Neue Einsatzmöglichkeit von Hardwarebeschleunigern für nachhaltigere KI-Modelle: Entwicklung und Evaluation der Boltzmann Maschinen auf einem physikinspirierten Hardwarebeschleuniger

Bachelorarbeit

submitted on March 18, 2024

Fakultät Wirtschaft und Gesundheit

Studiengang Wirtschaftsinformatik

Kurs WWI2021F

von

SIMON SPITZER

Betreuer in der Ausbildungsstätte:	DHBW Stuttgart:
⟨ Hewlett Packard GmbH ⟩	⟨ Prof. Dr., Kai Holzweißig ⟩
⟨ Dr. Fabian Böhm ⟩	⟨ der/des wissenschaftlichen Betreuerin/Prüferin ⟩
⟨ Research Scientist at Hewlett Packard Labs⟩	

Unterschrift der Betreuerin/des Betreuers

Contents

List of abbreviations

BM	Boltzmann Maschine
RBM	Restriced Boltzmann Maschine
DNN	Deep Neural Network
EBM	Energy Based Model
MCMC	Markov chain Monte Carlo
DNN	Deep Neural Networks
CPU	Central Processing Unit
GPU	Graphics Processing Unit
ASIC	Application Specific Integrated Circuit
FPGA	Field Programmable Gate Array

List of Figures

List of Tables

1 Einleitung

1.1 Motivation

In the research and development of generative AI models, the computing speed and energy efficiency are increasingly becoming the focus[1] The authors of Open AI confirm, that the growth rate of machine learning models surpassed the growth of efficiency within computerchips. The required computing power of the models double each 3-4 months but the power of computerchips, after Moore's Law double only every 2 years.[2] Focusing on current problems like rising energy consumption of datacenters and the associated greenhouse gas emissions, the search for more efficient solutions is essential for the future. Worldwide energy consumption of datacaenters increase yearly around 20-40%, which means that in 2022 about 1,3% of the total global energy consumption and about 1% of energy-related global greenhouse gas emissions was caused by them.[3] However, it is not clear how large the AI share contributed to the total numbers.

1.2 Problem statement

One approach that is already well known is the use of AI accelerators based on ASICs (application-specific integrated circuits) - i.e. circuits that are used application specific, such as Google TPUs (Tensor Processing Unit).[4] This is useful because the usage of multimodels for discriminating tasks compared to task specific models are more energy intense.[5] One promissing alternative concept in research is the usage on physics-inspired hardware accelerators, that are primarly used for optimization problems because of their ability to solve problems faster and more efficient than GPUs.[6] A scalable physics-inspired hardware accelerator (also called Ising-machine), that surpasses the power of existing standard digital computers, could have a large influence on practical applications for a variety of optimization problems.[7]

Such physics-inspired hardware accelerator offer, due to their special calculation method, potential for efficient processing of computationally intensive tasks. Specifically, the acceleration in contrast to digital computers is achieved by calculating the computationally intense tasks with analog signals. On top of that the implementation on dedicated hardware offers the possibility to exlpoit the parallelization of digitwal hardware accelerators and analog computation.[8]

[1] Vgl. Luccioni/Jernite/Strubell 2023, p. 1
[2] Vgl. Dario Amodei/Danny Hernandez 2024, p. 1
[3] Vgl. Hintemann/Hinterholzer 2022, p. 1
[4] Vgl. Wittpahl 2019, p. 39
[5] Vgl. Luccioni/Jernite/Strubell 2023, p. 5
[6] Vgl. Mohseni/McMahon/Byrnes 2022, p. 1
[7] Vgl. Mohseni/McMahon/Byrnes 2022, p. 1
[8] Vgl. Mohseni/McMahon/Byrnes 2022, p. 4

Interesting enough, despite their different applications, the energy function of the hardware accelerator that is used in Ising-machines shows big parallels to those used in Boltzmann Maschine (BM), therefore it can be suggested that Ising-machines could work well for AI.[9] Ising-machines strive to minimize their energy, which is defined by the pairwise interaction of binary variables (Spins).[10] In contrast, BMs are energy-based neuronal networks that are used for classification tasks by acllocating a skalar energy for each configuration of variables. Minimizing the total network energy is therefore equal with the solution of a optimization problem.[11] Current problems with BMs are the high complexity and high requirements for the all-to-allcommunication between the processing units, which causes the implemention on conventional digital computers to be inefficient, but also an inherently slow convergence in certain processes such as simulated annealing.[12] Theese challenges complicate the training and the usage of BMs, especially for large data volumes and complex optimization tasksl.[13] Nevertheless the similaritis of both models implicate, that Ising-machines could be able to execute this specific AI-model with higher energy efficiency and with higher computing speed. Currently there are only few concepts that exists on how to achieve a implementation of a BM within on a Ising-machine. The paper of the authors Mahdi, Nazm, BojnordiEngin and Engin Ipek is a promissing approach, However it could not be shown how a implementation on a real accelerator chip could function.

With the given background, the following central research question for this theis arises:

1. Can Boltzmann Machines be efficiently implemented on physics-inspired Hardware accelerators by analog noise injection?
    - What is the accuracy of the AI-model on the hardware accelerator?
        - Metric: Prediction accuracy and negative Likelihood
    - Comparison between other hardware accelerators, FPGA, GPU, or CPU within the literature in terms of energy efficiency and computing speed.
        - Metrics: Throughput (Samples/Sec), Energy usage (Energy/Operation)

It is therefore necessary to test whether this generative AI model is compatible with Ising machines machines and whether this solution is efficient or not.

1.3 Objective

The primary objective of this bachelor thesis ist the research and extension of a existing physics-inspired hardware accelerator(Ising machine) for the implementation and evaluation of BMs as an energy based AI-model. The aim is to answer the posed research question. In addition to

[9] Vgl. Cai et al. 2019, p. 10
[10] Vgl. Wang, T./Roychowdhury 2017, p. 1
[11] Vgl. Nazm Bojnordi/Ipek 2016, p. 2
[12] Vgl. Nazm Bojnordi/Ipek 2016, p. 1
[13] Vgl. Nazm Bojnordi/Ipek 2016, p. 2

that, it would be beneficial if rules for the influence of hyperparameters could be established since there is no data available for this new method.

To initially accomplish this objective it is necessary to establish a simulator pipeline to the hardware accelerator that translates BM on top of it. Die The simulator pipeline consists of an existing machine learning library and an existing hardware accelerator that are connected to each other. With the simulator pipeline it needs to be shown that it is possible for the hardware accelerator to realize BMs.

Within the simulator pipeline, the activation probabilities of individual neurons are measured on the simulated hardware. If this process proves successful, it is then expanded to simulate a complete neuronal network. The final step is that the hardware accelerator can be used for training and iterference and is comparable to conventional machine learning libraries. This phase includes a carefully adjustment and possibly extension of the existing accelerator to be compliant with the specific requirements of BMs.

If the simulator pipeline can be validated a workload consisting of a standard data set to recognize handwritten digits will be tested. The prediction accuracy, throughput (samples/sec) and the energy consumption (energy/operation) of the Boltzmann machines on the Ising hardware accelerator will be investigated as metrics in order to thereby answering the second part of the posed research questions.

1.4 Research method

Design Science Research

1. **Problemorientierung:** DSR fokussiert auf die Lösung praktischer Probleme, wie die Forschung zur Steigerung der Effizienz und Rechengeschwindigkeit in KI-Modellen.
2. **Artefakt Entwicklung:** Zentral in DSR ist die Entwicklung innovativer Artefakte. Die Arbeit zielt darauf ab, ein solches Artefakt in Form des physikinspirierten Hardwarebeschleunigers weiterzuentwickeln und für KI-Modelle einzusetzen.
3. **Iterative Evaluation:** Durch die iterative Vorgehensweise in DSR kann die Ausarbeitung der Lösung fortlaufend verbessert und angepasst werden, was für die Entwicklung und Optimierung von KI-Systemen entscheidend ist (ebenfalls das Konzept).
4. **Beitrag zur Wissensbasis und Praxisrelevanz:** DSR unterstützt die Generierung neuer Erkenntnisse und stellt sicher, dass Forschungsergebnisse sowohl theoretisch fundiert als auch praktisch anwendbar sind, was mit den Zielen Ihres Projekts im Einklang steht. Untermethodik könnte hierbei eine Simulation sein. Variabel, je nach Verlauf der Forschung.

