Universität Wien

Fakultät für Physik

SEMESTER: WS 2024

LP Computational Statistical Mechanics

REPORT:
Density of States of Neural Network

Constantin Pinterits

SUPERVISOR:

Univ.-Prof. Mag. Dr. Christoph Dellago

Contents

1 Introduction

The training process of an artificial neural network is accomplished by varying the parameters of the network to minimize the Loss/Energy function of the system. In a classical machine learning environment, the local and global minima of the Energy function are examined. In this project, the Energy landscape was investigated beyond its minima. This is achieved by calculating the Density of States, a quantity that is frequently used in statistical physics. For this purpose, the Wang Landau Algorithm is employed to approximate the Density of States and analyze its dependency on specific features of the given Dataset.

This method for connecting the Density of States with artificial neural networks has been proposed by Mele et al. [1] in their 2024 paper. The goal of this project was to reproduce and compare these findings. Beyond direct reproduction, additional simulations were conducted to examine the effects of varying specific parameters, not considered in the publication. These simulations were not part of the original scope but emerged as an intriguing avenue worth investigating, driven by curiosity about how certain modifications might influence the results.

2 Artificial neural Networks

In its most general form a neural network is defined as system of neurons/nodes, each of which is associated with a value called the state S of a neuron and connections between the neurons. Due to these connections neurons are able to influence the states of neighboring neurons and transport information. Within the scope of this work, two types of artificial neural networks will be used. One of the most basic networks, the single layer perceptron (SLP) and the multi layer perceptron (MLP). Both of these architectures are so called "feed-forward" networks. In these networks information can only pass in one direction, from input to output, without connections between neurons in the same layer or other kinds of feedback loops (recurrent networks). [2]

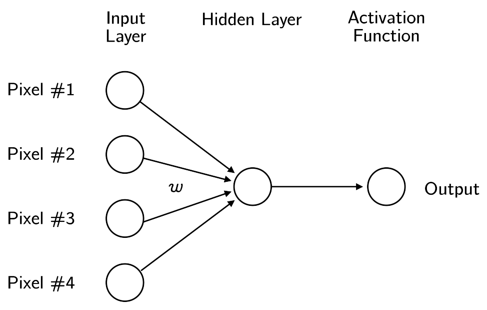


Figure 1: single layer perceptron architecture

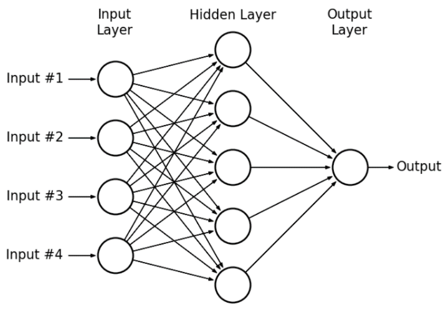


Figure 2: multi layer perceptron architecture

In the SLP as well as the MLP architectures, there are N nodes in the input layer, so that the neuron-states of the first layer can be associated with the components of an N-dimensional input vector $\boldsymbol{\xi}$. The output layer is also equal for SLPs and MLPs where for both cases the output layer only has a single neuron. For any given input vector $\boldsymbol{\xi}$ the output given by a neural network can be interpreted as a labeling process. This is achieved via the propagation of information due to the connections of the network. The state of a neuron that is fed information from N neurons in the previous layer can be described as

$$S_0^{n+1} = f\left(\sum_i w_i S_i^n\right) \tag{1}$$

Where w_i are the values of the weights connecting the nodes from the n-th layer to the node in the $(n+1)$-th layer. f is the activation function that limits the possible values the state S_0^{n+1} can take on. [2]

When looking at the network architecture of a single layer perceptron specifically Figure 1 it can be observed that the output layer is directly dependent on all neurons in the input layer and the connections to the singular output node. Therefore the output o reduces to the form [1]

$$o = f(\boldsymbol{w} \cdot \boldsymbol{\xi}) \tag{2}$$

Here $\boldsymbol{\xi}$ is the input vector of the system and $\boldsymbol{w}$ describes the connecting weights between the input and output layer in vector representation. In this case the process of assigning a label to the input vector is carried out as a vector-vector multiplication. In this project, all simulations use a binary classification task. Therefore, the chosen activation function is the sign function $f = \text{sgn}(x)$ which returns -1 or 1 as the label. [1]

