



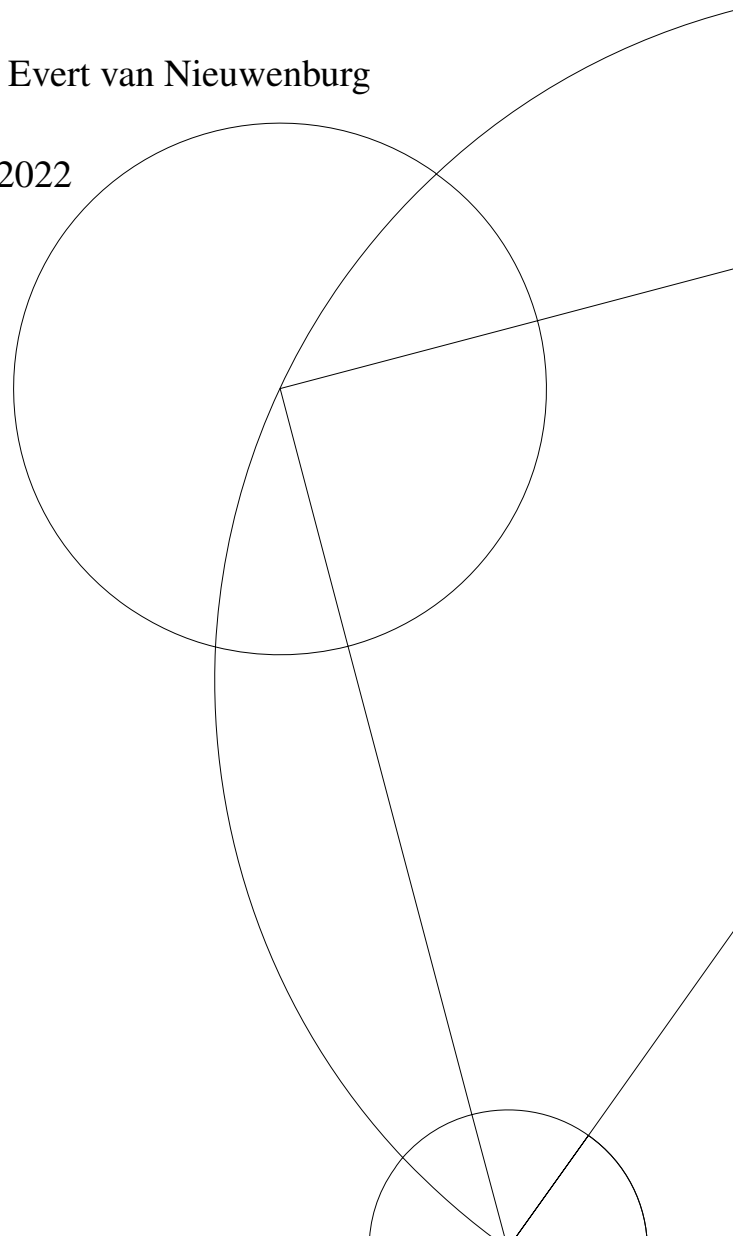
# Variational Autoencoder Analysis of Quantum Systems

Felix Frohnert

vlg978@alumni.ku.dk

Supervisors: Karsten Flensberg, Evert van Nieuwenburg

October 18, 2022



---

## ACKNOWLEDGEMENTS

---

This work has been supported by many people to whom I would like to express my sincere gratitude.

First, I would like to thank Karsten Flensburg for taking on the role of official supervisor for this thesis.

Second, I would like to thank Evert van Nieuwenburg for supporting me at every step of this research project. Your guidance during the writing of this thesis over the last few months deserves special recognition. With your feedback, patience, and encouragement at each meeting, I was able to learn so much more than I expected. For that, I want thank you very much!

Third, I would also like to take this opportunity to thank William Steadman, David Amaro, Dylan Sim, and Abhinav Anand. Working with you enabled me to grow and has had a great impact on me as a researcher over the past two years, which I greatly appreciate.

Finally, my parents, girlfriend, and friends deserve my deep gratitude for their continuous support.

---

## ABSTRACT

---

Representations learning of quantum systems is an important and challenging task at the interface of machine learning and quantum physics that has attracted increasing attention in recent years. Training unsupervised machine learning models to learn features that best describe data generated in a quantum physical process without human intervention can uncover novel representations of such data and provide us with valuable insights into how computers learn to efficiently describe quantum physical systems. In this work, we examine a deep generative model of quantum states where a variational autoencoder is trained on density matrices generated by a 2-qubit quantum circuit. The quantum states are parameterized by a single parameter and differ only in their entanglement properties. In a series of computational experiments, we investigate the learned representation of the model and its internal understanding of the data. We show that the model learns an interpretable representation of quantum states and is able to identify the relationship between the generative parameter and entanglement properties of density matrices. In particular, we find that the latent representation is based on the extraction of information corresponding to the entanglement spectrum. Thus, we are able to interpret how the model structures its latent space and find patterns that we can explain in terms of quantum physics. The results show how we can use and understand the internal representations of deep generative models in a complex scientific domain. The insights from this study represent a step toward interpretable machine learning of quantum states and can be applied to other physical systems.

---

## CONTENTS

---

1	INTRODUCTION	1
1.1	Context . . . . .	1
1.2	Motivation . . . . .	2
1.3	Thesis Outline . . . . .	4
2	BACKGROUND	5
2.1	Unsupervised Machine Learning . . . . .	5
2.1.1	Clustering . . . . .	6
2.1.2	Dimensionality Reduction . . . . .	6
2.2	Representation Learning . . . . .	7
2.2.1	Disentanglement . . . . .	7
2.2.2	Clustering . . . . .	8
2.2.3	Interpretability . . . . .	9
3	VARIATIONAL AUTOENCODERS	10
3.1	Generative Models . . . . .	10
3.2	Autoencoders . . . . .	11
3.2.1	Training . . . . .	12
3.3	Variational Autoencoder . . . . .	13
3.3.1	Problem Statement . . . . .	13
3.3.2	Architecture . . . . .	15
3.3.3	Training . . . . .	16
3.3.4	Reparameterization trick . . . . .	18

3.4	Regularized Variational Autoencoder . . . . .	20
3.4.1	Effect of $\beta$ . . . . .	21
4	MNIST DATA SET . . . . .	23
4.1	Data Set . . . . .	23
4.2	Architecture . . . . .	24
4.3	Results . . . . .	25
5	ISING MODEL DATA SET . . . . .	29
5.1	Ising model . . . . .	29
5.1.1	Phase Transition . . . . .	30
5.2	Data Set . . . . .	31
5.3	Architecture . . . . .	31
5.4	Results . . . . .	33
6	QUANTUM CIRCUIT DATA SET . . . . .	37
6.1	Parameterized Quantum Circuits . . . . .	38
6.1.1	Entanglement Measure . . . . .	40
6.2	Architecture . . . . .	41
6.3	Density Matrices . . . . .	42
6.4	Coherent Noise . . . . .	46
6.5	Incoherent Noise . . . . .	49
6.6	Random Unitary . . . . .	53
7	CONCLUSION . . . . .	62
8	FUTURE WORK . . . . .	64
8.1	Greenberger–Horne–Zeilinger States . . . . .	64
8.2	Representation Learning of Non-parameterized Quantum Circuits . . . . .	65
9	APPENDICES . . . . .	68
9.1	Machine Learning Vocabulary . . . . .	68

9.2	Neural Network Components . . . . .	71
9.3	Analysis Methods . . . . .	73
9.4	Generating random unitary operators . . . . .	74
9.5	Visualization Density Matrix Data Set . . . . .	77
9.6	Interpretation of latent variable $Z_1$ . . . . .	81
	Bibliography	82

---

## LIST OF FIGURES

---

1	Comparing disentangled and entangled latent representation . . . . .	8
2	Clustering of objects with matching color . . . . .	9
3	Interpretable latent space interpolation . . . . .	9
4	Text-to-image generation process of DALL-E . . . . .	11
5	Structure of regular autoencoder . . . . .	12
6	Use-case of variational autoencoder . . . . .	14
7	Structure of variational autoencoder . . . . .	15
8	Probability distribution represented by a latent variable $z$ . . . . .	16
9	Reparametrization trick . . . . .	19
10	Effect of $\beta$ on learned distribution . . . . .	22
11	Excerpt of sampled images from MNIST database . . . . .	23
12	Neural network architecture in MNIST experiments . . . . .	24
13	Learned latent representation of the MNIST data set . . . . .	28
14	Average magnetization at varying temperatures of 2D Ising model . . . . .	30
15	Spin configurations of 2D Ising model at varying temperatures . . . . .	31
16	Neural network architecture used in Ising experiment . . . . .	32
17	Learned latent representation of Ising data set . . . . .	33
18	Correlation between latent parameter and magnetization . . . . .	34
19	Average latent parameter and magnetization at varying temperatures . . . . .	35
20	Parameterized quantum circuit used in experiments . . . . .	39
21	Entanglement Entropy of circuit output states for $\theta \in [0, \pi]$ . . . . .	41

22	Neural network architecture used in quantum circuit experiments . . . . .	42
23	Structure of density matrix data set . . . . .	43
24	Correlation between $\theta$ and latent parameters . . . . .	44
25	Correlation between $\cos^2\frac{1}{2}\theta$ and latent parameters . . . . .	45
26	Structure of coherent noise data set . . . . .	46
27	Correlation between $\theta$ and latent parameters . . . . .	47
28	Correlation between $\cos^2\frac{1}{2}\theta$ and latent parameters . . . . .	48
29	Structure of incoherent noise data set . . . . .	50
30	Correlation between $\theta$ and latent parameters . . . . .	51
31	Correlation between $\cos^2\frac{1}{2}\theta$ and latent parameters . . . . .	52
32	Variance of $\cos^2\frac{1}{2}\theta$ . . . . .	52
33	Structure of random unitary data set . . . . .	54
34	Correlation between $\theta$ and latent parameters . . . . .	55
35	Correlation between $\cos^2\frac{1}{2}\theta$ and latent parameters . . . . .	56
36	Correlation between latent parameters . . . . .	57
37	Correlation between $\theta$ and latent parameters . . . . .	58
38	Latent parameter values and eigenvalues at varying angles . . . . .	59
39	Parameterized quantum circuit . . . . .	64
40	Entanglement entropy of circuit output states for $\theta, \phi \in [0, \pi]$ . . . . .	65
41	Illustration of random quantum circuits . . . . .	66
42	Structure of convolutional neural network layer . . . . .	72
43	Structure of fully connected neural network layer . . . . .	73
44	Illustration of kernel density estimation . . . . .	74
45	Uniformly sampled states . . . . .	75
46	Real component of 2000 states generated uniformly under Haar measure . . . . .	76
47	Visualization density matrix data set . . . . .	77



48	Visualization coherent noise data set . . . . .	78
49	Visualization incoherent noise data set . . . . .	79
50	Visualization random unitary data set . . . . .	80
51	Density matrices of different latent encodings . . . . .	81

---

LIST OF TABLES

---

1	Quantum logic gates used in the parameterized quantum circuit . . . . .	38
2	Quantum logic gates used in the non-parameterized quantum circuits . . . . .	67

---

## ACRONYMS

---

**AE** Autoencoder

**AI** Artificial Intelligence

**ELBO** Evidence lower bound

**GAN** Generative adversarial networks

**NN** Neural Network

**VAE** Variational autoencoder

**t-SNE** t-distributed stochastic neighbor embedding

**PCA** Principal component analysis

---

## INTRODUCTION

---

### 1.1 CONTEXT

Over the past decade, deep neural networks have revolutionized machine learning research. While manual feature engineering with specific domain expertise used to be required for performant models, novel powerful unsupervised representation learning algorithms have proven to be very successful in automatically extracting meaningful features from data. [1] This raises the question of what kind of internal representation the model learns and what features it considers important. Investigating this can be a difficult task: The complexity of deep neural networks makes the internal representation itself complex and often impossible for humans to interpret. In recent years, however, methods have been developed that are particularly well suited for learning meaningful internal representations. The model we focus on in this thesis is the variational autoencoder (VAE), as introduced in Kingma et al. (2013) [2]. Originally intended as a generative model for images, this architecture has shown that it is capable of extracting and representing generative factors, which can be considered dominant features, from a highly complex feature space in a human interpretable manner. For example, if a VAE is fed images of faces, it can automatically learn to encode the smile, hair color, or eye color in separate internal variables. With such a representation, we can vary each internal variable and observe the effects on the output, allowing us to analyze what types of features the model has learned.

## 1.2 MOTIVATION

In general, the state of a complex quantum physical system  $|\psi(\mathbf{X})\rangle$  is a function of one or multiple parameters  $\mathbf{x} \in \mathbf{X}$ . Varying such parameters influences the observable measurement outcomes of the system by generating states with specific physical features. An example would be how varying the external magnetic field in the transverse-field Ising model changes the measurement outcome of spins  $\sigma_z$ . Analyzing the relation between system and generative parameter with conventional dimensionality reduction methods usually results to an uninterpretable representation and does not derive complex concepts directly from the data. Efficient modeling of generative factors with a machine learning approach could generalize the learning of a complex quantum physical system through compositions of simpler abstractions. This will allow us to better understand the internal structure of data generated in quantum processes, explore novel compact representations of such data, and gain insight into the internal representation of quantum systems of an artificial intelligence (AI).

The machine learning model of interest in this study is the variational autoencoder [2], which has been extensively researched in many fields such as computer vision [3–5], natural language processing [6–8], and medical sciences [9–11]. In this unsupervised learning approach, we model data and form representations without explicit goals or direct assumptions about the underlying generative processes. The general technique is to use variational inference to learn a low-dimensional representation of data under certain constraints and regularizations.

The goal of this thesis is to apply this promising approach to complex data generated in quantum physical processes and explore the learned low-dimensional (latent) representation. By creating a data set with known generative factors and using methods such as  $\beta$ -VAE [12] to explore the internal learned latent representation, we gain insights into how machine learning models efficiently compress this type of data. The ultimate goal would be to discover a meaningful latent representation that would help us bridge the gap between representation learning and quantum physics.

With this idea in mind, we can formulate a simple use case: Take a set of quantum states that differ only in their quantum mechanical properties, such as entanglement. By training the unsupervised machine learning model on this data set, we obtain a latent representation for each state. Interpreting this learned representation gives us insight into how an unsupervised AI has learned to distinguish between the different quantum states and what kinds of quantities it has learned to extract from them. Below we propose some guiding questions that represent the interest in this work and that have guided us throughout the project:

- What is a good neural network architecture for learning quantum states?
- Are machine learning algorithms originally intended for image processing capable of learning to reconstruct quantum states correctly?
- What property of the quantum state data is the most amenable to a machine learning model?
- What data format of quantum states should we use as input to our model?
- Does a latent representation for quantum states exist and would it be human interpretable?

To date, VAEs (and other generative models) have been successfully used in various fields of physics. This includes state compression endeavors as in Rocchetto et al. (2018) [13], the study of phase transitions in Wetzel et al. (2017) [14], anomaly detection in particle physics experiments in Pol et al. (2019) [15], and geochemical pattern recognition in Xiong et al. (2022) [16]

There are several aspects that differentiate this thesis from previous studies: The system in Section 6 has not been subject to representation learning studies before. The formulation of the input data as density matrices differs from previous studies, which predominantly used state measurements. In addition, a novel constraint layer was introduced to the neural network architecture. Finally, a strong focus is placed on the interpretation of the learned latent representation, whereas previous studies often neglected this aspect and focused on the reconstruction process of the model, which does not fully exploit the potential of the information provided by the VAE.

### 1.3 THESIS OUTLINE

In the following, the structure of this thesis is detailed. The first chapter serves as an introduction to the topic of generative models in physics and motivates the chosen approach. In the second chapter, we present some preliminary information needed to understand the machine learning background. In the third chapter, we introduce several generative models of interest and gain an intuition for their hyperparameters. In the fourth chapter, we discuss the data set, model architecture, and learned representation of the MNIST database. We use this example unrelated to physics to gain a general understanding of how to interpret the results of a VAE. In the subsequent fifth chapter, we discuss the data set, model architecture, and learned representation of the Ising model. Applying a VAE to this simple classical system will help us understand how unsupervised machine learning methods are able to extract physically meaningful quantities from data. In the sixth chapter, we discuss the data set, model architecture, and learned representation of quantum states, which is the main merit of this thesis. Finally, in the last two chapters, we conclude and provide an overview of future research endeavors that build on the results of this work.

---

## BACKGROUND

---

In this chapter, we introduce important principles and formalisms of machine learning, as they will be frequently used throughout the thesis. As an addition to the general introduction, an overview of important vocabulary is presented in Appendix [9.1](#) and [9.2](#).

### 2.1 UNSUPERVISED MACHINE LEARNING

Unsupervised learning refers to algorithms that learn patterns from unlabeled data. [17] We do not explicitly assign labels or imply direct assumptions about the generative processes underlying the data, so a given machine learning model must independently recognize important features and form internal representations. Without human intervention, these algorithms have the ability to discover hidden patterns or data groupings. Suppose we are given a set of observations  $x$ . In an unsupervised learning approach, the machine learning model attempts to learn the probability distribution  $P(x)$  that generated the data, or some interesting features of that distribution.

Unsupervised learning algorithms are a well-established topic in computer science and have applications in a variety of fields. Two of the most important use cases are: clustering and dimensionality reduction. In the following, we will briefly explain the ideas underlying each method.



### 2.1.1 *Clustering*

Clustering describes a set of techniques aimed at discovering similarity structures in a collection of unlabeled data. Organizing objects into groups whose members are similar in some way can give us insights into the internal structure of the data. Common clustering algorithms include K-means clustering [18], Gaussian Mixture Models [19], DBSCAN [20], and Hierarchical clustering [21].

### 2.1.2 *Dimensionality Reduction*

Dimensionality reduction is the conversion of data from a high-dimensional to a low-dimensional space while retaining meaningful properties of the original data. Dimensionality reduction of data may be desirable for a variety of reasons and is generally divided into two categories: Feature selection and feature extraction. On the one hand, feature selection is simply about selecting a subset of features from the data set to reduce model complexity and remove noise caused by irrelevant features. An example of this is selecting features  $(X, Y)$  and omitting features  $(Z)$  in a data set with three features  $(X, Y, Z)$  to accomplish a particular task. Feature extraction, on the other hand, is about deriving information from the data set to create a new salient feature subspace. An example of this is to map features  $(X, Y, Z)$  to a single value using some function  $f(X, Y, Z) \rightarrow \mathbb{R}$ . The idea behind this is to compress the data with the goal of obtaining a more compact representation that contains only relevant information. As with feature selection, these techniques are also used to reduce the number of features in order to reduce the complexity of the model and remove noise. Common dimensionality reduction methods include principal component analysis (PCA) [22], t-distributed stochastic neighborhood embedding (t-SNE) [23], and autoencoders (AE) [24].

## 2.2 REPRESENTATION LEARNING

(Unsupervised) Representation learning is a subset of unsupervised techniques aimed at learning a (more useful) representation of data. Depending on the context, this can be viewed as learning a representation that preserves important content and omits unnecessary details of the data, which has led to recent advances in Deep Learning [25]: Improving the data efficiency in models [26], their robustness to noise [27], their ability to generalize [28], and to adapt to new tasks [29].

In recent years, there has been increasing interest in the use of deep generative models such as VAEs in the field of representation learning. In this approach, neural networks are used as feature extractors to capture the relationship between input features, and a probabilistic internal latent representation is learned that matches the probability distribution of the underlying generative factors of the observed raw input. This representation is at the core of the representation learning aspect of VAEs. Ideally, it represents new meaningful, interpretable, and generalizable features of the data that are discovered independently of humans. The question of what properties make a good representation is an active area of research, but the current consensus of (non-mutually exclusive) attributes as discussed in Bengio et al. (2021) [30] is summarized in the following sections.

### 2.2.1 *Disentanglement*

Learning a disentangled representation of a data set means that each factor of variation in the data is encoded in independent latent variables. With this, each dimension of the latent representation is independent and responds only to a single generative factor and is thus invariant to variations of other generative factors. [31, 32] In practice, this means that we can clearly identify the attribute that each latent dimension has learned, which helps us to find a clear interpretation of the learned latent representation. Figure 1 illustrates the landscape of the learned representation with an entangled and

a disentangled latent variable. In Figure 1a, we interpolate the learned representation of a data set containing geometric shapes with different colors. We vary the learned latent variable of the model and predict the corresponding output. In the disentangled case, the varied latent variable corresponds to a single generative factor: the color of the object. Figure 1b shows the case of an entangled latent variable that simultaneously encodes two generative factors: the color and shape of the object.

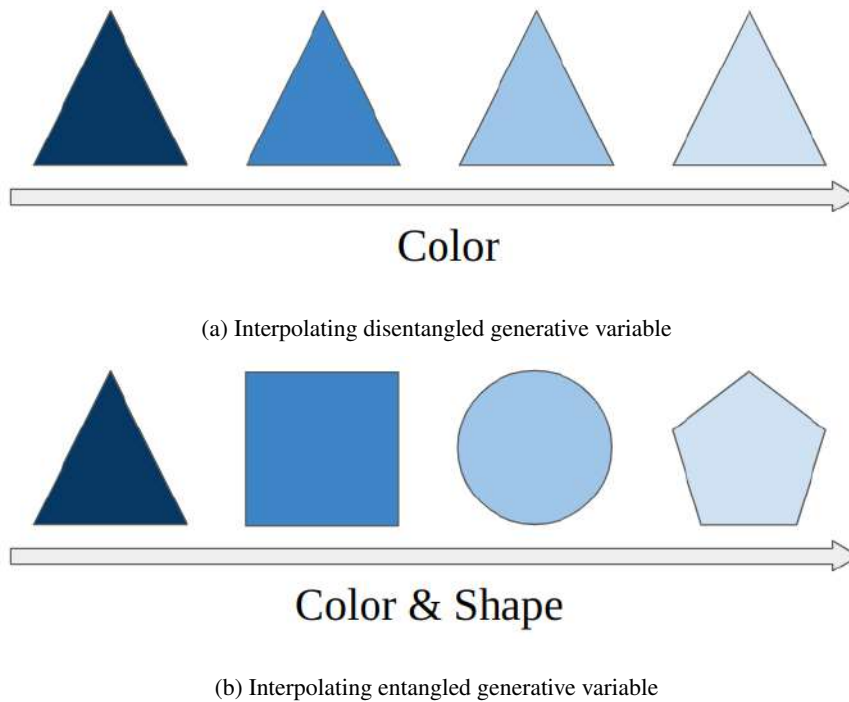


Figure 1: Comparing (a) disentangled and (b) entangled latent representation

### 2.2.2 Clustering

Learning a clustered representation means that data points with similar attributes are encoded in similar latent value domains. In practice, this is a natural consequence of learning an disentangled representation, where each latent dimension corresponds to a unique generative factor. Figure 2 illustrates the clustering of data points with matching colors in a single latent variable.

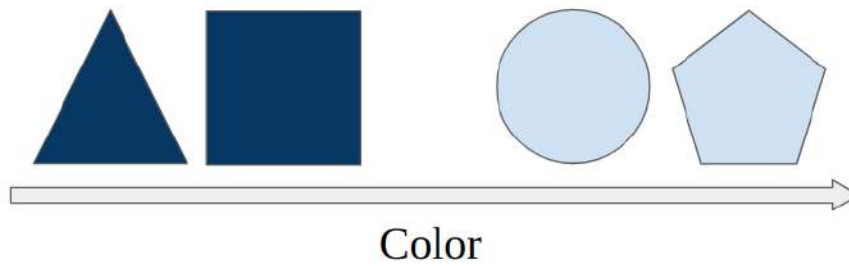


Figure 2: Clustering of objects with matching color

### 2.2.3 Interpretability

Representations are considered interpretable if humans can readily understand the reasons for the embedding decisions made by the model. [33] In practice, this can be understood as embedding data with smooth transitions between encoded features, as shown in Figure 3, where we interpolate the monotonic linear traversal of two latent variables. We consider this learned representation interpretable

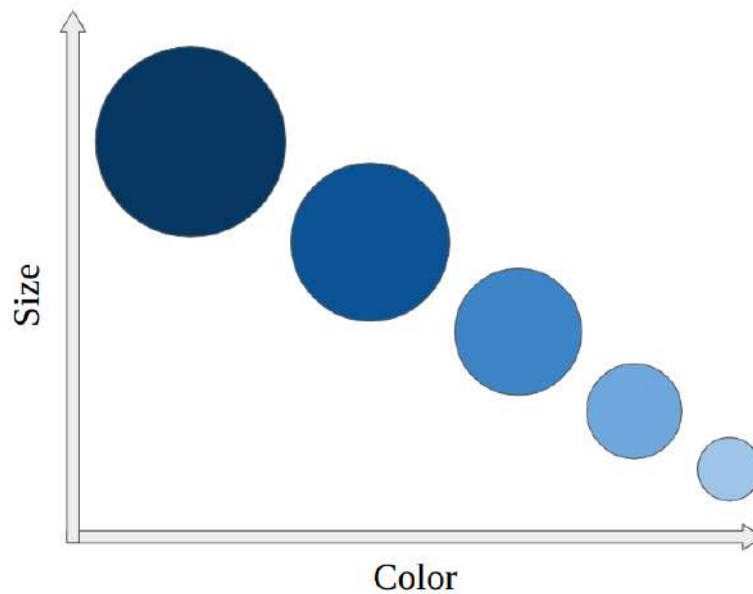


Figure 3: Interpretable latent space interpolation

because we can understand the result of changing a value of one of the latent variables in a linear way: The first variable smoothly changes the size of the object and the second variable changes its hue. In general, the interpretability of a learned representation can be improved in many cases by using machine learning visualization tools such as t-SNE. [23]

---

## VARIATIONAL AUTOENCODERS

---

In this chapter, we first give an overview of generative models in general and introduce the variational autoencoder as one such model that will serve as a framework for all subsequent implementations.

