Department of Physics and Astronomy

Heidelberg University

Master thesis

in Physics

submitted by

Tobias Schmale

born in Hildesheim

2021

Scalable Quantum State Tomography using Artificial Neural Networks

This Master thesis has been carried out by Tobias Schmale

at the

Kirchhoff-Institut für Physik, Heidelberg

under the supervision of

Priv.-Doz. Dr. Martin Gärttner

Skalierbare Quantenzustandstomographie mit künstlichen Neuronalen Netzen:

Moderne Quantensimulatoren können eine Vielzahl von Quantenzuständen präparieren, jedoch stellt das Auslesen dieser "Quantendaten" aufgrund seiner exponentiell skalierenden Natur nach wie vor eine Herausforderung dar. In dieser Arbeit wird dieses Problem angegangen, in dem eine Quantenzustandstomographiemethode entwickelt wird, deren zentraler Baustein ein neuronales Netzwerk ist. Dieses approximiert die Wahrscheinlichkeitsverteilung über die Ergebnisse einer informationell vollständigen Messung. Dabei zeigt sich eine hervorragende Darstellbarkeit prototypischer Grund- und Gleichgewichtszustände, unter Verwendung einer Anzahl an Variationsparametern, die nur polynomiell in der Systemgröße skaliert. Diese komprimierte Darstellung erlaubt die Rekonstruktion von Zuständen mit hohen Genauigkeiten, wobei Standardmethoden, wie die Maximum-Likelihood-Schätzung, übertroffen werden. Des Weiteren werden quadratische Mittelwertfehler von Observablen erreicht, die um bis zu einer Größenordnung kleiner sind als bei der direkten Schätzung aus experimentellen Daten. Mögliche Hindernisse, wie ein Scheitern der Rekonstruktion nicht-repräsentierbarer Zustände, sowie resultierende Einflüsse systematischer Fehler werden analysiert. Somit wird ein entscheidender Schritt gemacht, hin zur Anwendung der Methode im experimentellen Rahmen.

Scalable Quantum State Tomography using Artificial Neural Networks:

Modern day quantum simulators can prepare a wide variety of quantum states, yet the readout of this ”quantum data” still poses a challenge, due to its exponentially scaling nature. In this thesis, this problem is tackled by developing a quantum state tomography scheme which relies on approximating the probability distribution over the outcomes of an informationally complete measurement in a variational manifold represented by a convolutional neural network. An excellent representability of prototypical ground- and steady-states with this ansatz is shown, using a number of variational parameters that scales polynomially in system size. This compressed representation allows the reconstruction of states with high classical fidelities, outperforming standard methods such as maximum likelihood estimation. Furthermore, it achieves a reduction of the root mean square errors of observables by up to an order of magnitude compared to their direct estimation from experimental data. Possible pitfalls like the failure to reconstruct unrepresentable states and effects of biases are analysed, paving the way for an application of this tomography scheme in experimental settings.

Publications

Results of this thesis, in particular those from Section 3.2, as well as parts of Sections 3.1.2 and 2.3.1 are published in the following paper. The author of this thesis is the main author of said publication, hence parts are reproduced verbatim throughout this thesis, with permission of the coauthors.

> T. Schmale, M. Reh, and M. Gärttner. "Scalable quantum state tomography with artificial neural networks". In: *arXiv:2109.13776 [cond-mat, physics:physics, physics:quant-ph]* (Sept. 28, 2021)

Contents

1 Introduction

Quantum computers and quantum simulators are a hot topic of experimental and theoretical research. Both technologies promise to represent milestones in fields such as solid-state physics, particle physics, chemistry and even cosmology [1] and there is a multitude of platforms such as superconducting qubits, ion chains, Rydberg systems, cold atom systems and many more [1], that are continually maturing to powerful scientific instruments that make contributions to the stated fields. Like any experimental apparatus, these devices have to be validated, characterized and benchmarked, in order to ensure that simulations performed actually match physical reality. While parts of this validation process might happen on a component basis, e.g. ensuring that lasers or power supplies are within specification, there eventually comes a stage, where the prepared quantum states themselves must fall under scrutiny. It is at this point where the task becomes particularly challenging, especially when compared to the validation of classical ”states” or bit-registers. While a classical register might contain N bits of information (which is trivial to read out), describing the full quantum equivalent, i.e. a quantum state of N qubits, requires the knowledge of 2^N complex numbers. Furthermore, quantum mechanics is at its heart a probabilistic description of nature. This implies that the readout of a general quantum state is of statistical nature and requires a process in which data is collected. The consequence of these two facts, is that the readout of a quantum state is a highly challenging task, that takes an exponential effort to succeed.

Neural networks on the other hand, are a versatile numerical tool, that have proven highly useful in the application to tasks of even exponential computational complexity, such as protein folding [2], which some deem solved [3] by AlphaFold's [4] recent progress using machine learning approaches. The task of modelling human language is another, to which an application of neural networks has had a disruptive impact. The famous GPT-3 model can summarize texts and synthesize new ones with a quality that makes it hard to distinguish its output from human made ones [5].

It is therefore not surprising that neural networks have also found their way into the quantum sciences. Neural networks have been used to discriminate topological phases in quantum systems [6], for detecting phase transitions [7] and have been used as efficient parametrizations of wavefunctions [8], so-called neural quantum states (NQS). For the latter task, neural networks have been theoretically proven to be capable of outperforming the best standard methods, such as matrix-product states (MPS) [9, 10]. It is this success that motivates this thesis: solving the exponentially hard task of quantum state readout, termed *quantum state tomography* (QST), with the versatile tool that is ”neural networks”.

This is done by following the ideas of Carrasquilla et al. from [11]: The quantum state is injectively mapped to a probability distribution, which, in turn, gets approximated by a neural network. This probabilistic formulation in combination with Monte Carlo methods allows for observables to be efficiently extracted from the exponentially large quantum state. The neural network, being a powerful function approximator, is capable of learning this probability distribution from a very sparse dataset, alleviating the experimental burden of data collection. These are the two key ingredients, that will allow for a more efficient quantum state readout.

This thesis is structured as follows: In Section 2.1, the term *quantum state tomography* is defined more precisely. Section 2.2 will detail the mathematics of *positive operator valued measures* (POVMs), which is the probabilistic framework used to bridge the gap between quantum mechanics and neural networks. Section 2.3 will provide the necessary background on neural networks and illustrate the network architectures used throughout this thesis. In Section 2.4 all ingredients are put together and the main tomography scheme of interest is explained in detail, after a more extensive literature overview is given.

The results are split into three sections, in which the tomography scheme is benchmarked in the three scenarios of infinite synthetic data (Sec. 3.1), limited synthetic data (Sec. 3.2) and real-world experimental data (Sec. 3.3).

2 Theory and Methods

2.1 Quantum state tomography

Quantum measurements follow the Born rule [12]: If one prepares a mixed quantum state $\hat{\rho}$ and measures a hermitian observable $\hat{O} = \sum_i o_i \left|o_i\right\rangle\!\left\langle o_i\right|$, one randomly obtains the outcome o_i with probability

$$P(\text{obtain } o_i) = \mathrm{Tr}[\hat{\rho} \left|o_i\right\rangle\!\left\langle o_i\right|]. \tag{2.1}$$

The measurement updates the quantum state, projecting it onto the eigenspace of the obtained eigenvalue:

$$\hat{\rho} \xrightarrow{\text{obtained } o_i} \hat{\rho}_{\text{post-meas.}} = \frac{\left|o_i\right\rangle\!\left\langle o_i\right| \hat{\rho} \left|o_i\right\rangle\!\left\langle o_i\right|}{\mathrm{Tr}[\hat{\rho} \left|o_i\right\rangle\!\left\langle o_i\right|]} \qquad \text{(for non-degenerate } o_i\text{)}. \tag{2.2}$$

In general, this process is irreversible, as the density matrix gets projected onto a linear subspace with dimension 1, in the extreme case. This is often called the ”collapse of the wavefunction”. The original state is now destroyed and repeating this measurement gives no further information about the prepared state or measured observable. If one wants to reconstruct the density matrix of the original state, one has to prepare multiple copies of it, and measure each one independently. This is exactly the task which quantum state tomography (QST) aims to solve: given multiple independent copies of a quantum state $\hat{\rho}$, perform multiple measurements and process the resulting data in such a way, to infer the density matrix $\hat{\rho}$ of the state. The combination of the set of measurements and this post-processing will be referred to as a *tomography scheme.*

For an N qubit system, the Hilbert space is 2^N-dimensional. This means that $\hat{\rho}$ is a matrix with $2^N \times 2^N = 4^N$ complex entries. Neglecting normalization and taking into account that $\hat{\rho}$ is hermitian, this results in 4^N independent real parameters, all of which have to be estimated from experimental data. One therefore has to perform at least 4^N linearly independent measurements, of which each requires repeated execution to collect statistics [13]. The experimental burden of such *full* QST is thus of of exponential nature, i.e. the amount of data the experiment has to collect is exponential. In addition to that, the classical post-processing is of similar complexity: if the desired result is the full, exponentially large density matrix, the numerical complexity for obtaining this matrix is of course also exponential. This is why performing a successful state tomography may be thought of as one of the most challenging tasks related to quantum mechanics: it suffers two-fold from the exponential scaling of quantum mechanics, once in experiment and once in post-processing.

This task seems quite daunting and in fact is only feasible in full extent on relatively small systems. For larger systems, corners need to be cut. Over the years, several algorithms have been developed, which make some compromise regarding the generality of the tomography scheme, in order to avoid the exponential scaling. In Section 2.1.1 some criteria are laid out, which may be used to classify most of these algorithms. In Section 2.1.2 an overview over various existing tomography schemes is given.

2.1.1 Classification of tomography schemes

One can write down a few properties that a perfect tomography scheme might possess:

① Sub-exponential scaling in required experimental data.

② Sub-exponential scaling in classical post-processing.

③ ”Observable universality”, referring to the requirement that upon performing a successful tomography, any linear or non-linear quantum observable should be faithfully computable, without requiring further experimental data.

④ ”State universality” meaning that the algorithm should be indifferent to the (possibly mixed) target state that is prepared experimentally.

Obviously, no tomography scheme can exist that matches all four of these requirements to perfect accuracy. Most tomography schemes choose to give up one or more of these conditions, in order to gain with respect to the remaining.

One might argue whether a scheme that does not satisfy ”observable universality” should be called a *tomography* scheme. Strictly following the above criteria, an entanglement detection scheme for example would fall under the ”observable universality” violating *tomography* schemes, even though entanglement detection usually is not called ”tomography”. However, recent approaches like shadow tomography [14, 15] blur the lines between simple measurements of single observables and schemes resulting in full density matrices. Hence, it makes sense to explicitly mention ”observable universality” as a distinguishing property.

Using these criteria, the main target of this thesis may also be formulated more precisely: As the word ”scalable” in the title suggests, the goal is to construct a scheme, that gives an improvement w.r.t properties ① and ② while possibly making compromises on properties ③ and ④. To see that making such compromises are not too unusual, one may look at the properties of some more common tomography schemes.

2.1.2 Overview of tomography schemes

Linear inversion

The (mathematically) simplest tomography scheme relies on finding an orthonormal basis M_i for the set of hermitian operators on the Hilbert space. This allows for an expansion of the density matrix as

$$\hat{\rho}_{\mathrm{LI}} = c_i \hat{M}_i \qquad c_i \in \mathbb{R} \quad \forall i \in \{1, ..., 4^N\}. \tag{2.3}$$

In this thesis, the Einstein-convention is used, implying sums over repeated indices. Since all $\hat{M}_i$ are observables, the coefficients c_i may be measured directly:

$$\mathrm{Tr}\left[\hat{\rho}_{\mathrm{LI}} \hat{M}_i\right] = c_j \, \mathrm{Tr}\left[\hat{M}_j \hat{M}_i\right] = c_j \delta_{ij} = c_i. \tag{2.4}$$

An example for such a basis is are Pauli-strings, where the $\hat{M}_i$ are all 4^N products of the four operators $\{\mathbb{1}/2, \hat{\sigma}^x/2, \hat{\sigma}^y/2, \hat{\sigma}^z/2\}$ [13]. Not only is this procedure mathematically simple, but it is also easy to implement in experiments which allow individual qubit control, as measuring Pauli-strings only requires single qubit rotations and subsequent projections. However, this scheme violates several of the conditions from Section 2.1.1. The expectation values of 4^N operators have to be measured experimentally, violating condition ①. The subsequent sum in Eq. (2.3) is exponentially hard, thus condition ② is violated. Also, for finite measurement statistics, the resulting density matrix may be negative. A negative density matrix implies that some observables will not be captured accurately, thus hindering condition ③.

Maximum likelihood estimation

Maximum likelihood estimation (MLE) is the simplest extension of linear inversion that ensures positive density matrices. It seeks to find the physical density matrix that has the highest probability of yielding the dataset that was actually measured. The probability of obtaining the dataset is given by the product of the probabilities of obtaining the individual measurement outcomes o_i

$$\begin{aligned} P(\mathrm{Dataset} = \{o_1, o_2, ...\} | \hat{\rho}_{\mathrm{guess}}) &= \prod_{o_i \in \mathrm{Dataset}} P(o_i | \hat{\rho}_{\mathrm{guess}}) \\ &= \prod_{o_i \in \mathrm{Dataset}} \mathrm{Tr}[\hat{\rho}_{\mathrm{guess}} \, |o_i\rangle\langle o_i|] =: \mathcal{L}(\hat{\rho}_{\mathrm{guess}}). \end{aligned} \tag{2.5}$$

This quantity is called the *likelihood function*. MLE now seeks to maximize it w.r.t. $\hat{\rho}_{\text{guess}}$

$$\hat{\rho}_{\text{MLE}} = \underset{\hat{\rho}_{\text{guess}} \text{ physical d.m.}}{\text{argmax}} \mathcal{L}(\hat{\rho}_{\text{guess}}). \tag{2.6}$$

To perform this non-linear maximization two standard algorithms are used. The first parametrizes $\hat{\rho}_{\text{guess}}$ as $\frac{BB^\dagger}{\operatorname{Tr} BB^\dagger}$, with any complex matrix B of correct dimensions. This parametrization ensures positivity, normalization and hermiticity of the density matrix, therefore turning the optimization problem into an unconstrained one. This may be performed for example using standard gradient descent based methods. An alternative method is so called iterative MLE [16]. It works by first constructing the matrix

$$\hat{R}(\hat{\rho}) = \sum_{o_i \in \text{Dataset}} \frac{|o_i\rangle\langle o_i|}{\operatorname{Tr}[\hat{\rho}\,|o_i\rangle\langle o_i|]}.$$

Notice that if $\hat{\rho}$ is consistent with the dataset, the frequency with which o_i occurs in the dataset is proportional to $\operatorname{Tr}[\hat{\rho}\,|o_i\rangle\langle o_i|]$. Therefore $\hat{R}(\hat{\rho}) \propto \sum_i |i\rangle\langle i| = \mathbb{1}$. Starting with $\hat{\rho}_0 \propto \mathbb{1}$, one can iterate the equation

$$\hat{\rho}_{n+1} = \mathcal{N}\hat{R}(\hat{\rho}_n)\hat{\rho}_n\hat{R}(\hat{\rho}_n) \tag{2.7}$$

and by the previously made observation it is clear that this iteration has a fixed point when $\hat{\rho}$ is consistent with the dataset. Here $\mathcal{N}$ is a normalization constant that is recomputed after every iteration. Since $\hat{R}(\hat{\rho})$ is always a positive definite matrix, $\hat{\rho}_n$ is also positive definite for every n. This scheme finds application in theory [17] and experiment [18].