The primary goal in the training of artificial neural networks is to choose the parameters $\boldsymbol{w}$ in such a way, that when an input vector $\boldsymbol{\xi}$ is given to the network, the label assigned to the input will become equal to the correct label σ corresponding to $\boldsymbol{\xi}$. [2] In the context of correctly labeling inputs, we can introduce a measure called the Energy or Loss of a network to characterize the deviation of the network outputs o_i from the labels σ_i. This Energy can be defined for an SLP in the state $\boldsymbol{w}$ as follows.

$$E(\boldsymbol{w}) = \sum_i^P \Theta(-\sigma_i \cdot \text{sgn}(\boldsymbol{w} \cdot \boldsymbol{\xi}_i)) \tag{3}$$

For P different input vectors $\{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_P\}$ that are labeled $\{\sigma_1, \sigma_2, \ldots, \sigma_P\}$ either -1 or 1, the Energy is the number of misclassifications by the neural network in the state $\boldsymbol{w}$. Here Θ is the Heaviside step function which counts the number of times the assigned label o_i does not correlate with the true label σ_i of a given input $\boldsymbol{\xi}_i$. [3]

If a neural network is well trained for a specific task, the configuration of the weights vector, or weights matrices for more complex architectures, is in such a state that minimizes the number of misclasifications and therefor lies in a local or global minima of a Energy function $E(\boldsymbol{w})$. [1]

3 The Wang-Landau Algorithm

In this project however, we are not interested in simply finding a set of parameters that minimize the Energy for a given problem but to look at the Energy landscape on a wider spectrum. This is achieved by calculating the Density of States (DoS) for a given dataset. For a discrete system like the Energy values of a neural network the Density of States $g(E)$ is defined as the number of states that produce a certain Energy E. [1]

$$g(E) = \sum_i dx\ \delta(E_i - E) \tag{4}$$

For most systems, neural networks included, it is practically not feasible to calculate the energy for every possible microstate $\boldsymbol{w}$ that the system can take on. Even for a binary network where the values of the weights are restricted to $w_i \in \{-1, 1\}$, the number of accessible states grows exponentially with the length N of $\boldsymbol{w}$. So, the number of configurations for a single layer perceptron would be 2^N and even larger for more complex networks. Due to this fact, calculating the exact Density of States is practically not possible. [1]

A more efficient way of approximating the Density of States, is the Wang-Landau algorithm. The algorithm uses a series of random walks in configuration space that produce a flat histogram in Energy space. The power of the Wang Landau protocol lies in its ability to reconstruct the prior unknown Density of States for a system via a sequence of Monte Carlo simulations that produce an increasingly accurate estimation to the exact Density of States. [4]

To develop a deeper understanding of the Wang-Landau Algorithm and the properties used in the implementation, first the relation between the Density of States and the Probability Distribution of energy levels $P(E)$ needs to be discussed. $P(E)$ is given by the following expression [5].

$$P(E) = \int dx \ \rho(x) \ \delta(H(x) - E) \tag{5}$$

It describes the probability of finding a system in a particular energy state E by integrating over all possible configurations x, weighted by the density function $\rho(x)$. The Dirac delta function $\delta(H(x) - E)$ enforces that only configurations with $H(x) = E$ contribute to the probability distribution.

In the case where $\rho(x) = C$, so when all possible configurations are equally probable, the integral reduces to the Density of States multiplied by a constant. [6, p.143]

$$P(E) = C \int dx \ \delta(H(x) - E) = C \cdot g(E) \tag{6}$$

When the probability density is chosen to be $\rho(x) = \frac{C}{g(H(x))}$ instead, the probability distribution becomes a uniform function.

$$P(E) = \int dx \ \frac{C}{g(H(x))} \ \delta(H(x) - E) \tag{7}$$

$$P(E) = \frac{C}{g(E))} \underbrace{\int dx\ \delta(H(x) - E)}_{g(E)} = C \tag{8}$$

If the energy levels can be sampled in such a way that $P(E) = C$, that means that the inverse of the density function $\rho(x)$ gives a good approximation of the true Density of States $g(E)$. [4, 7]

In practice, this is achieved by implementing two histograms during simulation. $\tilde{g}(E)$ as the current approximation of the Density of States and $H(E)$ as a histogram that counts the number of occurrences of the energy states E, acting as the probability distribution function of the system. [7]

