Machine Learning Practical Exam (ID: 3)

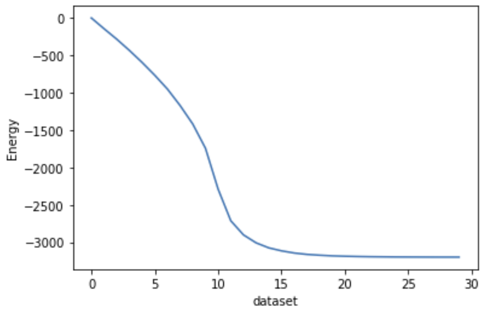
Luca Schinnerl
11545978

August 4, 2020

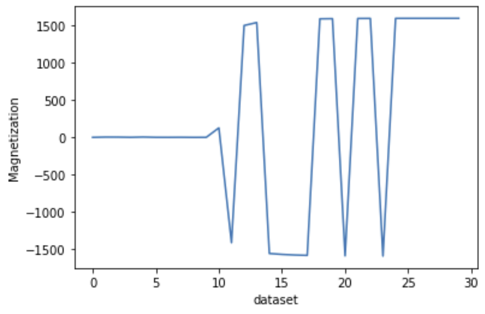
1 Generating appropriate data sets

First of all the data sets have to be examined. How do the different phases look like? We can plot the average energies (figure 1a) and magnetisation (figure 1b) for each temperature.

We can clearly see that the critical temperature is reached at data set 10. This means all data sets with the labels from 11 to 29 can be classified as being below the critical temperature, and all data sets with labels 0 to 9 can be classified as being above the critical temperature. Data set 10 can not be classified in a binary classification. The training and test data has been created with the function "train_test_split"



(a) Average energy of the spin configurations



(b) Average energy of the spin configurations

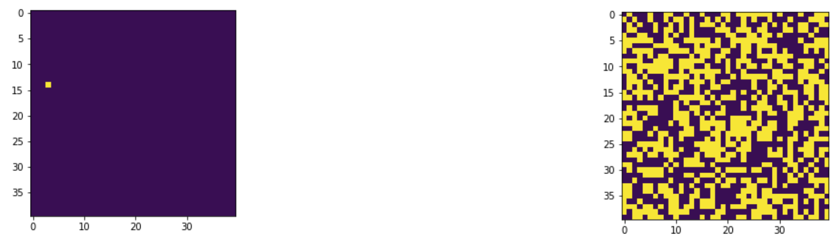
Figure 1: Plot of the average magnetisation and energy of the spin configurations at different temperatures

2 Principal Component Analysis

The X_train and X_test data sets have been dimensionally reduced to only 20 components. For this the data sets had to be flattened first. This means these reduced data sets can not be used in a convolutional architecture. The final data is a set of $29 * 1001$ spin configurations with 20 components each (instead of $40 * 40$)

3 Classification with mostly Dense Layers

Before starting to build a model, looking at the complexity of this classification problem can be very useful. For this we can plot two examples of the spin configurations with different labels. It can