As MLE relies on optimizing or multiplying exponentially large matrices it violates condition ②, and it typically needs a lot of data, violating condition ①. Apart from these scaling issues, MLE has a further issue: it tends to produce zero-eigenvalues, which are never compatible with any dataset [19]. Despite this, MLE is conceptually simple and versatile and is therefore one of the most readily used tomography schemes. It thus plays a crucial role throughout this thesis as a reference scheme.

Bayesian mean estimation

Bayesian approaches, like Bayesian Mean Estimation (BME), go yet another step further. While MLE seeks the state with the highest likelihood, Bayesian approaches take into account all states with high likelihood [19]. BME returns the mean of a posterior distribution

$$\hat{\rho}_{\text{BME}} = \int \hat{\rho} P(\hat{\rho}|\text{Data}) \text{d}\hat{\rho} \quad \text{with} \tag{2.8}$$
$$P(\hat{\rho}|\text{Data}) = \frac{P(\text{Data}|\hat{\rho}) P_0(\hat{\rho})}{P(\text{Data})} \equiv \frac{\mathcal{L}(\hat{\rho}) P_0(\hat{\rho})}{P(\text{Data})}.$$

Here $P(\hat{\rho}|\text{Data})$ is a probability distribution that quantifies how likely a given density matrix is, given the data. As opposed to MLE, this takes into account a prior distribution over all density matrices $P_0(\hat{\rho})$ that can encode any prior knowledge about the state or possible noise on the data. This scheme has the advantage of giving errorbars on its estimate and it does not give zero-eigenvalues as MLE does. However, due to the integral over the space of density matrices in Eq. (2.8), it is even more demanding resource-wise than MLE. It thus also violates condition ②. The sample complexity is difficult to estimate, as it depends on the choice of the prior [19], but for the general case where no assumptions about the target state are made, there is no reason to expect a better scaling than MLE, thus also condition ① is violated.

Schemes for limited sets of observables

There are a few schemes, that do not attempt a full reconstruction of the density matrix, thus violating condition ③. The culmination of these algorithms is the one by Huang, Kueng, and Preskill in [15], built upon works of Aaronson in [14, 20]. This scheme builds a "classical shadow"

$$\text{S}(\hat{\rho}, N) = \left\{ \hat{\rho}_1 = \mathcal{M}^{-1}(\hat{U}_1^\dagger \left|b_1\right\rangle\!\left\langle b_1\right| \hat{U}_1), ..., \hat{\rho}_N = \mathcal{M}^{-1}(\hat{U}_N^\dagger \left|b_N\right\rangle\!\left\langle b_N\right| \hat{U}_N) \right\} \tag{2.9}$$

resulting from computational basis measurements (bit strings $|b_i\rangle$) after applying random unitaries $\hat{U}_i$ to repeated copies of the target state. The mapping of $\hat{\rho}$ to $\hat{U}^\dagger \left|b\right\rangle\!\left\langle b\right| \hat{U}$ is thought of as a quantum channel $\mathcal{M}(\hat{\rho}) = \mathbb{E}\left[\hat{U}^\dagger \left|b_i\right\rangle\!\left\langle b_i\right| \hat{U}\right]$ [15]. The snapshots $\hat{\rho}_i$ are split into K bins, the sum over each bin is computed, resulting in K estimates for the (not necessarily positive) density matrix. Each of these estimates is used to compute a desired expectation value, and the median of these K expectation values is returned. This scheme requires

$$\mathcal{O}\left(\log(M) \max_i ||\hat{O}_i||^2_{\text{shadow}} \epsilon^{-2}\right)$$

measurement samples, to predict M expectation values $\text{Tr}\left[\hat{O}_1\hat{\rho}\right], ..., \text{Tr}\left[\hat{O}_M\hat{\rho}\right]$ up to an error ϵ [15]. For an observable acting on k qubits, the norm $||.||_{\text{shadow}}$ satisfies $||\hat{O}_i||^2_{\text{shadow}} \leq 4^k ||\hat{O}_i||^2_\infty$, the latter being the operator norm. Storing the classical shadows is efficient [15], thus apart from condition ③ none of the other conditions are violated. An example for an experimental application of this scheme can be found in [21].

Variational schemes

The class of variational algorithms is the most important in the context of this thesis, as the main algorithm developed and tested here belongs to this category. These schemes parametrize the density matrix in such a way that makes the numerical representation efficient, as required by property ②. The space in which they search for the target density matrix is therefore restricted, and condition ④ violated. The most notable of these schemes is matrix-product state (MPS) tomography [22–24]. Here the search space is restricted to states that are efficiently representable via MPS, i.e. those where the coefficient matrix $C_{i_1,\ldots,i_N}$ of a state

$$|\psi\rangle = \sum_{i_1,\ldots,i_N} C_{i_1,\ldots,i_N} |i_1\rangle \otimes \ldots \otimes |i_N\rangle$$

can be factored into a matrix-product

$$C_{i_1,\ldots,i_N} = \mathrm{Tr}\left[c_1 \cdot \ldots \cdot c_N\right], \tag{2.10}$$

where the c_i's are matrices of smaller rank than $C_{i_1,\ldots,i_N}$. The states that are efficiently representable can be classified using the entanglement entropy $S(\hat{\rho}) = -\mathrm{Tr}[\hat{\rho}\log\hat{\rho}]$. More precisely, it is well known that a dimension D of the c_i's allows for an entanglement entropy $S(\hat{\rho}_L) = \mathcal{O}(\log D)$ for 1D systems [25], where $\hat{\rho}_L$ is the reduced density matrix of a subsystem of length L. Therefore, MPS can only be used practically for states whose entanglement entropy does not scale with the length of L of the subsystem. These states are said to satisfy (1D) *area-law* scaling of entanglement, as opposed to *volume-law* scaling, where the entanglement would grow with the size of the subsystem. Generalizations to higher dimensions exist, but the contraction of Eq. (2.10) typically turns exponentially hard [25].

Compressed sensing [26, 27] and permutationally invariant tomography [28, 29] are two further common scheme that restrict their search space. The first assumes low-rank density matrices and the latter assumes permutation invariance.

This restriction of the search space is the key point that allows these schemes to achieve greater accuracies with less data than competing algorithms, as information about the target state is encoded *a priori* in the ansatz that is made. This prior information therefore does not need to be re-learned from the data or experiment, allowing for more efficient algorithms w.r.t. condition ①.

The most recently developed class of variational tomography schemes is that based on neural networks. The entire Section 2.4 is dedicated to these schemes.

2.2 POVM formalism

As stated in the introduction, a crucial step to specifying a tomography scheme is specifying the physical measurements that are repeatedly performed on the system. This is where POVMs come into play.

2.2.1 Motivation and definition

Quantum measurements as described in Section 2.1 are typically called *projective* measurements, because the key operators $\hat{P}_i = |o_i\rangle\langle o_i|$ are projection operators in the mathematical sense, satisfying $\hat{P}_i^2 = \hat{P}_i$. Notice that there are three essential properties of the operators $|o_i\rangle\langle o_i|$ that make equations (2.1) and (2.2) meaningful: They are self-adjoint, positive and sum up to the identity matrix. This ensures the expectation values in Eq. (2.1) may be interpreted as probabilities and that the post-measurement state in Eq. (2.2) remains a physical density matrix. Particularly, neither the projective nature nor the orthonormality of the $|o_i\rangle\langle o_i|$ is necessary [13]. We may thus write down a different set of measurement operators $\hat{M}_1, ..., \hat{M}_n$ that satisfy

$$\hat{M}_a = \hat{M}_a^\dagger, \quad \hat{M}_a \geq 0 \quad \text{and} \quad \sum_a \hat{M}_a = \mathbb{1}. \tag{2.11}$$

Such a set of operators is typically called a positive operator-valued measurement or *POVM*. The Born-rule translates simply[1]: the probability of obtaining outcome a is obtained via

$$P(a) = \mathrm{Tr}\left[\hat{M}_a \hat{\rho}\right]. \tag{2.12}$$

The formalism of POVMs becomes most useful in this thesis, if, in addition to the above properties, it is also informationally complete (IC). This means, that the POVM operators $\hat{M}_a$ span the entire space of hermitian operators on the desired Hilbert space. Consequently, every hermitian operator (and thus every density matrix $\hat{\rho}$) has a unique expansion

$$\hat{\rho} = c_a \hat{M}_a.$$

From here on, many explanations closely follow those from [11]. By multiplying with $\hat{M}_b$ and taking the trace, one can compute the coefficients c_a as follows

$$\begin{aligned} P(b) &\equiv \mathrm{Tr}\left[\hat{\rho}\hat{M}_b\right] = c_a \,\mathrm{Tr}\left[\hat{M}_a\hat{M}_b\right] \equiv c_a T_{ab}, \quad \text{where } T_{ab} = \mathrm{Tr}\left[\hat{M}_a\hat{M}_b\right] \\ \Rightarrow \quad c_a &= P(b) T_{ab}^{-1} \\ \Rightarrow \quad \hat{\rho} &= P(b) T_{ab}^{-1} \hat{M}_a. \end{aligned} \tag{2.13}$$

[1]For completeness sake: The post-measurement state is now proportional to $\sqrt{\hat{M}_a}\hat{\rho}\sqrt{\hat{M}_a}$. For details on why the square root appears here, see [13].

Notice that this is the generalization of the linear reconstruction of the density matrix from Eq. (2.4) to a non-orthonormal basis: Knowing the complete POVM probability distribution is equivalent to knowing the density matrix. The mapping between density matrices and POVM distributions given by equations (2.12) and (2.13) is only bijective if one restricts the space of probability distributions. While every density matrix has a corresponding POVM probability distribution, not every probability distribution gives a positive density matrix when inserted into Eq. (2.13). How one can parametrize positive POVM distributions and why this is not very useful in this context is discussed in Appendix B.2.

A key tool in Eq. (2.13) and everything that follows is the matrix T. Its elements quantify how strong any two POVM operators overlap with each other, which is why it is commonly called the *overlap matrix*. For measurements consisting of orthogonal projectors, it would be the identity matrix. Unfortunately, a POVM cannot be both informationally complete and orthogonal [30], i.e. a set of hermitian operators can only ever have at most two of the properties informationally complete, orthogonal and being all positive.

2.2.2 Choice of POVM

There are uncountably many POVMs one could choose from. However, in the context of tomography it makes sense to choose a POVM based on how easy it is to measure in an experiment. Many quantum computers or quantum simulators, that are available today can perform single qubit measurements. This implies that the measurement operators are product operators $\hat{M}_{\mathbf{a}} = \hat{M}^{(1)}_{a_1} \otimes ... \otimes \hat{M}^{(1)}_{a_N}$. It remains to determine the single-qubit POVM $\hat{M}^{(1)}_a$. The most natural measurements on most systems are Pauli-measurements, i.e. randomly choosing an axis x, y or z and projecting the qubit onto it. This measurement has the six outcomes

$$\hat{M}^{(1)}_a \in \left\{ \frac{1}{6} \left|\uparrow_*\right\rangle\!\left\langle\uparrow_*\right| , \frac{1}{6} \left|\downarrow_*\right\rangle\!\left\langle\downarrow_*\right| \;\middle|\; * \in \{x, y, z\} \right\}, \tag{2.14}$$

and is typically called the Pauli-6 POVM. However, these operators are not all linearly independent and result in an overlap matrix T that is not invertible. This is also clear because a single-qubit density matrix has 4 real degrees of freedom (up to normalization), one thus needs 4 (up to normalization) linearly independent operators satisfying the POVM conditions (2.11) to obtain a complete basis. One can fix this issue, by simply grouping three out of these six operators into one, resulting in e.g.

$$\begin{aligned} &\hat{M}^{(1)}_1 = \frac{1}{3} \left|\uparrow_x\right\rangle\!\left\langle\uparrow_x\right| , \quad \hat{M}^{(1)}_2 = \frac{1}{3} \left|\uparrow_y\right\rangle\!\left\langle\uparrow_y\right| , \quad \hat{M}^{(1)}_3 = \frac{1}{3} \left|\uparrow_z\right\rangle\!\left\langle\uparrow_z\right| \\ &\hat{M}^{(1)}_4 = \mathbb{1} - \hat{M}^{(1)}_0 - \hat{M}^{(1)}_1 - \hat{M}^{(1)}_2 . \end{aligned} \tag{2.15}$$

This is typically called the Pauli-4 POVM [11]. Physically, one would still measure the Pauli-6 POVM, but any $\downarrow$ outcome would be considered an $a = 4$ POVM outcome. This grouping of three measurement outcomes into one, effectively means that data is being discarded. This loss can be avoided by using multiple different groupings. There are $2^3 = 8$ possible single-qubit groupings that give valid POVMs, and per qubit this grouping may be different, resulting in 8^N possible POVMs that result from this scheme. However, in all analysis that follows, only the grouping shown above will be used. All comparisons will still be fair, because every tomography receives the same "reduced" dataset.