The Wang-Landau Algorithm is implemented as a random walk in configuration space. An initial configuration is proposed and the energy E_n of the system is calculated. Then, a new configuration is randomly chosen with the energy E_m. In order to accomplish a probability density function $\rho(x)$ that is the inverse of the true Density of States for the System in question, the random step is accepted with the Probability: [4]

$$P = \min \left[1, \frac{\tilde{g}(E_n)}{\tilde{g}(E_m)} \right] \tag{9}$$

Here, $\tilde{g}(E)$ is the current approximation for the DoS. [7] This means that a step will be accepted if the energy state E_m has been visited fewer times than the energy state E_n. If the energy state E_m has been visited more often than the state E_n, the step will be accepted with the probability $\tilde{g}(E_n)/\tilde{g}(E_m)$. [1]

The current approximation of the Density of States is modified after each time a step $E_n \rightarrow E_m$ is accepted and the Energy level E_m is visited. This is done via a modification factor $f > 1$ (typically $f = e^1$) that scales the entry in $\tilde{g}(E_m) \rightarrow \tilde{g}(E_m) \cdot f$ and the running histogram will be modified as $H(E_m) \rightarrow H(E_m) + 1$. If instead a move $E_n \rightarrow E_m$ is rejected, $\tilde{g}(E_n)$ will be modified in the same way and $H(E_n) \rightarrow H(E_n) + 1$. [1, 7]

This walk in configuration space is continued until the running histogram $H(E)$ is sufficiently "flat". At this point, the approximation of the Density of States $\tilde{g}(E)$ has an accuracy proportional to $\ln(f)$ [4] the modification factor f is then reduced by taking its square root $f \rightarrow \sqrt{f}$ and the histogram $H(E)$ is reset to an empty histogram. Then the next random walk is started, now with a finer modification factor. The process is stopped at some predefined value for the modification factor (for example $f_{final} = 10^{-8}$). When this value has been reacher and the running Histogram $H(E)$ has reached sufficient flatness, then the approximation of the Density of States should converge to the true value for the system. [1, 7]

4 Datasets

The goal of this project is to examine the properties of the Density of States for different types of Datasets, their size, resolution and other traits. Therefore two types of Datasets are used. The classic MNIST-Dataset and a synthetically generated Dataset in a teacher-student System, both of which will be discussed in more detail in the following sections. [1]

4.1 MNIST-Dataset

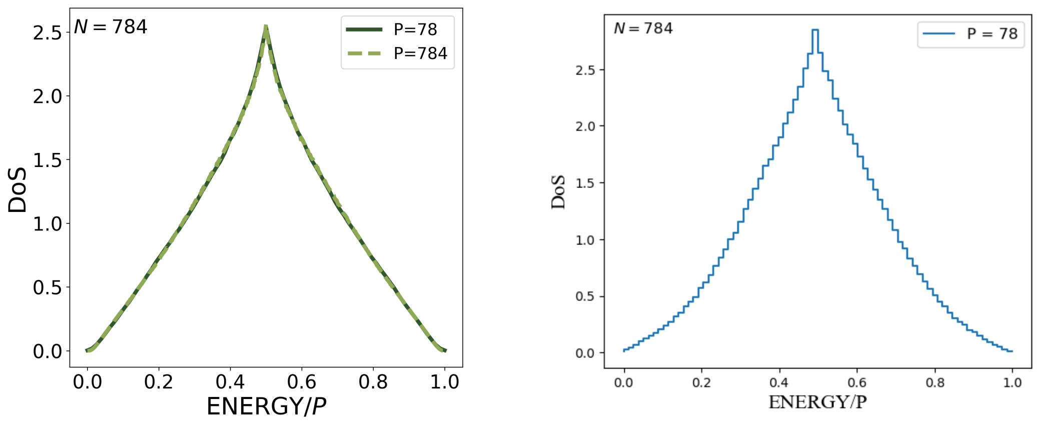
The MNIST Dataset is a collection of 70000 handwritten images of the digits $0-9$. This Dataset is split into 60000 training images and 10000 test images. Each depiction consists of 28×28 pixels, where a pixel can take on values between 0 and 255 (black and white). [8] For the purpose of this project, the Dataset was downloaded using the pytorch-package for Python. When initializing a Dataset to be analyzed, a specified amount A, B of images from two classes (most commonly 0 and 1) were randomly chosen from the training set. The images are normalized using the mean μ and standard deviation σ of the pixel-values in each class $x' = \frac{x-\mu}{\sigma}$ and are labeled with -1 and 1 for the two classes respectively. This technique is commonly used in the training of neural networks and ensures, that the images have a zero mean and unit variance. [9] For different types of analyses, the image resolution was also reduced from 28×28 pixels to 10×10 pixels.