1.5 Aufbau der Arbeit

2 Aktueller Stand der Forschung und Praxis

2.1 Ressourcenverbrauch bei KI-Modellen

2.1.1 Ressourcenverbrauch bei KI-Modellen

substantial challenges in high consumption of computational, memory, energy, and financial resources, especially in environments with limited resource capabilities[14]

Nachhaltigkeit

Stromverbrauch

Rechenleistung begrenzt, KI-Modelle wachsen schneller als verfügbare Leistung

2.2 Neural Networks - Boltzmann Machines

Over the past few years, the emergence of artificial neural networks has transformed the field of computer vision and extended its influence to other areas. These include natural language processing, game strategy development and execution (with examples in playing Atari and Go), and optimization of navigation tasks, such as determining the most efficient routes on maps.[15] Therefore, it is fair to say that neural networks are part of various important applications.[16] Particularly in the last two years, artificial intelligence has also garnered widespread interest from the public, especially regarding chatbots like ChatGPT and Google Bard.[17] An important feature of a neural network-based system that are inspired by our brain, is that they can learn and adapt to data.[18]

Internally, neural networks are computational models that consist of many simple processing units, called neurons that work together in parallel often structured within interconnected layers.[19] They consist out of a network architecture, which describes the layout and how the neurons are wired. Secondly, they have a optimization function which specifies the goals persued in the learning process.[20] Lastly, there is a training algorithm that varies all of the hyperparameters,

[14] cf. Bai et al. 2024, pp. 1–2
[15] cf. Cichy/Kaiser 2019, p. 305
[16] cf. Gawlikowski et al. 2023, p. 1513
[17] cf. Singh/Kumar/Mehra 2023, pp. 1–2
[18] cf. Cichy/Kaiser 2019, p. 305
[19] cf. Cichy/Kaiser 2019, p. 305
[20] cf. Durstewitz/Koppe/Meyer-Lindenberg 2019, p. 1583

like connection strengths between neurons, training iterations, the learning rate, etc..[21] The following figure 1 shows a typical neural network that consists out of a input layer, a hidden layer and an output layer with dots representig the neurons wirthin the network.

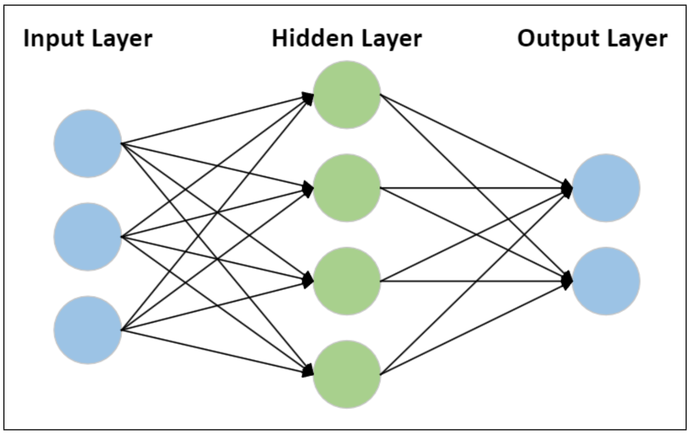

Fig. 1: figure of a neural network

Although, when these interconnected layers are stacked on top of each other, so multiple hidden layers are stacked on top of each other, the network is called deep.[22] In general, deep learning methods can be seen as subset of machine learning methods and are today's fundament of artificial intelligence allowing to solve more complex tasks.[23] Deep Neural Networks (DNN)s are constantly growing and currently have around 1200 interconnected layers that equal to more than 16 million neurons inside a network .[24] An example of a deep neural network is presented in figure 2 which shows the stacked layers in the middle of the network.

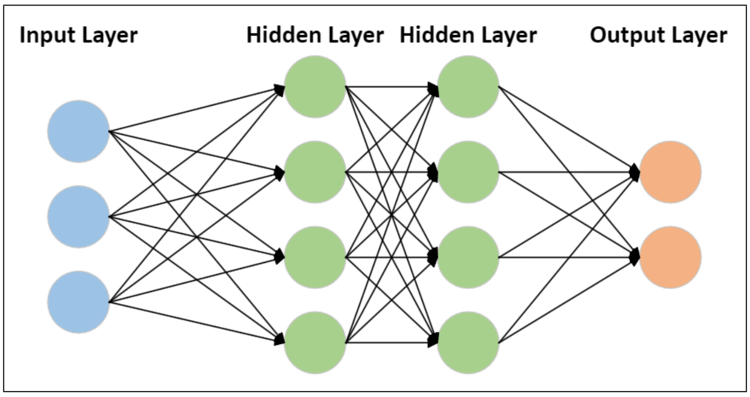

Fig. 2: figure of a neural network

Some examples for regression tasks within a DNN in the field of acomputer vision include object detection, medical image registration, head- and body-pose estimation, age estimation and visual

[21] cf. Durstewitz/Koppe/Meyer-Lindenberg 2019, p. 1583
[22] cf. Cichy/Kaiser 2019, p. 305
[23] cf. Durstewitz/Koppe/Meyer-Lindenberg 2019, p. 1583
[24] cf. Mall et al. 2023, p. 2

tracking.[25] Nowadays, verly large neural networks with millions of parameters can be created due to the research achievements made in the field of neural networks and deep learning leading to highly performing models.[26] Nonetheless, such models often have a negative effect on the environment in terms of unnecessary energy consumption and a limitation to their deployment on low-resource devices because they are excessively oversized and redundant.[27]

2.2.1 Energy-based models

An Energy Based Model (EBM) is a type of neural network that has special characteristics. One characteristic is that an EBM is a statistical model.[28] This probabilistic approach willingly uses uncertainty into the model calculations to draw the models inputs randomly from its underlying distribution.[29] This is done because the conventional deterministic method of backpropagation is known to potentially convert to local minimas, and requires long computation time.[30] As a result with conventional backpropagation more frequently incorrect classification would take place. The second characteristic is that an EBM is determined by an energy function that needs to be minimized in order to find the solution of the optimization problem.[31] Since 1982, those statistical neural network models have been continuously emerging in the machine learning field when J.J. Hopfield introduced the Hopfield Network.[32] Current developments include their use in reinforcement learning, potential replacements for discriminators in generative adversarial networks and for quantum EBMs.[33] In addition to that, Open AI showed that EBMs are useful models across a wide variety of tasks like achieving state-of-the-art out-of-distribution classification and continual online class learning to name a few.[34] The underlying idea behind EBMs is to establish a probabilistic physical system that is able to learn and memorize patterns but most importantly generalize it.[35] Especially, it involes learning an energy function $E_\theta(x) \in \mathbb{R}$, with x representing the configuration of the network, and assigning the low energy to observed data x_i and high energy to other values x.[36]

[25] cf. Gustafsson et al. 2020, pp. 325–326
[26] cf. Marinó et al. 2023, p. 152
[27] cf. Marinó et al. 2023, p. 152
[28] cf. Huembeli et al. 2022, p. 2
[29] cf. Uusitalo et al. 2015, pp. 25–27
[30] cf. Specht 1990, p. 109
[31] cf.Huembeli et al. 2022, p. 2; cf.Ranzato, a. et al. 2006, p. 1
[32] cf. Hopfield 1982
[33] cf.Verdon et al. 2019, p. 1; cf.Du/Lin/Mordatch 2021, p. 1
[34] cf. Du/Mordatch 2020, pp. 1–2
[35] cf. Huembeli et al. 2022, p. 2
[36] cf. Gustafsson et al. 2020, p. 330

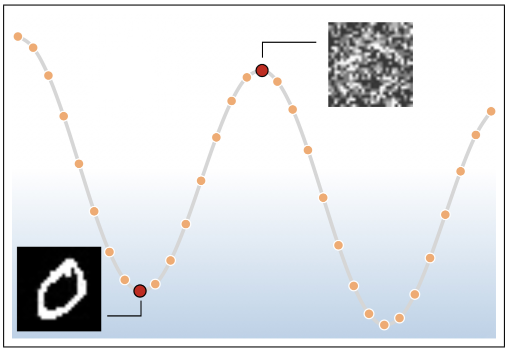

Fig. 3: Figure of a simplified energy landscape

In this figure 1 a simplified energy landscape is shown where the local minima corresponds to states that encode an MNIST digit.[37] It is visible that observed data settles in the local minimum of the energy landscape, in this case a clear 0. On the other hand close to the local maxima of the energy landscape the 0 is only barely recognizable and therefore got a higher energy value assigned to it. The assumption of the underlying distribution function $P(x)$ represents the probability distribution over the input data x, indicating how likely different configurations of x are under the models learned patterns:

$$P(x) = \frac{1}{Z} \exp\left(-\frac{E(x)}{T}\right), \tag{2.1}$$

where Z is the partition function to ensure that the density function normalizes to a total probability of 1 and T is interpreted as the temperature.[38] The partition function Z used in 2.1 is given by summing over all possible pairs of visible and hidden vectors[39]:

$$Z = \sum_x \exp\left(-\frac{E(x)}{T}\right) \tag{2.2}$$

The aim of the training in an EBM is to match the true probability distribution P_{data} as closely as possible with the internal probability distribution P_{model} learned by the model. What this means is that the specific aim is to adjust its parameters such that P_{model} becomes as close to P_{data} as possible, which shows the model has learned the distribution of the real world data. A practical method to achieve this goal is to use the KL divergence. KL divergence is a mathematical meassure that helps to meassure how close the predictions are by comparing the model's learned distribution to the true distribution of the data:

$$G = \sum_x P_{\text{data}}(x) \ln\left(\frac{P_{\text{data}}(x)}{P_{\text{model}}(x)}\right) \tag{2.3}$$

Here, P_{data} is the probability distribution when the network receives a speicific data input from the environmnet, while P_{model} represents the internal network running freely, also referred to as

[37] cf. Huembeli et al. 2022, p. 6
[38] cf. Huembeli et al. 2022, pp. 2–3
[39] cf. Hinton, G. E. 2012b, p. 4