An alternative to the Pauli POVMs is the product-SIC POVM. Its single particle operators project onto the four states

$$\begin{aligned} |\psi_0\rangle &= |0\rangle\,, & |\psi_1\rangle &= \frac{1}{\sqrt{3}}\,|0\rangle + \sqrt{\frac{2}{3}}\,|1\rangle\,, \\ |\psi_2\rangle &= \frac{1}{\sqrt{3}}\,|0\rangle + \sqrt{\frac{2}{3}}e^{\frac{2\pi \mathrm{i}}{3}}\,|1\rangle\,, & |\psi_3\rangle &= \frac{1}{\sqrt{3}}\,|0\rangle + \sqrt{\frac{2}{3}}e^{\frac{4\pi \mathrm{i}}{3}}\,|1\rangle\,, \end{aligned} \tag{2.16}$$

which form a tetrahedron in the Bloch sphere. For a single qubit, this POVM is symmetric, meaning that its overlap matrix is as close to the identity matrix as possible [30] and that the overlap between any distinct pair of operators is equal. In this sense, this POVM is more "uniform" than the Pauli-4 POVM, for which the $a = 4$ outcome will have the highest probability for most states. While this property might be beneficial, this POVM is much more difficult to measure experimentally, as the outcomes do not easily map to single qubit spin projections. Appendix B.1 explicitly shows how to measure this POVM by coupling to an ancillar qubit. For the stated reasons, only the Pauli-4 POVM will be used from here on, unless stated otherwise.

2.2.3 Observables in the POVM formalism

Observables may also be computed directly within the framework of POVMs, which can be seen by expanding an observable $\hat{O} = o_a \hat{M}_a$ using the POVM operators, computing its expectation value and using Eq. (2.13):

$$\begin{aligned} \mathrm{Tr}\Big[\hat{O}\hat{\rho}\Big] &= \mathrm{Tr}\Big[o_a \hat{M}_a P(b) T^{-1}_{bc} \hat{M}_c\Big] = o_a P(b)\,\mathrm{Tr}\Big[\hat{M}_a \hat{M}_c\Big] T^{-1}_{bc} \\ &= o_a P(b) \delta_{ab} = P(a) o_a. \end{aligned} \tag{2.17}$$

As in Eq. (2.13), the coefficients o_a can be computed using $o_a = T^{-1}_{ab}\,\mathrm{Tr}\Big[\hat{M}_b \hat{O}\Big]$. Similarly, non-linear quantities like the purity may be translated to POVMs:

$$\mathrm{Tr}\big[\hat{\rho}^2\big] = \mathrm{Tr}\Big[P(a) T^{-1}_{ab} \hat{M}_b P(c) T^{-1}_{cd} \hat{M}_d\Big] = \ldots = P(a) P(b) T^{-1}_{ab}.$$

The key point with both of these expressions is that they can be written as expectation values

$$\mathrm{Tr}\left[\hat{O}\hat{\rho}\right] = P(a)o_a = \langle o_a \rangle_{a\sim P(a)} \quad \text{and} \tag{2.18}$$
$$\mathrm{Tr}\left[\hat{\rho}^2\right] = P(a)P(b)T^{-1}_{ab} = \langle T^{-1}_{ab} \rangle_{a\sim P(a), b\sim P(b)}. \tag{2.19}$$

2.2.4 Some details on numerics

Monte Carlo sampling

Expectation values like the above can be approximated numerically using Monte Carlo (MC) techniques [31] as

$$\langle o_a \rangle_{a\sim P(a)} \approx \frac{1}{N_s}\sum_{a\in D} o_a =: E_{N_s}.$$

Here D is a dataset containing N_s samples, each having a probability $P(a)$ of appearing. Such Monte Carlo estimates come with an intrinsic error estimate. The standard deviation of the estimate E_{N_s} may be approximated using $\sigma_{N_s} = \frac{S_{N_s}}{\sqrt{N_s}}$ where S_{N_s} satisfies [31]

$$S^2_{N_s} = \frac{1}{N_s}\sum_{a\in D} o_a^2 - E^2_{N_s}. \tag{2.20}$$

Monte Carlo estimation thus offers a means of systematically estimating quantities that might otherwise be intractable to compute. However, not all observables can be efficiently evaluated this way. Sign problems, i.e. situations where large positive and negative portions of a sum have to cancel out, can lead to an exponentially large variance (2.20), and thus hinder the efficient computation of observables. Exactly this happens with purity-estimates such as (2.19) which are therefore not efficiently accessible for more than one or two qubits using this scheme.

Consequences of factorized POVMs

Using a factorized POVM as described in Section 2.2.2 has some nice consequences. All previous and following expression involving the POVM operators may be rewritten by replacing the indices $a \in \{1, ..., 4^N\}$ with multi-indices $\mathbf{a} = (a_1, ..., a_N) \in \{1, 2, 3, 4\}^{\times N}$, where an index a_i corresponds to qubit i. The trace property $\mathrm{Tr}[A \otimes B] = \mathrm{Tr}[A]\,\mathrm{Tr}[B]$ implies, that the overlap matrix T factorizes as $T_{\mathbf{ab}} = T_{a_1 b_1} \cdot ... \cdot T_{a_N b_N}$ and the same goes for its inverse. A product state $\hat{\rho} = \hat{\rho}_1 \otimes \hat{\rho}_2$ turns into a product distribution $P(\mathbf{a}) = P(\mathbf{a}_1)P(\mathbf{a}_2)$, which can be easily seen from the Born rule (2.12) and the stated trace property. Taking the partial trace becomes very easy on the POVM distribution, e.g.

$$\mathrm{Tr}_{\text{Qubits }1,3,5}[\hat{\rho}] \leftrightarrow \sum_{a_1,a_3,a_5} P_{a_1,\ldots,a_N}.$$

Furthermore, local observables, i.e. those acting only on a subsystem, also remain local. E.g. for an observable $\hat{O}^{1,2}$ acting on qubits 1 and 2

$$\mathrm{Tr}\left[\hat{O}^{1,2} \otimes \mathbb{1}^{3,\ldots,N} \hat{\rho}\right] = \ldots = o_{a_1,a_2} P(a_1, a_2),$$

i.e. local observables only require the knowledge of a marginal of the POVM distribution. For more details on this, as well as nice representations of these equations using tensor diagrams, see [11].

All of these consequences of the factorized POVM mean that many quantities, such as most local observables can be extracted efficiently, even from exponentially large state representations, as long as samples from the corresponding POVM distribution are available.

2.2.5 Simulating POVM measurements

By virtue of the Born rule (2.12), simulating measurements of a POVM on a state $\hat{\rho}$ amounts to generating samples from the POVM distribution $P(a_1, \ldots, a_N)$ corresponding to the target state $\hat{\rho}$. Since $P(a_1, \ldots, a_N)$ is a discrete, non-trivial, high-dimensional probability distribution, one has to resort to Markov Chains to generate samples from it. The procedure for doing so, goes as follows [32]:

1. Generate an arbitrary initial sample $\mathbf{a}$ and compute its probability $P(\mathbf{a})$.
2. Randomly change one of the components of the sample e.g. $\mathbf{a} = (a_1, \ldots, a_i, \ldots, a_N) \to (a_1, \ldots, a_i', \ldots, a_N) = \mathbf{a}'$ and compute the POVM probability $P(\mathbf{a}')$.
3. Draw a random number r uniformly between 0 and 1.
4. Return $\mathbf{a}'$ and replace $\mathbf{a}$ with $\mathbf{a}'$ if $r \leq \frac{P(\mathbf{a}')}{P(\mathbf{a})}$. Otherwise return $\mathbf{a}$.
5. Go to step 2. until sufficiently many samples have been returned.

This algorithm creates highly correlated samples, as only one component of $\mathbf{a}$ is updated per step. To remedy this, one has to discard most samples and only keep the sample after every s steps. The stride s is a hyperparameter. Also, the first few samples will generally have very small probabilities and therefore also have to be discarded. This is called burn-in [32].

2.3 Neural Networks

Neural networks (NNs) are the driving force behind many advances in modern computer science [33], robotics [34], physics [6], biology [4], medicine [35], linguistics [5], finance [36] and many more. The immense increase in computational power over the last 30 years has enabled researchers from many fields to develop deep, application specific machine learning architectures, and achieve results that are unmatched by competing approaches.

While neural networks were originally inspired by biological models for the brain [37], explaining the term *neural* network, many recent architectures have little to do with their biological ancestry. In a more modern definition, neural networks would be seen as a class of non-linear, highly parametrizable functions, that fulfil some sort of ”universal approximation theorem” [38] meaning that (in some limit of many parameters) they can approximate any function to arbitrary accuracy.

In this section, the most important neural network architectures will be explained, with a focus on those that are most relevant to this thesis. Some brief details on training neural networks will be covered, and finally the context for neural networks in quantum mechanics will be set.

2.3.1 Types of neural networks

There are a few important classes of neural networks to distinguish. The first classification is that of *generative* and *non-generative* neural networks. Generative neural networks are those that understand the notion of probabilities. Designed with the goal of learning exactly normalized probability distributions, their distinct feature is the ability to generate data according to a learned distribution. The most important example here is the Recurrent Neural Network (RNN). Non-generative models do not have this ability to generate data and can be seen as more general function approximators. The most important examples here are Restricted Boltzmann Machines (RBMs), simple Dense Neural Networks (DNNs) and Convolutional Neural Networks (CNNs). Further information about these architectures, that goes beyond the following introductions can be found in [39].

Restricted Boltzmann Machines

RBMs are the simplest class of neural networks. They consist of a set of visible binary neurons $\mathbf{v}$ which are connected to a set of hidden binary neurons $\mathbf{h}$ via a weight matrix $\mathbf{W}$ [40]. Visible and hidden neurons are subject to biases $\mathbf{b}$ and $\mathbf{c}$. RBMs encode the Boltzmann distribution $p(\mathbf{v}, \mathbf{h}) \propto \exp\left(W_{ij}h_iv_j + b_jv_j + c_ih_i\right)$, where the exponent is typically referred to as ”energy” [40]. After summing over the hidden neurons, the remaining distribution is given as [41]

$$p(\mathbf{v}) \propto \exp\left(b_i v_i + \sum_i \mathrm{softplus}(c_i + W_{ij} v_j)\right),$$

where $\mathrm{softplus}(x) = \log(1 + e^x)$. Sampling can be performed using Gibbs sampling, more on which can be found in [41].

Dense Neural Networks

Dense, or fully-connected neural networks can be thought of as a generalization to RBMs, consisting of multiple layers l of a form reminiscent of RBMs:

$$x_i^{l+1} = f^l(c_i^l + W_{ij}^l x_j^l),$$

where $f^l(x)$ is a non-linear function, $\mathbf{x^l}$ is the input to the l'th layer and $\mathbf{c}^l$ and $\mathbf{W}^l$ are bias and weight parameters respectively. If one considers the entries of $\mathbf{x}^l$ as "neurons", then each neuron from the l'th layer is connected to every neuron on the $(l+1)$'th layer, explaining the term *dense* network.

Recurrent Neural Networks

RNNs are commonly used to learn probability distributions. They make use of the fact that any multivariate probability distribution may be factored into a product of conditional probability distributions

$$p(x_1, x_2, ..., x_N) = p(x_1) \cdot p(x_2|x_1) \cdot ... \cdot p(x_N|x_{N-1}, ..., x_1). \tag{2.21}$$

Networks that make use of this fact are often called *autoregressive*. A RNN consists of a parametrizable, non-linear function, that takes a hidden state $\mathbf{h}$, as well as a single input component x_{i-1} and returns a new hidden state $\mathbf{h}'$ and a probability distribution for the next component $p(x_i)$. By sequentially sampling the components x_i and simultaneously iterating on the hidden state $\mathbf{h}$, probability distributions may be efficiently approximated [42].

Convolutional Neural Networks

CNNs are designed to deal with spatially correlated data, such as images. Typically, they are fed two dimensional images, but 1D and 2D CNNs are conceptually identical. CNNs perform iterated convolutions (here shown for the 1D case)

$$(\mathrm{conv}(\mathbf{x};\ \mathbf{k}, b))_i = f\left(b + \sum_{j=0}^{|\mathbf{k}|-1} k_j x_{j+i}\right), \quad \text{for } i \in \{0, ..., |\mathbf{x}| - |\mathbf{k}|\} \tag{2.22}$$

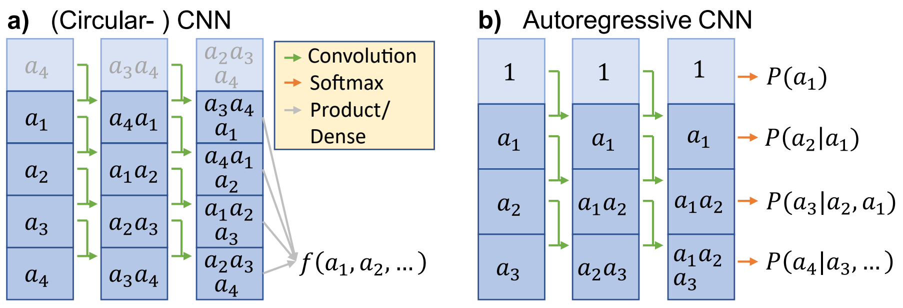

Figure 2.1: Schematic representation of the standard and autoregressive CNN, both for an input length of N=4.

of the input data $\mathbf{x}$ with a so-called kernel $\mathbf{k}$. Here the length of a vector $\mathbf{v}$ is denoted by $|\mathbf{v}|$, b is a bias and f^l is again a non-linear "activation" function acting element-wise on the results of the convolution. This can be thought of as taking dot-products of the kernel and translated section of the input vector. A graphical representation of this architecture can be seen in Fig. 2.1a. For a 2D CNN, $\mathbf{k}$ would be a matrix, and Eq. (2.22) would compute dot products between this matrix and translated submatrices of the then two-dimensional input $\mathbf{x}$.

Typically, multiple kernels are used per layer l resulting in multiple intermediate representations $\mathbf{x}^{l,m}$. The latter would then be computed via

$$\mathbf{x}^{l+1,m} = \sum_n \mathrm{conv}(\mathbf{x}^{l,n};\ \mathbf{k}^{l,m,n}, b^{l,m}). \tag{2.23}$$

This "dense"-like structure on kernel-level is *not* visible in Fig. 2.1. The axis indexed by m here is often called the *feature* dimension.