4.2 Teacher generated Dataset

The scenario of teacher generated Data involves a synthetic Dataset that is generated using a pre-defined or randomly chosen teacher vector $\boldsymbol{w_T}$ with binary entries $w_i \in \{-1, 1\}$, which labels other vectors, that have been generated in the same fashion. This labeling process is equivalent to the labeling done by a single layer perceptron as can be seen in (2) with the activation function again being $\text{sgn}(x)$. A certain number of vectors are randomly generated and the teacher vector $\boldsymbol{w_T}$ assigns each vector from this Dataset a label $\sigma \in \{-1, 1\}$. In contrast to the real-world Data from the MNIST set, we can be sure that there is a perfect solution that correctly labels the whole teacher generated Dataset, namely the a priori chosen teacher vector $\boldsymbol{w_T}$. [1]

5 Results

In the following chapter, the results of this project are presented. The simulations done in this work are largely based on a publication by Mele et al. from August of 2024 [1], therefore some simulation results will now be compared with the results found in the aforementioned paper. Furthermore, some additional results not looked at by Mele et al. will be presented.



(a) results by Mele et.al.

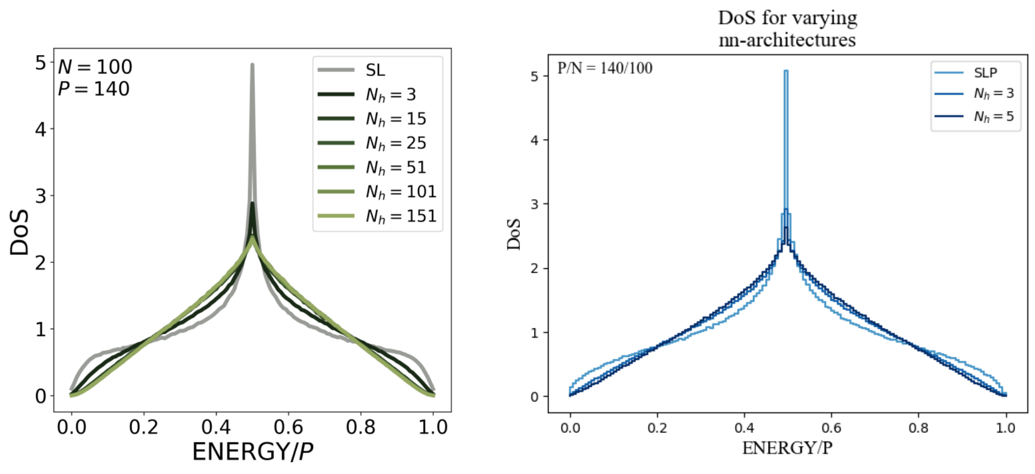
(b) results computed in this project

Figure 3: Comparison of the Density of State for a single layer perceptron

Figure 3 shows a comparison between the DoS as calculated by Mele et.al. and the DoS computed in this work. The system described by both graphs is the classification task by a single layer perceptron of a Dataset comprised of 28×28 pixel images from the MNIST Dataset. The number of pixles in one image (the length of one input vector) is given by $N = 784$ in both cases. P describes the number of images in the Dataset. In this project the case of $P = 78$ has been reproduced. The Datasets described by these figures also have the same number of images in both classes (i.e. 36 images of "0" and 36 images of "1"). In both figures there is a clear peak around the halfway point of the energy axis $E = 0.5$. This implies, that most of the randomly generated states of the neural network wrongly identify about half of the given input vectors. Only a small fraction of solutions can be found in a region of low energy.

With Figure 3b having a slightly higher peak and a less linear slope than

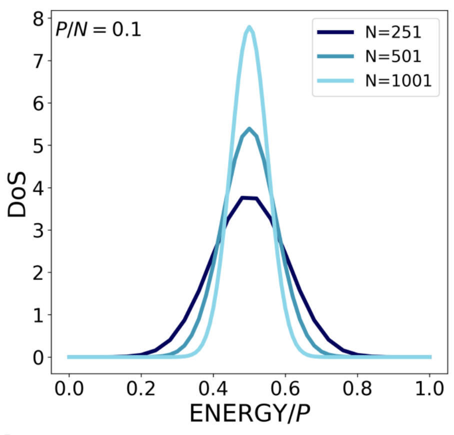
Figure 3a. This could be attributed to the fact, that both simulations use different Datasets, which can influence the peak value of the DoS, as some specific images might be more difficult to classify by a neural network than others. Nevertheless, the graphs show a clear agreement even though there are slight deviations.