### 3.1 GENERATIVE MODELS

Generative modeling refers to unsupervised machine learning models that generate synthetic data that falls within the probability distribution of some existing data of interest. [34] Prominent examples of this are the famous [This Person Does Not Exist website](#), [This Cat Does Not Exist website](#), or [Dall-E](#), which use complex generative models to generate realistic synthetic images.

Our intuition tells us that if such models are able to generate truly new realistic images that are indistinguishable from the training data, we conclude that it has extracted and learned some general concept of the data. The reason is that in order to generate new data, the model must recognize common patterns to match the training data distributions. In general, this learned concept can be interpreted as a combination of generative factors responsible for variation in the data. [34]

Figure 4 illustrates the performance of a state-of-the-art generative model as presented in Ramesh et al. (2022) [35]. In this example, the text on the left is first transformed into an encoding represented by a set of generative factors, that are chosen by the model, which are then used to produce an image with the desired properties. We propose a simplified set of possible generative factors that are extracted

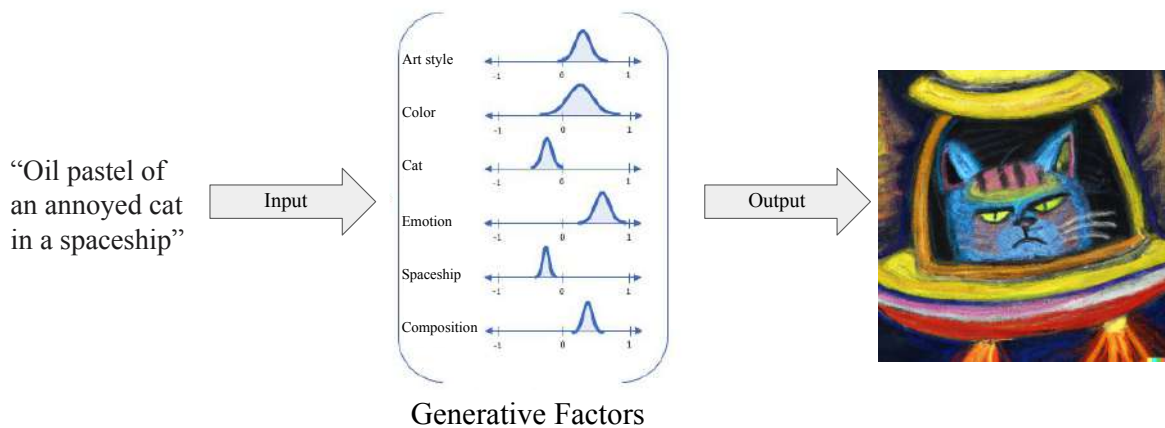


Figure 4: Simplified illustration of text-to-image generation process of DALL-E with generative factors.

from the text and contain variables that control the design of the cat, the spaceship, the art style, the color choices, and the composition. The ability of the model to display and combine these unique concepts tells us that the underlying framework has learned a concept of the keywords contained in the description.

There are a variety of approaches to generative modeling, depending on the purpose of the model, including VAEs [2], generative adversarial networks [36], and Boltzmann machines [37].

### 3.2 AUTOENCODERS

In machine learning, an autoencoder (AE) is a type of neural network that learns an efficient encoding of data through refinement by attempting to regenerate the input. [38] The compression is achieved by passing the input through a bottleneck of smaller dimension, called the latent space, ideally extracting salient features of the data by learning to ignore insignificant noise. AEs and their data compression abilities have found application in a variety of tasks including dimensionality reduction [24], image processing [39, 40], feature extraction [41], facial recognition [42], and anomaly detection [43].

Figure 5 visualizes the general AE architecture. It consists of two parts, the encoder and the decoder. The encoder is a neural network that compresses a  $d$ -dimensional input  $x$ , using a configuration of weights and biases  $\theta$ , into a  $d > d_h$ -dimensional representation. This "bottleneck" structure of the

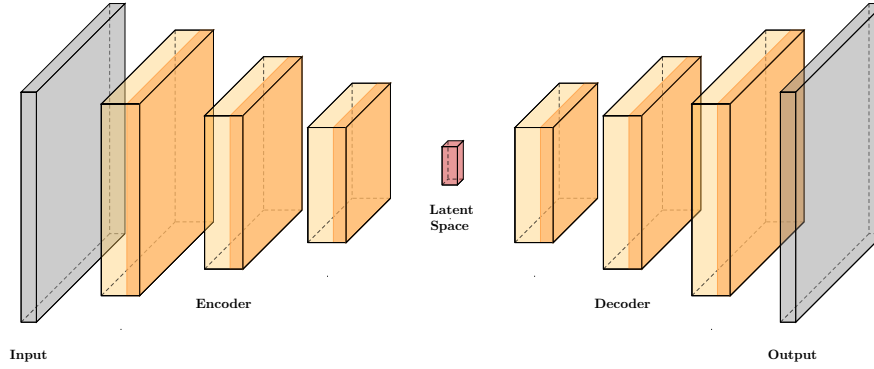


Figure 5: The basic scheme of an autoencoder. Input  $x$  is compressed by the encoder into the latent space. The decoder receives the information from the latent space and produces  $x'$  as similar as possible to  $x$ .

neural network forces the encoder to learn how to efficiently compress the data into the  $d_h$ -dimensional latent space and capture the most salient features of the data. This structure also prevents AEs from learning a simple identity map from input to reconstruction, making it a "undercomplete" model. [34] In the following the encoder is denoted as  $h = E_\theta(x)$ . The decoder is a neural network with weights and biases  $\phi$  that takes the  $d_h$ -dimensional data point and tries to reconstruct the original input. In the following, the decoder will be referred to as  $x' = D_\phi(h)$ .

### 3.2.1 Training

As stated before, the general goal of AEs is to reconstruct their input as well as possible. Formulating this in terms of an optimization problem, this means that our goal in training is to minimize the difference between the input  $x$  and the reconstruction  $x'$  by adjusting the neural network parameters. In order to train the model parameters in an iterative optimization process using backpropagation, we define a differentiable loss function in the following. [44]

$$L_{\phi,\theta}(x) = \mathbb{E}[l(x, D_\phi(E_\theta(x)))] \quad (1)$$

In this  $l$  is a function to measure the reconstruction quality of the model, commonly the mean square error or cross entropy, depending on the nature of the data.

### 3.3 VARIATIONAL AUTOENCODER

A general drawback of AEs is the irregularity of the learned latent representation of data, which makes them sub-optimal generative models. The optimization objective in eq. 1 does not penalize "disorganized" configurations of the latent space, since it focuses exclusively on encoding and decoding data with as little loss as possible. Therefore, we cannot rely solely on the architecture of the encoder, the degrees of freedom in the latent space, and the distribution of the input data to determine the learned latent representation, but must regulate the loss function itself.

In recent years, a multitude of AE variants were created aiming to force the learned representations to assume useful properties, one of them being Variational Autoencoders (VAE). They belong to the family of variational Bayesian methods and probabilistic graphical models, as they change the optimization objective from learning a simple mapping onto a fixed vector to finding an explicit probability distribution which approximates the input data's true distribution. In this approach each sample  $x$  from the data set is represented by a probability distribution in the latent space  $z$ . [2]

#### 3.3.1 Problem Statement

To gain a proper understanding of the VAE and its use cases, we will investigate a generic problem scenario combined with a walk-through of a specific example below. Suppose we have a number of samples  $x$  drawn from a data set  $\mathbf{X} = \{x^{(i)}\}_{i=1}^N$  and know that the data is generated by a process that involves latent variables  $z$ . As an example,  $\mathbf{X}$  consists of images, each displaying a shape with a certain color, e.g. a yellow circle. In the generative process, latent variables  $z^{(i)}$  are drawn from



a prior distribution  $p_{\theta^*}(z)$  and  $x^{(i)}$  is generated from the conditional distribution  $p_{\theta^*}(x|z)$ . In our example, we have two latent variables: one that controls the color and one that determines the shape displayed in an image. Suppose that a sample from the prior distribution represents the parameters for a red triangle, then the conditional distribution with parameters  $\theta^*$  generates the corresponding image. The problem scenario at hand is that we only have access to observation  $x^{(i)}$  whilst the true parameters  $\theta^*$  as well as the values of the latent variable  $z^{(i)}$  are unknown to us. Ideally, we would like to capture the relationship between latent variables and the data distribution with the true posterior density  $p_{\theta}(z|x) = \frac{p_{\theta}(x|z)p_{\theta}(z)}{p_{\theta}(x)}$  where  $p_{\theta}(x)$  is the marginal likelihood  $p_{\theta}(x) = \int p_{\theta}(z)p_{\theta}(x|z)dz$ . In our simple example, this can be interpreted as taking the image of a red triangle and inferring what underlying parameters have generated this composition. However, this integral and posterior are often intractable to efficiently compute for complicated likelihood functions  $p_{\theta}(x|z)$ . [2] For this reason, we use variational inference to estimate the intractable true posterior with an inference model  $p_{\theta}(z|x) \approx q_{\phi}(z|x)$ . During the optimization the parameters  $\phi$  of the inference model  $q_{\phi}(z|x)$  are trained jointly with the parameters  $\theta$  of the generative model  $p_{\theta}(x|z)$ . Figure 6 further illustrates this.

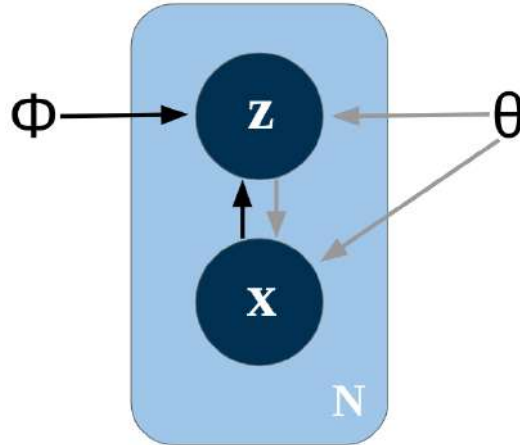


Figure 6: Visualization of the problem scenario. Gray lines denote the generative model  $p_{\theta}(z)p_{\theta}(x|z)$ , black lines denote the variational approximation  $q_{\phi}(z|x)$  of the intractable posterior  $p_{\theta}(z|x)$ . The variational parameters  $\phi$  are learned jointly with the generative model parameters  $\theta$  on  $N$  observations.

### 3.3.2 Architecture

The basic model architecture of a variational autoencoder is shown in Figure 7.

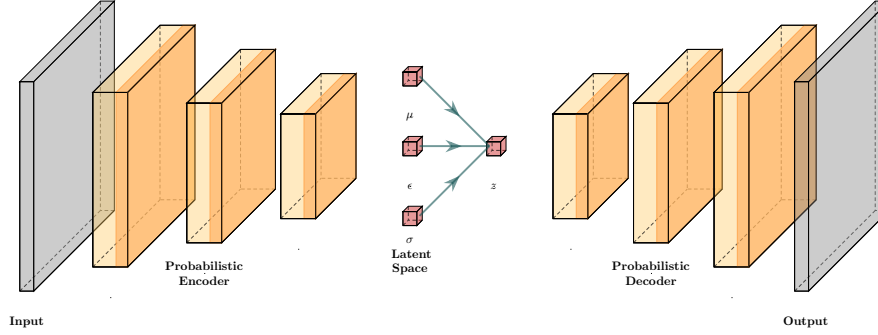


Figure 7: The basic scheme of a variational autoencoder. Input  $x$  is compressed by the encoder into the latent space. The decoder receives the information from the latent space and reconstructs  $x'$ .

The inference model  $q_{\phi}(z|x)$  is a neural network with weights and biases  $\phi$ , denoted as the probabilistic encoder. From a given data point  $x$ , the probabilistic encoder produces a distribution over the possible values of the latent variable  $z \sim q_{\phi}(z|x)$  from which  $x$  could have been generated. In this, " $\sim$ " represents sampling from a probability distribution. An important detail in this is the representation of the latent variables  $z$ . Kingma et al. (2013) [2] takes a univariate Gaussian assumption  $z \sim p_{\theta}(z|x) = \mathcal{N}(\mu, \sigma^2)$  on the true posterior. With this, we can express a sample from our learned approximate posterior distribution as:

$$z = \mu + \sigma \cdot \epsilon \quad (2)$$

This step and the role of  $\epsilon$  is explained in more detail in section 3.3.4. Furthermore, we illustrate how samples of  $z$  are used to visualize the learned probability distribution  $q_{\phi}(z|x^{(i)})$  of  $x^{(i)}$  in Figure 8.

Similarly, the generative model  $p_{\theta}(x|z)$  is a neural network with weights and biases  $\theta$ , denoted as the probabilistic decoder. From a latent variable  $z$ , the probabilistic decoder generates a distribution over the possible corresponding values of  $x$ .

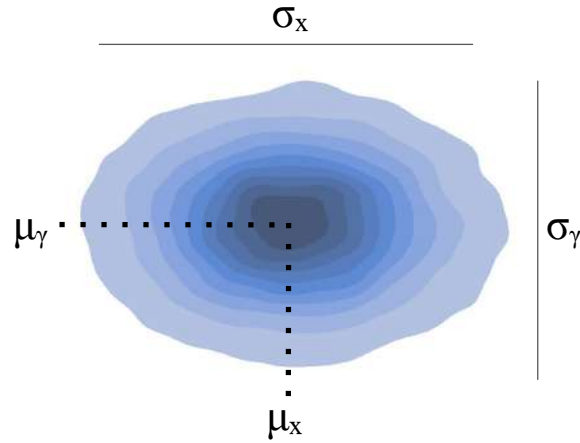


Figure 8: Illustration of the probability distribution  $q_{\phi}(z|x^{(i)})$  of the data point  $x^{(i)}$ . Shown are 2-dimensional latent variable samples  $z$  drawn from the inference model. In this visualization, the hue indicates the probability density.

### 3.3.3 Training

Motivated by the considerations presented in Section 3.3.1, the general goal is to learn parameters  $\theta$  that make the marginal probability distribution  $p_{\theta}(x)$  coincide with the true distribution. A common method of estimating parameters of an unknown probability distribution, given some observed data, is the so-called maximum likelihood estimation. [45, 46] We want to identify and assume a statistical model  $p_{\theta}(x)$  so that the observed data is to be most probable. This is achieved by tuning  $\theta$  towards maximizing the marginal likelihood of observing the data (under the assumption of our model).

$$\theta_{ML} = \arg \max_{\theta} \sum_{i=1}^N \log p_{\theta}(x^{(i)}) \quad (3)$$

The optimization objective of eq. 3 can be interpreted as taking samples  $x$  from the original distribution and maximizing the likelihood to observe them under the assumption of our model  $p_{\theta}(x)$ . As previously mentioned, in practice, the integration aspect of the explicit formulation of the likelihood in Section 3.3.1 makes this optimization problem computationally expensive and in most

cases intractable. In order to circumvent this computational difficulty, we express each term of the sum in terms of the variational lower bound on the marginal likelihood of data point  $\mathbf{x}^{(i)}$  [2] :

$$\log p_{\theta}(\mathbf{x}^{(i)}) \geq \mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \left[ \frac{p_{\theta}(\mathbf{x}^{(i)}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \right] := L_{\phi, \theta}(\mathbf{x}^{(i)}) \quad (4)$$

The variational lower bound can be simplified further.

$$L_{\phi, \theta}(\mathbf{x}^{(i)}) = -KL(q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\theta}(\mathbf{z})) + \mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \left( \log p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) \right) \quad (5)$$

We want to differentiate and maximize this lower bound with respect to both the variational parameters  $\phi$  and the generative parameters  $\theta$ , as a tractable approximation of the total marginal likelihood of the data under the model. Note that when phrasing this as a loss function, we wish to minimize the negative lower bound  $-L_{\phi, \theta}(\mathbf{x}^{(i)})$ .

We are now in a position to interpret the optimization objective. The likelihood term  $\log p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z})$  ensures that we are encoding as much information about the data as possible in latent variables. We can gain intuition on this by following its different computational steps: Starting with a data point  $\mathbf{x}$  and passing it through the encoder gives us an approximated posterior of latent variables  $\mathbf{z}$ , given  $\mathbf{x}$ . Now, we sample from this posterior and pass the samples to the generation network to reconstruct  $\mathbf{x}'$ . The likelihood term now evaluates whether the original input  $\mathbf{x}$  and the reconstruction  $\mathbf{x}'$  are likely to have stemmed from the same underlying probability distribution, which of course requires latent variables that could explain  $\mathbf{x}$  in the generation process. The only way we achieve this is by learning to invert the generation process in our inference network, which is what ultimately want to achieve, as we want to encode information about  $\mathbf{x}$  in our latent variable  $\mathbf{z}$ .

The Kullback–Leibler divergence term  $KL(q_{\phi}(\mathbf{z}|\mathbf{x})||p_{\theta}(\mathbf{z}))$  regularizes the approximate posterior to a chosen prior  $p_{\theta}(\mathbf{z})$ . [47] In general, the choice of the prior is based on an assumption about the optimal distribution of latent variables and can be freely chosen, giving us control over the properties of the learned approximate posterior.

A common choice for a prior is an isotropic Gaussian with zero mean and unit variance  $\mathcal{N}(0, 1)$ . For this prior, the covariance matrix is by definition the identity matrix, which results in the encoded latent space having independent components. Furthermore, with this we ensure that the latent representation of the input data is sufficiently diverse. Without the regularization term, the encoder may return either distributions with small variances (which would be more punctual distributions) or distributions with very different means (which would be very far apart in the latent space). The regularization causes the latent representation  $\mathbf{z}$  of similar input points to be close to each other, making the latent space continuous and meaningful.

Combining these two loss terms, we obtain a loss function that strikes a balance between expressiveness and conciseness, which makes the model reconstruct its input well, but also learns a simple latent space representation based on the chosen prior. [2]

#### 3.3.4 Reparameterization trick

As introduced in Section 3.3.2, the training process involves drawing samples from the approximate posterior to obtain latent parameters. This operation makes it impossible to use backpropagation for gradient descent methods to optimize the reconstruction loss, since it is a non-differentiable operation. To solve this, we introduce the so-called reparameterization trick. [2] Instead of directly drawing random samples from the Gaussian distribution  $\mathcal{N}(\mu, \sigma^2)$ , we can make use of the fact that we can express any Gaussian distribution in terms of the standard normal distribution  $\mathcal{N}(0, 1)$ .

$$\mathcal{N}(\mu, \sigma^2) = \mu + \sigma \cdot \mathcal{N}(0, 1) \quad (6)$$

Using this, we can sample latent parameters by sampling from the standard normal distribution  $\epsilon \sim \mathcal{N}(0, 1)$  and injecting  $\epsilon$  into the latent variable  $\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{\epsilon}$  as auxiliary noise, as in eq. 2. This process and its implementation is further visualized in Figure 9.

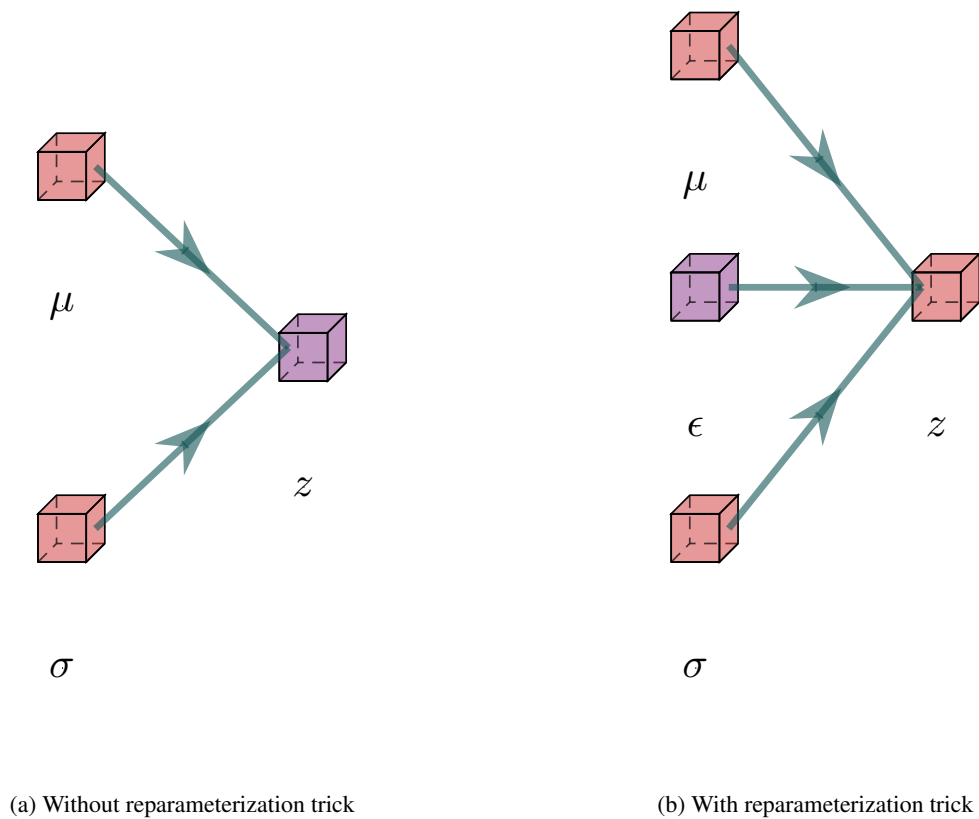


Figure 9: Process of computing latent variables  $z$ . Without the reparameterization trick in (a), the random nature of  $z$  prevents backpropagation over the parameters  $\mu$  and  $\sigma$ . The reparameterization in (b) allows the computation of a gradient across  $\mu$  and  $\sigma$ , since the remaining random variable  $\epsilon$  is injected into the latent variable  $z$  as an external input that can be treated as a per-sample constant. In this visualization, deterministic nodes are shown in red and random nodes in blue.

In this structure, the random variable  $\epsilon$  serves as an external input, which allows us to compute the gradient with respect to  $\mu$  and  $\sigma$  without involving any stochasticity during the parameter update. Finally, in the training process, we treat  $\epsilon$  as a constant for each sample, which is justified by the law of large numbers. Since our training data set is likely to be very large, we expect the introduced error to average over all samples. [2]

### 3.4 REGULARIZED VARIATIONAL AUTOENCODER

There are a number of extensions to the variational autoencoder architecture that allow adaptation to different domains and promise performance improvements. One drawback of regular VAEs is that they tend to map multiple generative factors to one dimension, whereas in an ideal scenario we want each latent dimension to encode a single generative factor for the sake of interpretability. The  $\beta$ -VAE architecture introduces an additional hyperparameter  $\beta$  that weights the KL divergence term of the standard VAE loss function and attempts to solve this issue. [12] Modifying eq. 5 yields:

$$L_{\phi, \theta}(\mathbf{x}) = \mathbb{E}_{z \sim q_{\phi}(z|\mathbf{x})} \log p_{\theta}(\mathbf{x}|z) - \beta \cdot KL(q_{\phi}(z|\mathbf{x}) || p_{\theta}(z)) \quad (7)$$

For different values of  $\beta$ , we impose different learning rules for the model:

- $\beta = 1$  standard VAE
- $\beta = 0$  standard AE
- $\beta \notin \{0, 1\}$  regularized VAE

The motivation for introducing a regularization coefficient  $\beta \notin \{0, 1\}$  can go in one of two directions: First, the application of  $\beta > 1$  is to promote the discovery of independent generative factors in separate latent dimensions, leading to the learning of a disentangled representation, as introduced in Section 2.2.1. In this approach, we favor the implicit independence pressure on the learned posterior due to the isotropic nature of the Gaussian prior over the reconstruction accuracy.

Second, the application of  $0 < \beta < 1$  is to avoid the so-called KL-vanishing problem. In this still poorly understood behavior, the term  $KL(q_{\phi}(z|\mathbf{x}) || p_{\theta}(z))$  becomes vanishingly small early in training, trapping the gradient descent in a local minimum of the loss function landscape. [48, 49] The general idea is that due to the random network initialization at the beginning of the training process, the latent parameters  $z$  are a poor representation of the true data distribution. Now, in the case of

$\beta \geq 1$ , the KL term of the loss function encourages the learned distribution of  $\mathbf{z}$  to move closer to the Gaussian prior. With this, the representation becomes even less representative of the corresponding observations. With each training epoch we repeat this process, making the reconstruction of  $\mathbf{x}$  more and more difficult. Finally,  $\mathbf{z}$  contains so little important characteristics of the observations that it effectively blocks the flow of information required for reconstruction. As a result, the model is forced to learn a simpler solution for the encoding-decoding process: It encodes all samples  $\mathbf{x}$  as unit Gaussians, which makes the KL loss value vanishingly small, hence the name. This poses a problem because no features are actually learned as the representation is almost identical to the uninformative Gaussian prior. Therefore, to avoid this, we favor the reconstruction accuracy over the implicit independence pressure on the learned posterior due to the isotropic nature of the Gaussian prior. The decision whether to set the hyperparameter  $\beta$  to  $\beta < 1$  or  $\beta > 1$  depends on a variety of factors such as the structure of the data set, model architecture, and initialization, and is therefore usually decided by trial and error.

#### 3.4.1 *Effect of $\beta$*

The addition of the hyperparameter  $\beta$  allows us to control the effect of the KL divergence between the approximated posterior and the imposed prior on the optimization process. Figure 10 illustrates the effects of different  $\beta$  values.