As is evident from Eq. (2.22), the dimension of interest of $\mathbf{x}$ shrinks with each convolution. To prevent this, one can artificially increase the size of this dimension by applying *padding* before performing the convolution. Common choices are to pad with a constant, or, if the target function satisfies periodic boundary conditions, one can pad with opposing entries along the desired dimension [43]. The latter case is shown in Fig. 2.1a.

In order to convert the last intermediate vector $\mathbf{x}^{L,m}$ on a CNN of depth L into a scalar, one has a few options. Most commonly, a dense layer with scalar output is applied to the feature dimension, resulting in a vector $\mathbf{x}^L$. Now, one can choose to apply another dense layer over the final dimension with exponential as output function f^L, or one can return $e^{\sum_i x_i^L}$, if one seeks to preserve translation invariance [43]. These two cases are referred to as the *Dense*- and *Product*-output layer in Fig. 2.1a.

In this thesis, the distinction between the "CNN" and the circular CNN, i.e. the "CCNN" is made. The prior refers to the standard CNN with constant padding and dense output layer, while the latter refers to the translation-invariant architecture with periodic padding and the product output layer.

For all use-cases throughout this thesis, all layers but the final are kept identical, resulting in three main hyperparameters that may be varied: the number of layers L, the kernel width k and the number of features per layer f. The number of variational parameters is dominated by the kernel tensor $\mathbf{k}^{l,m,n}$, resulting in

$$\text{Num. params} = \mathcal{O}(kLf^2) \tag{2.24}$$

parameters.

Autoregressive Convolutional Neural Networks

The following architecture is heavily inspired by [44]. With three minor modifications, the 1D CNN can be turned into an autoregressive CNN (ARCNN) that can exploit the factorization of probabilities into conditionals, just like the RNN. This enables the ARCNN to encode exactly normalized discrete probability distributions $p(\mathbf{x}),\ \mathbf{x} = (x_1, ..., x_N)$. Here x_i are the discrete components of $\mathbf{x}$, each with a value ranging from 1 to d_l, the local dimension. The modifications that are necessary are visible in Fig. 2.1b. First, the input dimension has to be shifted by one, resulting in the last component of the input vector to not be fed in as an input. Secondly, the padding has to be chosen constant. These modifications ensure, that dependencies on the input components only grow in one direction, just as is necessary for the factorization of probabilities in Eq. (2.21), as visible in Fig. 2.1b.

One can then map the local output of each final cell to a local probability distribution, for example by using a dense layer (with outputs y_i, d_l per site) along the feature dimension, followed by softmax layer which performs the computation

$$y_i \to \frac{e^{y_i}}{\sum_i e^{y_i}}.$$

Therefore, upon given an input of length N, the network therefore gives an output of the form:

$$\begin{array}{cccc}
\text{Site 1:} & \text{Site 2:} & ... & \text{Site } N: \\
P(x_1 = 1) & P(x_2 = 1 \mid x_1) & & P(x_N = 1 \mid x_{N-1}, ..., x_1) \\
P(x_1 = 2) & P(x_2 = 2 \mid x_1) & & P(x_N = 2 \mid x_{N-1}, ..., x_1) \\
... & & & \\
P(x_1 = d_l) & P(x_2 = d_l \mid x_1) & & P(x_N = d_l \mid x_{N-1}, ..., x_1)
\end{array}$$

These are $d_l \cdot N$ individual probabilities, conditioned on the inputs of the network. To evaluate the probability of e.g. $P(x_1 = 2, x_2 = 1, ...)$ one can now select the second entry of the first column $P(x_1 = 2)$, the first entry of the second column $P(x_2 = 1 \mid x_1)$, etc. and finally take the product of these values which results in the full probability, according to Eq. (2.21).

Exact, uncorrelated samples may be drawn by first passing a zero-vector $\mathbf{x} = (0, ..., 0)$ through the network and using the result to sample the first component x_1 of the result. This first component may be passed through the network as $\mathbf{x} = (x_1, 0, ..., 0)$, resulting in the conditional probabilities for the second component x_2 and so on. It therefore requires N passes through the network, to generate one sample.

Unfortunately, this simple extension of the CNN to an autoregressive structure fails for the 2D CNN, as certain regions would end up uncorrelated (see Appendix A.1 for a more detailed explanation). Hence, the ARCNN is limited to 1D systems.

Others

Other architectures such as the Transformer [45, 46], the LSTM [47], invertible neural networks [48], generative adversarial networks (GANs) [17] and variational autoencoders (VAEs) exist and have also found their way into physics applications, but they are not relevant in the context of this thesis.

2.3.2 Training

When it comes to training neural networks, the key quantity is the so called loss function $\mathcal{L}(\boldsymbol{\theta}, \mathbf{D})$. This function returns a scalar, that quantifies the performance of the network. One formulates the task that the network is supposed to learn, such that smaller values of the loss function correspond to a better performance of the network at its given task. The loss function depends on two main quantities. The first being the parameters $\boldsymbol{\theta}$ of the network. This is the object to be learned, i.e. one seeks those values for $\boldsymbol{\theta}$ that minimize the loss. The second input is a dataset $\mathbf{D}$. For a task of e.g. image classification, this dataset might consist of images and their corresponding labels (”Cat”, ”Dog”, ...).

The task of optimizing this loss function w.r.t the parameters $\boldsymbol{\theta}$ is an entire field of research of its own. In the context of this thesis, one relies on the fruits carried by decades of research on optimized training strategies. Standard neural networks, such as those described above, in conjunction with the task of optimizing an analytic loss function, are typically trained using gradient based methods. This means that one computes gradients of the loss function w.r.t the parameters $\boldsymbol{\theta}$, and iteratively updates the latter based on those gradients. In the simplest form, one simply takes steps in the direction opposite the gradients [49]:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \lambda \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_i, \mathbf{D}). \tag{2.25}$$

The step size λ is called the *learning rate*. This technique is called gradient descent [49].

The simplest extension to gradient descent it the *momentum* optimizer. It adjusts the update rule (2.25) by adding a velocity or momentum term [50]:

$$\begin{aligned} \boldsymbol{\theta}_{i+1} &= \boldsymbol{\theta}_i - \mathbf{v}_i, \\ \mathbf{v}_{i+1} &= \gamma \mathbf{v}_i + \lambda \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_i, \mathbf{D}). \end{aligned} \tag{2.26}$$

This can speed up convergence and help avoid local minima in the loss landscape. A damping factor $\lambda \approx 0.9 < 1$ ensures a steady state velocity is reached [50].

The current state of the art optimization algorithm is the ADAM algorithm [51], short for "adaptive moment estimation". It adaptively changes the learning rate λ for every parameter individually, based on all previously seen gradients. The latter allows the algorithm to effectively approximate the second order gradients of the loss function, without computing these explicitly [51]. This tends to increase the speed of convergence yet again, making it one of the most widely used optimization algorithms to date.

Commonly, not the entire dataset $\mathbf{D}$ is used to compute the weight updates (2.25) and (2.26). Rather, the dataset is randomly split into equally sized *batches* $D_1, D_2, ... \subset \mathbf{D}$, and for every weight-update i a new batch is used, e.g. for simple gradient descent (2.25)

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \lambda \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_i, D_i).$$

This has two main reasons: on the one hand it decreases the computational work load per iteration, and on the other hand it introduces stochasticity to the training process, which can help the network to *generalize* [39].

In the context of machine learning, the word *generalization*, used in the previous sentence, refers to scenario where a model trained on some limited dataset acquires knowledge about its underlying structure and is able to make predictions that go beyond the information contained in the data. Achieving this scenario is the ultimate goal most machine learning tasks strive for.

Furthermore, one may add an additional regularization term

$$\mu \cdot ||\boldsymbol{\theta}||^2$$

to the loss function. This term, commonly called *weight decay*, penalizes large network parameters and can further improve generalization performance and help prevent overfitting [39]. The importance of this term can be influenced by changing the weight decay coefficient μ.

2.4 Neural network quantum state tomography

2.4.1 Neural network quantum states and literature

In Sec. 2.1.2 some variational algorithms such as matrix product states were introduced as a means of representing quantum states of large systems efficiently on classical, resource-limited hardware. In 2017, Carleo and Troyer introduced neural networks as an entirely new variational approach [8]. As stated in Sec. 2.3.1, neural networks are an incredibly successful and efficient class of universal function approximators, that offer a great representational power, even with limited numbers of parameters. Motivated by this, the Neural Network Quantum State (NQS) was conceived. The idea is simple - in analogy to MPS (2.10), the coefficient tensor $C_{i_1,...,i_N}$ of a complex many-body state

$$|\psi\rangle = \sum_{i_1,...,i_N} C_{i_1,...,i_N} |i_1\rangle \otimes ... \otimes |i_N\rangle$$

is replaced by a neural network. This neural network accepts a configuration of single particle states $(i_1, ..., i_N)$ and returns the corresponding wave function amplitude $C_{i_1,...,i_N}$. Naturally, this is a complex quantity and multiple approaches have been studied for handling this fact. The original authors use a neural network with complex parameters [8]. This has the issue, that not all neural network frameworks can easily handle and differentiate w.r.t. complex network parameters [40]. A possible solution is to employ two different neural networks, one for the amplitude, and one for the phase of the wavefunction [40, 52]. Using this ansatz, a generative model can be used for the amplitudes, allowing spin configurations to be sampled.

A more recent, less adopted suggestion is to use one network to learn real, basis-dependent amplitudes and use a second network to adjust the parameters of the first, in order to represent the wave function in different bases [53].

Soon questions were asked on how to represent mixed quantum states with neural networks. Here, the community has settled on three main approaches. The first is a purification ansatz [54], that phrased the mixed quantum state as a pure state of a larger system, thus mapping the problem to a known one. This approach has been applied in [55–59]. Others have adopted approaches that directly operate on the elements of the density matrix themselves [17, 60, 61], at the cost of exponential scaling. A popular approach, pioneered by Carrasquilla et al. in [11], is to represent the mixed state using its POVM probability distribution. This has the advantage of allowing for an entirely probabilistic formulation, enabling efficient sampling of observables, all by using only one, real-valued network. The drawback is, of course, that positivity is not ensured, as explained in Sec. 2.2. This approach has found adoption in [46, 59].

Further approaches based on neural networks in combination with multiscale entanglement renormalization [62] and renormalization group methods [48] have been developed, while others seek to modify standard network architectures, to better encode desirable properties of specific target states and e.g. enhance the ability to capture specific correlators [63, 64].

These NN approaches have been applied to various typical problems in quantum mechanics, which include ground state search [8, 42, 44], even for frustrated spin systems [43], the simulation of quantum circuits [46] and typical quantum algorithms such as the Quantum Approximate Optimization algorithm [65], time evolution [8, 66–68], entanglement detection [69, 70], and of course state tomography [11, 17, 52, 53, 60, 71–74], with experimental implementation of the latter in [59, 75–77].

This literature on neural network assisted quantum state tomography can be classified according to the encoding of the quantum state within the network as described above, and the network architecture used. The pure-state approach with RBMs is used in [52, 59, 71, 75]. RBMs are further used in combination with basis dependent neural networks [53], low-rank approximations [73] and the purification ansatz [59, 73]. Tomography resulting in full density matrices has been carried out with CNNs [60], GANs [17] and RNNs [61]. The POVM approach has been applied to RNNs [11], RBMs [59] and Transformers [72].

The theory side has also made progress since NQS were first envisioned. It has been shown, that RBMs and MPS are equivalent in the sense that they can be transformed into each other [78] and the entanglement properties of RNNs and CNNs have been understood (at least for pure states), framing CNNs as a *generalization* to MPS [10], capable of encoding volume-law entanglement [9] unlike MPS.

For recent reviews on the field of NQS and their applications, see [79] and [80].

2.4.2 Description of the tomography scheme

With most details out of the way, one can now put all building blocks together, and describe the tomography scheme that is of interest in this thesis.

The big picture idea is the following: Use the POVM formalism as a state representation. This turns the task of finding the target state density matrix into the task of ”density estimation”, i.e. the reconstruction of a probability distribution from measured samples. Then, parametrize the POVM distribution using a neural network, and determine their variational parameters using the maximum likelihood method, i.e. by optimizing the same target as is done for MLE tomography in Section 2.1.2. As networks, use the two aforementioned CNN architectures, motivated by their previously mentioned superiority in the context of NQS [9, 10].

Notice, that this combination of network architecture and state encoding scheme is itself novel, as it is absent from the above literature list.

Now follows a more detailed explanation of the tomography scheme: Data from N_s POVM measurements is considered to be available in the form $\mathbf{D} = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_{N_s}\}$, where $\mathbf{a}_i = (a_1^i, a_2^i, ..., a_N^i)$ are independent tuples of single-site measurement outcomes a_j^i on an N qubit system. The ground truth POVM distribution $P(a_1, ..., a_N)$ underlying the target state is approximated by the CNN, i.e. the output of the CNN is interpreted as a variational probability distribution $P_{\boldsymbol{\theta}}(a_1, ..., a_N)$.

For the ARCNN, these probabilities may be directly evaluated as described in Sec. 2.3.1. The application of standard CNN, however, is not as straightforward. Its outputs are positive scalars $\mathrm{CNN}_{\boldsymbol{\theta}}(a_1, ..., a_N)$, which are not normalized to form a probability distribution. Thus, a manual normalization has to be introduced. Computing this exactly as

$$\mathcal{N} := \sum_{a_1,...,a_N} \mathrm{CNN}_{\boldsymbol{\theta}}(a_1, ..., a_N)$$

is not an option, since this sum contains exponentially many terms. It can be approximated efficiently using a Monte Carlo estimate, as in Sec. 2.2.4 by summing over a smaller batch containing $N_b \ll 4^N$ uniformly generated POVM samples:

$$\mathcal{N}_{\boldsymbol{\theta}}^{\mathrm{MC}} \approx \frac{4^N}{N_b} \sum_{\mathbf{a}_i \in \text{ Batch}} \mathrm{CNN}_{\boldsymbol{\theta}}(\mathbf{a}_i). \tag{2.27}$$

This quantity has to be re-computed after every training step. Using this, the CNN probability distribution may be written down as

$$P_{\boldsymbol{\theta}}(a_1, ..., a_N) = \frac{1}{\mathcal{N}_{\boldsymbol{\theta}}^{\mathrm{MC}}} \mathrm{CNN}_{\boldsymbol{\theta}}(a_1, ..., a_N). \tag{2.28}$$

The task now is to find the parameters $\boldsymbol{\theta}$ of the CNN, such that the CNN best approximates P, i.e.

$$P_{\boldsymbol{\theta}}(a_1, ..., a_N) \approx P(a_1, ..., a_N). \tag{2.29}$$

This approximation is found by maximizing the probability that the network reproduces the dataset; the same ansatz as used for MLE in Sec. 2.1. Therefore, the task is to maximize the likelihood function w.r.t. network parameters

$$P_{\boldsymbol{\theta}}(a_1, ..., a_N) \approx P(a_1, ..., a_N) \quad \leftrightarrow \quad \boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{\mathrm{argmax}} \prod_{\mathbf{a}_i \in \text{Dataset}} P_{\boldsymbol{\theta}}(\mathbf{a}_i).$$

To prevent numerical issues due to exponentially small numbers, the logarithm of the above function is taken, which does not shift the position of the maximum. Flipping the sign of the likelihood function turns the maximization into a minimization problem. Thus, the loss function used for training the network becomes

$$L(\boldsymbol{\theta}, \mathbf{D}) = \sum_{\mathbf{a}_i \in \mathbf{D}} \log P_\theta(\mathbf{a}_i). \tag{2.30}$$

This is a loss function predestined to be optimized with the ADAM optimizer from Section 2.3.2, which may be trained until Eq. (2.30) converges.