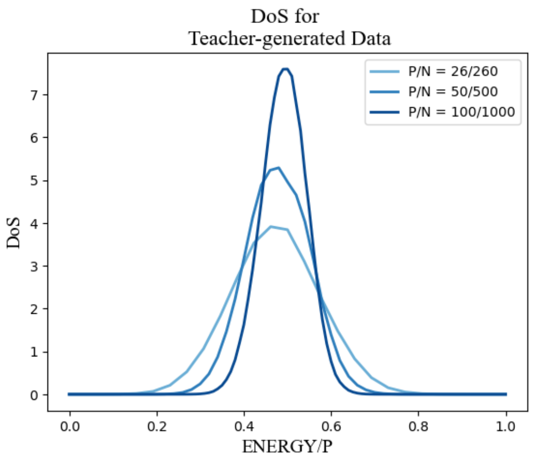
(a) results by Mele et.al. (b) results computed in this project

Figure 4: Comparison of the Density of State (DoS) for different NN-Architectures.

To analyze and compare the influence of the neural network architecture, Figure 4 shows the Density of States for a binary classification task of 140 images with a resolution of 10×10 pixels. The results by Mele et al. present the DoS for a single layer perceptron as well as one hidden layer perceptrons with the number N_h of hidden nodes in the hidden layer. In this project due to computational constraints, the DoS was calculated for the case of one hidden layer, for 3 and 5 hidden nodes. It can be observed that the peak value in both cases decreases very quickly with the number of hidden nodes in the first layer and converges towards a specific value. Furthermore, the Density of States of architectures with fewer hidden nodes show a bulge at low energy and high energy values. Looking at this from the perspective of a classification task, this might suggest, that for the described system the ratio of low energy solutions for a single layer perceptron is higher than that of a one hidden layer network. In other words, when generating a random configuration $\boldsymbol{w}$ of the single layer perceptron, it is more likely for $E(\boldsymbol{w})$ to be close to 0 than when generating a random configuration of a one hidden layer network. The results computed in this work are shown in Figure 4b and resemble the graphs published by Mele et al. in Figure 4a. Again however, there are some deviations to be considered. Especially the bulges at the low and high energy ranges are less prominent in Figure 4b. This deviation might also be attributed to the variation in the datasets used. Since picking the specific images compiled in one dataset happens randomly, these bulges have varied from dataset to dataset. (For additional information on this see the supplementary file)



(a) results by Mele et.al.



(b) results computed in this project

Figure 5: Comparison of the Density of State (DoS) for teacher generated Dataset.

In the last two Figures the analysis of real world data, specifically the MNIST Dataset, was presented. In contrast to this, Figure 5 shows the Density of States for the synthetically generated teacher-student Data. Here again Figure 5a and Figure 5b show the results published by Mele et al. and the results computed in this project respectively. The setup of the systems is the same for both figures. Three Datasets are generated using binary classification by a teacher vector as described in subsection 4.2 with a ratio of $P/N = 0.1$ and varying sizes of N. These datasets are then labeled by a single layer perceptron carrying out another binary classification task. The first thing to observe is that the graphs again show a symmetry around the midpoint of the energy axis $E = 0.5$. This suggests, that the Data is split evenly between vectors labeled -1 and vectors labeled 1, which is to be expected, since the randomly generated teacher vector labels the Dataset. As a result, the input vectors are linearly separable by construction [1].

In contrast to the previously discussed figures, these particular simulation results however do not have a sharp peak like in Figure 3 and Figure 4, but rather have the shape of a Gaussian curve. The graphs calculated in this project show a strong agreement with the results published by Mele et al., in both cases the height of the peak grows with the length of the input vectors N. When taking a closer look at the shape of the DoS there are however some differences to be observed. It seems that the peak values and width of the Gaussian curves do correlate, but that the graphs in Figure 5a seems to be slightly shifted towards the left in the two cases with a lower number of input vectors. This could be explained by the fashion in which the Data is generated. The teacher vector splits the randomly generated Dataset into two parts, however the size of which don't need to be equal. For a larger number of input vectors the ratio of elements in the first and second class P_1 and P_2 converges towards $P_1/P_2 \approx 1$. Which might be why the graph with $P = 100$ appears to be more centered when compared to the cases of $P = 26$ and $P = 50$.