"dreaming".[40] In the training process the asymmetric divergence G needs to be miminized and therefore the probabiliies of the model converge close to the ones in reality. To optimise the KL divergence the energy is adjusted, whereby data is assigned to low energy states (according to 2.1) and the training data receives high energy and therefore low probabiliies.[41]

NACH UNTEN As a side note it is worth mentioning that using the maximum likelihood estimator for Z is intractable due to the requirement of summing over all possible states, which leads to an exponential increase in the number of states for larger systems.[42]

2.2.2 concept of Boltzmann Maschines

A BM is a type of symmetrical EBM consisting of binary neurons $\{0, 1\}$.[43] The neurons of the network can be split into two functional groups, a set of visible neurons and a set of hidden neurons.[44] Therefore, the BM is a two-layer model with a visible layer ("v") and a hidden layer ("h").[45] The visible layer is the interface between the network and the environment. It receives data inputs during training and sets the state of a neuron to either $\{0, 1\}$ which represents activated or not activated. On the other hand, the hidden units are not connected to the environment and can be used to explain underlying constraints in the internal model of input vectors and they cannot be represented by pairwise constraints.[46] The connection between the individual neurons is referred to as bidirectional, as each neuron communicates with each other in both directions.[47] As early as 1985, one of the founding fathers of artificial intelligence, Geoffrey Hinton, was aware that an BM is able to learn its underlying features by looking at data from a domain and developing a generative internal model.[48]

Most machine learning models can be categorized in either generative or discriminative models. Both are strategies to estimate a probability that an specific object can be assigned to a category.[49] Discrimivative models estimate the probability distribution based on category labels that are given to specific objects.[50] On the other hand, a generative model differ as follows. They generate a probabilistic model of the underyling probability distribution for each category, which is assumed as the basis of the data, and in a following step they use Baye's rule to identify which category is very likely to have establihed the object.[51] An real world example would be the following: to predict if a movie will be a hit, you could analyze past box office successes to model characteristics shared by hits (generative approach), or assess immediate audience reactions to

[40] cf. Ackley/Hinton, G. E./Sejnowski, T. J. 1985, pp. 154–155
[41] cf. Zhai et al. 2016, pp. 2–3
[42] cf. Zhai et al. 2016, pp. 2–3
[43] cf. Amari/Kurata/Nagaoka 1992, p. 260
[44] cf. Ackley/Hinton, G. E./Sejnowski, T. J. 1985, p. 154
[45] cf. Salakhutdinov/Hinton, G. 2009, p. 448
[46] cf. Ackley/Hinton, G. E./Sejnowski, T. J. 1985, p. 154
[47] cf. Ackley/Hinton, G. E./Sejnowski, T. J. 1985, p. 149
[48] cf. Ackley/Hinton, G. E./Sejnowski, T. J. 1985, p. 148
[49] cf. Hsu/Griffiths 2010, p. 1
[50] cf. Gm et al. 2020, p. 2
[51] cf. Hsu/Griffiths 2010, p. 1

movie trailers and reviews to predict success without modeling historical data (discriminative approach). Therefore it can be said that BMs and EBMs are generative models. In the following figure 4, a general BM is depicted, where the upper layer embodies a vector of stochastic binary 'hidden' features, while the lower layer embodies a vector of stochastic binary 'visible' variables.[52]

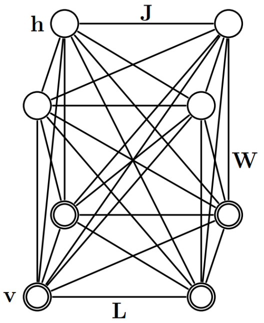

Fig. 4: figure of a general Boltzmann Machine

The model contains a set of visible units $v \in \{0,1\}$, and a set of hidden units $h \in \{0,1\}$ (see Fig. 1). The energy function of the BM with the states $\{v,h\}$ is defined as:

$$E(v,h;\theta) = -\frac{1}{2}v^T Lv - \frac{1}{2}h^T Jh - v^T Wh, \tag{2.4}$$

where $\theta = \{W, L, J\}$ are the model parameters.[53] W, L, J represent visible-to-hidden, visible-to-visible and hidden-to-hidden weights. In BM each neurons works towards minimizing the global energy by entering a particular neuron configuration representing a input to the machine and the system will find the minimum energy configuration that is compatible with the given input.[54] A simple method to find a local energy minimum involes to switch into wichever of the two states (on or off) of a neuron result in a lower energy given the current state of the other neurons.[55] Integrating the function 2.4 into the earlier introduced KL-divergence 2.2 and doing gradient descend a learning rule to update the weights and biases appers.[56] The gradient descent algorithm is commonly used in machine learning and is an iterative technique that adjusts the model parameters (weights and biases).[57] It progressively acquires the gradient of the energy function, methodically advancing towards the optimal solution and ultimately achieves the minimum loss function along with adjusted parameters.[58] Consequently, this leads to the specific learning rule[59]:

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}) \tag{2.5}$$

[52] cf. Salakhutdinov/Hinton, G. 2009, p. 449
[53] cf. Salakhutdinov/Hinton, G. 2009, p. 448
[54] cf. Ackley/Hinton, G. E./Sejnowski, T. J. 1985, p. 150
[55] cf. Fahlman/Hinton, G./Sejnowski, T. 1983, p. 110
[56] cf. Hinton, G. E. 2012b, p. 5
[57] cf. Wang, X./Yan/Zhang, Q. 2021, p. 11
[58] cf. Wang, X./Yan/Zhang, Q. 2021, p. 11
[59] cf. Hinton, G. E. 2012b, p. 5

The network can now update the weights “W” that exist between the neurons through the training rule based on the observations that served as input.[60] In this case, the square brackets represent expected values, as the training is based on the activation probability. In addition to that, the step sizes of updates to the weights are influenced by the learning rate ϵ within the iterative training process.

Performing exact training in this model is intractable because exact computation of the data predictions and the model predictions takes a time that is exponential in the number of hidden units.[61] When the number of hidden units is large compared to the number of visible units it is impossible to achieve a perfect model because of the totally connected network and the resulting 2^n possbilities.[62] Hereby, n represents the number of neurons in the network with each neuron being in one of the two states, the total sum of possibilities are 2^n. This leads back to the briefly mentioned constraint of equation 2.3, that is needed to calculate an activation probability of a neuron, which is required to update a weight in the training process shown in 2.5.

A specific example to demonstrate why it is intractable to calculate an activiation of a BM is the following. A fictional BM has 80 visible nodes and 120 hidden nodes and therefore the possbilities of states of neurons are 2^{200}, which is 1.61×10^{60}. To put this into perspective, the total atoms that exist on earth are only estimated to be around 1.33×10^{50}.[63] That means even if it would be possible to store one information per atom it would just not be enough.

As a result, instead of directly trying to train the model sampling methods are used that are able to estimate these activation probabiliies. This enables the training of BMs and RBMs.

2.2.3 Restriced Boltzmann Machines

As a simplification of the training problem Hinton and Sejnowski proposed Gibbs sampling as an algorithm to aporoximate both expectations.[64] Furthermore, the intralayer connections of the model got removed and the result is the so called RBM. To transform an BM into a RBM the diagonal elements L and J introduced earlier, are set to 0 and as a result the well-known model of a RBM establishes shown in fig.5.[65]

[60] cf. Barra et al. 2012, pp. 1–2

[61] cf. Salakhutdinov/Hinton, G. 2009, p. 449

[62] cf. Ackley/Hinton, G. E./Sejnowski, T. J. 1985, p. 154

[63] cf.Helmenstine 2022, p. 478-480; cf.Schlamminger 2014, p. 1

[64] cf. Ackley/Hinton, G. E./Sejnowski, T. J. 1985, pp. 158–165

[65] cf. Salakhutdinov/Hinton, G. 2009, p. 449

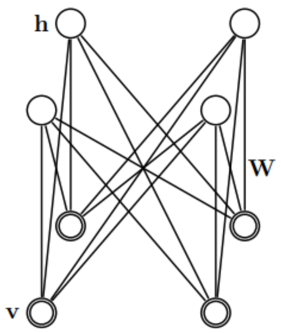

Fig. 5: Figure of a RBM

What can be recognized that no more visible-to-visible and hidden-to-hidden connections can be found in the model. The configuration of the visible and hidden units (v, h) therefore has also an updated energy function (Hopfield, 1982) given by:

$$E(v,h) = -\sum_{i \in \text{visible}} a_i v_i - \sum_{j \in \text{hidden}} b_j h_j - \sum_{i,j} v_i h_j w_{ij}, \tag{2.6}$$

where v_i, h_j are the binary states of a visible unit i and hidden unit j, a_i and b_j are their biases and w_{ij} is the weight between them.[66] Despite, compared to the fully connected BM, the RBM is less complex but the advantages of training surpasses the loss in expressivity possibilities.[67] The RBM has recently been drawing attention in the machine learning community beceause of its adaption and extention for various tasks such as representational learning, document modeling, image recognition and for serving as foundational components for deep networks including Deep Boltzmann Machines, Deep Belief Networks and hybrid models with CNNs.[68]

Training of RBMs

The training of RBMs can be established with the use of sampling methods that estimate the activation probabilities, which are needed to update the weights. There are currently two methods that can be chosen from: contrastive divergence and the Metropolis-Hastings algorithm. The goal of the techniques is to create a sequence of correlated steps from a random walk that, after enough iterations, makes it possible to sample a desired target probability distribution.[69] In the following part both methods will be explained in depth. Especially, they are interesting since they serve as baselines to compare against the new sampling method of a hopfield network that is to be achieved in the practical part of the thesis.

