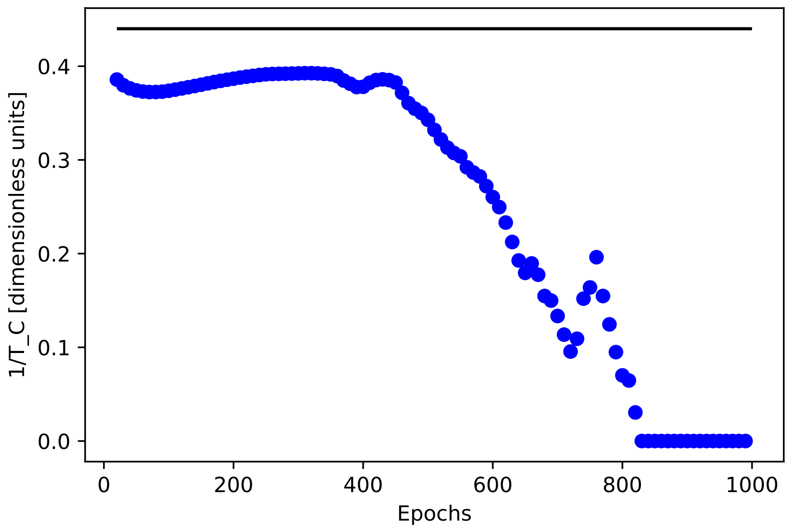


Figure 8: *The estimate for the critical inverse temperature vs. the number of epochs. After around 500 epochs the phase transistion in the weights becomes less clear and therefore the estimate of the critical temperature deviates further from the analytical result represented by the black line.*

4 Conclusion

We used a simple CNN to find the critical temperature of the zero field Ising model to be $T_C = 2.23(5)$ which agrees with the analytical result of $T_C = \frac{2}{\ln(1+\sqrt{2})} \simeq 2.2691\ldots$. The only information needed to find this critical temperature was the temperature of equilibrium configurations at a range of temperatures. This method therefore is able to find the critical temperature of a theoretical model without information regarding the phase transition.

For the zero field Ising model a very simple network consisting of only one convolutional layer and one fully connected layer was able to perform this correct determination of the critical temperature. An extension could be to repeat this analysis for the non zero field Ising model and construct the phase diagram for this model.

Using more complex networks it might be possible to extend this method to more complex models like the Potts model or to models of active systems like the Vicsek model.

References

[1] L. Onsager (1944), *"Crystal statistics. I. A two-dimensional model with an order-disorder transition"*, Physical Review, Series II, 65 (3–4): 117–149.

[2] D. Johnson, Phase Transitions, Finite-size Scaling and Renormalization Group, `http://schleife.web.engr.illinois.edu/teaching/mse485/lnotes/scaling.html`, retrieved on 2019-04-18.

[3] A. Tanaka & A. Tomiya (2017), *Detection of Phase Transition via Convolutional Neural Networks*, Journal of the Physical Society of Japan 86, 063001.