In Figure 10a, we illustrate the effect of a small  $\beta$ , which makes the reconstruction loss more significant in the optimization process. With this, we encourage the distribution to focus on describing the input data, which means that our learned distribution deviates from the prior to describe properties of the data. In Figure 10b, we visualize the effect of omitting the regularization with no  $\beta$ . Without regularization, our model is able to learn simple narrow distributions. With a small enough variance, this distribution effectively represents only a single value which results in an unsmooth latent representation. Finally, Figure 10c illustrates the effect of a large  $\beta$ , which places a larger importance on



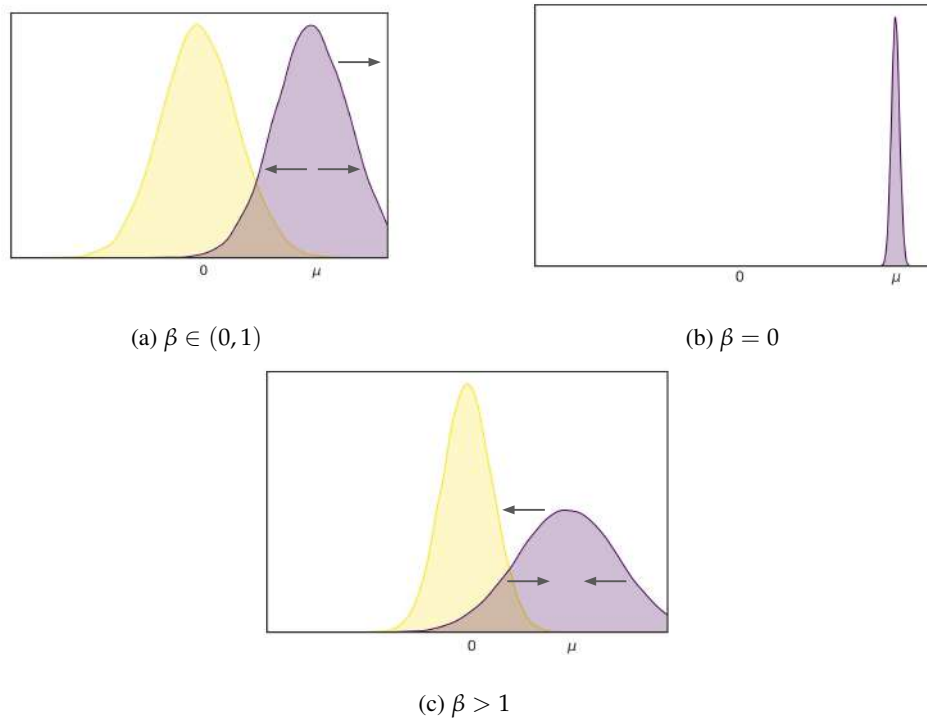


Figure 10: Illustrating the effects of the prior (yellow) on the learned distribution (blue) with varying  $\beta$  values.

the KL loss term in the optimization process. Thus, we encourage the distribution to match the prior.

In the case of a Gaussian this ensures sufficient variance to obtain a smooth latent space.

---

## MNIST DATA SET

---

The Modified National Institute of Standards and Technology (MNIST) database is a publicly available collection of handwritten digits. [50] Its simplicity, ease of use, and well-documented nature make this data set a popular starting example for various machine learning models. [51, 52] We will use the following experiments to gain intuition on the VAE training process, on the hyperparameter  $\beta$  and on how to interpret the learned latent representation.

### 4.1 DATA SET

The database consists of 70000 grayscale images, where each digit from 0 to 9 is represented by 28x28 pixels that we normalize to the range of  $[0, 1]$ .



Figure 11: Excerpt of sampled images from MNIST database

## 4.2 ARCHITECTURE

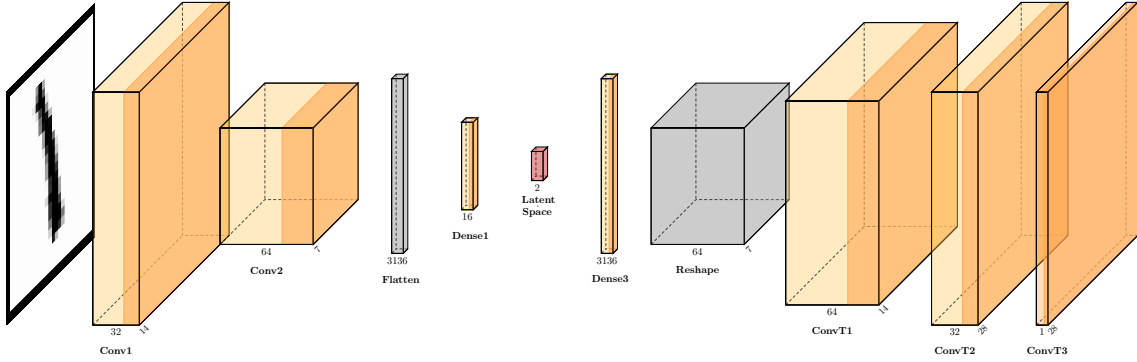


Figure 12: Illustration of the neural network architecture used in the MNIST database experiments. Layers with trainable weights such as convolutional layers and dense layers are colored yellow, where the additional dark yellow hue indicates their activation functions. Formatting layers without trainable parameters are colored gray. The latent space is represented by the color red and contains the reparametrization trick architecture.

Figure 12 visualizes the architecture of the neural network used in the proceeding experiments. Each image  $x \in \mathbb{R}^{28 \times 28}$  in the MNIST data set represents an input digit between 0 and 9. The first half of the encoder is a convolutional neural network (*Conv*) with 2 layers:

$$\mathbf{h} = \text{Conv}_2(\text{Conv}_1(x)) \quad (8)$$

The second half of the encoder is a fully connected layer (*Dense*) that maps the flattened output  $\mathbf{h}$  from the convolutional neural network to the parameters of the 2-dimensional latent space  $\mathbf{z}$ .

$$\mathbf{z} = \text{Dense}_1(\text{flatten}(\mathbf{h})) \quad (9)$$

A more in depth explanation of the neural network components is presented in Appendix 9.2. The decoder consist of a fully connected layer that rescales the sampled latent variables and is then reshaped to fit the transposed convolution.

$$\mathbf{x}' = \text{ConvT}(\text{ConvT}(\text{ConvT}(\text{reshape}(\text{Dense}_3(\mathbf{z})))))) \quad (10)$$

In this architecture, the activation functions of the intermediate layers are rectified linear units and the activation function of the last layer is a sigmoid to normalize the model output to the range  $[0, 1]$ .

### 4.3 RESULTS

In this section we will analyze  $\beta$ -VAE experiments on the MNIST data set in regard to the learned latent representation. In this initial investigation, we choose a 2-dimensional latent space as a starting point for a simple visualization without the need for further dimensionality reduction techniques. For each experiment, we found training on 100 epochs using the Adam optimizer [53] with a learning rate of  $10^{-3}$  and batch size 64 to produce converged loss values. The model was built and trained using the Keras machine learning package from TensorFlow [54].

In the first experiment, we test the  $\beta$ -VAE framework with  $\beta = 0$  on the MNIST data set. With this hyperparameter setting, we completely remove the  $KL$  term from the loss function in eq. 7 and reproduce the performance of a regular autoencoder as introduced in Section 3.2. Figure 13a visualizes the learned latent representation. Each point represents the predicted latent parameters  $\mathbf{z} = (Z_0, Z_1)$  of the encoded distribution of the corresponding input samples  $x$ . We observe that the range of values learned for the two latent variables  $Z_0$  and  $Z_1$  is broad and in both cases quite different. The first latent variable encompasses values in approximately  $Z_0 \in (-20, 2)$  and the second latent variable in  $Z_1 \in (-7, 18)$ . Furthermore, we note that the model learned distinct clusters for digit classes **0**, **2**, and **7**. For the remaining digit classes, we observe a representation with many overlaps. Additionally, we note that the identified clusters are of different densities; although each cluster contains the same number of digit samples, the cluster of the **0** digit class is represented in a more compact value interval than, for example, the **7** digit class. We can attribute a number of the observed properties to the non-regularization of the latent space. Without considering  $KL$  losses in the optimization process, the model focuses exclusively on encoding and decoding data with as little reconstruction loss as possible. Therefore, we learn a disorganized representation, since learning clusters with different

densities or clusters without human-interpretable structure is not penalized. It is important to note that the reconstruction quality of a disorganized latent representation can still be good, as the nonlinearity and deep nature of the neural network architecture enables the model to decode such a representation. This only negatively affects its ability as a generative model, as small changes in the latent variables can lead to very different and therefore unpredictable decoded results.

In the second experiment, we test the model with  $\beta = 1$  on the MNIST data set. This assigns the same importance to the  $KL$  term and the reconstruction term in the loss function in equation 7 and reproduces the performance of a regular VAE introduced in Section 7. Figure 13b visualizes the learned latent representation, where each point represents the predicted latent parameters  $\mathbf{z} = (Z_0, Z_1)$  of the encoded distribution of input samples  $\mathbf{x}$ . We note that the range of learned values for the two latent variables  $Z_0$  and  $Z_1$  is smaller compared to the previous experiment and includes values in  $Z_0 \in (-1, 4)$  and  $Z_1 \in (-4, 2)$ . We also note that the model has learned distinct clusters for the digit classes **0**, **1**, **3**, and **5** and an overlapping representation for the remaining digit classes. We also observe that the densities of the different digit classes are now more similar compared to the previous experiment. We can attribute the change in the properties of this learned representation to the introduction of regularization to the loss function. If we include minimization of the  $KL$  loss to the same extent as minimization of the reconstruction loss in the optimization process, the model focuses on finding a latent representation that leads to good reconstruction quality and tries to encode latent variables that are similar to samples from a normal distributions. In particular, this means that the plotted values are represented in a suitable range close to the mean of the prior (normal) distribution, which is what we observe.

In the third experiment, we test the model with  $\beta = 1.5$  on the MNIST data set. This assigns a higher importance to the regularization of the latent space compared to the reconstruction quality. Figure 13c shows the learned latent representation, where each point represents the predicted latent parameter value  $\mathbf{z} = (Z_0, Z_1)$  of the encoded distribution of input samples  $\mathbf{x}$ . We find that the range of learned values for the two latent variables  $Z_0$  and  $Z_1$  is similar to the previous experiment, with the

only difference being that the value intervals extend more uniformly to positive and negative numbers with  $Z_0 \in (-4, 4)$  and  $Z_1 \in (-3, 5)$ . In this plot, we also find that the model has learned unique clusters for many digit classes and only a small overlapping range. We also find that the densities of the different digit classes are now even more similar compared to the previous experiment. This shows that increasing the importance of the  $KL$  term in the loss function increases the regularization of the latent representation, with the model favoring an encoding distribution of the latent variables that resembles a normal distribution and focusing less on finding a latent representation that leads to good reconstruction quality. Intuitively, with this hyperparameter setting, we have achieved a for this data set unique equilibrium between the clustering nature of the reconstruction loss and the dense packing of the  $KL$  loss, forming distinct clusters that are easy to interpret.

In the forth experiment, we test the model with  $\beta = 15$  on the MNIST data set. This hyperparameter setting effectively removes the influence of the reconstruction term from the loss function in eq. 7. Figure 13d illustrates the learned latent representation. We note that the range of values learned for the two latent variables  $Z_0$  and  $Z_1$  is virtually identical and both encompass values in approximately  $Z_0, Z_1 \in (-4, 4)$ . Furthermore, we note that the model failed to learn any visible clusters for all digit classes from 0 to 9. By placing a very strong incentive on the regularization of the latent variables, we force the model to learn an encoded distribution that matches the prior distribution. In this experiment, this is a normal distribution  $\mathcal{N}(0, 1)$ . Thus, information about the actual properties of the samples is not regarded in the optimization; the learned latent representation visualizes the correlation between samples of two normal distributions.

In summary, we explored an unsupervised machine learning method to learn features that best describe images from the MNIST database. The neural network-based VAE is applied to 28x28 images of single integers without any a priori knowledge of the underlying structures or the number of unique digits in the data set. We note that when  $\beta$  is set correctly, the learned latent representation of the data is clustered, allowing identification of the different digit classes without prior knowledge of their existence.

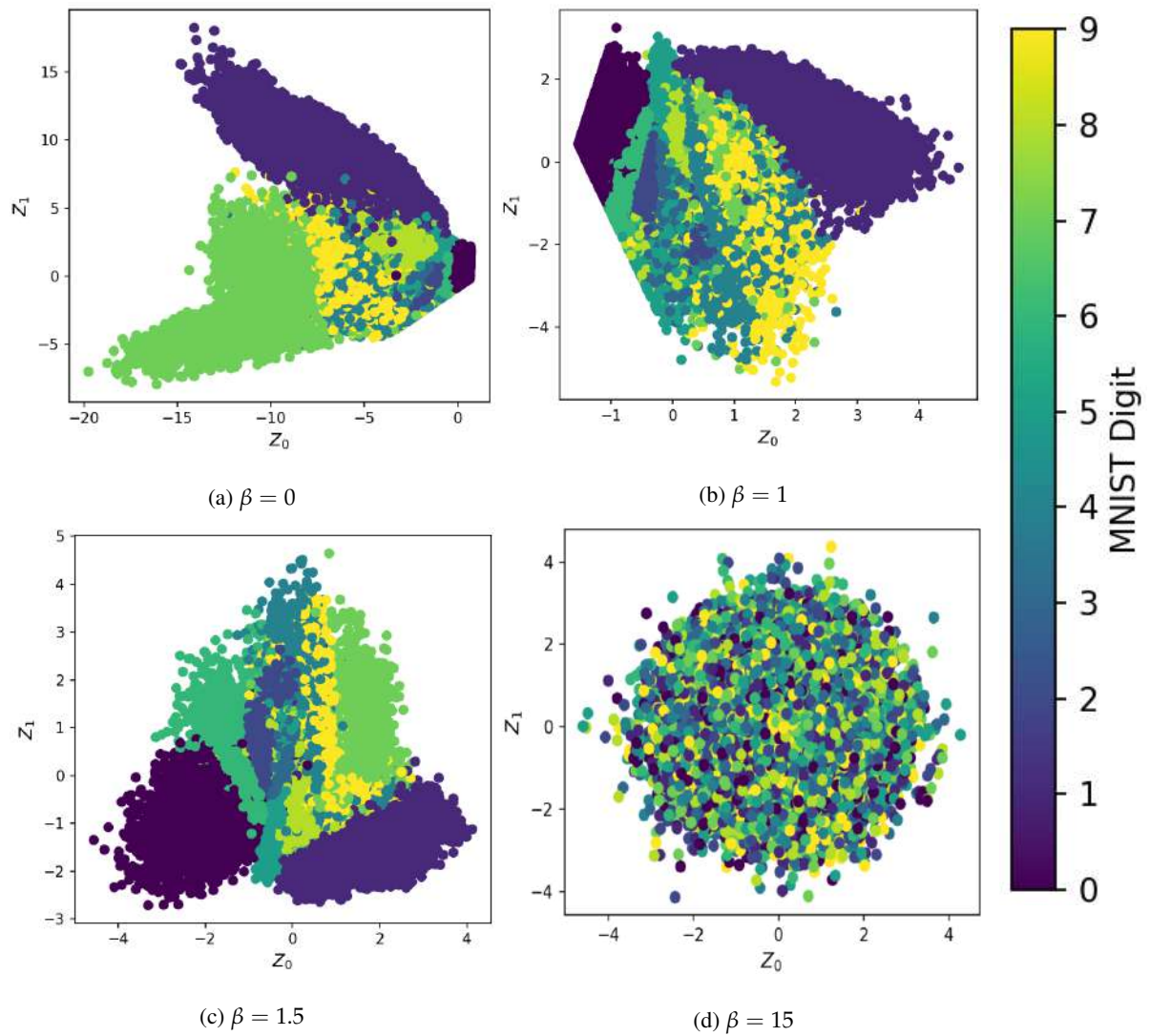


Figure 13: Distribution of digit classes in the learned 2-dimensional latent representation of the MNIST data set. Each marker represents the predicted latent parameter value  $z = (Z_0, Z_1)$  of an input digit  $x$  and the hue represents the associated digit class.

---

## ISING MODEL DATA SET

---

In this chapter, we will analyze the 2-dimensional Ising model and investigate its learned latent representation. This serves as an introduction of VAE based representation learning of classical systems and helps us gain intuition on how to extract and interpret a physically meaningful data representation. This model was chosen for its simplicity and well-researched nature in the field of unsupervised learning. [55–57] The goal of the subsequent chapter is to map raw Ising configurations at varying temperatures to a single latent variable that is able to discriminate between the samples using a generative factor inferred by the machine learning algorithm.

### 5.1 ISING MODEL

The 2-dimensional Ising model is a simple model of ferromagnetism showing a phase transition. [58] It consists of discrete variables representing the magnetic dipole moments of atomic "spins" that can be in one of two states  $\uparrow$  or  $\downarrow$ . These spins are arranged on a lattice, which allows interaction between neighbors. Neighboring spins with matching configurations are in an energetically favorable state and have lower energy than those that do not, so the system tends to the lowest energy. The



addition of heat disrupts this tendency, allowing for different structural phases. The Hamiltonian of the Ising model on a square lattice with vanishing external magnetic field is defined as follows.

$$H(S) = -J \sum_{\langle i,j \rangle} s_i s_j \quad (11)$$

In this formulation a spin configuration  $S = \{s_i\}_{i=1}^N$  assigns a spin to each lattice site  $i$  with  $s_i \in \{+1 = \uparrow, -1 = \downarrow\}$  and  $J = 1$  is defined as the uniform interaction strength.

### 5.1.1 Phase Transition

As mentioned in the previous section, we observe a phase transition when studying spin configurations at different temperatures. At temperatures below the critical temperature of  $T_c \approx 2.269$ , we observe that the spins become increasingly ordered and tend to point in the same direction. At temperatures above the critical temperature, the spins become increasingly disordered and tend to be in a random configuration. To quantify this phenomenon, we can measure the magnetization  $M(S) = \frac{1}{N} \sum s_i$ , which tends to  $\pm 1$  below  $T_c$  and 0 above  $T_c$ , as shown in Figure 14. Using this parameter and finite-size scaling, we can quantitatively identify the thermal phase transition by investigating increasingly larger system sizes and observing that the transition becomes sharper and sharper.

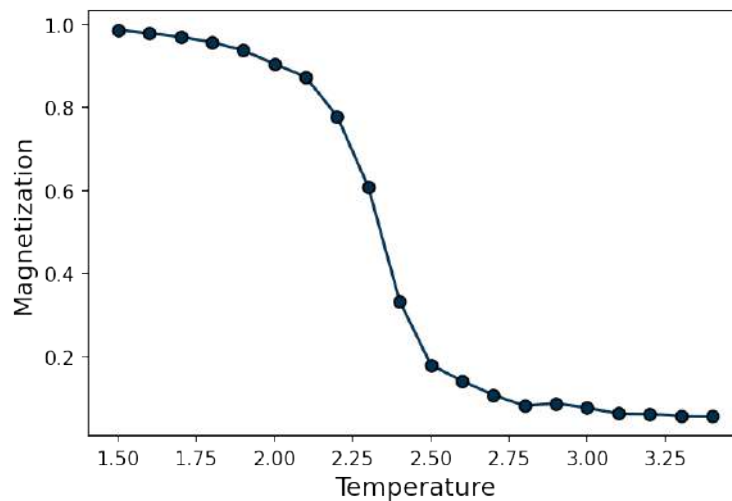


Figure 14: Average magnetization of  $N = 100$  samples at varying temperatures of the 2D Ising model.

## 5.2 DATA SET

The data set consists of 3000 2-dimensional images with binary pixel values, where black represents spin  $+1$  and white spin  $-1$ . Each image is a sample of the configuration of a  $40 \times 40$  Ising model. We vary the temperature in the range of  $T \in [1.5, 3.5]$  with step size  $\Delta T = 0.1$  and record 150 samples at a time using the Wolff algorithm. [59] Thus, temperature  $T$  is the singular generative factor of samples in the data set. The general structure and variation of the training data set is illustrated in Figure 15.

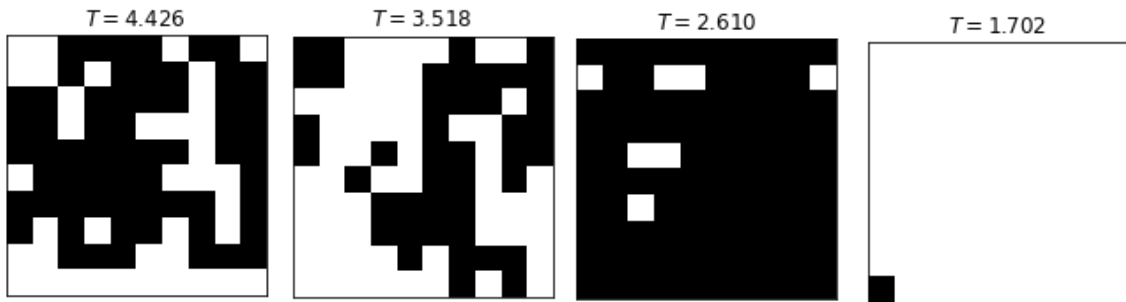


Figure 15: Example of spin configurations of the 2D Ising model at varying temperatures.

To further simplify the data set, we only consider states with positive magnetization. The sign of a given configuration with negative magnetization  $M(s) < 0$  is simply flipped with  $M(s)_+ = -M(s)$ .

## 5.3 ARCHITECTURE

Figure 16 visualizes the architecture of the neural network used in the proceeding experiment. Each image in the Ising data set represents a sampled binary spin configuration  $\mathbf{x} \in \mathbb{Z}_2^{40 \times 40}$ . The encoder is a single fully connected layer that receives a flattened spin configuration sample as input :

$$\mathbf{h} = \text{Dense}_1(\text{flatten}(\mathbf{x})) \quad (12)$$

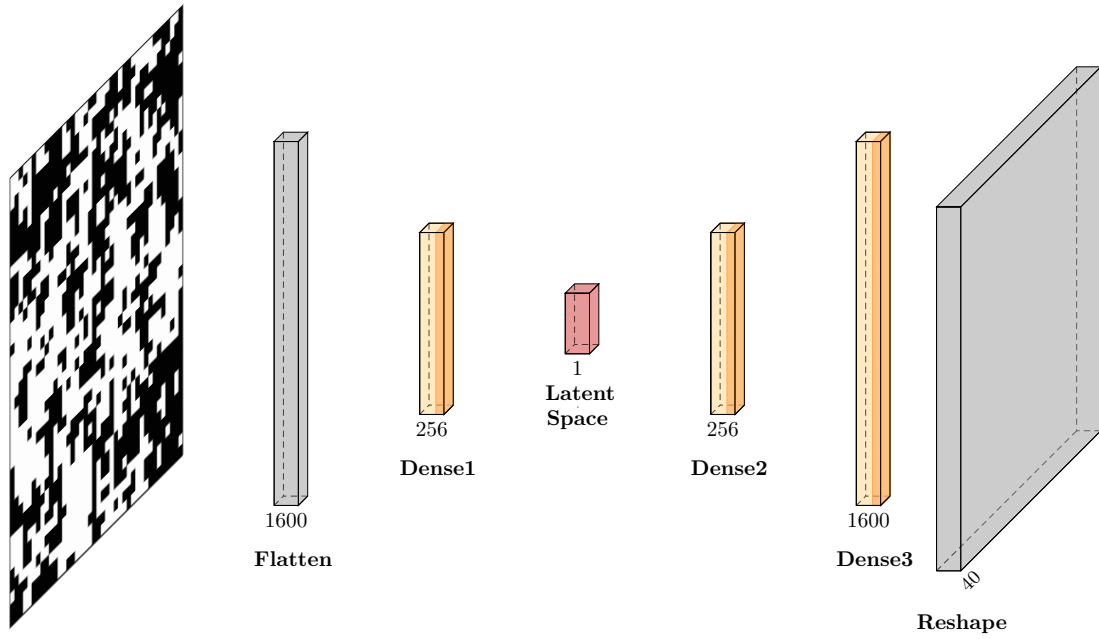


Figure 16: Illustration of the neural network architecture used in the Ising data set experiment. The dense layers with trainable weights are colored yellow, where the additional dark yellow hue indicates their activation functions. Formatting layers without trainable parameters are colored gray. The latent space is represented by the color red and contains the reparametrization trick architecture.

The encoder output  $h$  then gets mapped to the 1-dimensional latent space  $z$ . The decoder consist of two fully connected layers that rescale the sampled latent variable and reshapes the data to the original dimension.

$$\mathbf{x}' = \text{reshape}(\text{Dense}_3(\text{Dense}_2(\mathbf{z}))) \quad (13)$$

In this architecture, the activation functions of the intermediate layers are rectified linear units and the activation function of the last layer is a sigmoid to predict the probabilities for spin  $\uparrow$  or  $\downarrow$  in the Ising model.

## 5.4 RESULTS

In this section, we will analyze  $\beta$ -VAE experiments on the Ising data set in terms of the learned latent representation. In this initial experiment, we choose a 1-dimensional latent space as we want to investigate the capacity of the model to discover the single generative factor temperature  $T$ . We found that training on 100 epochs using the Adam optimizer with a learning rate of  $10^{-3}$ ,  $\beta = 1$ , and batch size 64 produces convergent loss values.