[66] cf. Hinton, G. E. 2012a, pp. 3–4
[67] cf. Huembeli et al. 2022, p. 4
[68] cf. Zhang, N. et al. 2018, p. 1186
[69] cf. Patrón et al. 2024, p. 1

Contrastive Divergence: Contrastive divergence is a special Gibbs Sampling training method developed by Geoffrey Hinton for the efficient training of RBMs.[70] In traditional, Gibbs sampling would have to generate a long chain of samples, until independent samples are obtained from the observed data distribution of the model.[71] The samples are needed for each iteration of the gradient ascent on the log-likelihood resulting in large computational costs.[72] To solve this issue contrastive divergence minimizes an approximation of the Kullback-Leibler divergence between the empirical distribution of the training data and the distribution generated by the model.[73] They way to achieve this is by initializing the Markov chain with the samples from the data distributon.[74] The outcome has been shown to heavily decrease the training time while only adding a small bias.[75] This allows to calculate the probabilities of equation 2.5. This entails initializing the visible units using an actual data input, such as an MNIST sample, and then commencing the subsequent steps with the hidden states. Often the process can be stopped after only sampling a very small number of steps.[76]

1. Forward Pass (positive phase)

During the forward pass using the Gibbs Sampling method, the visible units are set to a completely random state. Next up the hidden units are computed. The computation of the hidden units involves calculating their acitivation probabilities and performing an actual sampling with their calculated activation probabilities. With the RBM it is now easy to get an analytical calculated sample of $(\mathbf{v}_i\mathbf{h}_j)_{data}$.[77] Given an input data out of the training images, v, the binary state, j, of each hidden unit, h_j, is set to 1 with following probability:

$$p(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_i v_i w_{ij}), \tag{2.7}$$

where $\sigma(x)$ is the logistic sigmoid function with an unbiased sample. The sigmoid function is defined as $\sigma(x) = \frac{1}{1+\exp(-x)}$ and is needed because it is the underlying activation function of each neuron. A visual representation is shown in figure 6:

[70] cf. Hinton, G. E. 2012b, pp. 4–5

[71] cf. Huembeli et al. 2022, pp. 5–6

[72] cf. Upadhya/Sastry 2019, pp. 7–8

[73] cf. Mocanu et al. 2016, p. 246

[74] cf. Upadhya/Sastry 2019, pp. 7–8

[75] cf. Larochelle/Bengio 2008, p. 537

[76] cf. Larochelle/Mandel, et al. 2012, p. 646

[77] cf. Hinton, G. E. 2012b, p. 5

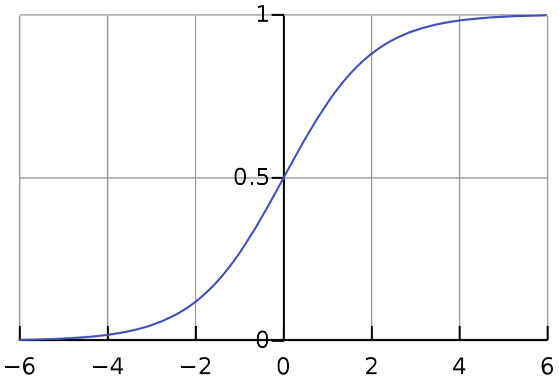

Fig. 6: Figure of a logistic sigmoid function RBM

The result is a set of probabilities that reflect how likely it is for each hidden unit to be on, wich stands for 1, or, which stands for 0, given the input data.[78] The sampling part of the positive phase uses the just calculated acitivation probabiliy of each hidden unit and performs a random experiment with it. That random experiment generates a uniform random number between 1 and 0 and if the random number is greater than the just calculated activiation probability the hidden unit is set to activated. Afterwards, the hidden unit is either activated or not activated and the training process continues with the new state of the hidden units.

2. Reconstruction (negative phase)

In this phase, the sampled hidden states are used to reconstruct the visible units. This is essentially a prediction of the input, which is how the model sees the input based on the just updated hidden units and is calulated with following probability:[79]

$$p(v_i = 1|\mathbf{h}) = \sigma(a_i + \sum_j h_j w_{ij}) \tag{2.8}$$

The sampling part of the negative phase uses the just calculated acitivation probabiliy of each visible unit and performs a random experiment, like in the positive phase. Now, the result is a prediction of the input in the visible nodes. Afterwards, a half forward pass is made to calculate the activation probability of a hidden unit again based on the activated or not activated visible units.

3. Updating the weights

Now, all the requirements to update the weights are satisfied and can be used within the equation 2.5. The delta that results is summed to the current weight and the internal model gets closer to prediciting the observed data. In total, one training iteration consists out of 1 Forward Pass, 1 Reconstruction and 0.5 Forward Pass again is accomplished. Repeating this training steps N

[78] cf. Huembeli et al. 2022, p. 6
[79] cf. Hinton, G. E. 2012b, p. 6

times for a suitable chosen N the model learns better, since more steps of alternating Gibbs sampling were performed.[80]

Metropolis-Hastings: The Metropolis-Hastings algorithm, often only called Metropolis algorithm, is a technique out of Markov chain Monte Carlo (MCMC) class techniques.[81] The Metropolis-Hastings method was invented by Metropolis et al. in 1953 when they noticed, that for an intractable distribution with too many states it can be seen as a limiting distribution of Markov chains.[82] The intractable distribution to handle with the Metropolis-Hastings technique in the case of RBMs is equation 2.3. An Interpretation of the method can be expressed as: "A visitor to a museum that is forced by a general blackout to watch a painting with a small torch. Due to the narrow beam of the torch, the person cannot get a global view of the painting but can proceed along this painting until all parts have been seen."[83] The version already adjusted for RBMs incorporates the following functionality of the Metropolis technique:

First, select a random or given configuarion x_{old} of a RBM that holds the states of all visible and hidden neurons.[84] Secondly, the energy of the configuration, noted as E_{old}, must be calculated using Equation 2.6, as previously introduced. Subsequently, this energy value is stored. Thirdly, the configuration gets updated by picking one random neuron and changing the state of it from 0 to 1 or vice versa.[85] This new configuration is stored as x_{new}. Following that the energy of the new configuration E_{new} is calculated and stored. Now the two energy values are compared and if $E_{\text{new}} <= E_{\text{old}}$ the new configuration will be accepted and $x_{\text{old}} = x_{\text{new}}$.[86] If $E_{\text{new}} > E_{\text{old}}$ then there are some extra steps to be followed:

The flip probability is calculated as $p = \exp\left(-\frac{E_{\text{new}} - E_{\text{old}}}{kT}\right)$. KT is interpreted as the temperature in the network and with higher temperature it increases the activation probability leading to an faster exploration through the landscape but with less details.[87] For RBMs KT is assumed to be 1.[88] In the following figure 7 the resulting probability function is shown.

[80] cf. Huembeli et al. 2022, p. 6

[81] cf. Patrón et al. 2024, p. 1

[82] cf. Metropolis et al. 1953, pp. 1087–1092

[83] cf. Robert 2016, p. 2

[84] cf. Beichl/Sullivan 2000, p. 65

[85] cf. Rosenthal 2009, p. 1

[86] cf. Patrón et al. 2024, pp. 1–2

[87] cf. Li et al. 2016, pp. 1–9

[88] cf. Hinton, G. 2014, p. 3

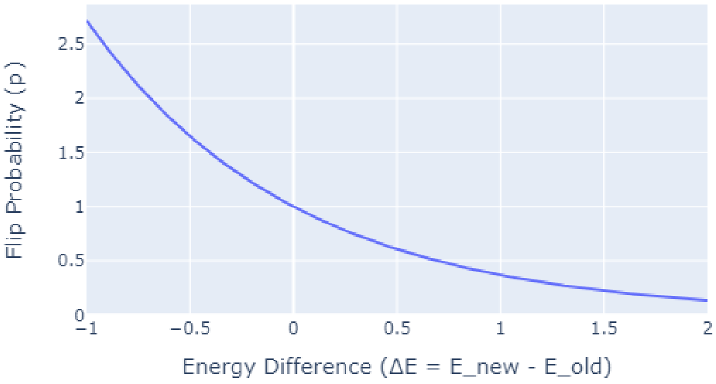

Fig. 7: Flip Probability Function in Metropolis-Hastings Algorithm

In the next step a uniform random number r between 0 and 1 is generated. After generating r the configuration will be accepted if $r \leq p$ (i.e., $x_{\text{old}} = x_{\text{new}}$).[89] Otherwise, a rejection takes place if $r > P$ (i.e., $x_{\text{old}} = x_{\text{old}}$).

Finally, the configuration x_{old} can be stored and the process repeats beginning from step 2 on.[90] After repeating enough times the activation probability for each neuron can be calculated by summing over all samples $(x_1 + x_2 + x_3 + \ldots)$ and the result is divided by the total number of samples.