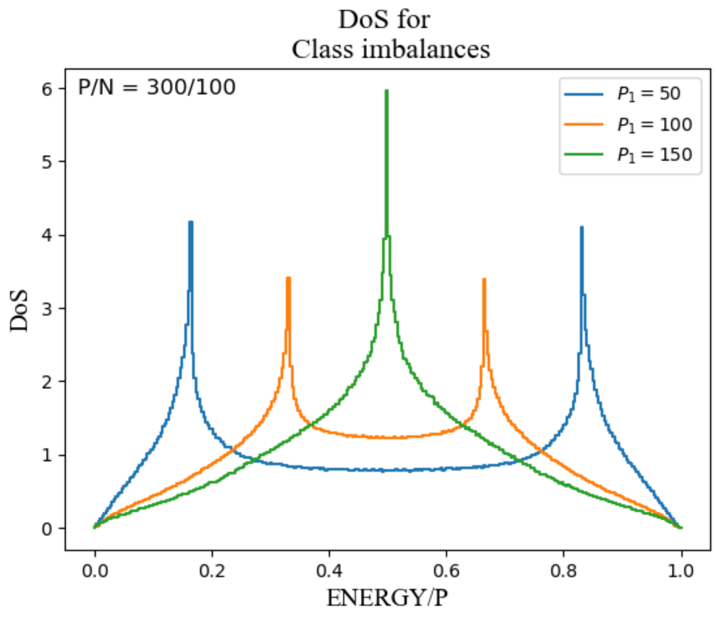
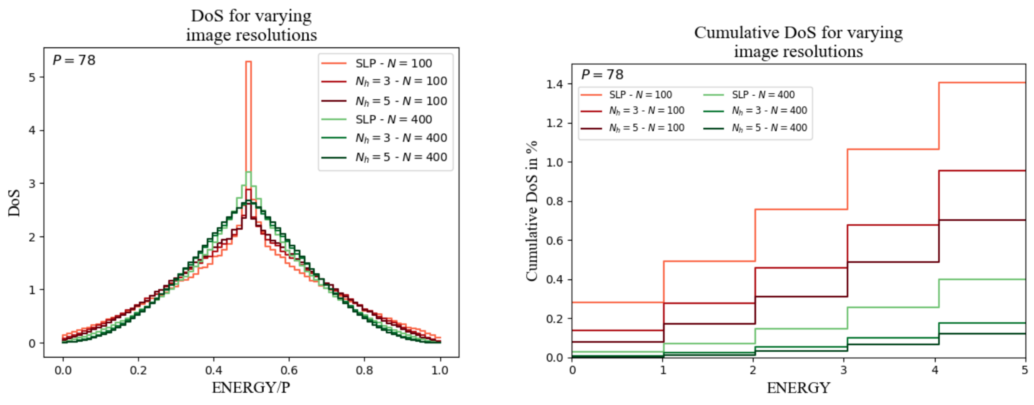


Figure 6: Density of States for class imbalances

The previously discussed simulations all had the property of a balance between the two classes labeled during the binary classification task. This was either achieved by design in Figure 3 and Figure 4 or as a result of probability as shown in Figure 5. We now want to look at the resulting Density of States when a class imbalance is introduced into the dataset. This means, that the amounts P_1 and P_2 of vectors in the classes labeled "-1" and "1" respectively, are not equal $P_1 \neq P_2$. In Figure 6 the results of the Wang-Landau algorithm can be seen for three distinct Datasets with $P_1 = 50$, $P_1 = 100$ and $P_1 = 150$ labeled by a single layer perceptron. A similar simulation has been published by Mele et al., however the MNIST Dataset was not used for that specific computation and therefore a comparison of the figures will not be included.

The green graph inFigure 6 represents a balanced Dataset similar to the case discussed in Figure 4, however with a larger number of input vectors $P = 300$. For this graph we can see the characteristic shape of the Density of State for a single layer perceptron with a sharp peak very few low energy states. When the class imbalance is introduced, the peak value shifts and also splits into two peaks. This of course is due to the symmetry of the system $E(\boldsymbol{w}) = E(-\boldsymbol{w})$. It can further be observed, that the peaks appear at the energy values corresponding to the number of vectors in the first and second class. Therefore it is very likely for a randomly generated state of the neural network to label close to P_1 vectors correctly. The same thing can also be said about the probability to label P_2 vectors correctly.