Here, the vector of single site outcomes a_i is treated as a 1D or 2D image (depending on system geometry). Neighbouring outcomes a_i and a_{i+1} are physically close to each other, and thus probably correlated, just as is common with neighbouring pixels of an image. This further motivates the use of a local neural network architecture such as the CNN. However, unlike with real images, the pixel values do not represent a magnitude. For an image, with 8-bit pixel values ranging from 0-255, a value of e.g. 100 is "close" to a value of e.g. 101, meaning that these pixel values may be treated similarly. However, for POVMs with local outcomes ranging from 0-3 (or 0-5 for the Pauli-6 POVM), this property does not hold, since measuring e.g. POVM outcome 2 is not at all "close" POVM outcome 3. To circumvent this problem, one has to encode the POVM outcomes in some different way. Common choices are the binary encoding, mapping

$$0 \to (0\ 0),\ 1 \to (0\ 1),\ 2 \to (1\ 0),\ 3 \to (1\ 1),$$

or the one-hot encoding

$$0 \to (0\ 0\ 0\ 1),\ 1 \to (0\ 0\ 1\ 0),\ 2 \to (0\ 1\ 0\ 0),\ 3 \to (1\ 0\ 0\ 0).$$

These mappings effectively increase the input shape by one dimension for the first layer, which can be undone using e.g. a subsequent dense layer along the extra axis. A short comparison of these encoding types can be found in the appendix in Sec. A.3.

After performing a successful tomography, i.e. finding a good approximation to the target distribution, samples may be generated using the neural network, either directly for the ARCNN as described in Sec. 2.3.1, or using Markov Chains 2.2.5 for the standard CNN. Due to the locality preserving nature of the POVM formalism all of this remains efficient, and local observables can be sampled from the neural network, allowing the prediction of new observables.

As stated in Sec. 2.2, the use of the POVM formalism has the consequence that the reconstructed density matrix is not necessarily positive definite. A non-positive density matrix will lead to the violation of quantum bounds on observables, such as $-1 \leq \langle \hat{\sigma}^z \rangle \leq 1$ or $\mathrm{Tr}[\hat{\rho}^2] < 1$. Therefore, one can generally not expect this tomography scheme to yield valid estimates for every possible observable. However, this is no reason to expect no advantage from this scheme at all: Since the data that is used for training the ansatz stems from a positive density matrix, the training process will steer the internally represented density matrix towards positive eigenvalues. Therefore, many observables will still be captured accurately and thus even follow their theoretical bounds.

In summary, this is a tomography scheme, that is

- variational, just like MPS is variational. This allows the exponentially large state to be compressed into a tractable number of parameters, making storage of the state efficient. Furthermore, this compressed representation is what can allow the state to be learned from fewer samples, as required for property ①. This is a direct trade-off with property ④, as the representable state space is necessarily restricted.

- probabilistic and local, making observables easy to approximate using Monte Carlo methods, as required for property ②.

- real, allowing for standard neural network packages to be applied.

- expressive, since CNNs are known to be capable of representing quantum mechanically interesting states.

From these considerations alone it is clear, that this scheme is sub-exponential in classical post-processing and therefore fulfils condition ② from the criteria given in Sec. 2.1.1. To see how this scheme fares regarding the other criteria, it has to be tested on some real-world tomography scenarios, which is done in the following chapters.

2.4.3 Details on implementation

This scheme is implemented in Python, using the Jax [81] framework for automatic differentiation and GPU compatibility as well as the flax library [82] for neural networks and optimization algorithms. The Adam optimization algorithm is contained in the flax library, which in turn makes use of jax to automatically differentiate neural networks w.r.t. their parameters efficiently on GPUs.

The used hyperparameters are listed in Table B.1 in the appendix. For the (C)CNN, the normalization batch size is chosen equal to the batch size. Since the further implementation of the neural networks follows standard designs, no additional details are given at this point.

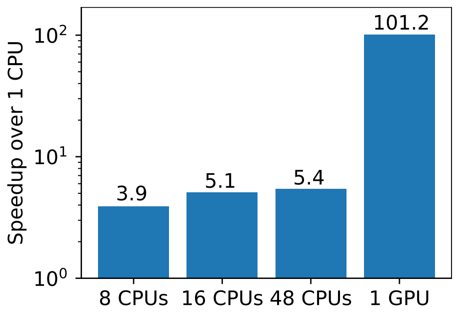

Figure 2.2: The speedup of training the neural networks on multiple `Intel Xeon 6252 Gold` CPUs on the `bwForCluster JUSTUS2` and a single `NVIDIA V100` GPU as compared to a single CPU.

During this thesis, most of the mathematics behind POVMs from Section 2.2 as well as Markov chains for sample generation were implemented in JAX. This allows for a tight integration of POVMs with the neural network libraries and a GPU compatible interface. Computation of POVM probabilities (2.12) requires the execution of large tensor contractions, i.e. generalized matrix multiplications, which is one of the tasks GPUs excel at. As stated in Sec. 2.2.5, generating samples using Markov chains is quite expensive numerically, as samples are generated sequentially and most samples have to be discarded. One may remedy this by running multiple independent Markov chains in parallel. The JAX library can also help with this, as it provides means to *vectorize* code, making the concurrent execution of multiple Markov chains possible using the massively parallel architecture of GPUs. A single `NVIDIA A100` GPU at the Juwels Booster Module Cluster [83] allows the concurrent execution of an excess of 100 Markov chains, each performing the tensor contractions for the evaluation of 16 qubit POVM probabilities. For 16 qubit Markov Chains, a stride of 32 and a burn-in of 10k samples is used.

The speedup that is achieved by the use of GPUs while training a 10 qubit state in shown in Fig. 2.2.

2.5 Ising ground states

For testing tomography schemes, one needs a class of states that can be used as meaningful test subjects. One class of states that finds its use throughout the thesis are ground states of the Transverse Field Ising Model (TFIM). The Ising Hamiltonian is defined as

$$\hat{H} = -J \sum_{\langle i,j \rangle} \hat{\sigma}_i^z \hat{\sigma}_j^z - B \sum_i \hat{\sigma}_i^x. \tag{2.31}$$

Its ground states are an interesting class of states to consider, not only because the Ising model is *the* prototypical spin-Hamiltonian, being subject of both experimental and theoretical studies. It is one of the simplest Hamiltonians one can write down, that is not trivial but still has an analytical solution. Furthermore, by varying a single parameter J/B, a wide variety of states with a rich range of properties may be achieved.

In the "small coupling" limit $J/B \ll 1$ the Hamiltonian is dominated by the field term $\sum_i \hat{\sigma}_i^x$, which aligns the spins along the x axis. In this limit, the ground state is thus a product state $|\psi\rangle = |+\ldots+\rangle$, which is uncorrelated and therefore "simple". Here, $|+\rangle$ is the $\uparrow$ eigenstate of the Pauli-x operator $\hat{\sigma}^x$.

In the opposite "strong coupling" limit, the coupling term $\sum_{\langle i,j\rangle} \hat{\sigma}_i^z \hat{\sigma}_j^z$ dominates, which gives the smallest energy when neighbouring spins point in the same direction along the z axis. Thus, for no external field B the ground state is two-fold degenerate, spanned by the states $|\uparrow\ldots\uparrow\rangle$ and $|\downarrow\ldots\downarrow\rangle$. For small fields and finite system sizes, the ground state is approximated well by the equal superposition of both previously given states, resulting in the so called "GHZ-state" $|\mathrm{GHZ}\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow\ldots\uparrow\rangle + |\downarrow\ldots\downarrow\rangle\right)$ [84]. This is a highly correlated, maximally entangled state and thus an interesting target state for neural network applications. In between, at coupling strength equal to the external field $J/B = 1$ lies the so-called critical state. It marks the transition point between the above two phases and is characterized by diverging correlation length, which makes this state interesting as well.

If one adds periodic boundary conditions (i.e. by adding a coupling term to the first sum that couples the first spin to the last spin), the model will produce translation-invariant ground states. This can be useful for testing the addition of such symmetries to the neural network ansatz.

The ground states are computed by exactly diagonalizing the Hamiltonian using the *Lanczos* method [85], which is implemented in the SciPy library [86], based on the ARPACK software [87].

3 Results

The results of this thesis are split into three parts. The first, Sec. 3.1, attempts to answer questions regarding the representability of quantum states within the neural network. What states are efficiently representable? Are there physical quantities, that influence whether a state can be represented using the network? Studies done in this section can be thought of as the "infinite data" case, because it is assumed that the target POVM distribution is known exactly.

In the second part of the results, Sec. 3.2, the tomography scheme is tested under conditions that better match those encountered on real experiments. Here finite dataset sizes are taken into account, and comparisons to competing tomography schemes are made.

In the third Section 3.3, the tomography scheme is applied to experimental data from measurements for [18], that was kindly supplied by Christian Roos.

Due to the chronological order in which the following data was collected, not all results from the first part will be applicable directly to the second, and not every experiment from the "finite data" section has a counterpart in the "infinite data" section. Furthermore, the ARCNN architecture was not yet known to the author, at the time when the experimental data was analysed, hence here the analysis is restricted to the CNN.

Unless stated otherwise, all error bars shown result from performing independent repetitions of runs or experiments.

3.1 Representability - Infinite data

Asking whether a given state is representable in a neural network is a different question from asking whether it is efficiently trainable from a (experimentally limited) dataset. In this section, an attempt is made to gain insight into the first question. To this end, it is assumed that the exact POVM distribution of the target state is known, corresponding to an infinitely large dataset.

In this situation, the Kullback-Leibler divergence

$$D_{KL} = \sum_{\mathbf{a}} P_{\text{Truth}}(\mathbf{a}) \log \frac{P_{\text{NN}}(\mathbf{a})}{P_{\text{Truth}}(\mathbf{a})} = \sum_{\mathbf{a}} P_{\text{Truth}}(\mathbf{a}) \log P_{\text{NN}}(\mathbf{a}) + \text{const.} \quad (3.1)$$

between the POVM distribution P_{truth} and the network encoded distribution can be used as a loss function for training, which is equivalent to the loss function in Sec. 2.4.2 in the infinite data limit. The "dataset" in this case corresponds to the set of all possible POVM outcomes. The sum in Eq. (3.1) may be split into batches, treating each batch as a Monte Carlo estimate.

After training, i.e. when the loss function has converged, its final value may be used as a quantifier for how well the neural network approximates the target distribution. This final value of the loss function is referred to as "final D_{KL}" or "residual D_{KL}". A smaller final D_{KL} implies a better approximation of the target state.

Analysing the representability of the neural networks can be approached from two different angles. The first is to fix a network architecture and apply it to a wide variety of quantum states. Then one may look for similarities between those states, that are represented with high accuracies. This scheme is applied in Sec. 3.1.1.

Alternatively one may fix a quantum state of interest and optimize the network architecture, to gain insight into what network size is necessary to represent said state. This scheme is applied in Section 3.1.2.