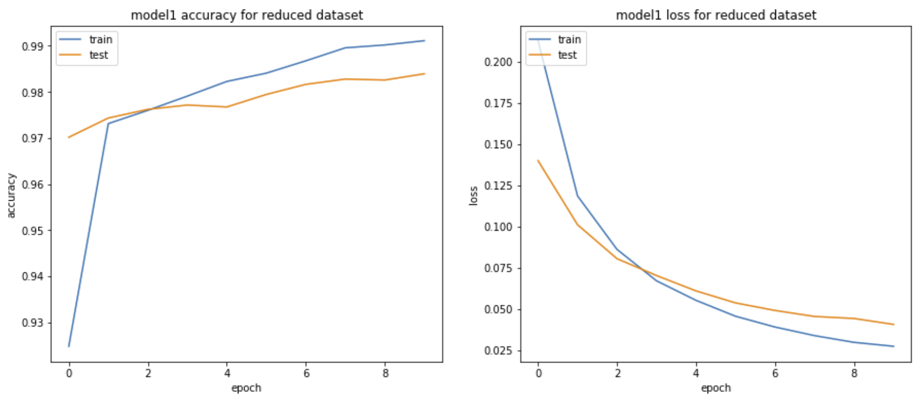


(a) Spin configuration with label 0

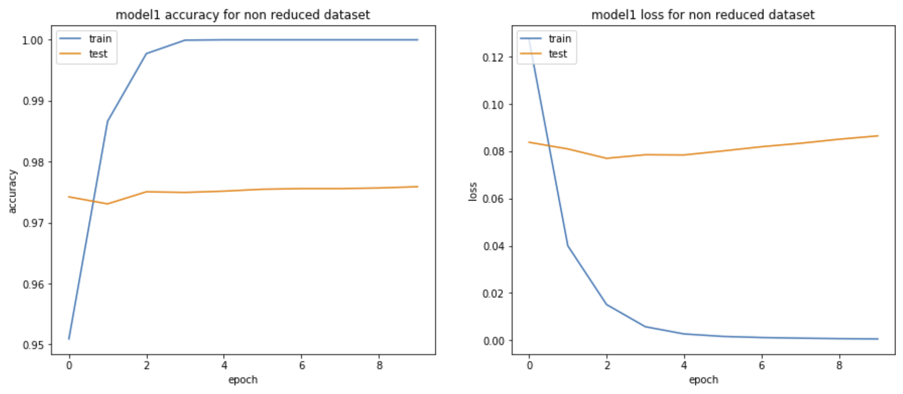
(b) Spin configuration with label 1

Figure 2: Two examples of spin configuration from the two different phases

immediately be seen that this is a relatively easy classification problem, as first of all it is binary and second of all a distinction can very easily be made between the two phases. In addition, with a small subset of the spin configurations, the the phases still look extremely different. This is why a simple ANN structure was chosen, much simpler than the one which was used for the MNIST data set for example. For the first model I chose an architecture with a single hidden layer with 100 nodes. Performance can be seen in figure 3. First things first, the choice of the Adam operator was successfully as it was able to find the minimum of the cost function very well, as training accuracy reached 1 after about 3 epochs on the non reduced data set. However, we can see that for the non reduced data set overfitting occurs, as while the training accuracy increases, the test accuracy does not change. To prevent overfitting I have implemented a dropout layer, batch normalisation and a simpler network (with only 10 hidden neurons). The best preforming combination in the end was a model with a single hidden layer with 100 neurons and a dropout layer with a rate of 0.2 trained and tested on the reduced data set. A validation accuracy of 0.9844 was reached while the training took 9.4 seconds (this could be reduced to 5 seconds as 10 epochs are not needed to find the minimum in the loss function).



(a) Performance on the reduced data set



(b) Performance on the non reduced data set

Figure 3: Performance of model 1

4 Classification with a convolutional architecture

As states before, a convolutional architecture can only be utilised when working with the 2D non reduced data set. I started of with a single convolutional layer and and added a max pooling layer later, which improved performance a lot. The dense part of this was the best best preforming model from last section. It was immediately noticeable that training took more than 15 times as long, while validation accuracy was a bit higher than with only dense layers (0.9955) as expected. From figure 4 it can be seen that the model is not very consistent and also overfitting occurs. To combat this, just as in the last section, model complexity has been reduced and dropout layers have been introduced. The best combination in the end was a model with a single Conv2D layer, a dense layer with 10 neurons and a dropout layer with a rate of 0.2. Overfitting has been mostly reduced and the validation accuracy even exceeded test accuracy. The validation accuracy is 0.996 while the training took about 130 seconds. Adding a second convolutional layer or changing some of the hyperparameter has not improved performance.

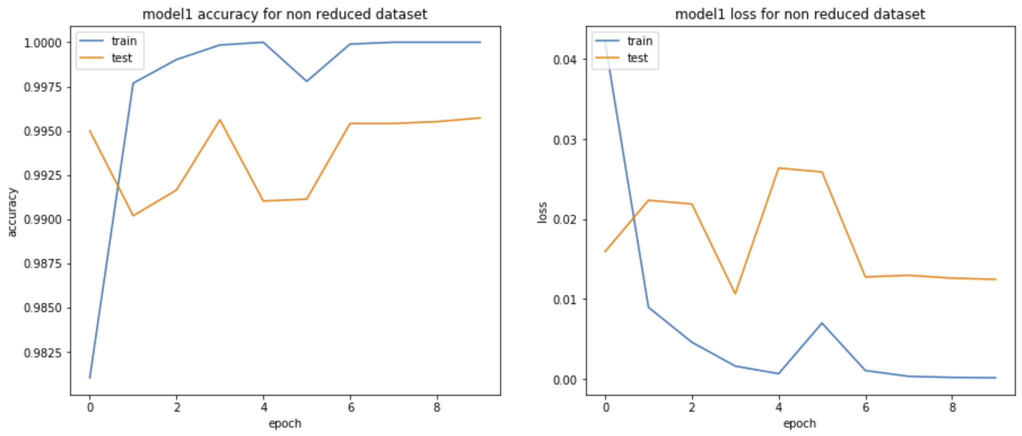


Figure 4: Performance of a model with one Conv2D, max pooling and dense layer with 10 hidden nodes

5 Random Forest classifier

To start off with, I tried a random forest with 100 estimators. I reduced the amount of estimators step by step to see if accuracy changes, which did not until reaching 20 estimators. The overall performance is extremely good, and was again best for the the reduced data set. Training took only 1.5 seconds while achieving a score of 0.986.
One more way to decrease computational cost is to introduce a maximal depth. Max depth of 10 on the non reduced data improves time a bit, but decreases the score also slightly. The score and time of the reduced data set did not change at all, which means that the depth the Random Forest uses is less than 11 here. One final way to improve performance is to preprocess the data and make them Standard Scaler. This again did not improve the performance at all. All in all the pest performance with a Random Forest is the default one on the reduced data set with 20 estimators

6 Comparison of the best approaches

In table 1 I have summarised the best model from each approach and whether or not best performance was reached with the reduced data set. All models had similar accuracy of 0.990 ± 0.06. Best accuracy was reached with the convolutional architecture, while best overall performance was reached with the Random Forest on the reduced data. The problem with the convolutional network is, that we are not able to utilise PCA here, making this classification a lot more complex than it has to be. However, as this is computationally most complex, the performance is also the best. My assumption from the beginning, that this classification is very simple, has been reassured by the fact that best performance was always found when working with the reduced data (when possible).

	Time	Accuracy	Reduced
Dense	9.4	0.984	Yes
Convolutional	130	0.996	No
Random Forest	1.5	0.986	Yes

Table 1: Performance of the best model from each approach