2.2.4 Current Problems witht BMs and RBMs

One general problem that occurs in the learning process of a BM is that it is both time-consuming and difficult.[91] This is because sampling from an undirected graphical model is not straightforward and therefore RBMs make use of MCMC proposed methods like Contrastive Divergence and Metropolis Hastings.[92] In addition to that, the selection of hyperparameters can be difficult since for the training of a practical model a large hyper-parameter space needs to be explored.[93] Hyperparmeters are: the learning rate, size of the hidden layer, number of training iterations iteration count per bias (sampling step size), initializing the weight sizes in the beginning but also the method for calculating activation probabilities (Contrastive Divergence, Metropolis Hastings, etc.). As a result, establishing a RBM with perfect hyperparameters is time consuming and can be seen as art. Furthermore, training can become unstable and predictions become inaccurate

[89] cf. Patrón et al. 2024, pp. 2–3
[90] cf. Patrón et al. 2024, p. 17
[91] cf. Fischer/Igel 2012, pp. 1–2
[92] cf. Fischer/Igel 2012, p. 2
[93] cf. Larochelle/Bengio 2008, p. 536

due to an incompatible selected temperature.[94] A lower temperature reduces the system's possbility to explore the energy landscape thoroughly, leading to the false selection of local minima instead of finding the global minimum. Vice versa a too high temperature can cause that the energy landscape is not explored enough and has gaps between it missing some minima or skipping the global maxima. Luckily, the temperature for RBMs is expected to be 1 and only for specific use cases it makes sense to adjust internal temperature.

To accelerate the training process of a RBM, it is crucial to address the most computationally demanding aspect: the matrix-vector multiplication involved in the sampling process. A possbilty of achieving this is using dedicated hardware, so called hardware accelerators for this problem. They are designed to tackle a specific task very efficiently, like matrix vector multiplications, which are widely used within most of the neural networks.[95] That is the reason why they are significant for the acceleration of this thesis and an interesting technology to look at.

2.3 Hardware accelerators

2.3.1 Current approaches in the field of AI and other solutions

Since Neural Networks and DNNs are growing their parameters at rapid rates they constantly achieve better and better resutls and are able to solv even more complex tasks.[96] This upcoming trend of growing network sizes exponentially also brings a dark side with it: An excessive increase in computational effort and memory size.[97] As a result Central Processing Unit (CPU)s can barely satisfy the required performance and specialized hardware accelerators are used to increase the performance of these Neural Networks.[98] In addition to that for many use cases, like autonomous driving, there are high energy, latency, and runtime predictability constraints CPUs are not able to meet.[99] There are different approches such as Graphics Processing Unit (GPU)s, Application Specific Integrated Circuit (ASIC)s, Field Programmable Gate Array (FPGA)s, but also new approaches like Quantum Computations or Photonic matrix multiplication are researched.[100] All of these methods have different use cases and get more and more application specific. The list of the sequence, sorted by application-unspecific to specific for established approches looks the following:

$$\text{CPU} \xrightarrow{\text{less flexible}} \text{GPU} \xrightarrow{\text{specialized}} \text{FPGA} \xrightarrow{\text{more customized}} \text{ASIC}$$

Currently the approaches can be segmented into three categories: **Firstly**, the design of data-driven digital circuits. It consits the shift from general-purpose GPUs to specialized dataflow

[94] cf. Huembeli et al. 2022, pp. 3–4
[95] cf. Lehnert et al. 2023, pp. 3881–3882
[96] cf. Baischer/Wess/TaheriNejad 2021, p. 1
[97] cf. Baischer/Wess/TaheriNejad 2021, pp. 1–2
[98] cf.Zhou et al. 2022, p. 1-2; cf.Baischer/Wess/TaheriNejad 2021, p. 2
[99] cf. Ahmad/Pasha 2020, p. 2692
[100] Zhou et al. 2022, p. 1-2; cf.Baischer/Wess/TaheriNejad 2021, p. 2

architectures like systolic arrays, which are used in Google's Tensor Processing Units (TPUs). These architectures are noted for their efficiency in performing deep learning operations by reducing control hardware and keeping data movement local.[101] **Secondly**, network structure optimizations. Hereby modifications to the neural networks themselves are made to improve hardware efficiency. One method is quantization, which simplifies arithmetic operations and reduces memory needs by using fixed-point representations of data and weights instad of using for example 32 bit floating points. The other one is pruning, which involves setting certain weights to zero to reduce the complexity of operations.[102] **Thirdly**, technology-driven designs. Current research into using novel circuitry and memory cells include memristive memory cells and silicon photonics, to further enhance performance and energy efficiency. They work by storing the network weighs and calculating the vector multiplications with analaog signals with technologies like crossbar arrays. While these technologies promise significant advantages, their practical application is still being explored.[103]

2.3.2 Asics

2.3.3 FPGA

2.4 Memristor Hopfield Network - ist ein spezielle ISING Maschine

The so called Memristor Hopfield Network, which is a totally new approach, is a hardware accelerator that uses analog signals in combination with electronical signals. Background is the current slow down or failure of Moore's law which causes issues improving performance and energy efficiency of conventional semiconductor electronical technology.[104]

[101] cf. Lehnert et al. 2023, p. 3883
[102] cf. Lehnert et al. 2023, p. 3883
[103] cf. Lehnert et al. 2023, p. 3883
[104] cf. Zhou et al. 2022, p. 2

Konzept (mit Energiefunktion), Probleme der Digitalrechner bzw. Unterschied zu Digitalrechner

Aktuelle Anwendung

Potentielle Einsatzgebiete für KI-Modelle

Parallelen Energiefunktion BM und ISING Maschine

2.4.1 Memristor

2.4.2 Hopfield Network

A Hopfield Network is an EBM and belongs to the field of recurrent neural networks.[105] The structure of the network consists of only one single layer with binary valued neurons inside.[106] Therefore, the neurons state can either be {1, 0} or {1, -1}. The connections between the neurons are symmetrical, which means that the weights of the connections are the same in either direction.[107] Initially, the primary applications of this type of network were to serve as storage for associative patterns and to facilitate pattern retrieval.[108] In practive given a query pattern a Hopfield Network can retrive a pattern that is most similar or an is an average of similar patterns.[109] In this paper the Hopfield Network's update function interests us because it possibly could be used to sample the intractable training of a RBM mentioned earlier. Surprisingly, since Hopfield networks were introduced by J.J Hopfield in 1982 the storage capacity got increased over time but the fundamentals stayed the same.[110] In following figure 6 an example of a Hopfield Network can be seen.[111]

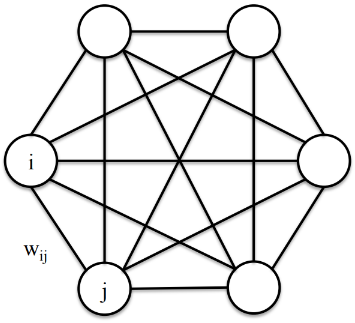

Fig. 8: Figure of a hopfield network

[105] cf. Dramsch 2020, p. 35

[106] cf. Ahad/Qadir/Ahsan 2016, p. 7

[107] cf. Ahad/Qadir/Ahsan 2016, p. 7

[108] cf. Ramsauer et al. 2021, p. 2

[109] cf. Ramsauer et al. 2021, p. 2

[110] cf.Hopfield 1982, p. 2554-2558; cf.Ramsauer et al. 2021, p. 2

[111] cf. Yao/Gripon/Rabbat 2013, pp. 1–2

The exemplary network has 6 neurons and bidirectional weigths W_{ij} between the neurons. In addition to that, a Hopfield network has no input or output layer.[112] The main goal is to find the values for each neuron in the network given a specific input that minimizes the total energy of the system.[113] The minimum energy is then equal to the state where the network is able to perform as a memory item.[114] This energy state can be calculated with the following energy equation[115]:

$$E = -\frac{1}{2} \sum_{i \neq j} T_{ij} V_i V_j. \tag{2.9}$$

This energy function invented by Hopfield has big similarities with a BM when comparing to the equation 2.4. This is one of the reasons why the execution on the Memristor Hopfield Network could work out. When comparing a Hopfield Network, they seek to achieve the effect of changing node activation on the overall energy of the network but BMs replace this with the probability of a certain node being activated on the network energy.[116] The second important reason to research the hopfield networks is for their updating process. Approximately, the acitivity rule for each neuron is to update its state as if it were a single neuron with the threshold activation function.[117]

$$s_i \leftarrow \begin{cases} +1 & \text{if } \sum_j w_{ij} s_j + b \geq \theta_i, \\ -1 & \text{otherwise.} \end{cases}$$

The state of the neuron will be updated to 1 if the sum over all weights multiplied with the states {1, -1} added to a bias b is greater than the threshold θ_i. In the case of our accelerator the threshold is set to 0 but in theory can be used as an hyperparameter.