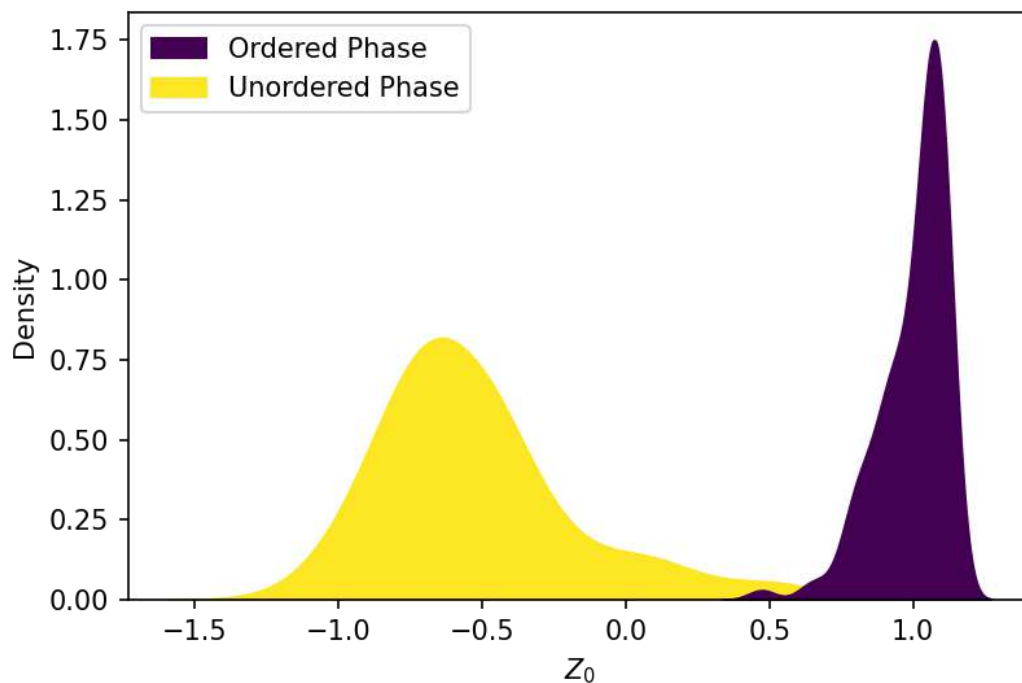


Figure 17: Kernel density estimation of learned latent parameter values  $z = Z_0$ . The yellow hue corresponds to data of the unordered phase and the blue hue correspond to data of the ordered phase.

Figure 17 visualizes the occurrence of latent parameters and illustrates the learned representation. Displayed are the density estimates of the latent parameter values  $z = Z_0$  of input samples  $x$ . The hue in the visualization indicates whether the predicted value corresponds to a sample from the ordered (blue) or unordered phase (yellow). We observe that the distribution of predicted parameters consists of mostly unique values from samples of both phases. Samples from the disordered phase occur most

frequently with an encoding around  $Z_0 \approx 1$ , while samples from the ordered phase most frequently occur around  $Z_0 \approx -0.6$ . From this observation we infer that the model has learned to distinguish samples according to the phase in which they were generated at.

However, what we cannot infer from this graph is *how* the model has learned this representation. Therefore, in the following experiments we investigate the mapping from the input spin configuration to the latent parameters of the model and examine which property of the input data is most amenable to the VAE.

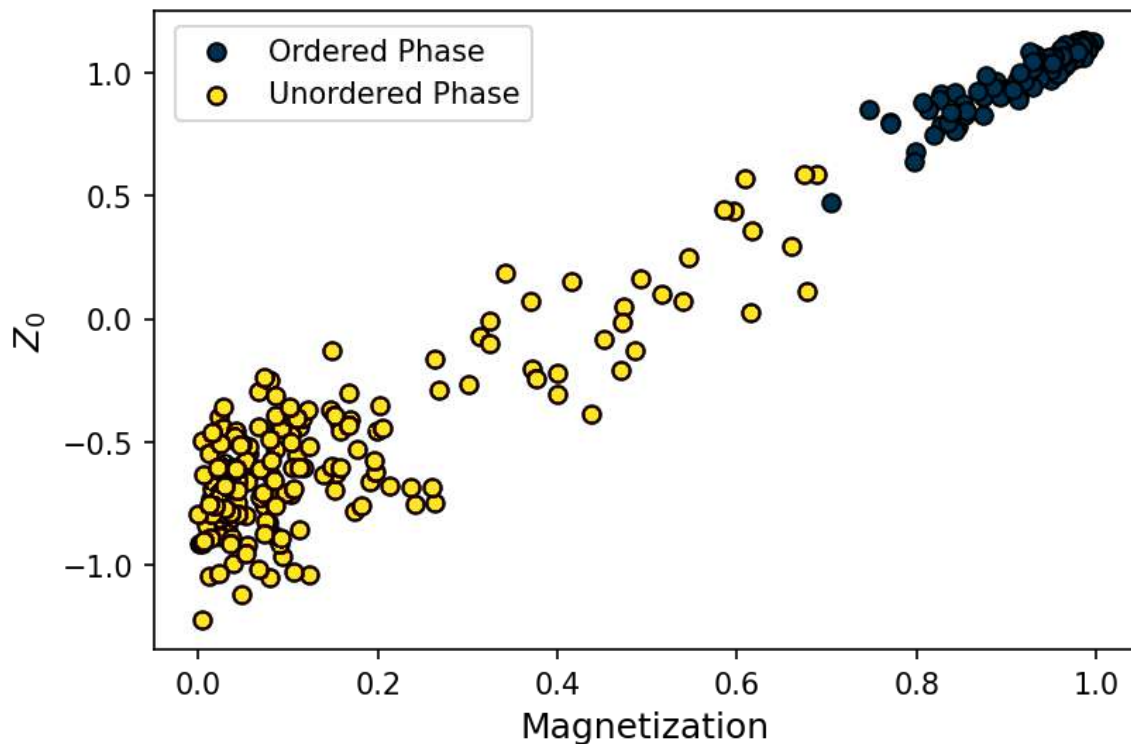


Figure 18: Correlation between latent parameter values  $z = Z_0$  and magnetization for each spin sample. Yellow dots indicate points in the unordered phase, while blue dots indicate points in the ordered phase.

Figure 18 illustrates the correlation between the learned latent parameter values and the magnetization of input samples. We observe two distinct clusters of data: A cluster of samples with magnetization in  $M(s) \in (0, 0.3)$  with latent parameter values of  $Z_0 \in (-1.1, -0.4)$  and a cluster of samples with magnetization in  $M(s) \in (0.8, 1.0)$  with latent parameter values of  $Z_0 \in (0.8, 1.0)$ . This again reiterates the information shown in the previous Figure 17; The VAE has learned to assign

unique parameter ranges for ordered and unordered phase samples. Viewing all markers as a whole in Figure 18 illustrates new information: We observe an approximately linear relationship between the magnetization of a given sample and its latent parameter. This gives us important insight into the learned map from input to latent parameter. The observed linear relation visualizes that learned representation does not only distinguish between samples from both phases but also matches the magnetization  $M(S)$  up to a constant. From this we infer that the learned map is based on a quantity closely related to the magnetization of a given sample. The exact relationship between  $z$  and  $M(S)$  is explored in Figure 19.

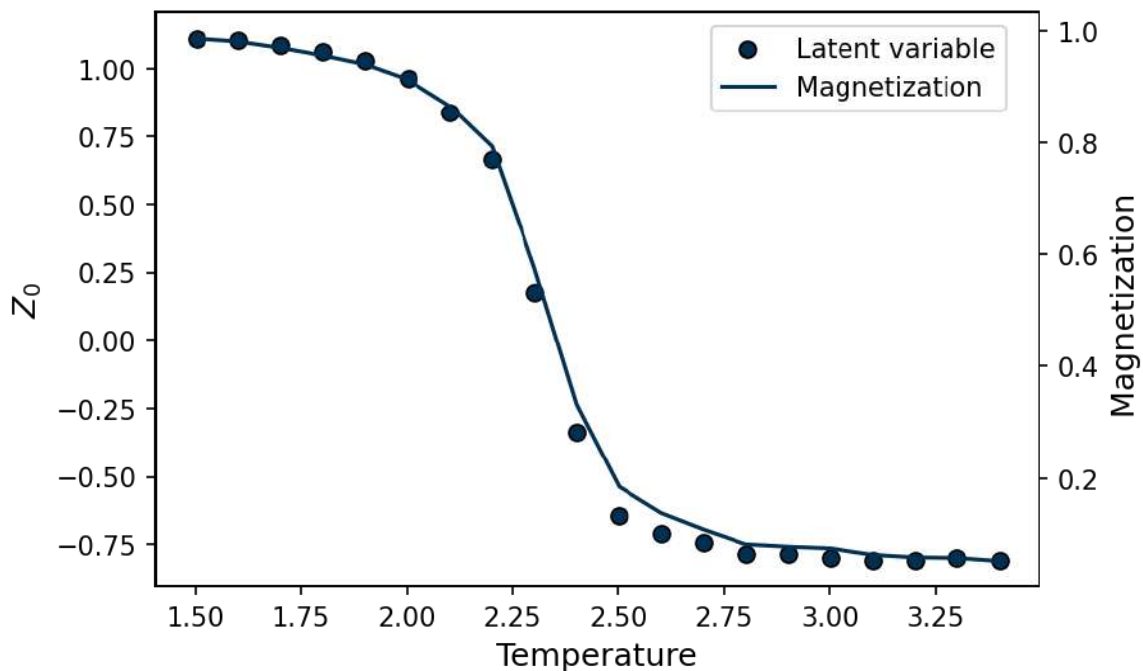


Figure 19: Average latent parameter value  $z = Z_0$  and magnetization of  $N = 150$  configuration samples generated at temperatures in the interval  $T \in [1.5, 3.4]$ .

In Figure 19, we visualize the average magnetization and latent parameter value as a function of temperature. We observe that both curves have the same shape as a function of temperature and only differ in an offset. This further exemplifies that the representation is closely related to magnetization.

In the following, we interpret this result. To reiterate the problem scenario: We took a model with no prior knowledge of the Ising model and trained it with samples of spin configuration at different

temperatures. This training data set is defined by a single generative factor, the temperature  $T$  at which the samples were generated at, that determines the distribution of  $\pm 1$  for each sample. As introduced in Section 7, our model attempts to learn an efficient encoding-decoding method during training. The architecture presented in Figure 16 requires the model to encode into and decode from a single latent variable. An optimization strategy for this is to learn a mapping from a given sample to a single generative factor. In this experiment, each temperature is uniquely defined by its magnetization. Therefore, by computing the magnetization of a given sample, we can distinguish between samples generated at different temperatures, which accounts for the variance in the data set. With this intuition, learning to compute a quantity proportional to the magnetization of each sample is the optimal strategy to structure the latent representation. Note that it is not necessary to *exactly* calculate the magnetization, since a map with an offset with respect to magnetization can still capture the behavior of the underlying generative factor. As an addition, although this was not the focus of this experiment, we can use the fact that the model has learned to approximate the order parameter magnetization to identify the phase transition at  $T_c \approx 2.269$ . Even without prior knowledge of this order parameter, we are able to use the results to derive the position of the phase transition, since the latent parameters also exhibit a steep change at  $T_c$ .

In summary, we have explored an unsupervised machine learning technique to learn features that best describe configurations of the two-dimensional Ising model. The neural network-based VAE is applied to sampled Ising configurations and has no a priori knowledge of its Hamiltonian or the order parameter. We find that the predicted latent parameters is based on a quantity closely related to the known order parameter. With this, the machine learning model has learned to approximate a physical quantity in an human interpretable manner. Moreover, the latent representations of the models are clustered, which allows identification of phases without prior knowledge of their existence.

---

## QUANTUM CIRCUIT DATA SET

---

In this chapter, we will analyze the states generated by a simple quantum circuit and study the learned latent representation. This is the main merit of this thesis and serves as an introduction to representation learning of quantum states.

The purpose of using a VAE in this context is to extract a low-dimensional representation of quantum states that would otherwise be difficult to compare directly without prior domain knowledge of methods derived from quantum mechanics used to accomplish the same tasks. In this case, we are interested in distinguishing states with varying amounts of entanglement. The motivation to use a VAE to encode and decode quantum states lies in the desire to automate the parameterization of states without using conventional quantum mechanical methods, but instead, the neural network shall learn and discover important features directly from the states themselves. The latent representations will be a small set of variables that should be able to discriminate between states, relying on the assumption that proximities in the latent representations accounts for structural similarity between states in their original density matrix representation. In this way, the VAE is used as an alternative to conventional quantum mechanical methods to extract entanglement information from the states. The unique feature of this work is the use of density matrices as input to the machine learning model and the attempt to interpret the learned latent representation. Many works in the literature use (informationally complete) measurements of states as input and do not attempt to interpret the learned representation.



## 6.1 PARAMETERIZED QUANTUM CIRCUITS

Parameterized quantum circuits (PQCs) are a central component of many variational quantum algorithms (VQAs) with applications in quantum chemistry, combinatorial optimization, and machine learning tasks. [60, 61] The general concept behind these VQAs is to employ a PQC to generate a trial wavefunction on a quantum device which is measured to minimize some objective function. [62] In these examples, the quantum circuit serves as a framework for quantum computation, including the initialization of qubits  $|0\rangle$ , the application of a sequence of quantum gates  $U(\theta)$ , and extracting the final state  $U(\theta)|0\rangle = |\psi(\theta)\rangle$ . The PQC in this study consist of two quantum logic gates: The Hadamard gate and the Controlled-Ry gate.

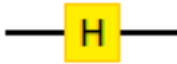
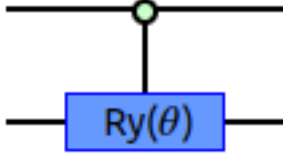
Gate	Visual	Operation
Hadamard		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Controlled-Ry		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta/2 & 0 & -\sin\theta/2 \\ 0 & 0 & 1 & 0 \\ 0 & \sin\theta/2 & 0 & \cos\theta/2 \end{bmatrix}$

Table 1: Quantum logic gates used in the parameterized quantum circuit

In the illustration in Figure 20 we visualize the compiled circuit. The horizontal axis is time, starting at the left hand side and ending at the right. Horizontal lines are qubits. The items that are connected by these lines are gates performed on the qubits. [63]

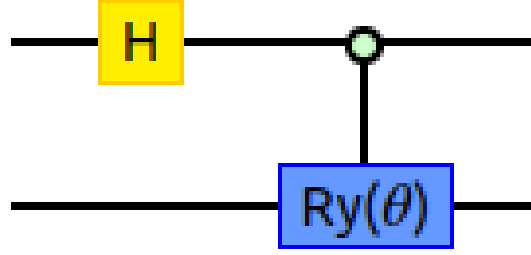


Figure 20: Parameterized quantum circuit consisting of a Hadamard and controlled  $R_y(\theta)$  gate.

The 2-qubit circuit consists of two operations: Initially, both qubits are initialized to the state  $|0\rangle$ .

Second, we apply a Hadamard gate to the first qubit which puts its state into superposition:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad (14)$$

The combined state of both qubits after this operation can be described as follows.

$$\mathbb{1}|q_1\rangle \otimes H|q_0\rangle = (\mathbb{1} \otimes H)|00\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} |00\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |0+\rangle \quad (15)$$

Third, we use the controlled  $R_y(\theta)$  gate to entangle both qubits. Depending on the angle  $\theta$ , we change the final state  $U(\theta)|0\rangle = |\psi(\theta)\rangle$  of the circuit we extract.

$$CR_y|0+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2} \end{pmatrix} = |\psi(\theta)\rangle \quad (16)$$

This yields the following density matrix.

$$\rho(\theta) = |\psi(\theta)\rangle \langle \psi(\theta)| = \frac{1}{2} \begin{pmatrix} 1 & 0 & \cos\frac{1}{2}\theta & \sin\frac{1}{2}\theta \\ 0 & 0 & 0 & 0 \\ \cos\frac{1}{2}\theta & 0 & \cos^2\frac{1}{2}\theta & \sin\frac{1}{2}\theta\cos\frac{1}{2}\theta \\ \sin\frac{1}{2}\theta & 0 & \sin\frac{1}{2}\theta\cos\frac{1}{2}\theta & \sin^2\frac{1}{2}\theta \end{pmatrix} \quad (17)$$

This parametric family of states is of interest to us as they are defined by a single generative factor, namely the angle  $\theta$ . By varying  $\theta$ , we vary the amount of entanglement entropy of the final state of the two qubits: The transition that occurs with the change of  $\theta$  is that at  $\theta = 0$  we have a fully separable state and at  $\theta = \pi$  we have a maximally entangled Bell state. [63]

### 6.1.1 Entanglement Measure

We measure the degree of quantum entanglement between two subsystems with the so-called entanglement entropy measure. Given a pure bipartite quantum state of the combined system, it is possible to obtain a reduced density matrix describing knowledge of the state of a subsystem. [63] We compute entanglement entropy by measuring the Von Neumann entropy of the reduced density matrix for any of the subsystems. If it is non-zero, i.e. the subsystem is in a mixed state, it indicates the two subsystems are entangled as indicated by eq. 19

$$S(\rho_A) = -\text{Tr}(\rho_A \log \rho_A) \quad (18)$$

$$= -\sum \lambda_i \log(\lambda_i) \quad (19)$$

with  $\rho_A = \text{Tr}_B(\rho_{AB})$  is the reduced density matrix for partition  $A$  and  $\lambda_i$  are the eigenvalues of the reduced density matrix. In the following, we will calculate the entanglement entropy of the output states of the circuit in eq. 17 by starting with the calculation of the partial trace.

$$\rho_A = \frac{1}{2} \begin{pmatrix} 1 & \cos\frac{1}{2}\theta \\ \cos\frac{1}{2}\theta & 1 \end{pmatrix} \quad (20)$$

To access the eigenvalues of the matrix, we diagonalize it below:

$$diag = \frac{1}{2} \begin{pmatrix} 1 - \cos\frac{1}{2}\theta & 0 \\ 0 & 1 + \cos\frac{1}{2}\theta \end{pmatrix} \quad (21)$$

The eigenvalues are then simply:

$$x_0 = \frac{1}{2}(1 - \cos\frac{1}{2}\theta) \quad (22)$$

$$x_1 = \frac{1}{2}(1 + \cos\frac{1}{2}\theta) \quad (23)$$

Finally, using eq. 19, we can calculate the entanglement entropy of the two subsystems. The values of entanglement entropy as a function of theta are shown in Figure 21.

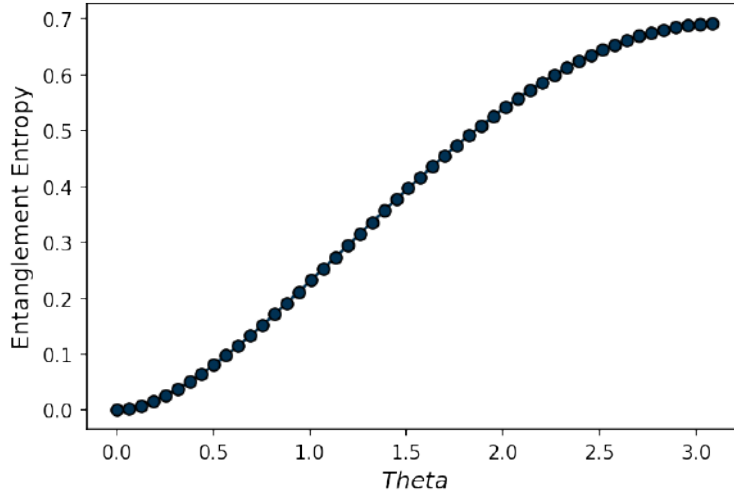


Figure 21: Entanglement Entropy of circuit output states for  $\theta \in [0, \pi]$

## 6.2 ARCHITECTURE

Figure 22 visualizes the architecture of the neural network used in the proceeding experiment. Each input data point is a density matrix  $x \in \mathbb{R}^{4 \times 4}$ . The encoder consists of three fully connected layers.

$$h = \text{Dense}_3(\text{Dense}_2(\text{Dense}_1(\text{flatten}(x)))) \quad (24)$$

The output  $h$  then gets mapped to the latent space  $z$ . The decoder consist of a three fully connected layers that rescale the sampled latent variable.

$$x^* = \text{Dense}_6(\text{Dense}_5(\text{Dense}_4(z))) \quad (25)$$

In this architecture, the activation functions of the intermediate layers are hyperbolic tangent functions and the activation function of the final layer is a linear unit. Finally, the constraint block enforces that output is a positive semidefinite hermitian operator of trace one, ensuring that the learned output is always a valid density matrix. This novel design choice removes the need of the network to learn any constraints of the system, and focuses the training effort on extracting meaningful quantities from the density matrices.

$$x' = \text{constraint}(x^*) \quad (26)$$

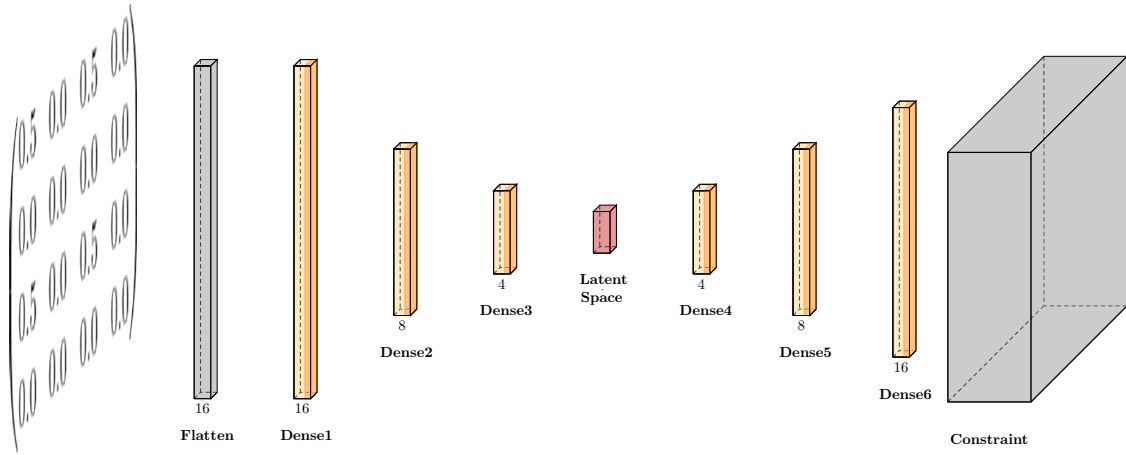


Figure 22: Illustration of the neural network architecture used in the quantum circuit data set experiment.

The dense layers with trainable weights are colored yellow, where the additional dark yellow hue indicates their activation functions. Formatting layers without trainable parameters are colored gray. The latent space is represented by the color red and contains the reparametrization trick architecture.

### 6.3 DENSITY MATRICES

#### DATA SET: DENSITY MATRICES

In the first experiment, the data set consists of 5000 density matrices  $\rho(\theta)$ , each of which represents the final state of the circuit shown in Figure 20. The generative angles are chosen in the interval  $\theta \in [0, \pi]$  with a step size of  $\Delta\theta = \frac{\pi}{5000}$ . The general structure and the variation of the training data set are illustrated in Figure 23. An additional visualization can be found in Appendix 47.

#### RESULTS: DENSITY MATRIX

In this section, we will analyze  $\beta$ -VAE experiments on the density matrix data set in terms of the learned latent representation. In this initial investigation, we choose a 1-dimensional latent space as a starting point, as we know that in theory the data set has the angle  $\theta$  as its singular generative factor.

$$\rho(\theta) = \frac{1}{2} \begin{pmatrix} 1 & 0 & \cos\frac{1}{2}\theta & \sin\frac{1}{2}\theta \\ 0 & 0 & 0 & 0 \\ \cos\frac{1}{2}\theta & 0 & \cos^2\frac{1}{2}\theta & \sin\frac{1}{2}\theta\cos\frac{1}{2}\theta \\ \sin\frac{1}{2}\theta & 0 & \sin\frac{1}{2}\theta\cos\frac{1}{2}\theta & \sin^2\frac{1}{2}\theta \end{pmatrix}$$

(a) Density matrix as function of  $\theta$ 

<p>(b) <math>\theta = 0\pi</math></p> $\begin{pmatrix} 0.50 & 0.00 & 0.50 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.50 & 0.00 & 0.50 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \end{pmatrix}$	<p>(c) <math>\theta = 0.2\pi</math></p> $\begin{pmatrix} 0.50 & 0.00 & 0.48 & 0.15 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.48 & 0.00 & 0.45 & 0.15 \\ 0.15 & 0.00 & 0.15 & 0.05 \end{pmatrix}$
<p>(d) <math>\theta = 0.7\pi</math></p> $\begin{pmatrix} 0.50 & 0.00 & 0.23 & 0.45 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.23 & 0.00 & 0.10 & 0.20 \\ 0.45 & 0.00 & 0.20 & 0.40 \end{pmatrix}$	<p>(e) <math>\theta = \pi</math></p> $\begin{pmatrix} 0.50 & 0.00 & 0.00 & 0.50 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.50 & 0.00 & 0.00 & 0.50 \end{pmatrix}$

Figure 23: Illustration of (a) structure of data set and (b-e) density matrix samples generated at varying angles  $\theta$ 

For the following experiment, we found that training on 400 epochs using the Adam optimizer with a learning rate of  $10^{-3}$ , batch size of 64, and  $\beta = 0.25$  produces convergent loss values.

Figure 24 illustrates the correlation between the learned latent parameter values and the angle  $\theta$  of the corresponding input density matrices. We observe that the model has learned to assign unique latent parameter values to density matrices generated at each angle. This means the learned map from input to latent representation must be based on a quantity that discriminates between samples as a function of the generative factor  $\theta$ . The hue corresponding to the entanglement entropy of each data point follows the same behavior, as it is a function of the angle  $\theta$ .