3.1.1 Random states

In this section, an attempt of classifying the CNN-representable states is made, by fixing a network architecture and training it on several thousand *random* states. The random states considered here are of the form

$$\hat{\rho}_{\text{random}} = p \left|\psi_{\text{Haar}}\right\rangle\!\left\langle\psi_{\text{Haar}}\right| + (1-p)\hat{\rho}_{\text{random mixed}}. \tag{3.2}$$

Here $p \in [0, 1]$ is a uniformly chosen random number, $|\psi_{\text{Haar}}\rangle$ is a *Haar* random pure state [88] and $\hat{\rho}_{\text{random mixed}}$ is a random density matrix generated according to [89], that has a high probability of having full rank. While this parametrization only covers mixtures of pure and highly mixed states, this ensures that states of any purity are covered. These states are generated for $N = 6$ qubits and a CNN with $f = 6$ features per layer, a kernel width of $k = 6$ and $L = 2$ layers is chosen as a test subject. This is a network architecture with 301 variational parameters used for approximating a POVM distribution with $4^N = 4096$ probabilities. It is therefore reasonable to assume that there will be some variation in the quality of approximation. The latter is quantified by the KL-divergence D_{KL} (3.1) between the POVM distribution of the random state and the network distribution after convergence. One can then look for correlations between D_{KL} and various properties of the random target state. The quantities that were considered are listed in Table 3.1. "Histogram density" here refers to the fraction of non-zero bins when grouping the POVM probabilities into a histogram. This therefore quantifies how many "different probabilities" appear in the POVM distribution: for a uniform

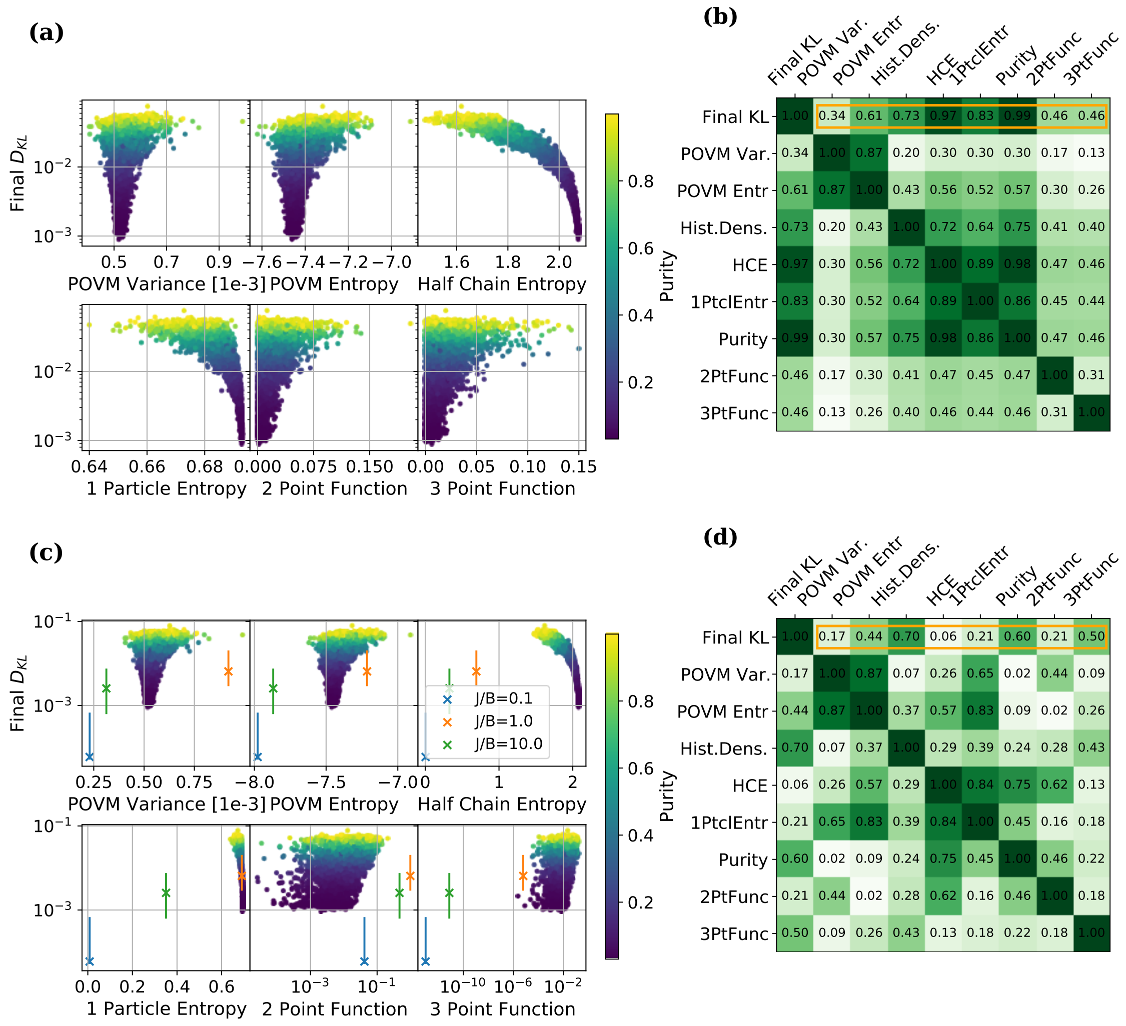

Figure 3.1: (a) The representability of 3000 random quantum states (3.2), expressed in terms of the final D_{KL}, is plotted against quantities from Table 3.1. **(b)** Correlation matrix corresponding to (a). Only the absolute value is shown. Coefficients highlighted in orange are the relevant influences on the representability. **(c/d)** Same data shown as in (a/b) but including Ising states with three different coupling strengths J/B with a weight of 3 % of the entire dataset each.

distribution this quantity would be almost zero, while a value of 1 would be taken if every outcome had a unique probability.

The resulting correlation plots can be seen in Figure 3.1(a) and 3.1(b). On first sight, this might suggest some interesting correlations: POVM entropy, histogram density, half-chain entropy, single site entropy and purity all seem to strongly

	Name	Definition
Physical properties	Half-chain Entropy	$\mathrm{Tr}[\hat{\rho}' \log \hat{\rho}']$ for $\hat{\rho}' = -\mathrm{Tr}_{1,\dots,N/2}[\hat{\rho}]$
	1-Ptcl. Entropy	$\frac{1}{N} \sum\limits_{i=1}^{N} \mathrm{Tr}[\hat{\rho}_i \log \hat{\rho}_i]$ for $\hat{\rho}_i = \mathrm{Tr}_{\{1,\dots,N\} \setminus i}[\hat{\rho}]$
	2-Point funct.	$\frac{1}{N} \sum_{i=1}^{N} \left\langle \hat{\sigma}_i^z \hat{\sigma}_{i+1}^z \right\rangle$
	3-Point funct.	$\frac{1}{N} \sum_{i=1}^{N} \left\langle \hat{\sigma}_i^z \hat{\sigma}_{i+1}^z \hat{\sigma}_{i+2}^z \right\rangle$
	Purity	$\mathrm{Tr}[\hat{\rho}^2]$
POVM distribution properties	Variance	$\langle P_a^2 \rangle - \langle P_a \rangle^2 = \sum_a P_a^2 - 1$
	Entropy	$\sum_a P_a \log P_a$
	"Histogram Density"	see text

Table 3.1: Tracked quantities in Fig. 3.1 for analysing the representability of random quantum states in the CNN.

correlate with the approximation quality D_{KL}. A look at the correlation matrix in Fig. 3.1b, however, reveals that most of the stated quantities also strongly correlate with each other. The purity in particular shows almost identical correlations with all other quantities, hinting at the fact, that *it* might be the quantity with most influence over the approximation quality. In fact, purity is also the property, that correlates strongest with D_{KL}.

However, most of these statements should be taken with a grain of salt, which becomes evident when including some typical pure ground states into the dataset. Figures 3.1(c) and 3.1(d) show the same types of correlation plots, but include three typical Ising ground states, one from each phase and one at the critical point, each with a weight of $\approx 3\%$ in the entire set of considered states. The inclusion of these states gives two important insights: First of all, the random states are all very similar. For many of the quantities of interest, the 3 Ising states show a much wider variety in properties than the 3000 random states. This suggests that the random states might not be the most interesting class of states and the physically interesting states are somewhat "special". Note, that this also gives hope that neural networks might be able to capture these "special" states and find efficient parametrizations for these. Given that these simple ground states offer a much wider variety of properties, random states will not be used any more, from here on.

The second new insight is that most correlations that seemed to be present for the random states disappear, when including the Ising states. Only two quantities out of those analysed remain with a correlation coefficient of greater than ½, namely the histogram density and the purity. It is intuitively clear, that the histogram density correlates strongly with representability: a low histogram density means (by definition) that many POVM outcomes have same or similar probabilities. Therefore "generalization" in this case is much simpler, only requiring the knowledge of which outcomes have the same probabilities. This leaves the purity as the main result of this analysis, allowing for the following statement: *States with higher purity tend to be harder to represent.* This statement also holds for Ising ground states themselves, which become easier to represent if dephasing noise is added to the density matrix. This is shown in the appendix in Fig. A.2. Unfortunately, this is not a very helpful statement in the context of this thesis, as most candidate states for tomography will probably be almost pure, with only a small degree of mixing.

3.1.2 Representability - Ising ground states

CNN Phenomenology

In the following two sections, the representability of the CNN is quantified by fixing the translation-invariant Ising ground states as a class of quantum states and analysing the network architecture that is necessary to represent it accurately.

As a first test, a small CNN architecture for four spins is fixed, and the ratio J/B is scanned from 0 to 5. The results may be seen in Figure 3.2, and are representative of all further tests performed using the CNN: The states for small physical coupling, i.e. $J/B < 1$ are easy to represent in the network, meaning that the accuracy after training is high. For strong coupling $J/B > 1$, the network performs worse. This matches intuition nicely: The product-like states at small coupling result in a product-like POVM distribution which is easily approximated using a CNN due to the exponential output function. The stronger the coupling, the more correlated the state, thus the POVM distribution is more correlated and in turn harder to approximate.

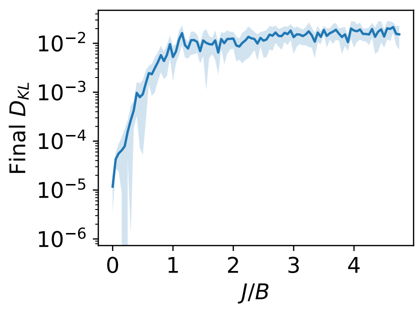

Figure 3.2: The mean D_{KL} to the ground truth target after training is shown for states of the Ising model phase transition, for a four qubit system and a CNN with $f = 4, L = 2$ and $k = 4$.

A more interesting question is that of scalability, i.e. how must the size of the network scale with the number of qubits, in order for the state to be approximated efficiently. To this end, several architectures

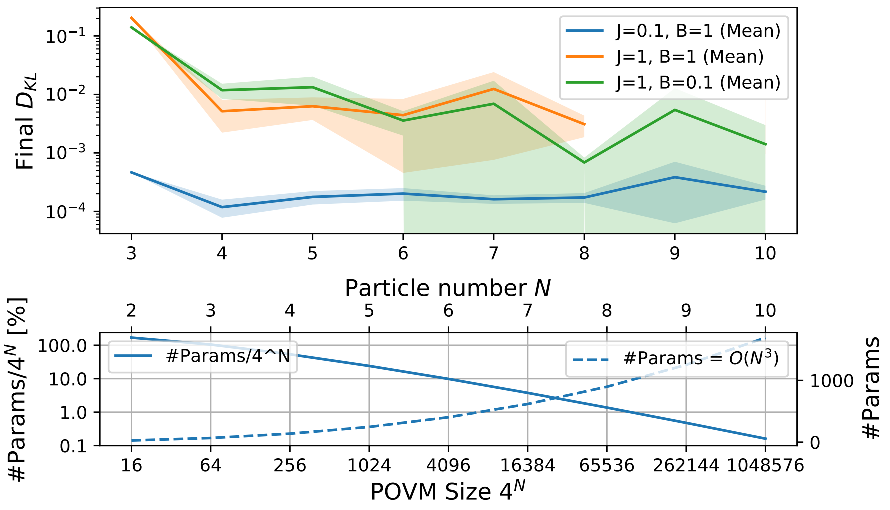

Figure 3.3: Training three typical states from the Ising phase transition for different system sizes, using a CNN scaling of $f = N, k = N/2$ and $L = 4$, resulting in an $\mathcal{O}(N^3)$ scaling in hyperparameters.

are tested for different numbers of qubits N. Hereby it is found, that a scaling of the feature-dimension $f = N$, the kernel width $k = N/2$ and the network depth $L = 4 =$ const. seems to suffice, at least for small N. This can be seen in Fig. 3.3. The D_{KL} is constant or even decreasing in N for all example states from the 3 typical Ising regimes, hinting at a scalable representation. The scaling used in Fig. 3.3 results in a scaling of the number of parameters of $\mathcal{O}(N^3)$ (see Eq. 2.24). Fig. A.3 in the appendix shows a scaling of $\mathcal{O}(N^2)$ that is not sufficient for representing these Ising ground states. For the small coupling regime however, it turns out that even a constant number of parameters suffices to learn the state approximately, as can be seen in the appendix in Fig. A.4.

Notice that in Figures 3.2 and 3.3 the mean and standard deviations over independent runs are shown which is suboptimal for quantities that vary over many orders of magnitude. In this situation a median would have been more appropriate and would show even better results.[1]

It turns out, that this scaling of the CNN *for GHZ states* holds only for the small particle numbers shown so far. In fact, the training of the CNN will fail entirely for GHZ states in larger systems. Here one has to resort to the ARCNN. This is detailed further in Appendix A.4, with a possible explanation based on the sampling of the normalization given in Appendix A.5.

[1]This is because the mean will be dominated by the largest contributions, i.e. the mean D_{KL} is close to the maximum D_{KL} of repeated runs.

CNN and ARCNN systematics

These results so far are of course very phenomenological in nature. One can gain more insight, by looking at the propagation of information in the networks itself using Fig. 2.1. For each layer of the CNN, the dependence of a particular input site is propagated to the next $k-1$ neighbouring sites. Thus, the maximum physical distance correlations are correctly propagated for the (C)CNN with product output layer is given by

$$d_{\max}^{\mathrm{CNN}} = (k-1) \cdot L \,. \tag{3.3}$$

For the CNN with dense output layer this bound is not valid, as the final dense layer mixes all physical sites, similar to an RBM. For the ARCNN, the same logic for the spreading of dependencies applies, but no input is required to sample outcomes of the first site, resulting in a maximum correlation length that is one larger than for the CNN

$$d_{\max}^{\mathrm{ARCNN}} = (k-1) \cdot L + 1 \,. \tag{3.4}$$

The validity of both bounds can be empirically verified by e.g. tracking the value of a zz correlator as a function of physical distance for different network architectures, as is done in Fig. 3.4 and 3.5. Here the ground states of the translation invariant Ising model are trained for critical coupling and strong coupling. These states being translation invariant explains why the true correlators show a reflection symmetry at distances of half the system size, since going a distance d to the right is the same as going a distance of $N-d$ to the left. The CCNN, as evident from its name, has this translation invariance enforced in its architecture, explaining why correlations are captured correctly again for distances greater than $N-d_{\max}$.