Since every neuron's output is an input to all the other neurons the order of the updates need to be specified.[118] There is the possbility to update all neurons synchronous or asynchronous. There is no study that shows what update method leads to better results. Therefore, this paper follows the asynchronous option and ensures to do enough iterations, so that every neuron has at least updated once before moving on. In addition to that, the idea of the updating method of the accelerator is slightly different. The idea behind this is to inject noise into the system so that the activation function could work together with the activation function that a RBM needs to perform. In detail the idea is to add a normal gaussian distribution $g(x)$ on top of the activation

[112] cf. Yao/Gripon/Rabbat 2013, p. 3
[113] cf. Ahad/Qadir/Ahsan 2016, p. 7
[114] cf. Ahad/Qadir/Ahsan 2016, p. 7
[115] cf. Hopfield 1982, p. 2556
[116] cf. Ahad/Qadir/Ahsan 2016, p. 7
[117] cf. MacKay 2003, p. 506
[118] cf. MacKay 2003, p. 506

function.[119] As a result the new statistical updating function looks like the following:

$$s_i \leftarrow \begin{cases} +1 & \text{if } \sum_j w_{ij} + b + g(x) \geq \theta_i, \\ -1 & \text{otherwise.} \end{cases}$$

Now the system could potentially be used to update the states of the neurons within a RBM. Since the success of this method is not guaranteed or tested in literature yet the practical part first needs to validate if this concept is feasible.

2.4.3 Crossbar

2.4.4 Output Hopfield Network

2.4.5 Noisy HNN

Picture is out of following paper.[120]

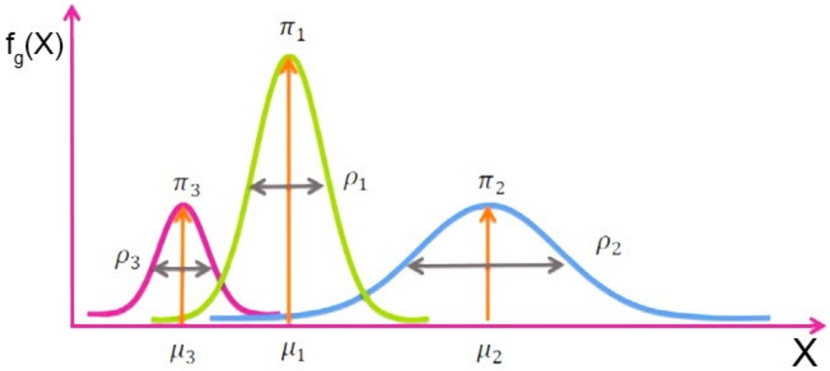

Fig. 9: figure of a general Boltzmann Machine

[119] cf. Böhm et al. 2022, pp. 4–5

[120] cf. Gm et al. 2020, p. 3

3 Zielspezifikation und Darlegung der Forschungsmethodik

3.1 Zielspezifikation (genauer als in Einleitung, Metriken erwähnen, Erfolg meiner Methode bewerten, Welcher Teil der Forschungsfrage wird beantwortet?)

3.2 Design Science Research

3.3 Zielsetzung(ohne gneaue Metriken nennen, generell halten)

3.4 Laborexperiment für die Umsetzung

4 Implementierung/Laborexperiment der Simulator Pipeline

Hopfield Netzwerk aktivierungsfunktion der Updating methode

-> Konzeptionell Art des Updates mit keiner Temperatur wie bei MCMC Unterschied von MCMC zu Hopfield Netzwerk -> Zufällige Konfiguration und minimale Energie finden. Jedoch hat ein Hopfield Netzwerk keine Temperatur

-> Starte zufällige Konfiguration -> Wähle ein Neuron aus und Berechne Summe und addiere mit Bias, -> Update wenn thresshold überschritten 1 und dann auf 0 -> Speichern der neuen Konfuguration -> Starte iteration von gespeicherter Konfiguration -> Am Ende habe ich 10000 Vektoren (Die Konfigurationen) -> V1 Neuron wurde so und so oft aktiviert und ich muss average über das neuron und habe dadurch die Aktivierungswarscheinlichkeit.

-Aktivierungsfunktion einfügen (Binary Step und verfleich zu sigmoid von Abb.4)

-Testen der Aktivierungsfunktion, wenn ich ein Neuron trainiere und dann Mitteln - Von vornerein auf Netzwerk Basis arbeiten mit mehren Neuron, jedoch für 1 Neuron testen

4.1 Zielsetzung und Forschungsmethodik

4.2 Aufbau der Simulator Pipeline

4.3 KI-Bibliothek Scikit-Learn

5 Evaluation der BM auf dem physikinspiriertem Hardwarebeschleuniger

5.1 Zielsetzung und Forschungsmethodik

5.1.1 Prediction Accuracy

5.1.2 Troughput (Samples/Sec)

5.1.3 Energieverbrauch (Energy/Operation)

5.2 Vergleichen mit anderen Hardwarebeschleuniger, FPGA, GPU oder CPU aus der Literatur

6 Kritische Reflexion und Ausblick

6.1 Evaluation der Erkenntnisse in Bezug auf die Zielsetzung der Arbeit

6.2 Kritische Reflexion der Ergebnisse und Methodik

6.3 Zielsetzung(ohne gneaue Metriken nennen, generell halten)

6.4 Ergebnisextration für Theorie und Praxis (evtl. mit 6.4 Zusammenlegen)

6.5 Ausblick

Appendix

List of appendices

Appendix 1: So funktioniert's

Um den Anforderungen der Zitierrichtlinien nachzukommen, wird das Paket `tocloft` verwendet. Jeder Anhang wird mit dem (neu definierten) Befehl **anhang**{Bezeichnung} begonnen, der insbesondere dafür sorgt, dass ein Eintrag im Anhangsverzeichnis erzeugt wird. Manchmal ist es wünschenswert, auch einen Anhang noch weiter zu unterteilen. Hierfür wurde der Befehl **anhangteil**{Bezeichnung} definiert.

In Anhang 1/1 finden Sie eine bekannte Abbildung und etwas Source Code in **??**.

Anhang 1/1: Wieder mal eine Abbildung

Fig. 10: Mal wieder das DHBW-Logo.

List of references

Ackley, D. H./Hinton, G. E./Sejnowski, T. J. (1985): A Learning Algorithm for Boltzmann Machines. In: *Cognitive Science* 9.1, pp. 147–169. ISSN: 0364-0213. DOI: 10.1016/S0364-0213(85)80012-4. URL: https://www.sciencedirect.com/science/article/pii/S0364021385800124 (retrieval: 02/16/2024).

Ahad, N./Qadir, J./Ahsan, N. (2016): Neural Networks in Wireless Networks: Techniques, Applications and Guidelines. In: *Journal of Network and Computer Applications* 68, pp. 1–27. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2016.04.006. URL: https://www.sciencedirect.com/science/article/pii/S1084804516300492 (retrieval: 02/28/2024).

Ahmad, A./Pasha, M. A. (2020): Optimizing Hardware Accelerated General Matrix-Matrix Multiplication for CNNs on FPGAs. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 67.11, pp. 2692–2696. ISSN: 1558-3791. DOI: 10.1109/TCSII.2020.2965154. URL: https://ieeexplore.ieee.org/abstract/document/8954788?casa_token=FflZ-99s30MAAAAA:l-4hcDRIsH2x9gRN3bGMy8BAo1nbQbrJEhqZpdRnAR5IJSe2naviSLmKFiAuYV_yuWVlAPPPdb (retrieval: 03/18/2024).

Amari, S./Kurata, K./Nagaoka, H. (1992): Information Geometry of Boltzmann Machines. In: *IEEE Transactions on Neural Networks* 3.2, pp. 260–271. ISSN: 1941-0093. DOI: 10.1109/72.125867. URL: https://ieeexplore.ieee.org/abstract/document/125867 (retrieval: 02/16/2024).

Bai, G./Chai, Z./Ling, C./Wang, S./Lu, J./Zhang, N./Shi, T./Yu, Z./Zhu, M./Zhang, Y./Yang, C./Cheng, Y./Zhao, L. (2024): Beyond Efficiency: A Systematic Survey of Resource-Efficient Large Language Models. DOI: 10.48550/arXiv.2401.00625. arXiv: 2401.00625 [cs]. URL: http://arxiv.org/abs/2401.00625 (retrieval: 02/23/2024). preprint.

Baischer, L./Wess, M./TaheriNejad, N. (2021): Learning on Hardware: A Tutorial on Neural Network Accelerators and Co-Processors. DOI: 10.48550/arXiv.2104.09252. arXiv: 2104.09252 [cs]. URL: http://arxiv.org/abs/2104.09252 (retrieval: 03/18/2024). preprint.

Barra, A./Bernacchia, A./Santucci, E./Contucci, P. (2012): On the Equivalence of Hopfield Networks and Boltzmann Machines. In: *Neural Networks* 34, pp. 1–9. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2012.06.003. URL: https://www.sciencedirect.com/science/article/pii/S0893608012001608 (retrieval: 02/16/2024).

Beichl, I./Sullivan, F. (2000): The Metropolis Algorithm. In: *Computing in Science & Engineering* 2.1, pp. 65–69. ISSN: 1558-366X. DOI: 10.1109/5992.814660. URL: https://ieeexplore.ieee.org/document/814660 (retrieval: 02/27/2024).