The crux of this analysis is now in the interpretation of the extracted representation as presented by the single latent variable  $Z_0$ . Ideally, we want to understand how the model identifies the map from a given input density matrix to the unique latent value for each angle. Furthermore, we want to investigate if we have learned to extract a particular property of the input data to distinguish the samples according to the angle at which they were generated at. To this end, we correlate the learned latent parameter values with different elements of the density matrices.

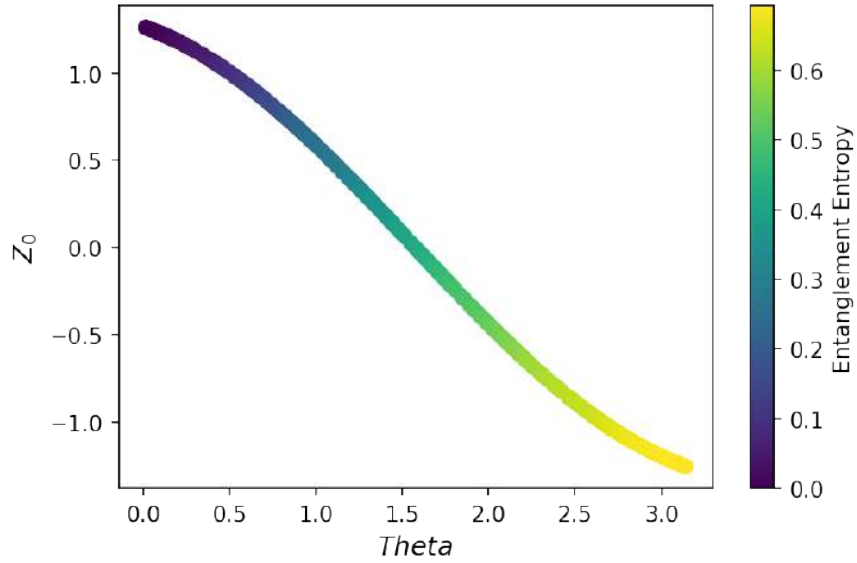


Figure 24: Correlation between latent parameter value  $z = Z_0$  and generative angle  $\theta$  for density matrix samples. The hue indicates the entanglement entropy of the density matrix corresponding to the respective learned latent parameter.

Figure 25 shows the correlation between the learned latent parameter values and the  $\rho_{3,3} = \cos^2 \frac{1}{2} \theta$  entry of the input density matrices. We observe an approximately linear relationship between the two, except for slight deviations in the regions of  $\theta \approx 0$  and  $\theta \approx \pi$ . A general statement we can derive from this is that the learned latent representation is based on a quantity that is closely related, up to a constant, to this entry of the density matrix. Considering the general learning goal as introduced in Section 3.3.3, we interpret this in the following way: The model attempts to learn an efficient encoding-decoding method during training by identifying a mapping from a given sample to a single latent variable. For this data set, each density matrix has a unique structure as a function of the angle  $\theta$ . With this intuition, learning to extract information related to a function of  $\theta$  of each sample is the optimal strategy for structuring the latent representation, as it allows us to distinguish between density matrices generated at different angles. Based on the structure shown in Figure 23, we know that it is theoretically sufficient to consider the individual elements

$$\rho_{3,1} = \rho_{1,3} = \cos \frac{1}{2} \theta \quad \rho_{3,3} = \cos^2 \frac{1}{2} \theta \quad \rho_{4,1} = \rho_{1,4} = \sin \frac{1}{2} \theta \quad \rho_{4,4} = \sin^2 \frac{1}{2} \theta \quad (27)$$

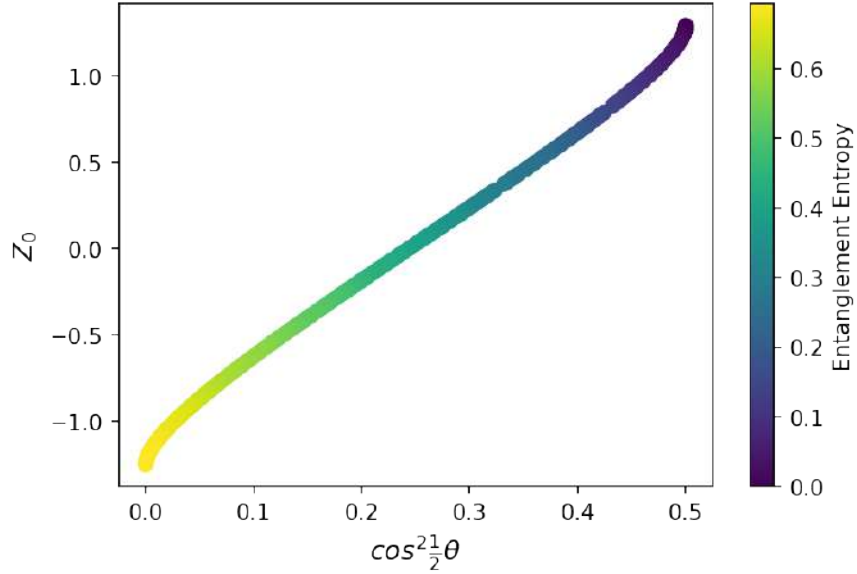


Figure 25: Correlation between latent parameter value  $z = Z_0$  and  $\rho_{3,3} = \cos^2 \frac{1}{2} \theta$  entry of density matrix samples. The hue indicates the entanglement entropy of the density matrix corresponding to the respective learned latent parameter.

of the density matrix to identify this optimal map. This is because each of the expressions in eq. 27 provides a unique output for a particular generative angle in  $\theta \in [0, \pi]$ . Therefore, each element represents a unique mapping between the input sample and the generative factor  $\theta$ . The linear relation in Figure 25 illustrates that the learned representation is closely related to the  $\rho_{3,3} = \cos^2 \frac{1}{2} \theta$  element. The question of *why exactly* this specific element is modeled over others is difficult to answer as it depends on several factors. These include the initialization of the network weights, the inner workings of the gradient descent algorithm, and the nonlinear nature of the network, all of which make it difficult to derive general statements about this choice. The finding that the VAE learns to extract a surface artifact by focusing on a single element of the density matrix is not what we originally wanted to achieve, as we want the model to extract some more complex quantities from the data set. This problem of neural networks and their tendency to learn superficial statistical regularities instead of higher level abstractions is a well-known problem, as discussed in Jo et al. (2017) [64]. In order to circumvent this, we attempt to remove the unique map from  $\theta$  to  $\cos^2 \frac{1}{2} \theta$  in order to encourage the model to extract a more useful quantity from the density matrices in the next section.



## 6.4 COHERENT NOISE

## DATA SET: COHERENT NOISE

In the second experiment, our goal is to introduce coherent noise by adding unitary operations to the density matrix samples that maintain the purity of the output quantum state. For this, we introduce a data set of 5000 density matrices  $\rho(\theta)$  extracted from the circuit shown in Figure 20. The generative angles are chosen in the interval  $\theta \in [0, \pi]$  with a step size of  $\Delta\theta = \frac{\pi}{200}$ . At each angle, we draw  $n = 25$  samples and introduce a small error  $\theta + \kappa$  with  $\kappa \in [-\frac{1}{2}, \frac{1}{2}]$  as a uniformly distributed random number. With this systematic error, we remove the unique map from generative angle  $\theta$  to density matrix. The general structure and the variation of the training data set are illustrated in Figure 26 and an additional visualization is presented in Appendix 48.

$$\rho(\theta) = \frac{1}{2} \begin{pmatrix} 1 & 0 & \cos\frac{1}{2}(\theta + \kappa) & \sin\frac{1}{2}(\theta + \kappa) \\ 0 & 0 & 0 & 0 \\ \cos\frac{1}{2}(\theta + \kappa) & 0 & \cos^2\frac{1}{2}(\theta + \kappa) & \sin\frac{1}{2}(\theta + \kappa)\cos\frac{1}{2}(\theta + \kappa) \\ \sin\frac{1}{2}(\theta + \kappa) & 0 & \sin\frac{1}{2}(\theta + \kappa)\cos\frac{1}{2}(\theta + \kappa) & \sin^2\frac{1}{2}(\theta + \kappa) \end{pmatrix}$$

(a) Density matrix as a function of  $\theta$ 

<p>(b) <math>\theta = 0\pi</math></p> $\begin{pmatrix} 0.50 & 0.00 & 0.50 & 0.05 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.50 & 0.00 & 0.49 & 0.05 \\ 0.05 & 0.00 & 0.05 & 0.01 \end{pmatrix}$	<p>(c) <math>\theta = 0.2\pi</math></p> $\begin{pmatrix} 0.50 & 0.00 & 0.45 & 0.22 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.45 & 0.00 & 0.40 & 0.20 \\ 0.22 & 0.00 & 0.20 & 0.10 \end{pmatrix}$
<p>(d) <math>\theta = 0.7\pi</math></p> $\begin{pmatrix} 0.50 & 0.00 & 0.26 & 0.42 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.26 & 0.00 & 0.14 & 0.22 \\ 0.42 & 0.00 & 0.22 & 0.36 \end{pmatrix}$	<p>(e) <math>\theta = \pi</math></p> $\begin{pmatrix} 0.50 & 0.00 & 0.05 & 0.50 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.05 & 0.00 & 0.01 & 0.05 \\ 0.50 & 0.00 & 0.05 & 0.49 \end{pmatrix}$

Figure 26: Illustration of (a) structure of data set and (b-e) density matrix samples generated at varying angles  $\theta$

## RESULTS: COHERENT NOISE

In this section, we will analyze  $\beta$ -VAE experiments on the density matrix data set modified with coherent noise in terms of the learned 1-dimensional latent representation. For the following experiment, we found that training on 400 epochs using the Adam optimizer with a learning rate of  $10^{-3}$ , batch size of 64, and  $\beta = 0.25$  produces convergent loss values.

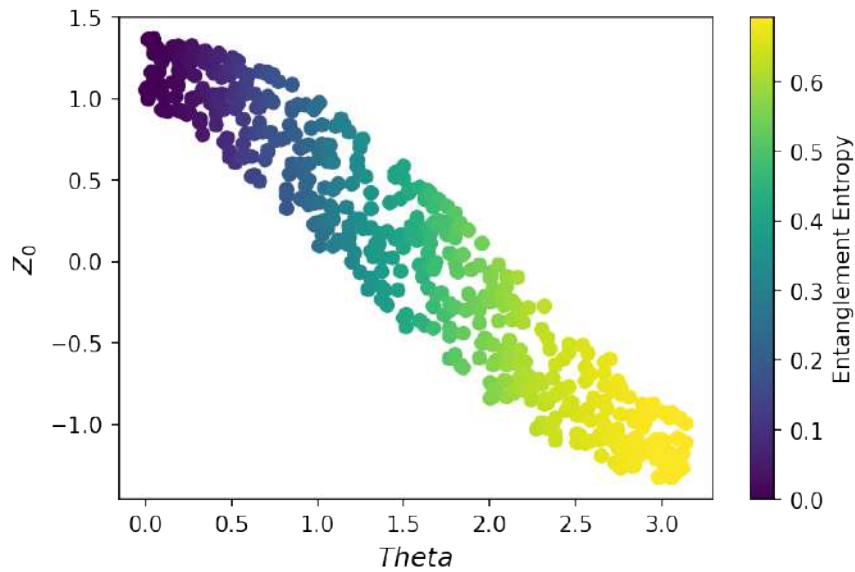


Figure 27: Correlation between latent parameter value  $z = Z_0$  and generative angle  $\theta$  for density matrix samples. The hue indicates the entanglement entropy of the density matrix corresponding to the respective learned latent parameter.

Figure 27 illustrates the correlation between the learned latent parameter values trained with coherent noise and the angle  $\theta$  of the corresponding input density matrices. In this experiment, it can be observed that the model has learned to represent samples from each angle in a latent parameter value range instead of a single point. This value interval captures the uncertainty introduced by the noise. From this, we can infer that the unique map of input to angle has been obstructed by the introduction of coherent noise, since two identical latent parameters can now represent states generated with different angles. For example, the learned encoding for states generated at  $\theta = 0$  and  $\theta = 1$  both includes  $Z_0 = 1$ , which means that our model is unable to distinguish between the two states based

on the encoded value. In the following, we again interpret the latent representation to understand how the VAE learned to encode the input and what quantity it extracts from the states.

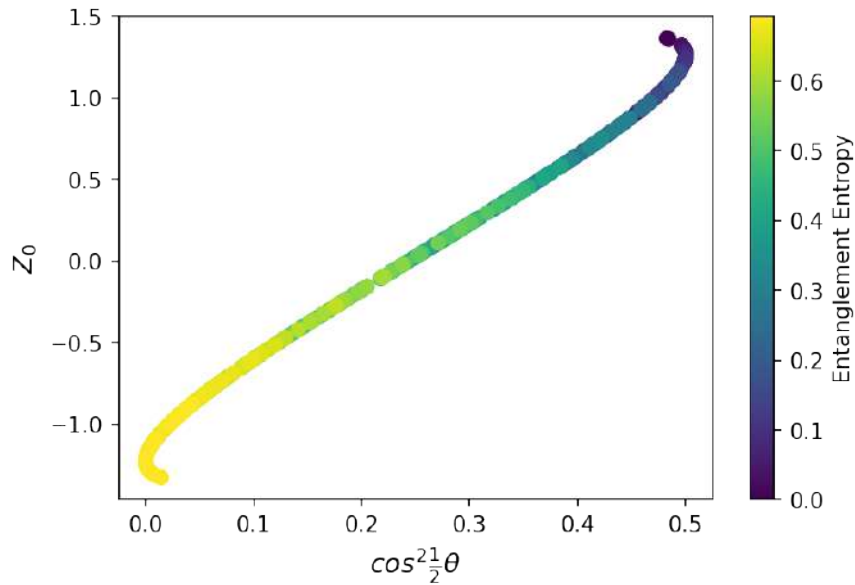


Figure 28: Correlation between latent parameter value  $z = Z_0$  and  $\cos^2\frac{1}{2}\theta$  entry of density matrix samples.

The hue indicates the entanglement entropy of the density matrix corresponding to the respective learned latent parameter.

Figure 28 shows the correlation between the learned latent parameter values and the  $\rho_{3,3} = \cos^2\frac{1}{2}\theta$  entry of the density matrices. Surprisingly, we again observe a mostly linear relationship between the two. This suggests that the model persists in learning a representation based on simply extracting a single entry from the density matrices. From this observation, we infer that the model has learned to encode the coherent noise as a constant offset, which did not affect its ability to discover the same mapping between input and generative factor  $\theta$ . Therefore, in the next experiment, we introduce a different type of noise with the same goal of removing the ability to discover this simple map and instead encourage it to discover a more abstract representation.

## 6.5 INCOHERENT NOISE

## DATA SET: INCOHERENT NOISE

In the third experiment, our goal is to introduce incoherent noise by adding unitary operations to our density matrix samples that do not necessarily maintain the purity of the output quantum state. For this, we introduce a data set consisting of 5000 density matrices  $\rho(\theta)$  that were extracted from the circuit shown in Figure 20. The generative angles are chosen in the interval  $\theta \in [0, \pi]$  with a step size of  $\Delta\theta = \frac{\pi}{200}$ . At each angle, we draw  $n = 25$  samples subject to incoherent noise. In practice, this type of noise originates from the quantum circuit becoming entangled with the environment, thus leading to outputs that are to a certain degree random, regardless of what basis we measure in. We model incoherent noise as a unitary transformation using Kraus operators  $K_i$  with some associated probability satisfying the condition  $\sum_i K_i^\dagger K_i = \mathbb{1}$  with

$$D(\rho) = \sum_i \frac{1}{p_i} K_i^\dagger \rho K_i \quad (28)$$

In this, the probability is give by  $p_i = \text{Tr}(K_i^\dagger \rho K_i)$ . [63] In the following experiment, we will introduce noise that generalize the bit flip and phase flip errors. It describes a transformation that applies Pauli operators with probability  $p = 0.3$  and applies the identity with probability  $1 - p$ . The Kraus operators of this transformation are given by

- $K_0 = \mathbb{1} \otimes \sqrt{1-p} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
- $K_1 = \mathbb{1} \otimes \sqrt{p/3} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
- $K_2 = \mathbb{1} \otimes \sqrt{p/3} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
- $K_3 = \mathbb{1} \otimes \sqrt{p/3} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$

The general structure of the training data set is illustrated in Figure 29 and Appendix 49.

$$\rho(\theta) = \sum_i \frac{1}{2^{p_i}} K_i^+ \begin{pmatrix} 1 & 0 & \cos\frac{1}{2}\theta & \sin\frac{1}{2}\theta \\ 0 & 0 & 0 & 0 \\ \cos\frac{1}{2}\theta & 0 & \cos^2\frac{1}{2}\theta & \sin\frac{1}{2}\theta\cos\frac{1}{2}\theta \\ \sin\frac{1}{2}\theta & 0 & \sin\frac{1}{2}\theta\cos\frac{1}{2}\theta & \sin^2\frac{1}{2}\theta \end{pmatrix} K_i$$

(a) Density matrix as function of  $\theta$ 

<p>(b) <math>\theta = 0\pi</math></p> $\begin{pmatrix} 0.43 & 0.00 & 0.32 & 0.00 \\ 0.00 & 0.07 & 0.00 & 0.05 \\ 0.32 & 0.00 & 0.43 & 0.00 \\ 0.05 & 0.05 & 0.00 & 0.07 \end{pmatrix}$	<p>(c) <math>\theta = 0.2\pi</math></p> $\begin{pmatrix} 0.43 & 0.01 & 0.30 & 0.08 \\ 0.01 & 0.07 & 0.00 & 0.05 \\ 0.30 & 0.00 & 0.40 & 0.09 \\ 0.08 & 0.05 & 0.09 & 0.10 \end{pmatrix}$
<p>(d) <math>\theta = 0.7\pi</math></p> $\begin{pmatrix} 0.39 & 0.02 & 0.14 & 0.24 \\ 0.02 & 0.11 & 0.00 & 0.02 \\ 0.14 & 0.00 & 0.18 & 0.13 \\ 0.24 & 0.02 & 0.13 & 0.32 \end{pmatrix}$	<p>(e) <math>\theta = \pi</math></p> $\begin{pmatrix} 0.38 & 0.00 & 0.00 & 0.27 \\ 0.00 & 0.12 & 0.00 & 0.00 \\ 0.05 & 0.00 & 0.12 & 0.05 \\ 0.27 & 0.00 & 0.00 & 0.38 \end{pmatrix}$

Figure 29: Illustration of (a) structure of data set and (b-e) density matrix samples generated at varying angles  $\theta$ 

## RESULTS: INCOHERENT NOISE

In this section, we will analyze  $\beta$ -VAE experiments on the density matrix data set modified with incoherent noise in terms of the learned 1-dimensional latent representation. For the following experiment, we found that training on 400 epochs using the Adam optimizer with a learning rate of  $10^{-3}$ , batch size of 64, and  $\beta = 0.25$  produces convergent loss values.

Figure 30 illustrates the correlation between the learned latent parameter values trained with incoherent noise and the angle  $\theta$  of the corresponding input density matrices. The relationship shown is more complex compared to Figure 24 and 27; for angles towards  $\theta \rightarrow 0$  and  $\theta \rightarrow \pi$ , the learned ranges of latent parameter values become progressively wider, while the learned range of latent parameters for angles in the direction  $\theta \rightarrow \frac{\pi}{2}$  become increasingly clustered. In the following, we will again examine how the model learned this representation. Figure 31 shows the correlation between the learned latent parameter values and the  $\rho_{3,3} = \cos^2\frac{1}{2}\theta$  entry of the density matrices. We again observe a near linear relationship between the two, suggesting that the VAE is still persistent in learning a

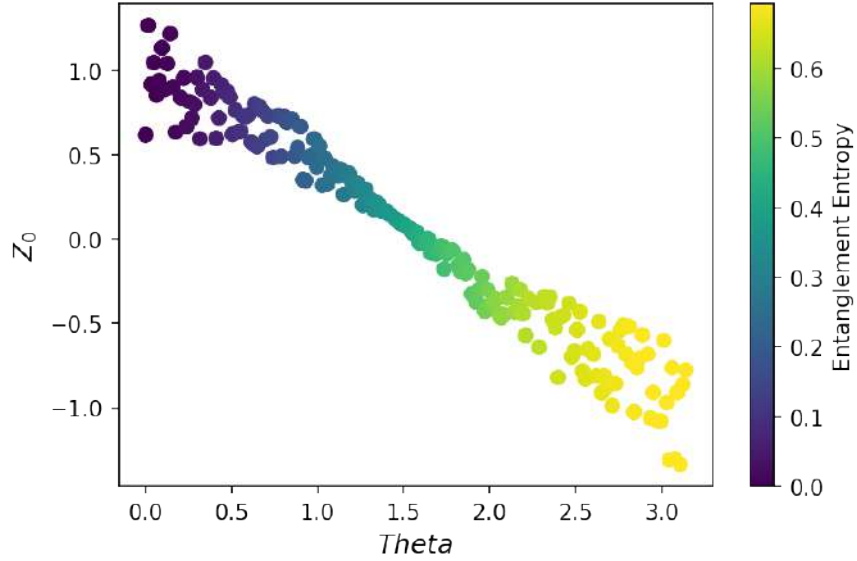


Figure 30: Correlation between latent parameter value  $z = Z_0$  and generative angle  $\theta$  for density matrix samples. The hue indicates the entanglement entropy of the density matrix corresponding to the respective learned latent parameter.

representation based on extracting a single entry from the density matrices. Having established the dependence of the latent variables on the  $\rho_{3,3} = \cos^2 \frac{1}{2} \theta$  entry of the density matrices, we can explain the varying distributions of the recorded latent parameter values in Figure 30.

In Figure 32, we illustrate the variance of the  $\rho_{3,3} = \cos^2 \frac{1}{2} \theta$  entry of  $N = 200$  samples with incoherent noise at varying angles  $\theta$ . We note that for angles towards  $\theta \rightarrow 0$  and  $\theta \rightarrow \pi$  the variance increases and towards  $\theta \rightarrow \frac{\pi}{2}$  it decreases. This is consistent with the behavior observed in Figure 30, which explains the observed variation in learned latent parameter values. On closer inspection, we can attribute this behavior to the Pauli-X error introduced with  $K_1$ . In general, this error changes the position of the elements of the density matrix and applies a constant multiplier. We know that in the density matrix without noise, with angles towards  $\theta \rightarrow \frac{\pi}{2}$  the entries of the density matrix become more and more similar, since  $\cos \frac{1}{4} \pi = \sin \frac{1}{4} \pi$ . Conversely, the elements of the density matrix become increasingly dissimilar towards angles  $\theta \rightarrow 0, \pi$ . The idea now is that if the matrix elements are similar from the beginning, their rearrangement will change the overall structure between samples less, as shown by the minimum variance in Figure 32. In summary, the model is persistent to use the

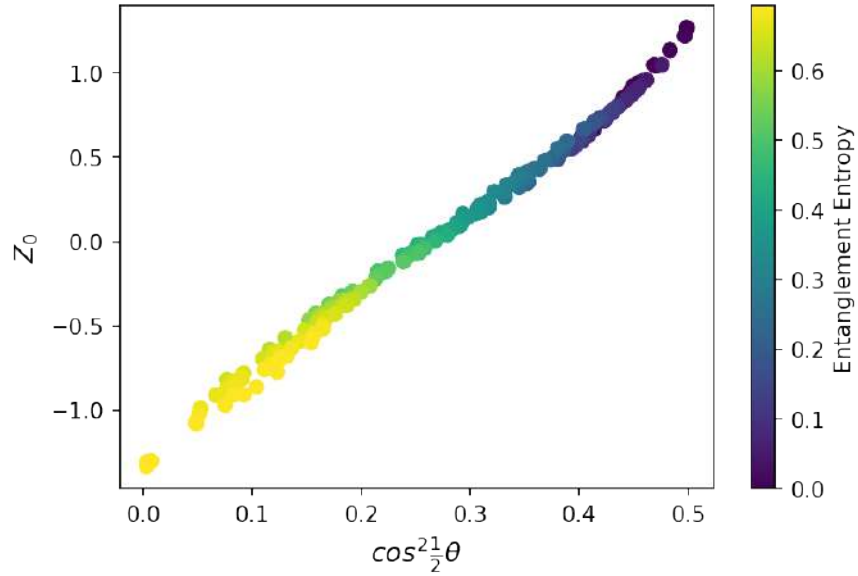


Figure 31: Correlation between latent parameter value  $z = Z_0$  and  $\rho_{3,3} = \cos^2 \frac{1}{2} \theta$  for density matrix samples.

The hue indicates the entanglement entropy of the density matrix corresponding to the respective learned latent parameter.