In general, correlations beyond $d_{\max}$ are not captured correctly, due to the arguments given above. However, notice that qubits separated by distances greater than $d_{\max}$ are not necessarily uncorrelated. In fact, after only one convolution with a kernel width of two, in principle all-to-all correlations are possible: following the structure of Fig. 2.1, after one such convolution all neighbouring pairs of sites share a dependency and are thus correlated: i.e. (a_1, a_2) are correlated, (a_2, a_3) are correlated, and so are (a_3, a_4) and so on. But since both a_3 and a_1 share a dependency with a_2, also a_1 and a_3 are correlated. Following this logic, even a_1 and a_4 can be correlated and so can all other sites, even if they do not share a mutual dependency on some other site. In the extreme case this effect can help the network to generalize, like for the GHZ state in Fig. 3.4. Here, even some networks with $d_{\max} < N$ capture all shown correlation correctly. This can be understood by the rather simple structure of the GHZ state: if one knows all correlations between one pair of qubits, one knows all correlations between all pairs of qubits, i.e. every

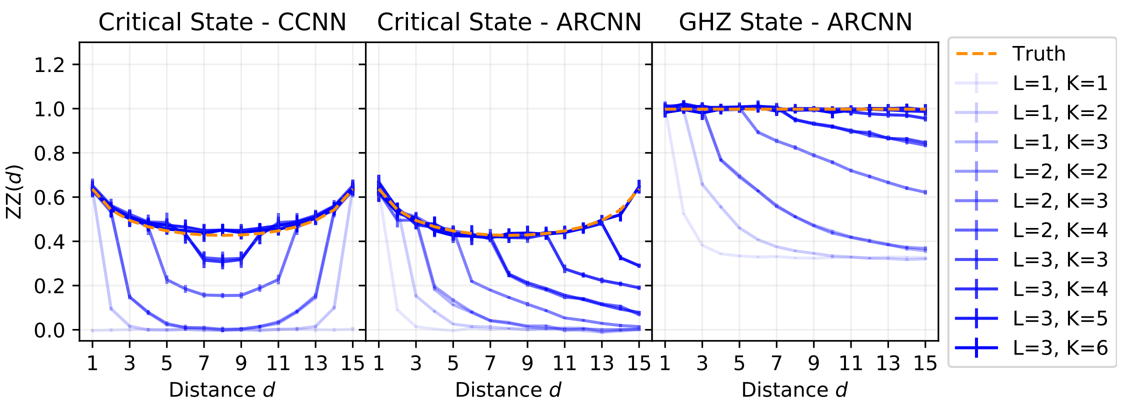

Figure 3.4: The zz correlator is shown as a function of physical distance as learned by different configurations of the CCNN and the ARCNN on an $N = 16$ qubit translation-invariant Ising ground state at critical and at strong coupling. These correlations are only captured correctly for all states up to the distance d_{max} from equations (3.3) and (3.4).

Regime	Scaling of NN parameters	Comment (supporting Figures)
$J/B \ll 1$	$\mathcal{O}(1)$	(3.3, A.4)
$J/B = 1$	$\mathcal{O}(N^3)$	(3.3, 3.4, A.3, A.5, A.6)
$J/B > 1$	$\mathcal{O}(N^3)$	valid for (C)CNN if $N \leq 10$ (3.3), valid for ARCNN for all tested N (3.4)

Table 3.2: Summary of the requirements for efficiently representing 1D Ising ground states in both network architectures, assuming the bounds from Eq. (3.4) and (3.3) are met.

qubit is indistinguishable from any other qubit. Due to their convolutional nature, these networks are able to capture this simplicity and exploit it.

It remains to emphasize, that the implication that the quantity d_{max} makes, only goes one way: while a d_{max} that is too small will probably lead to a bad representation of a state, a good representation is of course not guaranteed by a sufficiently large d_{max}.

Using the formula for the numbers of parameters of the CNNs $\mathcal{O}(kLf^2)$ (2.24) and the scaling $f = N$ for the feature dimension, that is also used for Figures 3.4 and 3.5, the constraint $d_{\text{max}} = N$ implies that the number of parameters scales like $\mathcal{O}(N^3)$, just as was empirically found in the previous section for the strongly correlated states.

A summary of which network and which scaling is required to learn states from the three Ising regimes is found in Table 3.2.

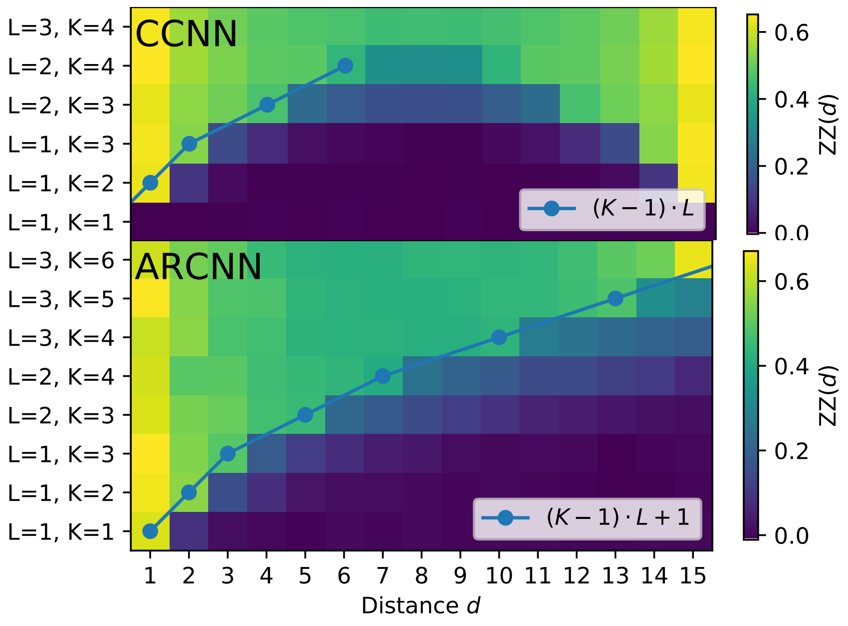

Figure 3.5: Same data as Fig. 3.4. Here different hyperparameter configurations are shown on the vertical axis and the strength of the correlation is shown as color. The blue lines indicate the correlation bounds given by $d_{\max}$ from equations (3.3) and (3.4). Beyond these lines, the captured correlations decay. Both panels show the translation-invariant Ising critical state.

3.2 Generalisation - Limited data

With some basic facts regarding the representational power of the CNNs out of the way, one can now benchmark the tomography scheme of interest in a more experimentally relevant setting. In this section, quantum measurements on various physical systems are simulated and finite datasets are used for state reconstruction.

In order to make quantitative statements on the performance of this tomography scheme, it is compared to two alternatives: the first is MLE (see Section 2.1.2), as it presents the go-to choice for many small scale qubit systems [16, 18]. Secondly it is compared to direct estimation of observables from the dataset (see Sections 2.2 and 2.2.4), inspired by [71].

The analysis is done according to the following scheme: a target wave function or density matrix is computed exactly and its POVM distribution P_{truth} is obtained. From this distribution, N_s samples are drawn (typically $N_s = 10^3 - 10^5$ for 16 qubit systems) using Markov Chains (see Sec. 2.2.5). This amounts to simulating measurements on the target state. These samples are used to train the network, resulting in the network approximation $P_{\text{NN}}(\mathbf{a})$ of the target distribution. For small systems, i.e. those where it is feasible, MLE is performed on the samples, yielding a MLE-estimate for the density matrix, of which the POVM distribution $P_{\text{MLE}}(\mathbf{a})$ is computed. One may then ask, which of these two estimates is closer to the ground truth target distribution, using the classical infidelity[2]

$$D_{\text{NN/MLE}} = 1 - \sum_{\mathbf{a}} \sqrt{P_{\text{NN/MLE}}(\mathbf{a}) P_{\text{Truth}}(\mathbf{a})}. \tag{3.5}$$

This allows to quantify, which of the two estimates is better, by using the quotient $D_{\text{NN}}/D_{\text{MLE}}$. If it is less than one, the network gives the better estimate for the target state compared to MLE, and vice versa.
For systems where MLE is infeasible, one may instead look at the root mean square (RMS) error of local observables

$$\text{RMS}_{\text{NN/Data}} = \sqrt{\left\langle (O_{\text{NN/Data}} - O_{\text{Truth}})^2 \right\rangle} \tag{3.6}$$

as a smaller RMS error on many observables implies a better approximation of the target state. Here, $\langle \cdot \rangle$ indicates the expectation value over independent datasets.

These observables can either be sampled from the training dataset itself, or from the network-encoded distribution, from which $5 \cdot 10^5$ samples are drawn for 16 qubit systems. In this situation the NN acts in a way of replacing the measured dataset with a larger, network-generated one, aiming to decrease statistical measurement noise. Here one can ask, whether the NN gives an advantage for the estimation of observables, by looking at the quotient $\text{RMS}_{\text{NN}}/\text{RMS}_{\text{Data}}$.

[2]This similarity measure is chosen over the previously used D_{KL} for easier comparisons to previous studies such as [11] during development.

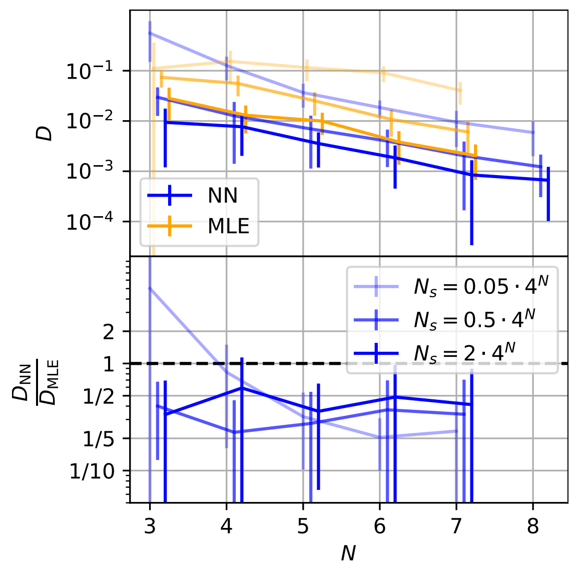

Figure 3.6: *Upper:* Residual classical infidelity D (3.5) of CCNN and MLE on 1D Ising ground state with periodic boundary conditions, $J/B = 1$. Shown for different dataset sizes N_s. *Lower:* Residual network infidelity normalised to residual MLE infidelity.

3.2.1 Transverse Field Ising Model

As a start, this method is again benchmarked on ground states of a translation-invariant TFIM (2.31), here using the standard CCNN. This serves as a proof of concept in an idealized scenario, as the translation invariance is directly encoded in the neural network. Since experimentally prepared states are rarely exactly translation-invariant, and the comparison of a symmetrized network to an unsymmetrized reference is somewhat unfair, the use of networks with enforced symmetries is refrained from for the later examples.

Figure 3.6 shows the method being applied to small, i.e. MLE-suitable 1D states at the Ising critical point, for dataset sizes N_s given as a fraction of the POVM size 4^N. One can achieve a reduction of infidelity by a factor 2-5, depending on system and dataset size. The figure shows one main trend: the network advantage shrinks for increasing dataset size N_s. This is also an expected result, as MLE has to outperform any variational approach in the limit of infinite dataset size. This behaviour holds for all following systems.

In Fig. 3.7 the method is applied to a 4×4 Ising lattice. The histograms show how enhancing the dataset using the NN can lead to a reduced variance of observable estimates, and thus a reduced error. For the network advantage, one sees two trends: An increased advantage for small datasets, similar to the previous example, as well as an improved performance for smaller coupling strengths. The latter result matches that from Section 3.1. If J/B becomes too large, training the network becomes unstable on such small datasets and the advantage disappears entirely.

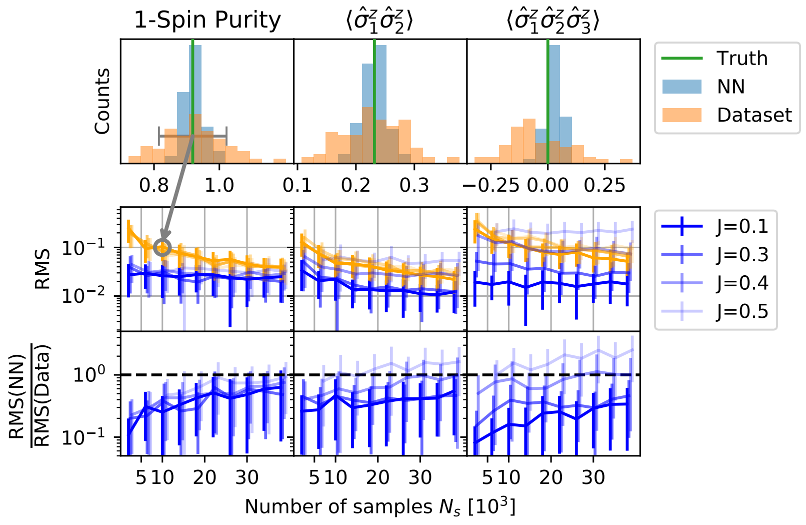

Figure 3.7: Comparing the CCNN against the plain dataset by estimating three local observables on a 4×4 Ising lattice with periodic boundaries. *Upper:* Histogram of said observables for independent datasets and network initializations, with $J/B = 0.3$ and $N_s = 10\text{k}$. *Middle:* RMS Errors of observables for varying coupling strengths and dataset sizes. *Lower:* Comparison of NN and plain dataset in terms of the quotient of their respective RMS errors.

3.2.2 Long-range interacting ion chain with dissipation

For a more experimentally motivated [90] example, one may look at ground states of a 16-site ion chain Hamiltonian,

$$\hat{H} = -J \sum_{i \neq j} \frac{\hat{\sigma}_i^z \hat{\sigma}_j^z}{|i-j|^{1.1}} - B \sum_i \hat{\sigma}_i^x. \tag{3.7}$$

with long range interactions, open boundary conditions and small added ($\approx 3\%$) dephasing noise. The latter reduces the coherences of the state, hence the target density matrix is given by $\hat{\rho}_{\text{Target}} = 0.97 \left|\psi_0\right\rangle \left\langle\psi_0\right| + \frac{0.03}{2^{16}}\mathbb{1}$. The dephasing noise makes the state not only relevant, as such noise is experimentally hard to avoid [90], but it also makes use of the networks capability of representing mixed states, as so far only pure states have been considered. Moreover, the studies on the random states in Sec. 3.1.1 showed a better performance of the network on less pure states, further motivating this choice of target state. The Hamiltonian only differs from the TFIM by additional coupling terms $\hat{\sigma}_i^z \hat{\sigma}_j^z$. Therefore, it behaves very similar to a TFIM with slightly stronger coupling J/B.