Böhm, F./Alonso-Urquijo, D./Verschaffelt, G./Van der Sande, G. (2022): Noise-Injected Analog Ising Machines Enable Ultrafast Statistical Sampling and Machine Learning. In: *Nature Communications* 13.1 (1), p. 5847. ISSN: 2041-1723. DOI: 10.1038/s41467-022-33441-3. URL: https://www.nature.com/articles/s41467-022-33441-3 (retrieval: 02/15/2024).

Cai, F./Kumar, S./Van Vaerenbergh, T./Liu, R./Li, C./Yu, S./Xia, Q./Yang, J. J./Beausoleil, R./Lu, W./Strachan, J. P. (2019): Harnessing Intrinsic Noise in Memristor

Hopfield Neural Networks for Combinatorial Optimization. DOI: 10.48550/arXiv.1903.11194. arXiv: 1903.11194 [cs]. URL: http://arxiv.org/abs/1903.11194 (retrieval: 02/15/2024). preprint.

Cichy, R. M./Kaiser, D. (2019): Deep Neural Networks as Scientific Models. In: *Trends in Cognitive Sciences* 23.4, pp. 305–317. ISSN: 1364-6613, 1879-307X. DOI: 10.1016/j.tics.2019.01.009. pmid: 30795896. URL: https://www.cell.com/trends/cognitive-sciences/abstract/S1364-6613(19)30034-8 (retrieval: 02/23/2024).

Dario Amodei/Danny Hernandez (2024): AI and Compute. URL: https://openai.com/research/ai-and-compute (retrieval: 02/15/2024).

Dramsch, J. S. (2020): 'Chapter One - 70 Years of Machine Learning in Geoscience in Review'. In: *Advances in Geophysics*. Ed. by Ben Moseley/Lion Krischer. Vol. 61. Machine Learning in Geosciences. Elsevier, pp. 1–55. DOI: 10.1016/bs.agph.2020.08.002. URL: https://www.sciencedirect.com/science/article/pii/S0065268720300054 (retrieval: 02/28/2024).

Du, Y./Lin, T./Mordatch, I. (2021): Model Based Planning with Energy Based Models. DOI: 10.48550/arXiv.1909.06878. arXiv: 1909.06878 [cs, stat]. URL: http://arxiv.org/abs/1909.06878 (retrieval: 02/19/2024). preprint.

Du, Y./Mordatch, I. (2020): Implicit Generation and Generalization in Energy-Based Models. DOI: 10.48550/arXiv.1903.08689. arXiv: 1903.08689 [cs, stat]. URL: http://arxiv.org/abs/1903.08689 (retrieval: 02/23/2024). preprint.

Durstewitz, D./Koppe, G./Meyer-Lindenberg, A. (2019): Deep Neural Networks in Psychiatry. In: *Molecular Psychiatry* 24.11 (11), pp. 1583–1598. ISSN: 1476-5578. DOI: 10.1038/s41380-019-0365-9. URL: https://www.nature.com/articles/s41380-019-0365-9 (retrieval: 02/23/2024).

Fahlman, S./Hinton, G./Sejnowski, T. (1983): Massively Parallel Architectures for AI: NETL, Thistle, and Boltzmann Machines., p. 113. 109 pp.

Fischer, A./Igel, C. (2012): An Introduction to Restricted Boltzmann Machines. In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Ed. by Luis Alvarez/Marta Mejail/Luis Gomez/Julio Jacobo. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 14–36. ISBN: 978-3-642-33275-3. DOI: 10.1007/978-3-642-33275-3_2.

Gawlikowski, J./Tassi, C. R. N./Ali, M./Lee, J./Humt, M./Feng, J./Kruspe, A./Triebel, R./Jung, P./Roscher, R./Shahzad, M./Yang, W./Bamler, R./Zhu, X. X. (2023): A Survey of Uncertainty in Deep Neural Networks. In: *Artificial Intelligence Review* 56.1, pp. 1513–1589. ISSN: 1573-7462. DOI: 10.1007/s10462-023-10562-9. URL: https://doi.org/10.1007/s10462-023-10562-9 (retrieval: 02/23/2024).

Gm, H./Gourisaria, M. K./Pandey, M./Rautaray, S. S. (2020): A Comprehensive Survey and Analysis of Generative Models in Machine Learning. In: *Computer Science Review* 38, p. 100285. ISSN: 1574-0137. DOI: 10.1016/j.cosrev.2020.100285. URL: https://www.sciencedirect.com/science/article/pii/S1574013720303853 (retrieval: 03/08/2024).

Gustafsson, F. K./Danelljan, M./Bhat, G./Schön, T. B. (2020): Energy-Based Models for Deep Probabilistic Regression. In: *Computer Vision – ECCV 2020*. Ed. by Andrea Vedaldi/

Horst Bischof/Thomas Brox/Jan-Michael Frahm. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 325–343. ISBN: 978-3-030-58565-5. DOI: 10.1007/978-3-030-58565-5_20.

Helmenstine, A. (2022): How Many Atoms Are in the World? Science Notes/Projects. URL: https://sciencenotes.org/how-many-atoms-are-in-the-world/ (retrieval: 02/21/2024).

Hintemann, R./Hinterholzer, S. (2022): Data Centers 2021: Data Center Boom in Germany Continues - Cloud Computing Drives the Growth of the Data Center Industry and Its Energy Consumption. DOI: 10.13140/RG.2.2.31826.43207.

Hinton, G. (2014): 'Boltzmann Machines'. In: *Encyclopedia of Machine Learning and Data Mining*. Ed. by Claude Sammut/Geoffrey I. Webb. Boston, MA: Springer US, pp. 1–7. ISBN: 978-1-4899-7502-7. DOI: 10.1007/978-1-4899-7502-7_31-1. URL: https://link.springer.com/10.1007/978-1-4899-7502-7_31-1 (retrieval: 03/16/2024).

Hinton, G. E. (2012a): 'A Practical Guide to Training Restricted Boltzmann Machines'. In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon/Geneviève B. Orr/Klaus-Robert Müller. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 599–619. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_32. URL: https://doi.org/10.1007/978-3-642-35289-8_32 (retrieval: 02/15/2024).

– **(2012b)**: 'A Practical Guide to Training Restricted Boltzmann Machines'. In: *Neural Networks: Tricks of the Trade*. Ed. by Grégoire Montavon/Geneviève B. Orr/Klaus-Robert Müller. Vol. 7700. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 599–619. ISBN: 978-3-642-35288-1 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_32. URL: http://link.springer.com/10.1007/978-3-642-35289-8_32 (retrieval: 02/15/2024).

Hopfield, J. J. (1982): Neural Networks and Physical Systems with Emergent Collective Computational Abilities. In: *Proceedings of the National Academy of Sciences* 79.8, pp. 2554–2558. DOI: 10.1073/pnas.79.8.2554. URL: https://www.pnas.org/doi/10.1073/pnas.79.8.2554 (retrieval: 02/19/2024).

Hsu, A./Griffiths, T. (2010): Effects of Generative and Discriminative Learning on Use of Category Variability. In: *Proceedings of 32nd Annual Conference of the Cognitive Science Society*.

Huembeli, P./Arrazola, J. M./Killoran, N./Mohseni, M./Wittek, P. (2022): The Physics of Energy-Based Models. In: *Quantum Machine Intelligence* 4.1, p. 1. ISSN: 2524-4914. DOI: 10.1007/s42484-021-00057-7. URL: https://doi.org/10.1007/s42484-021-00057-7 (retrieval: 02/19/2024).

Larochelle, H./Bengio, Y. (2008): Classification Using Discriminative Restricted Boltzmann Machines. In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. New York, NY, USA: Association for Computing Machinery, pp. 536–543. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390224. URL: https://dl.acm.org/doi/10.1145/1390156.1390224 (retrieval: 02/22/2024).

Larochelle, H./Mandel, M./Pascanu, R./Bengio, Y. (2012): Learning Algorithms for the Classification Restricted Boltzmann Machine. In: *The Journal of Machine Learning Research* 13, pp. 643–669.

Lehnert, A./Holzinger, P./Pfenning, S./Müller, R./Reichenbach, M. (2023): Most Resource Efficient Matrix Vector Multiplication on FPGAs. In: *IEEE Access* 11, pp. 3881–3898. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3234622. URL: https://ieeexplore.ieee.org/document/10007836?denied= (retrieval: 03/16/2024).

Li, G./Deng, L./Xu, Y./Wen, C./Wang, W./Pei, J./Shi, L. (2016): Temperature Based Restricted Boltzmann Machines. In: *Scientific Reports* 6.1 (1), p. 19133. ISSN: 2045-2322. DOI: 10.1038/srep19133. URL: https://www.nature.com/articles/srep19133 (retrieval: 02/27/2024).

Luccioni, A. S./Jernite, Y./Strubell, E. (2023): Power Hungry Processing: Watts Driving the Cost of AI Deployment? DOI: 10.48550/arXiv.2311.16863. arXiv: 2311.16863 [cs]. URL: http://arxiv.org/abs/2311.16863 (retrieval: 02/15/2024). preprint.

MacKay, D. J. C. (2003): Information Theory, Inference and Learning Algorithms. Cambridge University Press. 694 pp. ISBN: 978-0-521-64298-9. Google Books: AKuMj4PN_EMC.