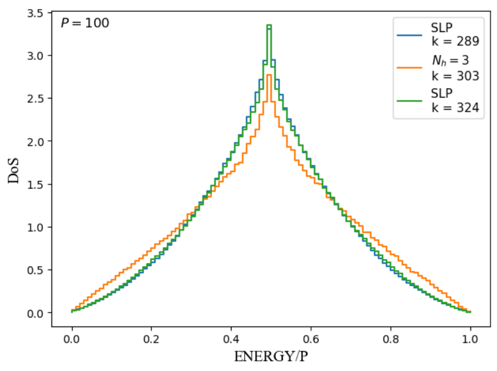
(a) Density of States for different image-resolutions

(b) cumulative sum of the Density of States

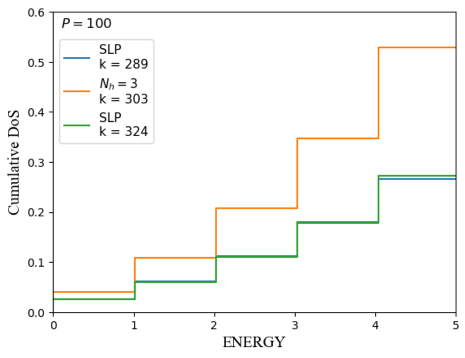
Figure 7: Comparison of the Density of State for varying image resolutions

In Figure 7 an until now overlooked variable of the MNIST-Dataset is analyzed, namely the resolution of the input images. As previously discussed, the images in the MNIST-Dataset come in the form of 28×28 pixels. To reduce the length N of the input vectors and with it the amount of possible states of the neural network (see section 3) and computation time, the image resolution has already been reduced for the simulations in Figure 4 and Figure 6. The graphs shown in Figure 7a present the DoS for a single layer perceptron and one hidden layer networks with 3 and 5 neurons performing a binary classification task between images of 0s and 1s. The Data displayed in green represents the Density of States for images reduced to the size of 20×20 pixels, the red Data represents the DoS of 10×10 pixel images. In all cases, the size of the Dataset is kept at $P = 78$ with a balanced distribution between the two classes $P_1 = P_2$. Figure 7b on the other hand shows a different representation of the Data. In this graph, the cumulative sum of the Density of States is presented and zoomed in on the low energy spectrum. The graph can be interpreted as the probability (in %) to mislabel at most n images of the 78 images in the Dataset. So for example a single layer perceptron executing a binary classification task on 78 images with a resolution of $10 \times 10 = 100$ pixels has a probability of about 1.4 % to label 5 or fewer images incorrectly. In contrast for the case of an SLP with 20×20 images, this probability drops to 0.4 %. The cumulative sum of the Density of State demonstrates two very important aspects of the classification tasks. For the systems described, a single layer perceptron has the highest probability of generating a low energy state. This probability reduces for architectures with a higher number of nodes. It is also easy to see, that not only the structure of a neural network has an effect on the DoS but also the resolution of the images provided/the length of the input vectors. For a higher resolution of input images, the probability of finding low energy states reduces for all neural network structures. Interestingly, it seems that the probability of generating a state with low energy reduces with the number of accessible states, as both the structure of a network as well as the length of the input vectors raise the amount of possible states of a neural network.

Motivated by these findings an additional simulation has been carried out to investigate the probability of encountering low energy configurations for systems with the same approximate number of accessible states.



(a) Density of States for different image-resolutions



(b) cumulative sum of the Density of States

Figure 8: Simulations on different architectures with similar number of states

Figure 8 shows the Density of State, as well as the cumulative Density of State for two single layer perceptrons and a one hidden layer network with $N_h = 3$. The goal of this simulation is to examine if the number of low energy states for more complex neural network structures is higher than the number for single layer perceptrons if both systems have a similar number of accessible states 2^k. For a SLP the number of possible configurations is 2^N with $k = N$ being the length of one input vector. For a one hidden layer Network the exponent k becomes $k = (N+1) \cdot N_h$ with N_h being the number of nodes in the hidden layer. The resolutions of the images used has been adjusted to ensure that k is similar for all three simulations. The single layer perceptrons classify images of 17×17 and 18×18 pixels as an upper and lower bound respectively, while the one hidden layer network carries out a classification task on 10×10 images. This results in the exponent k taking on the values 289, 303 and 324.