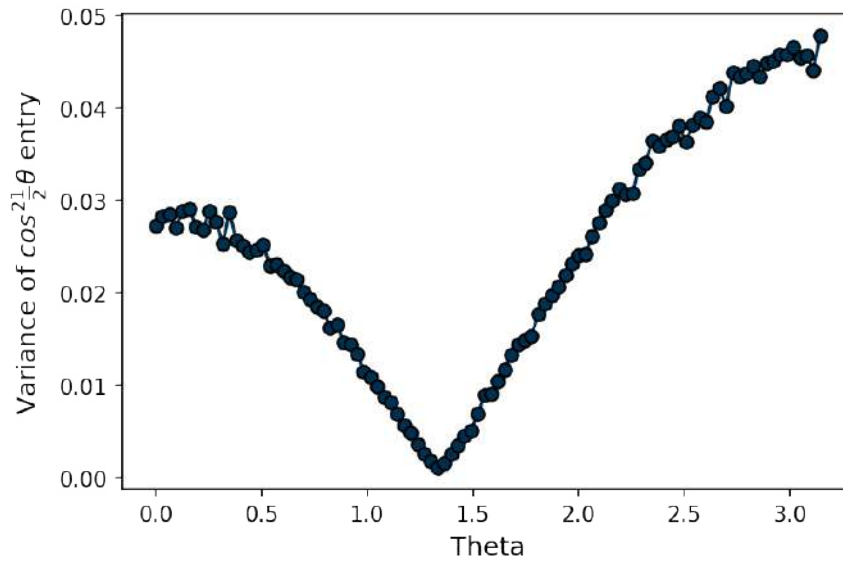


Figure 32: Visualization of the variance (200 samples) of the  $\rho_{3,3} = \cos^2 \frac{1}{2} \theta$  entry at different angles.

$\rho_{3,3} = \cos^2 \frac{1}{2} \theta$  entry for its learned representation. Again, learning a superficial statistical regularity structure is not in our interest, as we want to extract a more abstract quantity. In the next experiment, we will introduce a more stringent error to prohibit the model from learning this simple map.

## 6.6 RANDOM UNITARY

## DATA SET: RANDOM UNITARY

The fourth data set consists of  $10^4$  density matrices  $\rho(\theta)$  that were extracted from the circuit shown in Figure 20. The generative angles are chosen in  $\theta \in [0, \pi]$  with a step size of  $\Delta\theta = \frac{\pi}{100}$ . Furthermore, we draw  $n = 100$  samples at each angle subject to the following randomization scheme. We apply random local 4x4 unitary operators to the output density matrices of the circuit  $\tilde{\rho}(\theta)$ . [65]

$$\rho(\theta) = (U_a \otimes U_b) \tilde{\rho}(\theta) (U_a \otimes U_b) \quad (29)$$

The motivation is to introduce a randomization procedure with local unitaries that encourages the neural network to extract features related to correlation between the subpartitions of the density matrices that are insensitive to these local transformations. [66] The derivation of the generation of the 1-local unitary single-qubit random operators  $U_a$  and  $U_b$  is presented in Section 9.4. To gain intuition on the randomization, we consider the example of computing an arbitrary observable below.

$$\langle O \rangle = \text{Tr}(\rho O) \quad (30)$$

Applying the random unitary method changes this to the following expression:

$$\text{Tr}(U\rho U^\dagger O) = \text{Tr}(\rho U^\dagger O U) = \langle O' \rangle \quad (31)$$

With  $\langle O' \rangle \neq \langle O \rangle$ , this shows that observables are generally not preserved under this transformation. To understand what information is actually preserved, consider an arbitrary state of the 2-qubit circuit.

$$|\psi\rangle = |\psi_a\rangle \otimes |\psi_b\rangle \quad (32)$$

The application of the randomization procedure to the statevector can be represented as follows.

$$U_a \otimes U_b |\psi\rangle = U_a |\psi_a\rangle \otimes U_b |\psi_b\rangle \quad (33)$$



which results in the following density matrix.

$$\rho = (U_a |\psi_a\rangle \otimes U_b |\psi_b\rangle)(\langle\psi_a| U_a^\dagger \otimes \langle\psi_b| U_b^\dagger) \quad (34)$$

In order to compute the entanglement entropy of the state as introduced in Section 6.1.1, we take the partial trace with respect to subsystem  $b$ .

$$\text{Tr}_b \rho = U_a |\psi_a\rangle \langle\psi_a| U_a^\dagger = \rho_a \quad (35)$$

The eigenvalues of  $\rho_a$  are identical to the eigenvalues of the unmodified reduced density matrix  $|\psi_a\rangle \langle\psi_a|$ . This shows that the transformation preserves the entanglement entropy of the state. [67]

The general structure and variation of the training data set are illustrated in Figure 33 and Appendix 50.

$$\rho(\theta) = (U_a \otimes U_b) \begin{pmatrix} 1 & 0 & \cos\frac{1}{2}\theta & \sin\frac{1}{2}\theta \\ 0 & 0 & 0 & 0 \\ \cos\frac{1}{2}\theta & 0 & \cos^2\frac{1}{2}\theta & \sin\frac{1}{2}\theta\cos\frac{1}{2}\theta \\ \sin\frac{1}{2}\theta & 0 & \sin\frac{1}{2}\theta\cos\frac{1}{2}\theta & \sin^2\frac{1}{2}\theta \end{pmatrix} (U_a \otimes U_b)$$

(a) Density matrix as function of  $\theta$

<p>(b) <math>\theta = 0\pi</math></p> $\begin{pmatrix} 0.11 & 0.01 & -0.31 & -0.02 \\ 0.01 & 0.00 & -0.02 & 0.00 \\ -0.31 & -0.02 & 0.88 & 0.07 \\ -0.02 & 0.00 & 0.07 & 0.01 \end{pmatrix}$	<p>(c) <math>\theta = 0.2\pi</math></p> $\begin{pmatrix} 0.04 & 0.15 & 0.03 & 0.12 \\ 0.15 & 0.55 & 0.12 & 0.46 \\ 0.03 & 0.12 & 0.03 & 0.10 \\ 0.12 & 0.46 & 0.10 & 0.38 \end{pmatrix}$
<p>(d) <math>\theta = 0.7\pi</math></p> $\begin{pmatrix} 0.41 & -0.07 & -0.48 & 0.09 \\ -0.07 & 0.01 & 0.09 & -0.02 \\ -0.48 & 0.09 & 0.56 & -0.10 \\ 0.09 & -0.02 & -0.10 & 0.02 \end{pmatrix}$	<p>(e) <math>\theta = \pi</math></p> $\begin{pmatrix} 0.04 & 0.10 & -0.07 & -0.16 \\ 0.10 & 0.22 & -0.16 & -0.37 \\ -0.07 & -0.16 & 0.12 & 0.27 \\ -0.16 & -0.37 & 0.27 & 0.62 \end{pmatrix}$

Figure 33: Illustration of (a) structure of data set and (b-e) density matrix samples generated at varying angles  $\theta$

## RESULTS: RANDOM UNITARY I

In this section, we will analyze  $\beta$ -VAE experiments on the density matrix data set modified with random unitaries in terms of the learned latent representation. With the newly added randomization,

we first investigate a 1-dimensional latent space. For the following experiment, we trained the model for 200 epochs using the Adam optimizer with a learning rate of  $10^{-3}$ , batch size 64, and  $\beta = 0.75$ .

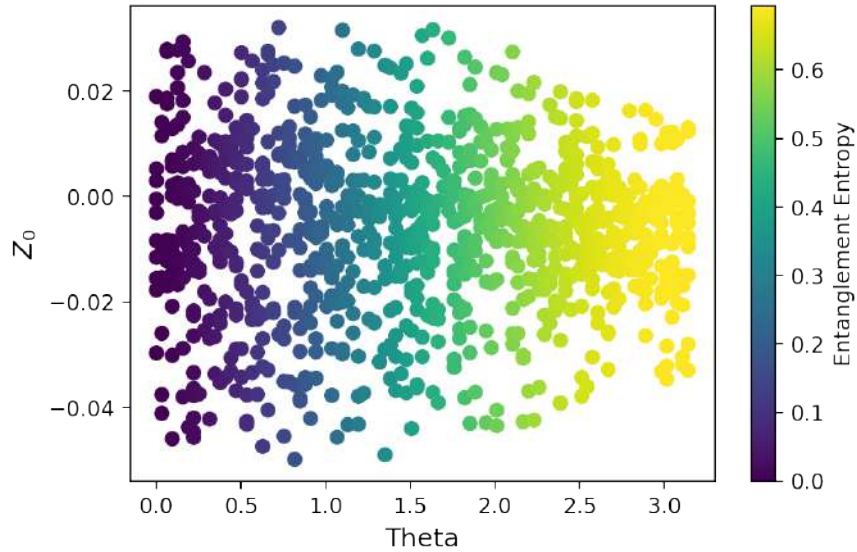


Figure 34: Correlation between latent parameter value  $z = Z_0$  and generative angle  $\theta$  for density matrix samples. The hue indicates the entanglement entropy of the density matrix corresponding to the respective learned latent parameter.

Figure 34 illustrates the correlation between the learned latent parameter values and the angle  $\theta$  of the corresponding input density matrices trained with the unitary randomization scheme. We note that the model has failed to assign unique latent parameter values depending on the angle the input density matrix was generated at. For instance, with a latent parameter value of  $Z_0 = 0$ , we could not infer whether its corresponding density matrix was generated at  $\theta = 0$  or any other angle. Based on this observation, we conclude that the model has failed to learn a representation that discriminates between samples generated at different angles due to the randomization scheme. In previous experiments in Sections 6.3,6.4,6.5 we learned that the model learned to extract information from a single entry of the density matrix to structure its latent space, so we will investigate whether this is again the case. Figure 35 illustrates the correlation between the learned latent parameter value and the  $\rho_{3,3} = \cos^2 \frac{1}{2} \theta$  entry of the density matrices. We do not observe a linear relationship between the two, suggesting that the model is learning a representation based on a different quantity. This means that the introduction

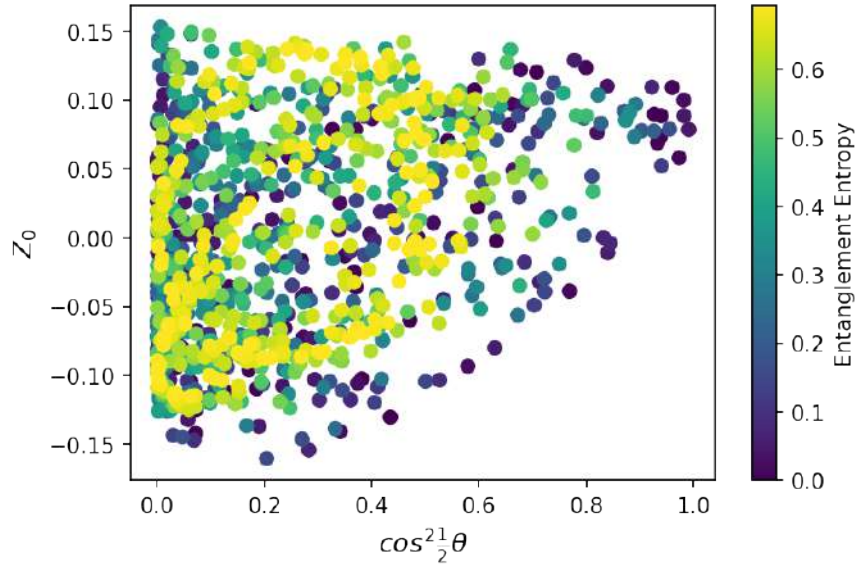


Figure 35: Correlation between latent parameter value  $z = Z_0$  and  $\cos^2\frac{1}{2}\theta$  entry of density matrix samples.

The hue indicates the entanglement entropy of the density matrix corresponding to the respective learned latent parameter.

of the randomization scheme successfully prevented the model from extracting information only from the  $\rho_{3,3} = \cos^2\frac{1}{2}\theta$  entry of the density matrices to structure its latent representation. Looking back at our target scenario, the introduction of randomization was aimed at preventing the model from extracting surface statistical regularities. Based on the observations in Figure 35, we can see that this proposition is met. Moreover, the unitary randomization scheme ought to promote the extraction of quantities that are invariant to the randomization, in particular the entanglement entropy of a state. This proposition clearly is not met, as samples with a given entanglement entropy value are assigned multiple latent parameter values, as can be seen in Figure 34. The lack of structure in the learned representation may indicate that the model is unable to find a suitable map that satisfies both the reconstruction quality and regularization loss. Therefore, in the next experiment, we increase the capacity of the representation by increasing the latent space to 2-dimensions.

## RESULTS: RANDOM UNITARY II

In this section, we will analyze  $\beta$ -VAE experiments on the density matrix data set modified with random unitaries in terms of the learned latent representation. In this experiment, we extend the model to a 2-dimensional latent space. For the following experiment, we trained the model for 200 epochs using the Adam optimizer with a learning rate of  $10^{-3}$ , batch size 64, and  $\beta = 0.75$ .

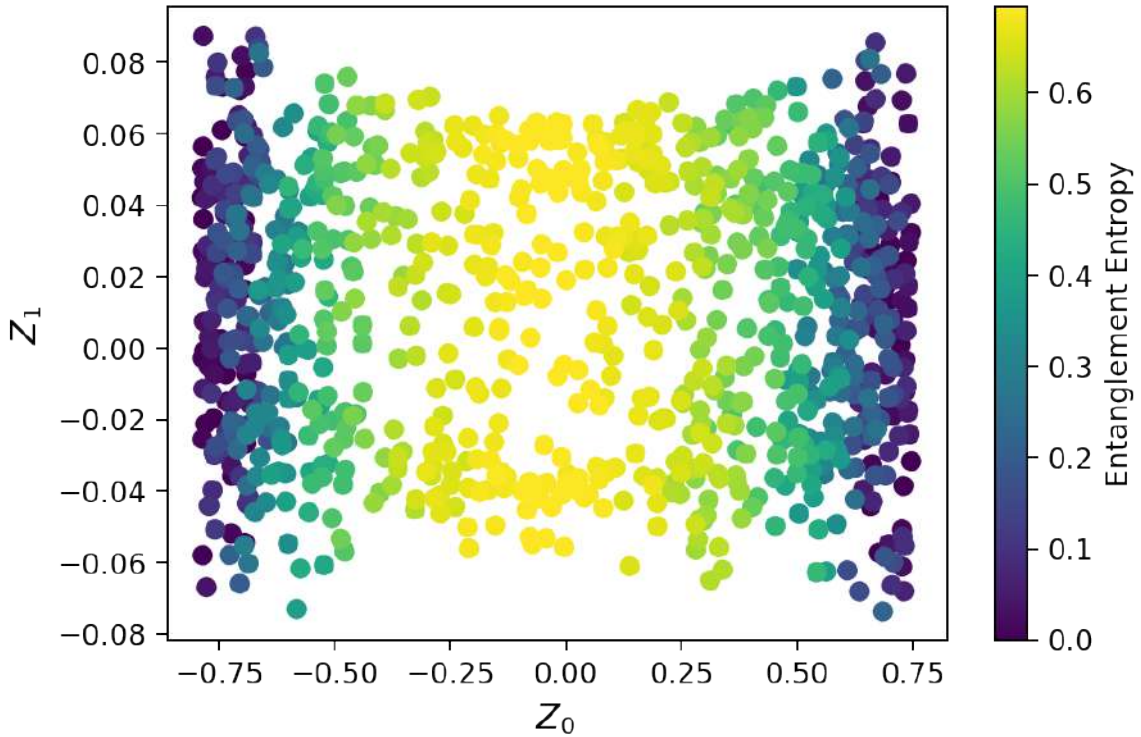


Figure 36: Correlation between latent parameter values  $z = (Z_0, Z_1)$ . The hue indicates the entanglement entropy of the density matrix corresponding to the respective learned latent parameters.

Figure 36 illustrates the correlation between the learned latent parameter values trained with the unitary randomization scheme. We note that the model has learned to assign unique pairs of latent parameter values ranges to the density matrices generated at each angle. In this, we observe that the first latent parameter  $Z_0$  is affected by the generative angle of the input density matrices: States generated near  $\theta \approx 0$  are encoded at  $Z_0 \approx \pm 0.75$ . As  $\theta$  increase, samples are continuously encoded more towards the center  $Z_0 \rightarrow 0$ , with the absolute minimum of  $Z_0 \approx 0$  being reached at  $\theta = \pi$ . The

second latent parameter  $Z_1$  is not directly affected by the generative angle  $\theta$ . It encodes a given density matrix in a range of  $Z_1 \in [-0.08, 0.08]$ . Again, the crux of the analysis of this learned representation is in the interpretation of the extracted latent space as presented by the two latent variables  $(Z_0, Z_1)$ . Ideally, we want to understand how the model identifies the map from a given input density matrix to the unique latent value pair for each angle. Furthermore, we want to investigate if we have learned to extract a particular property of the input data to distinguish the samples according to the angle at which they were generated at. To begin this analysis, we investigate the correlation of both latent variables to the generative angle  $\theta$ .

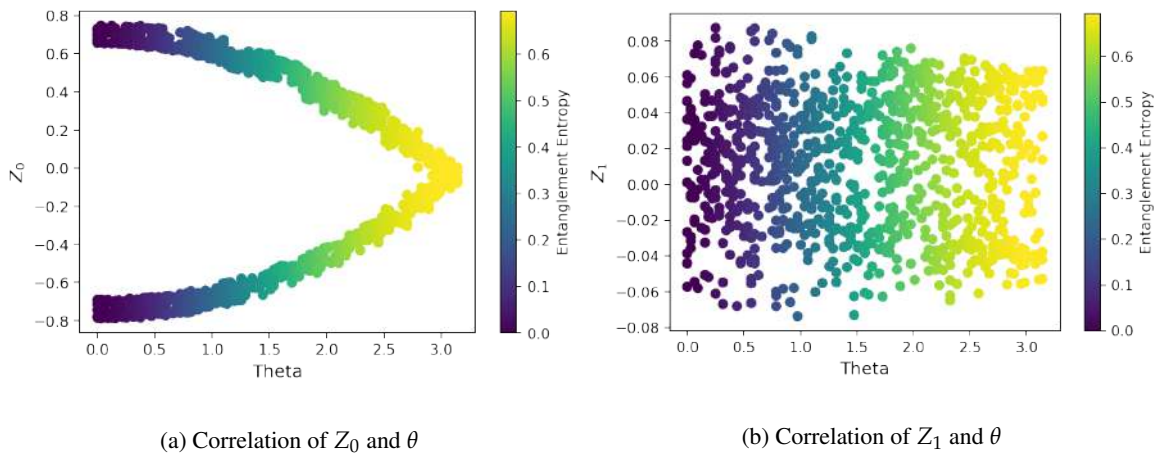


Figure 37: Correlation between  $\theta$  and (a) latent parameter value  $Z_0$  and (b) latent parameter value  $Z_1$ . The hue indicates the entanglement entropy of the density matrix corresponding to the respective learned latent parameter.

Figure 37 illustrates the correlation between the two learned latent parameter values and the angle  $\theta$  of the corresponding input density matrices. In this, Figure 37b shows the correlation between  $Z_1$  and  $\theta$ . We note that the model is not able to assign unique latent parameter values as a function of the angle at which the input density matrix was generated. Since there is no direct correlation between  $Z_1$  and  $\theta$ , we defer further investigation of how this latent variable is used to account for specific noise patterns in the encoding-decoding process to the Appendix 9.6.

In Figure 37a, we observe the correlation of  $Z_0$  and  $\theta$ . We note that the model has learned to assign two latent parameter value ranges to density matrices generated at each angle. The learned parameters

are symmetric with respect to  $Z_0 = 0$ . In the learned representation, samples generated at  $\theta \approx 0$  are encoded between  $Z_0 \approx \pm 0.75$  while samples generated towards  $\theta \approx \pi$  are encoded with increasingly converging values  $Z_0 \approx \pm 0.1$ . Firstly, this means that the model has found a map from input to single latent variable that distinguishes samples generated at different angles, as seen by the unique value ranges. Secondly, the model has learned to distinguish the samples into two categories. In the following, we examine how samples from the two categories  $Z_0 < 0$  and  $Z_0 > 0$  correlate with quantities extracted from the density matrices they encode.

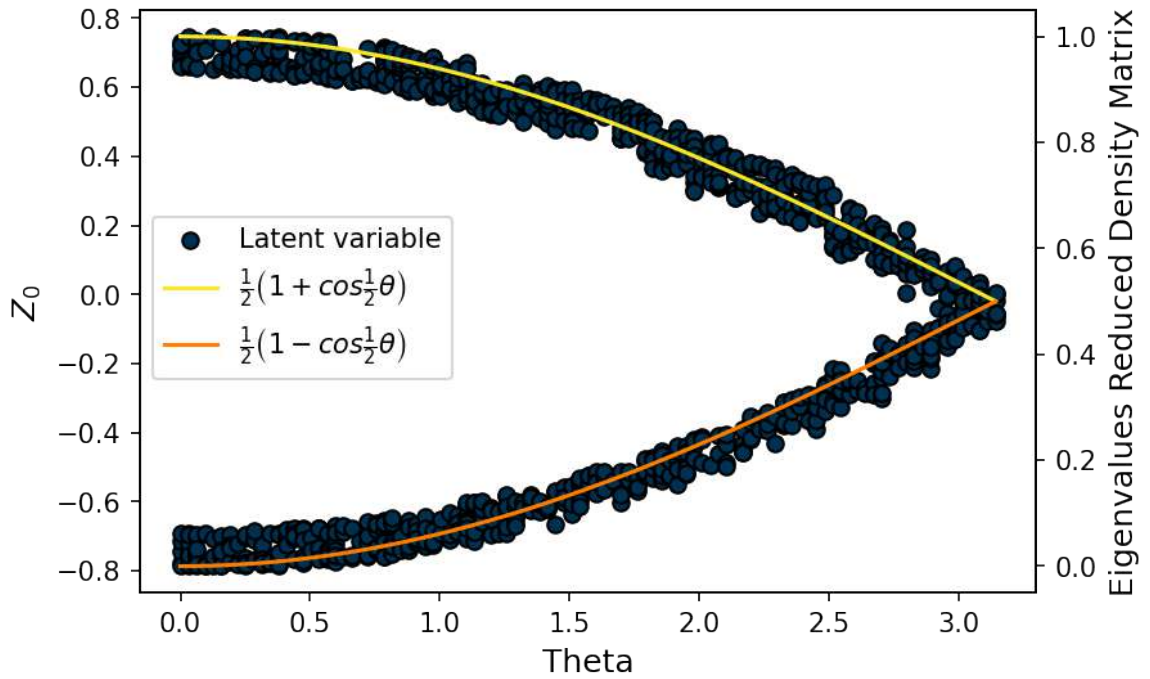


Figure 38: Latent parameter values  $Z_0$  and eigenvalues  $\frac{1}{2} (1 \pm \cos \frac{1}{2} \theta)$  of reduced density matrix  $Tr_B(\rho(\theta)) = \rho_A(\theta)$  at angles in  $\theta \in [0, \pi]$ .

In Figure 38, we display the learned latent parameter values  $Z_0$  and the two eigenvalues of the reduced density matrices that they encode as a function of the generative angle  $\theta$ . We note that the curve of the subset of latent parameters  $Z_0 > 0$  is proportional to the eigenvalues  $x_1 = \frac{1}{2} (1 + \cos \frac{1}{2} \theta)$  as given in eq. 23. Similarly, the curve of the subset of latent parameters  $Z_0 < 0$  is proportional to the eigenvalues  $x_0 = \frac{1}{2} (1 - \cos \frac{1}{2} \theta)$  as introduced in eq. 22. This observation illustrates that the learned representation in  $Z_0$  encodes information correlated to the eigenvalues of the reduced density

matrix. Learning to encode the spectrum of eigenvalues of the reduced density matrices is equivalent to learning to encode information about the entanglement spectrum of states. [63] From this relation, we conclude that the model has learned the equivalent of extracting entanglement information from the density matrices to structure the latent variable  $Z_0$ . We interpret this result as follows.

To reiterate the problem scenario: We took a model with no prior knowledge of quantum states and trained it with noisy samples of density matrices. In this, the data set is defined by a single generative parameter, namely the angle  $\theta$  at which the samples were generated. This parameter determines the value of the entanglement entropy for each sample. As introduced in Section 7, our model attempts to learn an efficient encoding-decoding process during training. The architecture shown in Figure 22 used in this experiment requires the model to encode into and decode from two latent variables. An optimization strategy for this is to learn a mapping from a given sample to the generative parameter using two latent variables. In this experiment, a density matrix generated at a given angle is uniquely defined by its entanglement entropy value. However, the randomization scheme prevents the discovery of simple structure-property relationships. Therefore, the optimal strategy for structuring the latent space is to use one latent variable to account for noise patterns in the data set and use a second latent variable to account for information about the entanglement spectrum of a given sample. With this representation, we can distinguish between samples generated at different angles, which accounts for variance unrelated to noise in the data set, and still be able to satisfy the noise constraints.

A natural question that arises from these observations is: How does the model encode information about two separate eigenvalue curves in a single latent variable? The intuition on this is that computing one eigenvalue of the reduced density matrix uniquely defines the second eigenvalue. This is because  $x_0 = \frac{1}{2}(1 - \cos\frac{1}{2}\theta)$  and  $x_1 = \frac{1}{2}(1 + \cos\frac{1}{2}\theta)$  always sum to  $x_0 + x_1 = 1$ . *How exactly* the model identifies the mapping from the input to  $Z_0 > 0$  that is proportional to  $x_1 = \frac{1}{2}(1 + \cos\frac{1}{2}\theta)$  and  $Z_0 < 0$  that is proportional to  $x_0 = \frac{1}{2}(1 - \cos\frac{1}{2}\theta)$  remains hidden. As it does not directly affect the interpretation of the latent representation we leave this investigation for future studies.

The final topic of discussion is the question of why the model with only a single latent variable was not able to find the same map. The challenge with this data set is that the specific noise used makes it unlikely that a single latent variable will learn to encode information related to the entanglement spectrum as a generative factor, as the reconstruction loss would be poor. This is because a large number of states are associated with a single entanglement entropy value, as introduced by the random unitary randomness scheme. If one increases the number of latent variables, it is possible to base the encoding on a combination of properties of the input density matrices. In this experiment, the latent variable  $Z_1$  contributes to satisfy the noise constraints in the encoding-decoding process, and  $Z_0$  encodes information related to the generative angle  $\theta$ . In this case, this information is correlated with the entanglement spectrum of  $\rho(\theta)$ .