Mall, P. K./Singh, P. K./Srivastav, S./Narayan, V./Paprzycki, M./Jaworska, T./Ganzha, M. (2023): A Comprehensive Review of Deep Neural Networks for Medical Image Processing: Recent Developments and Future Opportunities. In: *Healthcare Analytics* 4, p. 100216. ISSN: 2772-4425. DOI: 10.1016/j.health.2023.100216. URL: https://www.sciencedirect.com/science/article/pii/S2772442523000837 (retrieval: 02/23/2024).

Marinó, G. C./Petrini, A./Malchiodi, D./Frasca, M. (2023): Deep Neural Networks Compression: A Comparative Survey and Choice Recommendations. In: *Neurocomputing* 520, pp. 152–170. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2022.11.072. URL: https://www.sciencedirect.com/science/article/pii/S0925231222014643 (retrieval: 02/23/2024).

Metropolis, N./Rosenbluth, A. W./Rosenbluth, M. N./Teller, A. H./Teller, E. (1953): Equation of State Calculations by Fast Computing Machines. In: *The Journal of Chemical Physics* 21.6, pp. 1087–1092. ISSN: 0021-9606. DOI: 10.1063/1.1699114. URL: https://doi.org/10.1063/1.1699114 (retrieval: 02/27/2024).

Mocanu, D. C./Mocanu, E./Nguyen, P. H./Gibescu, M./Liotta, A. (2016): A Topological Insight into Restricted Boltzmann Machines. In: *Machine Learning* 104.2, pp. 243–270. ISSN: 1573-0565. DOI: 10.1007/s10994-016-5570-z. URL: https://doi.org/10.1007/s10994-016-5570-z (retrieval: 02/22/2024).

Mohseni, N./McMahon, P. L./Byrnes, T. (2022): Ising Machines as Hardware Solvers of Combinatorial Optimization Problems. DOI: 10.48550/arXiv.2204.00276. arXiv: 2204.00276 [physics, physics:quant-ph]. URL: http://arxiv.org/abs/2204.00276 (retrieval: 02/15/2024). preprint.

Nazm Bojnordi, M./Ipek, E. (2016): Memristive Boltzmann Machine: A Hardware Accelerator for Combinatorial Optimization and Deep Learning, p. 13. 1 p. DOI: 10.1109/HPCA.2016.7446049.

Patrón, A./Chepelianskii, A. D./Prados, A./Trizac, E. (2024): On the Optimal Relaxation Rate for the Metropolis Algorithm in One Dimension. DOI: 10.48550/arXiv.2402.11267. arXiv: 2402.11267 [cond-mat, physics:math-ph]. URL: http://arxiv.org/abs/2402.11267 (retrieval: 02/27/2024). preprint.

Ramsauer, H./Schäfl, B./Lehner, J./Seidl, P./Widrich, M./Adler, T./Gruber, L./ Holzleitner, M./Pavlović, M./Sandve, G. K./Greiff, V./Kreil, D./Kopp, M./Klambauer, G./Brandstetter, J./Hochreiter, S. (2021): Hopfield Networks Is All You Need. DOI: 10.48550/arXiv.2008.02217. arXiv: 2008.02217 [cs, stat]. URL: http://arxiv.org/abs/2008.02217 (retrieval: 02/28/2024). preprint.

Ranzato, M. a./Poultney, C./Chopra, S./Cun, Y. (2006): Efficient Learning of Sparse Representations with an Energy-Based Model. In: *Advances in Neural Information Processing Systems*. Vol. 19. MIT Press. URL: https://proceedings.neurips.cc/paper/2006/hash/87f4d79e36d68c3031ccf6c55e9bbd39-Abstract.html (retrieval: 03/07/2024).

Robert, C. P. (2016): The Metropolis-Hastings Algorithm. DOI: 10.48550/arXiv.1504.01896. arXiv: 1504.01896 [stat]. URL: http://arxiv.org/abs/1504.01896 (retrieval: 02/27/2024). preprint.

Rosenthal, S. (2009): Optimal Proposal Distributions and Adaptive MCMC. In: *Handbook of Markov Chain Monte Carlo*.

Salakhutdinov, R./Hinton, G. (2009): Deep Boltzmann Machines. In: *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*. Artificial Intelligence and Statistics. PMLR, pp. 448–455. URL: https://proceedings.mlr.press/v5/salakhutdinov09a.html (retrieval: 02/16/2024).

Schlamminger, S. (2014): A Cool Way to Measure Big G. In: *Nature* 510.7506 (7506), pp. 478–480. ISSN: 1476-4687. DOI: 10.1038/nature13507. URL: https://www.nature.com/articles/nature13507 (retrieval: 02/21/2024).

Singh, S. K./Kumar, S./Mehra, P. S. (2023): Chat GPT & Google Bard AI: A Review. In: *2023 International Conference on IoT, Communication and Automation Technology (ICICAT)*. 2023 International Conference on IoT, Communication and Automation Technology (ICICAT), pp. 1–6. DOI: 10.1109/ICICAT57735.2023.10263706. URL: https://ieeexplore.ieee.org/abstract/document/10263706?casa_token=JMHwBzQgxnwAAAAA:70OnfYs5ECetZhuq8D_F3QXyua1Xu65rL0a_Ywve3mch00UAeSs0yVjhWCUvDuBpMX83NAbpUpM (retrieval: 02/23/2024).

Specht, D. F. (1990): Probabilistic Neural Networks. In: *Neural Networks* 3.1, pp. 109–118. ISSN: 0893-6080. DOI: 10.1016/0893-6080(90)90049-Q. URL: https://www.sciencedirect.com/science/article/pii/089360809090049Q (retrieval: 03/07/2024).

Upadhya, V./Sastry, P. (2019): An Overview of Restricted Boltzmann Machines. In: *Journal of the Indian Institute of Science* 99. DOI: 10.1007/s41745-019-0102-z.

Uusitalo, L./Lehikoinen, A./Helle, I./Myrberg, K. (2015): An Overview of Methods to Evaluate Uncertainty of Deterministic Models in Decision Support. In: *Environmental Modelling & Software* 63, pp. 24–31. ISSN: 1364-8152. DOI: 10.1016/j.envsoft.2014.09.017. URL: https://www.sciencedirect.com/science/article/pii/S1364815214002813 (retrieval: 03/07/2024).

Verdon, G./Marks, J./Nanda, S./Leichenauer, S./Hidary, J. (2019): Quantum Hamiltonian-Based Models and the Variational Quantum Thermalizer Algorithm. arXiv: 1910.02071 [quant-ph]. URL: http://arxiv.org/abs/1910.02071 (retrieval: 02/19/2024). preprint.

Wang, T./Roychowdhury, J. (2017): Oscillator-Based Ising Machine. DOI: 10.48550/arXiv.1709.08102. arXiv: 1709.08102 [physics]. URL: http://arxiv.org/abs/1709.08102 (retrieval: 02/15/2024). preprint.

Wang, X./Yan, L./Zhang, Q. (2021): Research on the Application of Gradient Descent Algorithm in Machine Learning. In: *2021 International Conference on Computer Network, Electronic and Automation (ICCNEA)*. 2021 International Conference on Computer Network, Electronic and Automation (ICCNEA), pp. 11–15. DOI: 10.1109/ICCNEA53019.2021.00014. URL: https://ieeexplore.ieee.org/document/9603742 (retrieval: 03/08/2024).

Wittpahl, V., ed. **(2019)**: Künstliche Intelligenz: Technologie | Anwendung | Gesellschaft. Berlin, Heidelberg: Springer. ISBN: 978-3-662-58041-7 978-3-662-58042-4. DOI: 10.1007/978-3-662-58042-4. URL: http://link.springer.com/10.1007/978-3-662-58042-4 (retrieval: 02/15/2024).

Yao, Z./Gripon, V./Rabbat, M. (2013): A Massively Parallel Associative Memory Based on Sparse Neural Networks. In.

Zhai, S./Cheng, Y./Lu, W./Zhang, Z. (2016): Deep Structured Energy Based Models for Anomaly Detection. In: *Proceedings of The 33rd International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, pp. 1100–1109. URL: https://proceedings.mlr.press/v48/zhai16.html (retrieval: 02/19/2024).

Zhang, N./Ding, S./Zhang, J./Xue, Y. (2018): An Overview on Restricted Boltzmann Machines. In: *Neurocomputing* 275, pp. 1186–1199. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.09.065. URL: https://www.sciencedirect.com/science/article/pii/S0925231217315849 (retrieval: 02/15/2024).

Zhou, H./Dong, J./Cheng, J./Dong, W./Huang, C./Shen, Y./Zhang, Q./Gu, M./Qian, C./Chen, H./Ruan, Z./Zhang, X. (2022): Photonic Matrix Multiplication Lights up Photonic Accelerator and Beyond. In: *Light: Science & Applications* 11.1, p. 30. ISSN: 2047-7538. DOI: 10.1038/s41377-022-00717-8. URL: https://www.nature.com/articles/s41377-022-00717-8 (retrieval: 03/18/2024).

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Thema: *Mein Titel* selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

(Ort, Datum)

(Unterschrift)