The Results for the Density of States is shown in Figure 8a. It is clear to see, that both simulations using the single layer perceptron have very similar results. The graph of the one hidden layer system has the same general shape but a lower peak and higher values in the low energy range. This becomes especially apparent in Figure 8b when looking at the low energy states more closely. The cumulative sum for both single layer perceptrons is approximately equal while the one hidden layer network has a much higher probability of finding configurations in the low energy range.

These results however should be approached with a degree of caution. By simply keeping the number of accessible states 2^k at a constant value, the resolution of the images has been changed significantly and therefore making the classification task less complicated. When giving the same Data to a single layer perceptron, we would expect to find a distribution similar to Figure 4 where the number of low energy states decreases with the complexity of the network. The findings from Figure 8 should be seen as a first investigation into the Density of State for networks with a similar number of possible states and not be taken at face value.

6 Discussion

The goal of this project was to reproduce the Density of States (DoS) simulations published by Mele et al. and compare the results to assess how well they match. This approach enables an investigation into the energy landscape of artificial neural networks on a broader scale than just the local minima. By following a similar computational approach, the aim was to verify whether the key features observed in the original study could be replicated using the same methodology.

The results obtained from the simulations in this project show a strong qualitative agreement with the findings of Mele et al., since the overall shape of the curves closely matches those reported in the original study. This consistency suggests that the computational approach used here successfully reproduces the key features of the system. However, some notable differences can be observed. Specifically, the low- and high energy regions in Figure 4b display lower values than the comparable data in Figure 4a.

These discrepancies may stem from a variety of factors, including differences in numerical implementations and variations in the exact composition of analyzed datasets. Especially the tails that can be seen for the single layer perceptron, can vary significantly with the constellation of images analyzed. Even with consistent general parameters such as the number of images P, the resolution N and the imbalance between classes, the neural network used might have difficulties finding low energy configurations if the Dataset provided includes images with unclear or overlapping characteristics which make clear identification challenging. Some further data on this is provided in the supplementary file.

Despite these variations, the overall agreement in the functional form and structure of the results strongly indicates that the underlying behavior remains consistent. This reinforces the reliability of the methods employed while also highlighting the importance of carefully considering numerical effects when comparing results across different implementations.

In addition to reproducing the published results, further variations in the simulation setup were explored to examine how different parameter choices might affect the outcome. Specifically, simulations were conducted where (1) the image resolution of the analyzed dataset was varied and (2) the number of accessible states was kept constant for two types of neural networks.

The second approach, however, comes with a trade-off: keeping this parameter fixed changes the properties of the input dataset, notably the resolution of the provided MNIST images, which could make the results less meaningful or even difficult to interpret. While these variations were not intended as a direct extension of Mele's work, they were an interesting way to test how sensitive the method is to certain changes. The results of these additional simulations may not necessarily lead to any strong conclusions, but they offer insight into the inner workings of neural networks and potential directions for further exploration.

The code developed for this project is available on GitHub:

`https://github.com/CON5T1/ANN_DOS`

References

[1] M. Mele, R. Menichetti, A. Ingrosso, and R. Potestio, "Density of states in neural networks: an in-depth exploration of learning in parameter space," 2024. [Online]. Available: https://arxiv.org/abs/2409.18683

[2] T. L. H. Watkin, A. Rau, and M. Biehl, "The statistical mechanics of learning a rule," *Reviews of Modern Physics vol. 65, pp. 499–556*, 1993.

[3] H. S. Seung, H. Sompolinsky, and N. Tishby, "Statistical mechanics of learning from examples," *Physical review A, Atomic, molecular, and optical physics*, 1992.

[4] F. Wang and D. P. Landau, "Efficient, multiple-range random walk algorithm to calculate the density of states," *Physical Review Letters*, vol. 86, no. 10, p. 2050, 2001.

[5] S. Chakraborty, "Some applications of dirac's delta function in statistics for more than one random variable," *Applications and Applied Mathematics [electronic only]*, vol. 3, 01 2008.

[6] N. W. Ashcroft and N. D. Mermin, *Solid State Physics.* Saunders College, 1976.

[7] F. Wang and D. P. Landau, "Determining the density of states for classical statistical models: A random walk algorithm to produce a flat histogram," 2001. [Online]. Available: https://arxiv.org/abs/cond-mat/0107006

[8] Y. Pei and L. Ye, "Cluster analysis of mnist data set," *Journal of Physics: Conference Series*, 2022.

[9] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, 2020.