As an addition, although this was not the focus of this experiment, we can use the fact that the model has learned to approximate the entanglement spectrum to estimate the entanglement entropy value of a given state. We achieve this by normalizing  $Z_0$  to the range  $[0, 1]$  and using the formula in eq. 19. In this,  $Z_0$  values previously encoded with  $Z_0 > 0$  stand for  $\lambda_0$  and  $Z_0 < 0$  for  $\lambda_1$ . Therefore, even without prior knowledge of the entanglement spectrum, we are able to use the results to derive an approximation of the entanglement entropy value of states, since the latent parameters are proportional entanglement spectrum.

In summary, we have explored an unsupervised machine learning technique to learn features that best describe 2-qubit quantum states. The neural network-based VAE is applied to sampled density matrices and has no a priori knowledge of their generative process or underlying structure. We find that the latent representation encodes information equivalent to the entanglement spectrum of the quantum states. With this, the machine learning model has learned to differentiate between states by approximating a physical quantity that matches our human intuition on how to accomplish such a task.



---

## CONCLUSION

---

In this thesis, we have studied the  $\beta$ -VAE framework, its application to representation learning in quantum physics, and its learned low-dimensional descriptions of quantum states. We created a data set of 2-qubit quantum states with varying amounts of entanglement parameterized by a generative angle  $\theta$ . Our key findings of the study of this data set are summarized in the following:

First, statistical surface regularities of quantum states, such as matrix elements that are a function of the generative parameter  $\rho_{i,j}(\theta)$ , are the most amenable property for our machine learning model. They provide the VAE with a unique mapping from the input to the ground truth generative factor that is not hindered by the introduction of "simple" types of noise into the data.

Second, learning to extract higher-level abstractions, such as approximating information about the entanglement spectrum, is possible given sufficient expressiveness of the model, but requires a randomization scheme. In particular, the randomization must disguise observable measurements as a function of the generative parameter.

Third, the use of density matrices as input offers interpretive potential. With this data formulation, we can relate the encoding-decoding process of the model with physical quantities intuitively.

Fourth, latent representations for quantum state data exists and can be interpreted by humans. The final learned representation matches the entanglement spectrum of the states, which corresponds to our intuition of how to distinguish states with different degrees of entanglement.

Fifth, physical information can be extracted from the learned representation. Although no domain-specific knowledge was provided during optimization, we are able to use the final learned representation to infer the entanglement entropy value of states with a given encoding value.

With this, our results constitute a step towards interpretable machine learning of quantum states.

---

## FUTURE WORK

---

### 8.1 GREENBERGER–HORNE–ZEILINGER STATES

In Section 6.6, we presented representation learning of 2-qubit states generated by a simple quantum circuit. A logical extension is to increase the complexity of the model by adding one qubit and introducing a second parameterized gate, as shown in Fig. 39. With this circuit, we generate

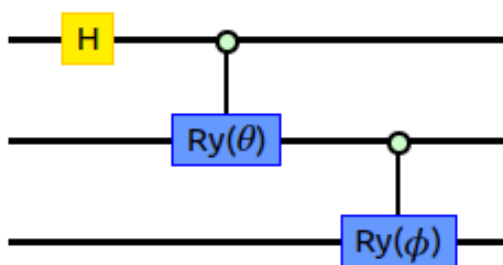


Figure 39: Parameterized quantum circuit consisting of a Hadamard and two controlled  $R_y$  gates

3-qubit-8x8-output density matrices as a function of the two angles  $\theta$  and  $\phi$ . At angles  $\theta = \phi = 0$  we obtain a fully separable output state,  $\theta = \phi = \pi$  we obtain a maximally entangled Greenberger–Horne–Zeilinger state. Both angles influence the entanglement entropy of the final output state, as shown in Figure 40. With this model we can generate a data set with two generative factors and repeat the analysis process from Section 6.6. The overall question in this experiment would be

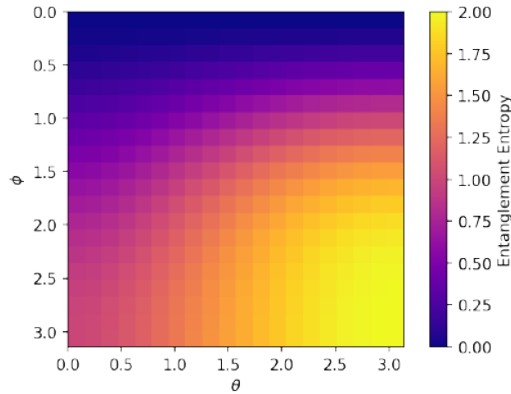


Figure 40: Entanglement entropy of circuit output states for  $\theta, \phi \in [0, \pi]$

if a variational autoencoder is able to find a representation that encompasses the variation in both generative factors by extracting some quantum property of the density matrices.

## 8.2 REPRESENTATION LEARNING OF NON-PARAMETERIZED QUANTUM CIRCUITS

While working with the proposed parameterized quantum circuit in Figure 20, a logical follow-up question arose: how would a generative model for non-parameterized quantum circuits work and how could we use a learned representation? In general, the connection between the structure of circuits and their entanglement properties is difficult to understand intuitively. For this reason, it is beneficial for researchers to develop computational methods that aid in the design of new quantum circuits while providing a conceptual understanding of the so-called structure-property relationship.

The idea we have developed is to consider a large number of quantum circuits from individual quantum logic gates in table 2. One can then generate a data set of random quantum circuits with a sequence of devices, as shown in Figure 41.

One can encode the setup of a quantum circuit sequentially as a series of one-hot column vectors  $x_t$  in a matrix where  $x = [x_1, \dots, x_t] \in \mathbb{R}^{t \times g}$ . Here,  $t$  is maximum circuit length (number of gates) and  $g$  is the number of gates in the toolbox. In any experiment, every possible device on any path or path combination is represented as a one-hot vector  $x_t \in \{0, 1\}^g$ . For example, the experiment in Figure

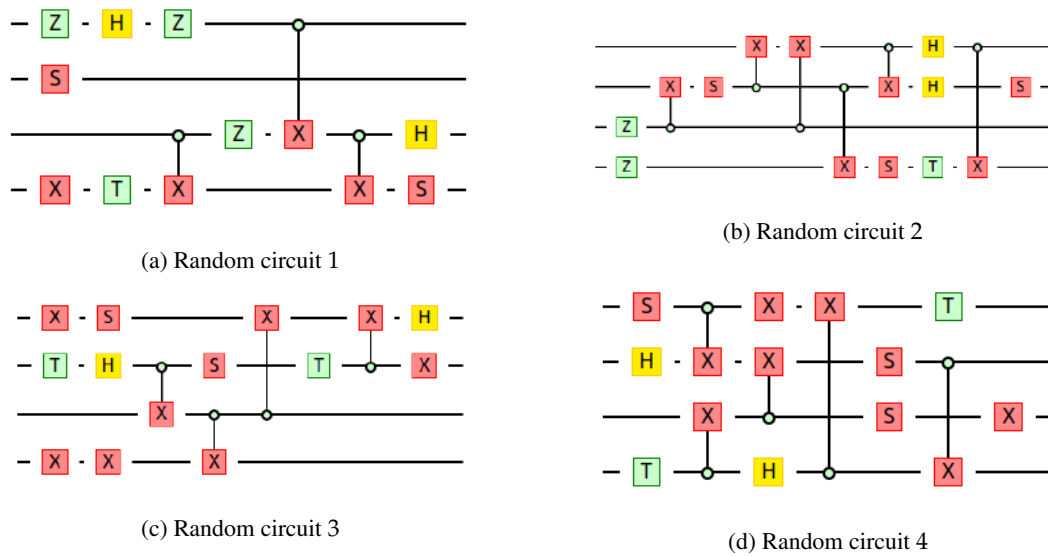


Figure 41: Illustration of random quantum circuits

41a would have twelve one hot vectors for each device. With this structure in mind, the following research question may be investigated:

- Can a generative model trained on this data set create novel quantum circuit setups?
- Is it possible to learn the distribution of entangled output states of the training input?
- Can we use a generative model to search for circuits with specific properties such as highly entangled output states?
- Would the learned latent representation of quantum circuits be human interpretable?


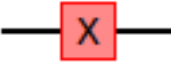
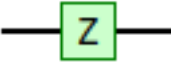
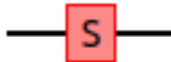
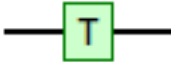

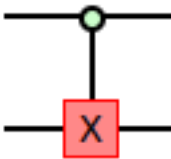
Gate	Visual	Operation
Identity		$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Pauli-X		$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Z		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Phase		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Hadamard		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
CNOT		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

Table 2: Quantum logic gates used in the non-parameterized quantum circuits

---

## APPENDICES

---

### 9.1 MACHINE LEARNING VOCABULARY

This glossary defines general machine learning terms and important concepts used throughout this work; the presented information is inspired from Goodfellow et al. (2015) [34] and Foster (2019) [68].

- **Batch:**

The set of samples used in a single training iteration (a gradient update) of the model.

- **Batch Size:**

The number of samples in a batch that determines the amount of data the model processes simultaneously in a single iteration. Both too large and too small of a batch size generally result in poor performance: If the model is trained on too many samples at once, it will be too "coarse", resulting in poor generalization. Training the model on too few samples leads to sample bias, overfitting the batch distribution rather than the actual distribution of the data set.

- **Backpropagation:**

A widely used algorithm for performing gradient descent on neural networks with the goal of optimizing the network parameters. Firstly, the output values of each node are computed and stored in a forward pass through the network. Secondly, the partial derivative of the loss with respect to each parameter is computed in a backward pass through the network.

- **Deep Neural Network:**

A type of neural network consisting of multiple layers.

- **Training Convergence:**

A terminology that refers to a state reached during training in which the optimization loss changes little with each iteration after a certain number of iterations, and additional training on the current data does not further improve the model.

- **Epoch:**

A complete training run over the entire data set, so that each example was seen once. The total number of training iterations in an epoch can be computed by taking the size of the data set and dividing by the batch size.

- **Latent Space:**

An embedding of data with lower dimensionality than the feature space from which the data points originate.

- **Learning Rate:**

At each training iteration, the gradient descent algorithm multiplies the gradient by the learning rate. This controls how much we adjust the weighting updates of our network parameters with respect to the loss gradients obtained by backpropagation.

- **Gradient Descent:**

A technique to minimize loss by computing the gradients of loss with respect to the model's parameters, conditioned on training data. Informally, gradient descent iteratively adjusts parameters, gradually finding the best combination of weights and bias to minimize loss.

- **Hyperparameter:**

Comprehensive term for parameters that are set before training a model.



- **Iteration:**

A single update of a model's weights during training. An iteration consists of computing the gradients of the parameters with respect to the loss on a single batch of data.

- **Training and testing set:**

When developing machine learning models, it is common to divide the entire data set randomly into subsections: A training set with about 80% of samples and a test set with 20%. In this way, the model can be trained and tested with separate data to test its generalization ability. When the trained model generalizes to new data samples that come from the same data distribution as the samples on which it was originally trained, we have determined that the model has actually "learned" something and is not just mimicking the patterns in the training data.

- **Optimization Objective:**

A metric that the model aims optimize during training. It is used to define the loss of the model which can then be used for parameter estimation that computes the difference between estimated and true values for an instance of data. Common objective functions include mean squared loss and cross-entropy loss.

- **Kullback–Leibler Divergence:**

A measure of difference of two probability distributions  $A$  and  $B$ : Small values indicate little loss of information when describing the observations of  $B$  using  $A$ . We denote it as

$$KL(A \parallel B) = \sum_x P(x) \log \left( \frac{P(x)}{Q(x)} \right).$$

## 9.2 NEURAL NETWORK COMPONENTS

- **Activation function:**

A function that takes in the weighted sum of all of the inputs from the previous layer and then generates an output value to the next layer. This aspect of the neural network is the main source of non-linearity between input and output. The activation functions used in this thesis are:

- **Hyperbolic tangent:**

$$f(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \text{ which maps values into the range } [-1, 1].$$

- **Rectified linear unit (ReLU):**

$$f(x) = \max(0, x) \text{ which only retains positive input values.}$$

- **Sigmoid:**

$$f(x) = \frac{1}{1 + \exp(-x)} \text{ which maps values into the range } [0, 1].$$

- **Convolutional layer:**

As the name implies, a convolution layer applies a convolution operation to its input. Here we start with the element-wise multiplication of the convolution filter with slices of the input matrix. This is followed by the summation of all values in the resulting product matrix. In Figure 42, we visualize a single convolutional operation of a 5x5 input: We take a slice of the input matrix with the same rank and size as the convolutional filter and apply an element wise multiplication of both and sum the resulting product matrix. Expressing the operation of this layer mathematically yields the following expression:

$$\text{Conv}(x) = f(w \odot x + b) \tag{36}$$

with  $b$  and  $w \in \mathbb{R}^{n \cdot l \cdot d}$  as the trainable bias and convolution filter tensor, respectively. In this,  $n$  are the number of filters each with length  $l$  and  $d$  features. Furthermore,  $f$  is the activation function and  $\odot$  is the convolution operator. Finally, the complete convolutional layer then

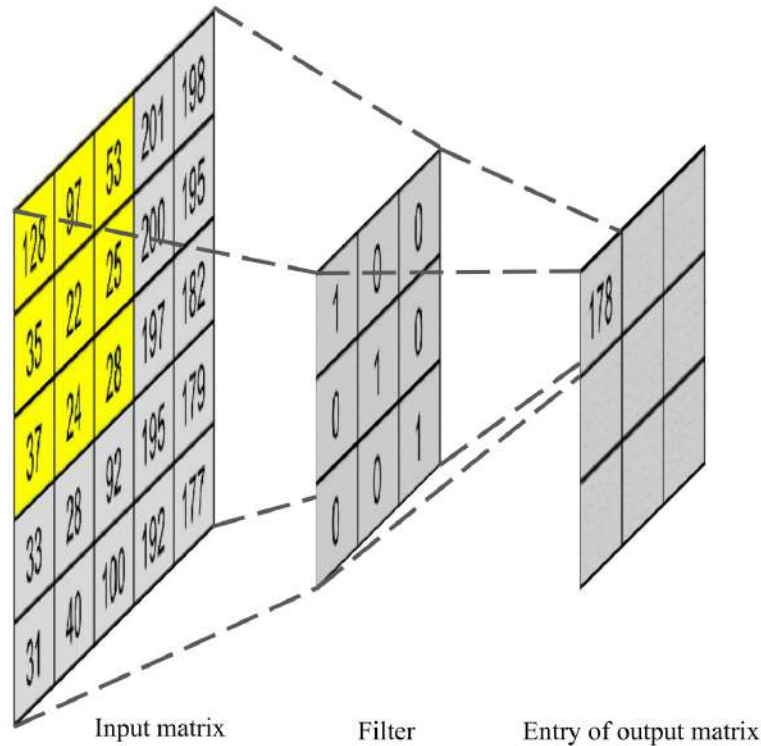


Figure 42: Illustration of a single convolution operation of a convolutional layer. We apply the filtering operator to a subset of the input density matrix and take the sum to obtain an element of the output matrix.

consists of a series of these convolutional operations, each acting on a different slice of the input matrix. [69]

- **Dense Layers:**

A dense layer (fully connected layer) is a layer in which every node is connected to every node in the subsequent layer. In this, the layer takes one or more input values, computes the weighted sum of the inputs, and applies an activation function to yield a single output value.

The following illustrates the performance of a single dense layer: Figure 43 shows a dense layer that takes three inputs, multiplies each by trainable weights, and outputs a single value.

Expressing the operation of this layer mathematically yields the following expression:

$$dense(x) = f\left(\sum_i w_i x_i + b\right) \quad (37)$$

with  $b$  and  $w_i$  as the trainable bias and weight, respectively.

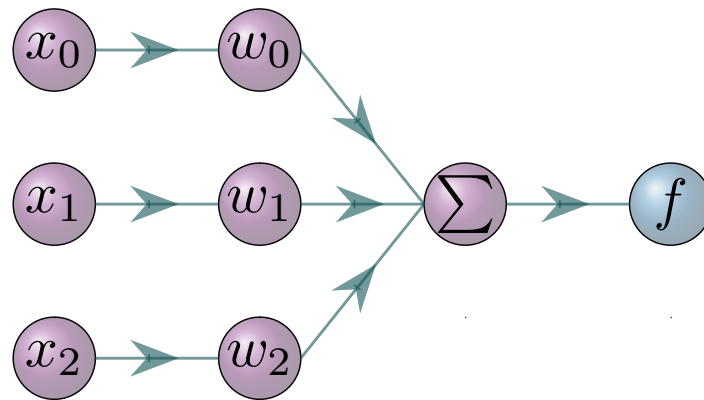


Figure 43: Illustration of a fully connected neural network layer. We apply the dot product between the input  $x$  and the weights  $w$ . We then take the resulting number and apply the activation function  $f$  to obtain the final output.

### 9.3 ANALYSIS METHODS

- **Kernel Density Estimation:**

As an alternative to histograms to approximate a representation of a distribution of data in a discrete manner, we utilize the closely related kernel density estimation to find a smooth and continuous representations. The general concept is to estimate the probability density function  $f$  of observations  $X$  using the kernel density estimator. [70]

$$\hat{f}(x) = \frac{1}{Nh} \sum_{i=1}^N K\left(\frac{x - x_i}{h}\right) \quad (38)$$

In this  $K$  is the so-called kernel and  $h$  is a smoothing parameter called bandwidth. The underlying idea of Eq. 38 is to impose a kernel, for example a normal distribution with a certain variance, on each data point. An estimate of the probability density function is then the sum of the normal distributions, as visualized in Figure 44.

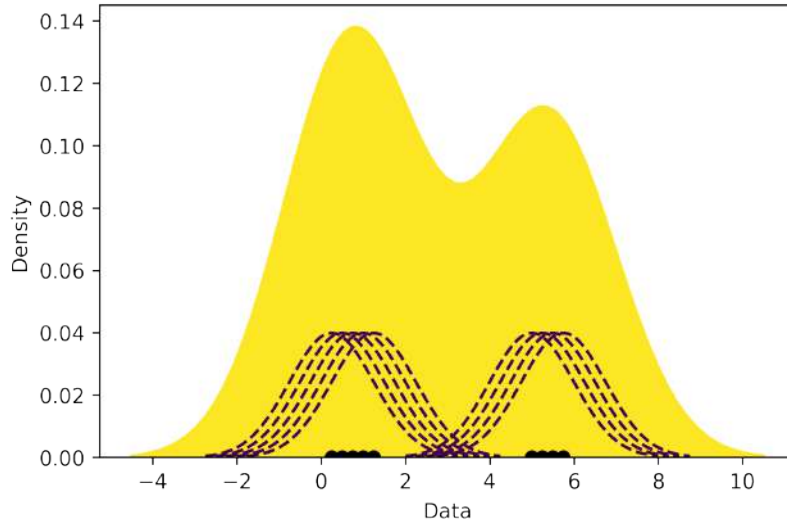
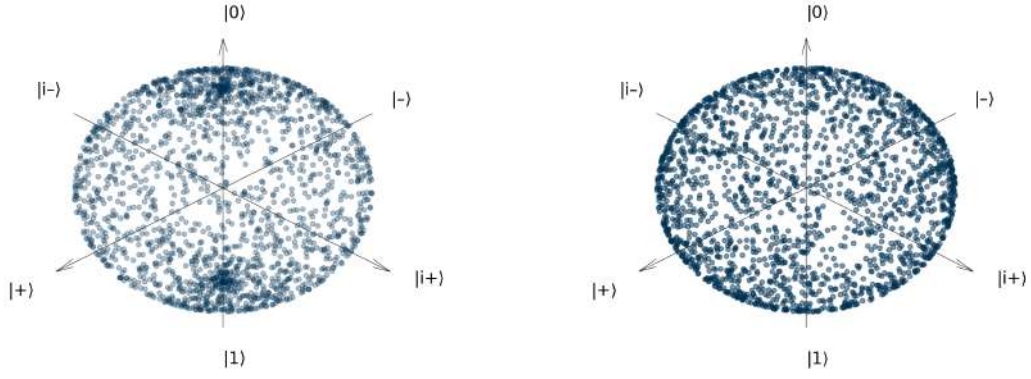


Figure 44: Illustration of the nine individual kernels of the data set  $[0.25, 0.50, 0.75, 1.00, 1.25, 5.00, 5.25, 5.50]$  drawn as blue curves, and the kernel density estimate in yellow. Individual data points are indicated by markers on the horizontal axis.

#### 9.4 GENERATING RANDOM UNITARY OPERATORS

In the following, we detail an algorithm used to generate random local  $N \times N$  unitary operators that are distributed uniformly according to the Haar measure. The motivation for this is to introduce additional randomization that provides the correct performance for a confusion scheme: The the neural network is ideally forced to extract features that are insensitive to these local transformations. The algorithm consists of the following steps, as detailed in Nordenvaad et al. (2004) [71].

- Generate matrix  $Z = X + iY$ , where  $X, Y$  are  $N \times N$  matrices with random entries that are normally distributed with zero mean and unit variance
- Compute QR-decomposition  $Z = QR$
- Compute diagonal matrix  $\Lambda$  with  $\Lambda_{i,i} = \frac{R_{i,i}}{|R_{i,i}|}$
- Compute  $U = Q\Lambda$  which is uniformly distributed under Haar measure



(a) 2000 states generated uniformly in  $\omega, \phi, \theta \in [0, 2\pi)$  (b) 2000 states generated uniformly under Haar measure

Figure 45: Illustration of Bloch vectors of (a) uniformly sampled angles and states uniform distributed over Bloch sphere.

The meaning of "uniformly distributed according to the Haar measure" is demonstrated in the following example. In general, operations in quantum mechanics are described by unitary matrices. For example, the general expression for the rotation of a single qubit is the following.

$$U(\omega, \phi, \theta) = \begin{pmatrix} \exp(-i(\phi + \omega/2)\cos(\theta/2)) & -\exp(-i(\phi - \omega/2)\sin(\theta/2)) \\ -\exp(-i(\phi - \omega/2)\sin(\theta/2)) & \exp(i(\phi + \omega/2)\cos(\theta/2)) \end{pmatrix} \quad (39)$$

Now suppose that we want to apply single-qubit rotations to the fixed base states  $|0\rangle$ , with the goal of uniformly reaching points on the Bloch sphere. The naive approach of sampling  $\omega, \phi, \theta$  from the uniform distribution  $[0, 2\pi)$  is shown in Figure 45a. We note that despite our uniformly distributed parameter samples, the distribution of Bloch vectors is not uniform. We observe clustering around the poles  $|0\rangle$  and  $|1\rangle$  of the sphere and sparseness around the equator. The reason for this is that the area  $dA$  of a differential surface element in spherical coordinates is  $dA(d\theta, d\phi) = r^2 \sin(\phi) d\phi d\theta$ . Therefore, near the poles of the sphere ( $\phi = 0$  and  $\phi = \pi$ ), the differential area element determined by  $d\theta$  and  $d\phi$  becomes smaller, since  $\sin(\phi) \rightarrow 0$ . So we should take fewer points near  $\phi = 0$  and  $\phi = \pi$  and more points near  $\phi = \pi/2$  to achieve a uniform distribution on the sphere. This is exactly what we achieve with the algorithm described above, which samples the elements of the unitary group,

as can be seen in Figure 45b. We observe that utilizing the improved measure, the qubit states are distributed more uniformly on the Bloch sphere.

As a simplification, we restrict the process described above to generating only states with real entries for the randomization scheme used in the experiments. Without this step, the neural network would require additional input neurons to account for the imaginary component of the density matrices, since machine learning models cannot accept imaginary numbers as input. The workaround needed to solve this problem would be to split a given imaginary number  $z = x + iy$  into a real component  $x$  and a complex component  $y$  and use separate neurons as input. This would make it difficult to compare the performance of experiments with different types of noise, since the architecture of the underlying models would be different. For example, in the case of randomizing 2-qubit states, 16 additional input neurons would be required to account for the imaginary components. A set of random states generated with this simplification is shown in Figure 46.

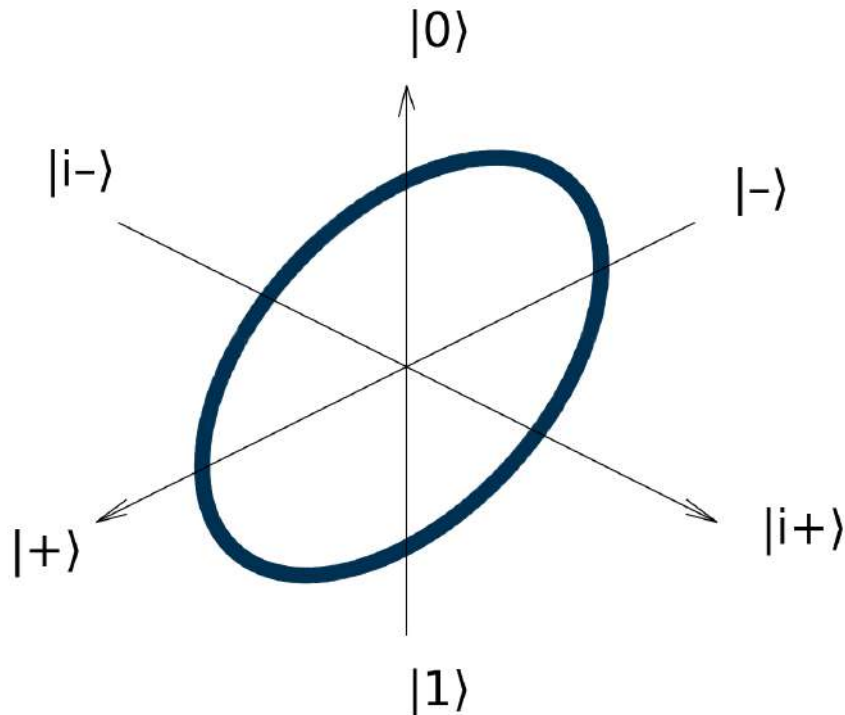
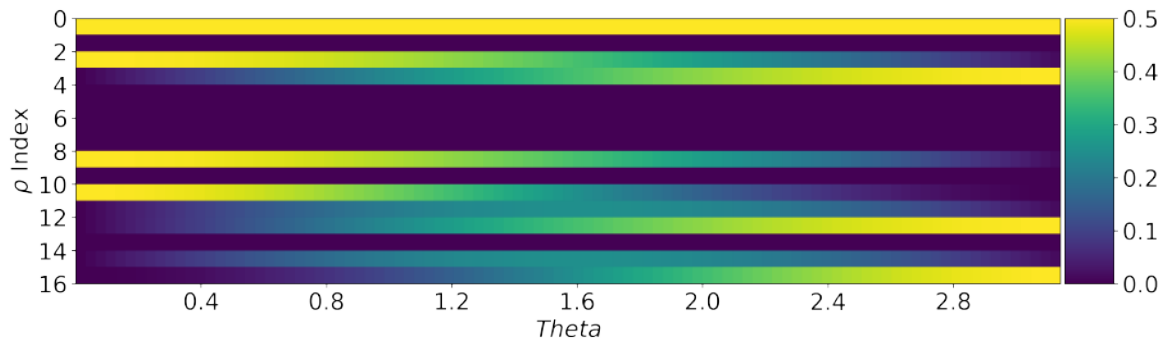


Figure 46: Real component of 2000 states generated uniformly under Haar measure

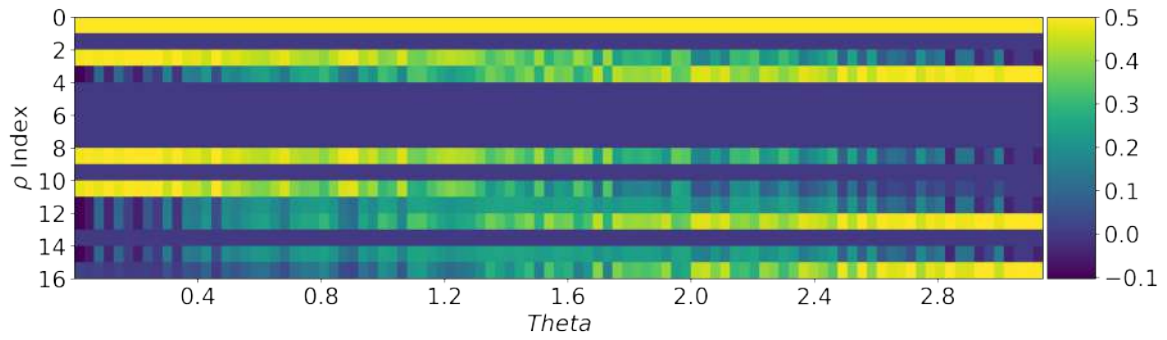
## 9.5 VISUALIZATION DENSITY MATRIX DATA SET

$$\frac{1}{2} \begin{pmatrix} 1 & 0 & \cos\frac{1}{2}\theta & \sin\frac{1}{2}\theta \\ 0 & 0 & 0 & 0 \\ \cos\frac{1}{2}\theta & 0 & \cos^2\frac{1}{2}\theta & \sin\frac{1}{2}\theta\cos\frac{1}{2}\theta \\ \sin\frac{1}{2}\theta & 0 & \sin\frac{1}{2}\theta\cos\frac{1}{2}\theta & \sin^2\frac{1}{2}\theta \end{pmatrix}$$

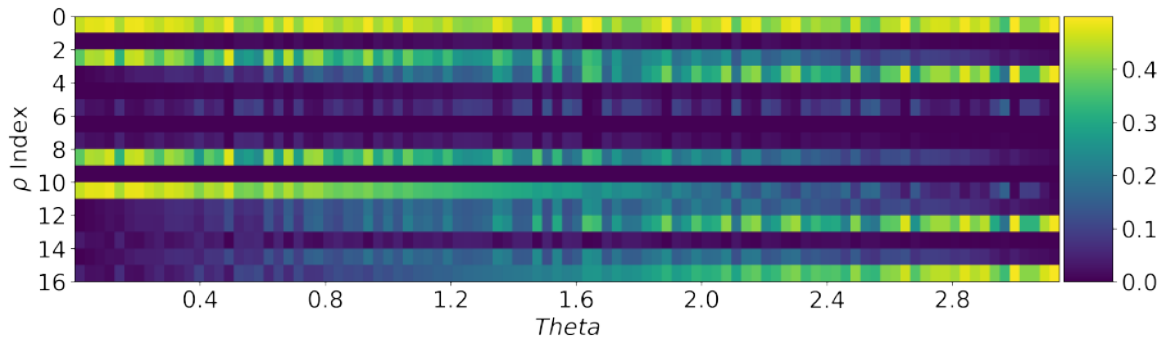
(a) Density matrix as function of  $\theta$ (b) Density matrix elements sampled at angles in  $\theta \in [0, \pi]$ Figure 47: (a) Density matrix and (b) samples generated at angles  $\theta$  using no additional randomization scheme



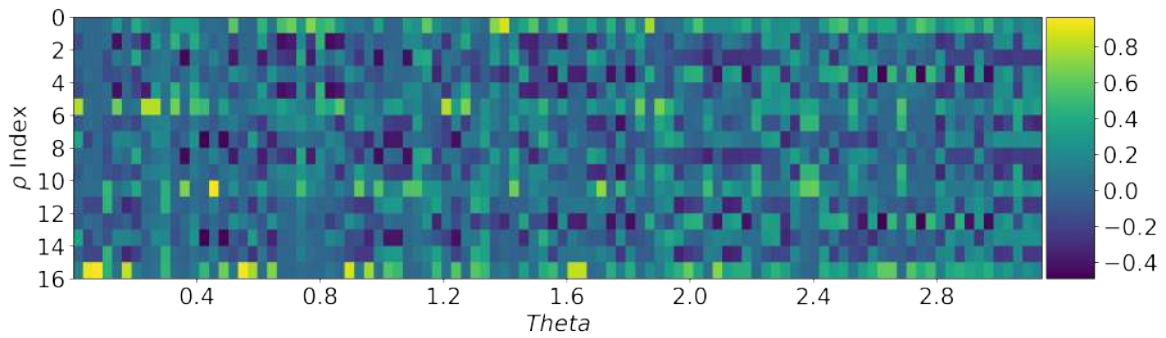
$$\frac{1}{2} \begin{pmatrix} 1 & 0 & \cos\frac{1}{2}(\theta + \kappa) & \sin\frac{1}{2}(\theta + \kappa) \\ 0 & 0 & 0 & 0 \\ \cos\frac{1}{2}(\theta + \kappa) & 0 & \cos^2\frac{1}{2}(\theta + \kappa) & \sin\frac{1}{2}(\theta + \kappa)\cos\frac{1}{2}(\theta + \kappa) \\ \sin\frac{1}{2}(\theta + \kappa) & 0 & \sin\frac{1}{2}(\theta + \kappa)\cos\frac{1}{2}(\theta + \kappa) & \sin^2\frac{1}{2}(\theta + \kappa) \end{pmatrix}$$

(a) Density matrix as function of  $\theta$ (b) Density matrix elements sampled at angles in  $\theta \in [0, \pi]$ Figure 48: (a) Density matrix and (b) samples generated at angles  $\theta$  using coherent noise randomization

$$\sum_i \frac{1}{2^{p_i}} K_i^\dagger \begin{pmatrix} 1 & 0 & \cos \frac{1}{2} \theta & \sin \frac{1}{2} \theta \\ 0 & 0 & 0 & 0 \\ \cos \frac{1}{2} \theta & 0 & \cos^2 \frac{1}{2} \theta & \sin \frac{1}{2} \theta \cos \frac{1}{2} \theta \\ \sin \frac{1}{2} \theta & 0 & \sin \frac{1}{2} \theta \cos \frac{1}{2} \theta & \sin^2 \frac{1}{2} (\theta) \end{pmatrix} K_i$$

(a) Density matrix as function of  $\theta$ (b) Density matrix elements sampled at angles in  $\theta \in [0, \pi]$ Figure 49: (a) Density matrix and (b) samples generated at angles  $\theta$  using incoherent noise randomization

$$(U_a \otimes U_b) \begin{pmatrix} 1 & 0 & \cos\frac{1}{2}\theta & \sin\frac{1}{2}\theta \\ 0 & 0 & 0 & 0 \\ \cos\frac{1}{2}\theta & 0 & \cos^2\frac{1}{2}\theta & \sin\frac{1}{2}\theta\cos\frac{1}{2}\theta \\ \sin\frac{1}{2}\theta & 0 & \sin\frac{1}{2}\theta\cos\frac{1}{2}\theta & \sin^2\frac{1}{2}(\theta) \end{pmatrix} (U_a \otimes U_b)$$

(a) Density matrix as function of  $\theta$ (b) Density matrix elements sampled at angles in  $\theta \in [0, \pi]$ Figure 50: (a) Density matrix and (b) samples generated at angles  $\theta$  using random unitary randomization

9.6 INTERPRETATION OF LATENT VARIABLE  $Z_1$ 

To gain insights into the latent representation of  $Z_1$ , we examine density matrices encoded near the edge cases  $Z_1 > 0.05$  and  $Z_1 < -0.05$  in Figure 51.

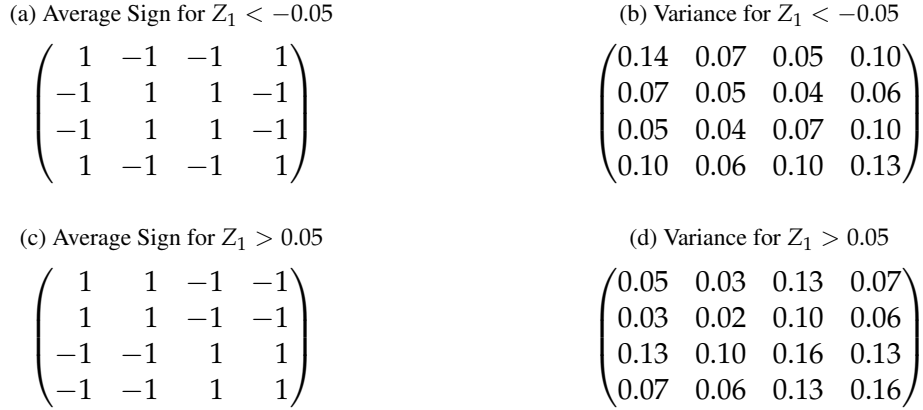


Figure 51: Average sign of density matrices with latent encoding of (a)  $Z_1 < -0.05$  and (c)  $Z_1 > 0.05$ .

Variance of density matrix entries with latent encoding of (b)  $Z_1 < -0.05$  and (d)  $Z_1 > 0.05$ .

Figure 51a illustrates the average sign of 43 density matrices for which the trained encoder predicted latent parameters of  $Z_1 < -0.05$ . Similarly, Figure 51c illustrates the average sign of 84 density matrices that were encoded with latent parameters of  $Z_1 > 0.05$ . We find that in each respective case the position of the signs coincides. Figures 51b and 51d illustrate the respective variance of the same density matrix samples. We note that the variance is generally nonzero. This shows that the coincidence of the positions of the signs is not a mere consequence of matching numerical values of the density matrix entries. From these observations we infer that the model learns to assign the same latent parameter values for density matrices with some common structure, such as matching sign positions. To complete this analysis, we revisit the information in Figure 37b, where we found that there is no obvious correlation between the generative factor  $\theta$  and the latent parameter  $Z_1$ . We now know that this does not mean that the latent variable is unimportant to the encoding-decoding process: rather, the dispersion of  $Z_1$  contributes to the different noise patterns introduced by the randomization scheme. Here, the interpolation of density matrices of a given latent parameter value leads to a common structure, but does not guarantee matching properties that are a function of  $\theta$ .

---

## BIBLIOGRAPHY

---

1. Bengio, Y., Courville, A. & Vincent, P. *Representation Learning: A Review and New Perspectives* 2012. <https://arxiv.org/abs/1206.5538>.
2. Kingma, D. P. & Welling, M. *Auto-Encoding Variational Bayes* 2013. <https://arxiv.org/abs/1312.6114>.
3. Pu, Y. *et al.* *Variational Autoencoder for Deep Learning of Images, Labels and Captions* in *Advances in Neural Information Processing Systems* (eds Lee, D., Sugiyama, M., Luxburg, U., Guyon, I. & Garnett, R.) **29** (Curran Associates, Inc., 2016). <https://proceedings.neurips.cc/paper/2016/file/eb86d510361fc23b59f18c1bc9802cc6-Paper.pdf>.
4. Walker, J., Doersch, C., Gupta, A. & Hebert, M. *An uncertain future: Forecasting from static images using variational autoencoders* in *European Conference on Computer Vision* (2016), 835–851.
5. Hou, X., Shen, L., Sun, K. & Qiu, G. *Deep feature consistent variational autoencoder* in *2017 IEEE winter conference on applications of computer vision (WACV)* (2017), 1133–1141.
6. Kusner, M. J., Paige, B. & Hernández-Lobato, J. M. *Grammar Variational Autoencoder* in *Proceedings of the 34th International Conference on Machine Learning* (eds Precup, D. & Teh, Y. W.) **70** (PMLR, June 2017), 1945–1954. <https://proceedings.mlr.press/v70/kusner17a.html>.
7. Chien, J.-T. *Deep Bayesian natural language processing* in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts* (2019), 25–30.

8. Li, J. *et al.* *Generating classical chinese poems via conditional variational autoencoder and adversarial training in Proceedings of the 2018 conference on empirical methods in natural language processing* (2018), 3890–3900.
9. Kim, J.-H. *et al.* Representation Learning of Resting State fMRI with Variational Autoencoder. *bioRxiv*. eprint: <https://www.biorxiv.org/content/early/2021/01/30/2020.06.16.155937.full.pdf>. <https://www.biorxiv.org/content/early/2021/01/30/2020.06.16.155937> (2021).
10. Baumgartner, C. F. *et al.* *Phiseg: Capturing uncertainty in medical image segmentation in International Conference on Medical Image Computing and Computer-Assisted Intervention* (2019), 119–127.
11. Uzunova, H., Schultz, S., Handels, H. & Ehrhardt, J. Unsupervised pathology detection in medical images using conditional variational autoencoders. *International journal of computer assisted radiology and surgery* **14**, 451–461 (2019).
12. Higgins, I. *et al.* *beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework in International Conference on Learning Representations* (2017). <https://openreview.net/forum?id=Sy2fzU9gl>.
13. Rocchetto, A., Grant, E., Strelchuk, S., Carleo, G. & Severini, S. Learning hard quantum distributions with variational autoencoders. *npj Quantum Information* **4**, 28. ISSN: 2056-6387. <https://doi.org/10.1038/s41534-018-0077-z> (June 2018).
14. Wetzel, S. J. Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders. *Phys. Rev. E* **96**, 022140. <https://link.aps.org/doi/10.1103/PhysRevE.96.022140> (2 Aug. 2017).
15. Pol, A. A., Berger, V., Germain, C., Cerminara, G. & Pierini, M. *Anomaly Detection with Conditional Variational Autoencoders in 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)* (2019), 1651–1657.

16. Xiong, Y., Zuo, R., Luo, Z. & Wang, X. A physically constrained variational autoencoder for geochemical pattern recognition. *Mathematical Geosciences* **54**, 783–806 (2022).
17. Hinton, G. & Sejnowski, T. J. *Unsupervised learning: foundations of neural computation* (MIT press, 1999).
18. Li, Y. & Wu, H. A Clustering Method Based on K-Means Algorithm. *Physics Procedia* **25**. International Conference on Solid State Devices and Materials Science, April 1-2, 2012, Macao, 1104–1109. ISSN: 1875-3892. <https://www.sciencedirect.com/science/article/pii/S1875389212006220> (2012).
19. Reynolds, D. in *Encyclopedia of Biometrics* (eds Li, S. Z. & Jain, A.) 659–663 (Springer US, Boston, MA, 2009). ISBN: 978-0-387-73003-5. [https://doi.org/10.1007/978-0-387-73003-5\\_196](https://doi.org/10.1007/978-0-387-73003-5_196).
20. Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (AAAI Press, Portland, Oregon, 1996), 226–231.
21. Nielsen, F. in, 195–211 (Feb. 2016). ISBN: 978-3-319-21902-8.
22. F.R.S., K. P. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **2**, 559–572 (1901).
23. Van der Maaten, L. & Hinton, G. Visualizing data using t-SNE. *Journal of machine learning research* **9** (2008).
24. Wang, Y., Yao, H. & Zhao, S. Auto-encoder based dimensionality reduction. *Neurocomputing* **184**. RoLoD: Robust Local Descriptors for Computer Vision 2014, 232–242. ISSN: 0925-2312. <https://www.sciencedirect.com/science/article/pii/S0925231215017671> (2016).

25. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* 2018. <https://arxiv.org/abs/1810.04805>.
26. Gens, R. & Domingos, P. M. *Deep Symmetry Networks* in *Advances in Neural Information Processing Systems* (eds Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. & Weinberger, K.) **27** (Curran Associates, Inc., 2014). <https://proceedings.neurips.cc/paper/2014/file/f9be311e65d81a9ad8150a60844bb94c-Paper.pdf>.
27. Gowal, S. *et al.* *Achieving Robustness in the Wild via Adversarial Mixing with Disentangled Representations* 2019. <https://arxiv.org/abs/1912.03192>.
28. Higgins, I. *et al.* *DARLA: Improving Zero-Shot Transfer in Reinforcement Learning* 2017. <https://arxiv.org/abs/1707.08475>.
29. Achille, A. *et al.* *Life-Long Disentangled Representation Learning with Cross-Domain Latent Homologies* 2018. <https://arxiv.org/abs/1808.06508>.
30. Bengio, Y., Courville, A. & Vincent, P. *Representation Learning: A Review and New Perspectives* 2012. <https://arxiv.org/abs/1206.5538>.
31. Bengio, Y. *Deep Learning of Representations: Looking Forward* 2013. <https://arxiv.org/abs/1305.0445>.
32. Higgins, I. *et al.* *Towards a Definition of Disentangled Representations* 2018. <https://arxiv.org/abs/1812.02230>.
33. Miller, T. *Explanation in Artificial Intelligence: Insights from the Social Sciences* 2017. <https://arxiv.org/abs/1706.07269>.
34. Goodfellow, I. J., Bengio, Y. & Courville, A. *Deep Learning* <http://www.deeplearningbook.org> (MIT Press, Cambridge, MA, USA, 2016).
35. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C. & Chen, M. *Hierarchical Text-Conditional Image Generation with CLIP Latents* 2022. <https://arxiv.org/abs/2204.06125>.



36. Goodfellow, I. *et al.* *Generative adversarial nets* in *Advances in neural information processing systems* (2014), 2672–2680.
37. Hinton, G. E. Boltzmann machine. *Scholarpedia* **2**, 1668 (2007).
38. Liou, C.-Y., Cheng, W.-C., Liou, J.-W. & Liou, D.-R. Autoencoder for words. *Neurocomputing* **139**, 84–96 (2014).
39. Gondara, L. *Medical image denoising using convolutional denoising autoencoders* in *2016 IEEE 16th international conference on data mining workshops (ICDMW)* (2016), 241–246.
40. Cheng, Z., Sun, H., Takeuchi, M. & Katto, J. *Deep convolutional autoencoder-based lossy image compression* in *2018 Picture Coding Symposium (PCS)* (2018), 253–257.
41. Meng, Q., Catchpoole, D., Skillicom, D. & Kennedy, P. J. *Relational autoencoder for feature extraction* in *2017 International Joint Conference on Neural Networks (IJCNN)* (2017), 364–371.
42. Hinton, G. E., Krizhevsky, A. & Wang, S. D. *Transforming auto-encoders* in *International conference on artificial neural networks* (2011), 44–51.
43. Morales-Forero, A. & Bassetto, S. Case Study: A Semi-Supervised Methodology for Anomaly Detection and Diagnosis. *2019 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 1031–1037. <https://doi.org/10.1109/IEEM44572.2019.8978509> (2019).
44. Ruder, S. *An overview of gradient descent optimization algorithms* 2016. <https://arxiv.org/abs/1609.04747>.
45. Myung, I. J. Tutorial on maximum likelihood estimation. *Journal of mathematical Psychology* **47**, 90–100 (2003).
46. Pan, J.-X. & Fang, K.-T. in *Growth curve models and statistical diagnostics* 77–158 (Springer, 2002).

47. Kullback, S. & Leibler, R. A. On Information and Sufficiency. *The Annals of Mathematical Statistics* **22**, 79–86. <https://doi.org/10.1214/aoms/1177729694> (1951).
48. Bowman, S. R. *et al.* Generating Sentences from a Continuous Space. <https://arxiv.org/abs/1511.06349> (2015).
49. Fu, H. *et al.* Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Association for Computational Linguistics, Minneapolis, Minnesota, June 2019), 240–250. <https://aclanthology.org/N19-1021>.
50. Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* **29**, 141–142 (2012).
51. *Udacity Intro to TensorFlow for Deep Learning* <https://www.udacity.com/course/intro-to-tensorflow-for-deep-learning--ud187>. Accessed: 2022-09-03.
52. *Theano Tutorial* <https://lasagne.readthedocs.io/en/latest/user/tutorial.html>. Accessed: 2022-09-03.
53. Kingma, D. P. & Ba, J. *Adam: A Method for Stochastic Optimization* 2014. <https://arxiv.org/abs/1412.6980>.
54. Chollet, F. *et al.* *Keras* <https://github.com/keras-team/keras>. 2022.
55. Wetzel, S. J. Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders. *Phys. Rev. E* **96**, 022140. <https://link.aps.org/doi/10.1103/PhysRevE.96.022140> (2 Aug. 2017).
56. Walker, N., Tam, K.-M. & Jarrell, M. Deep learning on the 2-dimensional Ising model to extract the crossover region with a variational autoencoder. *Scientific Reports* **10**, 13047. ISSN: 2045-2322. <https://doi.org/10.1038/s41598-020-69848-5> (Aug. 2020).

57. Yevick, D. Variational autoencoder analysis of Ising model statistical distributions and phase transitions. *The European Physical Journal B* **95**. <https://doi.org/10.1140%2Fepjb%2Fs10051-022-00296-y> (Mar. 2022).
58. Chaikin, P. M. & Lubensky, T. C. *Principles of Condensed Matter Physics* (Cambridge University Press, 1995).
59. Wolff, U. Collective Monte Carlo Updating for Spin Systems. *Phys. Rev. Lett.* **62**, 361–364. <https://link.aps.org/doi/10.1103/PhysRevLett.62.361> (4 Jan. 1989).
60. Peruzzo, A. *et al.* A variational eigenvalue solver on a photonic quantum processor. *Nature Communications* **5**, 4213. ISSN: 2041-1723. <https://doi.org/10.1038/ncomms5213> (July 2014).
61. Farhi, E., Goldstone, J. & Gutmann, S. *A Quantum Approximate Optimization Algorithm* 2014. <https://arxiv.org/abs/1411.4028>.
62. Anand, A., Kristensen, L. B., Frohnert, F., Sim, S. & Aspuru-Guzik, A. *Information flow in parameterized quantum circuits* 2022. <https://arxiv.org/abs/2207.05149>.
63. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).
64. Jo, J. & Bengio, Y. *Measuring the tendency of CNNs to Learn Surface Statistical Regularities* 2017. <https://arxiv.org/abs/1711.11561>.
65. Lundberg, M. & Svensson, L. *The Haar measure and the generation of random unitary matrices in 2004 IEEE Sensor Array and Multichannel Signal Processing Workshop: 18 - 21 July 2004, Barcelona, Spain* Godkänd; 2004; 20100217 (andbra) (IEEE Communications Society, 2004), 114–118. ISBN: 0-7803-8545-4. <http://urn.kb.se/resolve?urn=urn:nbn:se:ltu:diva-31088>.
66. Mezzadri, F. How to generate random matrices from the classical compact groups. <https://arxiv.org/abs/math-ph/0609050> (2006).

67. Gavreev, M. A., Mastiukova, A. S., Kiktenko, E. O. & Fedorov, A. K. Learning entanglement breakdown as a phase transition by confusion. *New Journal of Physics* **24**, 073045. <https://doi.org/10.1088%5C%2F1367-2630%5C%2Fac7fb2> (July 2022).
68. Foster, D. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play* ISBN: 9781492041917. <https://books.google.de/books?id=RKegDwAAQBAJ> (O'Reilly Media, 2019).
69. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks* **61**, 85–117. <https://doi.org/10.1016%2Fj.neunet.2014.09.003> (Jan. 2015).
70. Sheather, S. J. Density Estimation. *Statistical Science* **19**, 588–597. ISSN: 08834237. <http://www.jstor.org/stable/4144429> (2022) (2004).
71. Nordenvaad, M. & Svensson, L. *The Haar measure and the generation of random unitary matrices* in (Aug. 2004), 114–118. ISBN: 0-7803-8545-4.