Since this is naturally a 1D system, the ARCNN is best used. Here the NN

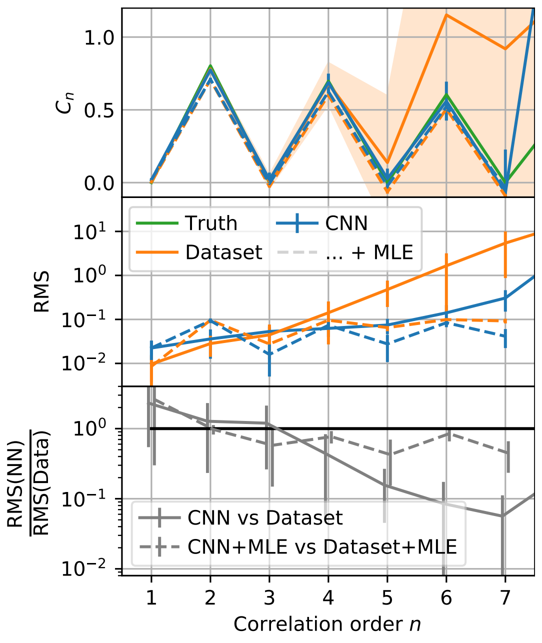

Figure 3.8: Benchmarking the ARCNN by evaluating observables from Eq. (3.8) and comparing to those obtained from plain dataset, on a length 16 ion chain with 3% dephasing noise, $J/B = 0.6$ and open boundaries, with a dataset size of $N_s = 10\text{k}$. Dashed lines show results of applying local MLE directly to dataset (orange) and to a neural network enhanced dataset (blue). *Upper:* Observables according to Eq. (3.8), *Middle:* residual RMS error, *Lower:* residual network RMS normalized to dataset RMS.

advantage is studied as a function of correlation order. This is interesting, as higher order correlators are typically harder to estimate from samples, due to the variance of the POVM-observable scaling exponentially in correlation order. Thus, higher moments require a better approximation of the state, providing a sensitive benchmark for the quality of the state representation. Specifically, powers n of the Pauli-Z operator

$$C_n := \frac{1}{16-n+1} \sum_{i=1}^{16-n+1} \left\langle \hat{\sigma}_i^z \hat{\sigma}_{i+1}^z ... \hat{\sigma}_{i+n-1}^z \right\rangle \tag{3.8}$$

are looked at. Local observables, like the terms in Eq. (3.8), only depend on the reduced density matrix of the subsystem they act on. Thus, these observables can in principle be estimated by performing MLE on this subsystem only. This "local MLE" is shown when applied to the pure dataset and to the NN enhanced dataset in Fig. 3.8 in addition to the previous benchmarks. Using the NN generated dataset, one sees a reduction in RMS to a degree, that allows sampling for correlators of three orders higher, than what is possible with the plain dataset. When applying the local MLE to both original and NN generated dataset, this advantage is reduced significantly, but does not disappear. However, one should emphasize that the ARCNN is on par with MLE, at a greatly reduced computational complexity. Notice also the peak in RMS at a correlation order of 2 for MLE, leading to an increased RMS compared to plain sampling. This is found to be systematic, which is why this comparison to local MLE is omitted from further studies.

3.2.3 2D driven dissipative system

As a final system, steady states of a 4×4 TFIM with spontaneous decay are considered, motivated by ongoing research into phase diagrams of open quantum systems [91], with potential applications to Rydberg systems. These states again make use of the ability of the ansatz of representing mixed quantum states. The network is reverted to the standard, 2D CNN with dense output layer, since the system is two-dimensional and *not* translation-invariant. Now, no symmetries are encoded in the network. The Monte-Carlo wave function (MCWF) approach, implemented in the QUTIP library [92], is used to simulate the dynamics under the Lindblad master equation

$$\dot{\hat{\rho}} = -i[\hat{H}, \hat{\rho}] + \gamma \sum_j \left(\hat{L}_j \hat{\rho} \hat{L}_j^\dagger - \frac{1}{2} \left\{ \hat{L}_j^\dagger \hat{L}_j, \hat{\rho} \right\} \right), \tag{3.9}$$

$$\text{with} \qquad \hat{H} = J \sum_{\langle ij \rangle}^{N} \hat{\sigma}_i^z \hat{\sigma}_j^z + h_x \sum_i^N \hat{\sigma}_i^x, \quad J = 1.25\gamma, \tag{3.10}$$

$$\hat{L}_j = \hat{\sigma}^- = \frac{1}{2} \left(\hat{\sigma}_j^x - i \hat{\sigma}_j^y \right), \tag{3.11}$$

until a steady state is reached. The density matrix that is obtained using 1000 pure-state trajectories is treated as the exact target. To simulate measurements, POVM samples are generated for every pure state from the ensemble.

This system undergoes a dissipative phase transition [91] from a $|\downarrow\rangle$ polarized state at small fields h_x to a strongly dephased state at strong fields. The transition between these two phases is visible as a peak in the correlation length

$$\xi_z^2 = \sum_{i,j} |\vec{r}_i - \vec{r}_j|^2 \left(\langle \hat{\sigma}_i^z \hat{\sigma}_j^z \rangle - \langle \hat{\sigma}_i^z \rangle \langle \hat{\sigma}_j^z \rangle \right). \tag{3.12}$$

This phase transition is shown as obtained by sampling the correlation length on the 1D diagonal of the 2D lattice, once directly from the training data, as well as from a NN enhanced dataset in Fig. 3.9. The network is able to capture the phase transition, as a peak in the correlation length at $h_x/\gamma \approx 2$ is clearly visible. At the dashed grey line, $\uparrow$ and $\downarrow$ are exchanged in the POVM that the CNN uses, ensuring that the target state does not contain exact zeros in its POVM distribution.

For small h_x, the steady state tends towards an eigenstate of the observable in question, thus the variance of sampling this observable is significantly reduced, hence the network has no advantage here. For the limit of large h_x one can see a huge variance in the sampled correlation length, which the CNN trades for a small systematic error, i.e. bias. The overall effect is that the CNN bias + CNN variance lead to a significantly *smaller* RMS error as compared to the plain dataset. Notice that this bias also shrinks with the dataset size (Fig. 3.9 insets).

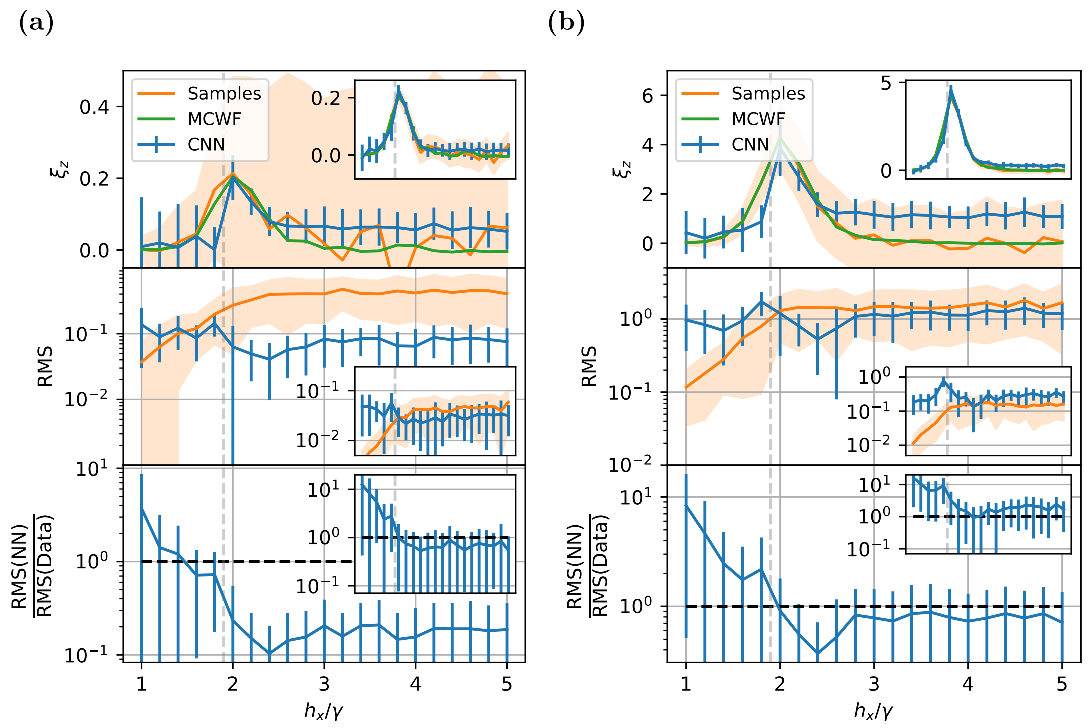

Figure 3.9: Dissipative phase transition of a 4×4 TFIM with spontaneous emission at $J = 1.25\gamma$ with open boundaries. $N_s = 1\text{k}$ (insets: 100k). Grey vertical line: Switch $\uparrow$ and $\downarrow$ in POVM for CNN. *Upper:* Observables according to Eq. (3.12), *Middle:* residual RMS error, *Lower:* residual network RMS normalized to dataset RMS. **(a)** Eq. (3.12) summed only over one diagonal of the system, **(b)** summed over entire system.

Depending on the observable of interest, the bias can have a more severe effect than depicted. When computing the correlation length over the 1D diagonal, as in Fig. 3.9(a), the corresponding sum in Eq. (3.12) is a weighted average of $\binom{4}{2} = 6$ connected correlators of the form $\langle \hat{\sigma}_i^z \hat{\sigma}_j^z \rangle - \langle \hat{\sigma}_i^z \rangle \langle \hat{\sigma}_j^z \rangle$. After sampling, one may consider each of these connected correlators as a random variable with a variance and a bias. For the plain dataset, this bias is of course zero. However, when evaluating Eq. (3.12) over the entire lattice as in Fig. 3.9(b), the sum contains $\binom{16}{2} = 120$ terms, with roughly similar variance and bias. For the sampled case, by simple addition of probability distributions, the variance of the ”entire sum” is thus reduced by a factor of $\sqrt{120/6} \approx 4.5$ compared to the ”small sum”. Due to the bias, the network is not able to make use of this self-averaging effect, resulting in a significantly reduced advantage. This can be seen in Fig. 3.9b. In fact, the network shows an advantage when computing the correlation length for *any* four qubit subsystem (and not just the one shown in Fig. 3.9(a)), and the advantage only disappears when looking at larger systems. Thus, the network advantage is the biggest, if one is interested in the expectation values of individual correlators, and might be smaller if large sums over many similarly distributed correlators are involved as the bias inherent to the variational approach worsens the averaged results.

3.3 Cluster states - Experimental data

Finally, the tomography scheme is tested on some real experimental data. For this thesis, the results of a full tomographic measurement were gratefully received from Christian Roos. The data was originally collected for [18], with the intention of demonstrating measurement-based quantum computing. This type of quantum computing relies on highly entangled, so-called cluster states. Concretely, the five-qubit state

$$2\left|\psi\right\rangle = -\mathrm{i}\left|\text{-000-}\right\rangle + \left|\text{+111-}\right\rangle + \left|\text{-111+}\right\rangle + \mathrm{i}\left|\text{+000+}\right\rangle \tag{3.13}$$

was prepared experimentally using ion chains. Here $|1\rangle$ and $|0\rangle$ correspond to the $+1$ and -1 eigenstates of the Pauli z operator and $|\text{+}\rangle$ and $|\text{-}\rangle$ similarly for the Pauli x operator. This state is equivalent up to local unitaries to the state

$$2\left|\psi'\right\rangle = \left|0\text{+++}0\right\rangle + \left|1\text{ - - - }0\right\rangle + \left|0\text{ - - - }1\right\rangle + \left|1\text{+++}1\right\rangle. \tag{3.14}$$

The latter state may be generated by preparing a product $|\text{+}\rangle$ state and applying a controlled phase gate

$$\mathrm{CPHASE} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{pmatrix}$$

to qubit pairs $(1,2)$, $(1,3)$, $(1,4)$ as well as $(5,2)$, $(5,3)$, $(5,4)$. States prepared using such product state initialization followed by pair-wise entangling gates are commonly called cluster states.

For all 3^5 combinations of Pauli bases $xxxxx, xxxxy, \ldots, zzzzz$, 100 projective measurements were performed in the experiment, and the number of occurrences counted for each of the 2^5 possible outcomes $00000, 00001, \ldots, 11111$. Thus the raw data that was made available is a $3^5 \times 2^5$ table of counts. Upon normalizing with the total number of counts, this data directly corresponds to $3^5 \cdot 100 = 24\,300$ samples of the Pauli-6 POVM distribution.

Reconstruction Method	Negative Log-Likelihood	Quantum Fidelity	Classical Infidelity	Negativity
Pauli-4 MLE	8.621	0.740	0.015	-
Pauli-6 MLE	8.605	0.843	0.013	-
Pauli-4 LI	8.358	0.833	0.023	3.15
Pauli-4 CNN	8.624	0.358	0.032	1.01
Pauli-4 CNN (synth. Data)	8.627 ± 0.012	0.357 ± 0.033	0.029 ± 0.002	0.86 ± 0.07

Table 3.3: Results for reconstructing the density matrix for the given experimental dataset using four different techniques, as well as CNN with $f = 10, k = 4$ and $l = 2$ trained on synthetic dataset for reference. Fidelities computed w.r.t. experimental target (3.13). Since only one experimental dataset exists, no error bars are given.

3.3.1 Analysis of the full dataset

To ensure, that the format of the provided data is understood, it is first analysed by classical means, with the goal of reproducing quantities from the original publication. To this end, maximum likelihood estimation is performed using the full dataset, i.e. the Pauli-6 POVM. Table 3.3 shows the quantum fidelity to the experimental target state (3.13). This matches exactly the fidelity quoted in the original publication of 0.843 ± 0.005, implying that the data is understood and the MLE algorithms are comparable.

Since the exact state prepared in the experiment is not known, but only measurements resulting from it, there is no ground truth target to compare tomography schemes against. The next best thing one can do for assessing the quality of state reconstructions, is to inspect the likelihood function. This is quoted in Tab. 3.3 as ”Negative Log-Likelihood”. Lower number here are better, indicating that the resulting density matrix of a particular tomography scheme is more likely to have produced the obtained dataset. Table 3.3 also shows the results of reconstructing the density matrix using the Pauli-4 POVM, once using MLE and once by direct linear inversion (LI, which is not possible using the Pauli-6 POVM). To obtain the Pauli-4 dataset, the measurement counts from the full dataset are regrouped according to Sec. 2.2. Using MLE with the Pauli-4 POVM is worse than using the Pauli-6 POVM, which is expected, since regrouping the data to Pauli-4 implies a loss of information. Linear inversion, on the other hand is not constrained by the positivity of the density matrix, which allows the likelihood to be even better than the Pauli-6 MLE case. The trade-off is of course that the sum of the negative eigenvalues of the density matrix (i.e. the ”Negativity”) is not negligible.

Finally, the CNN is trained on the dataset. For 5 qubits, the density matrix contains 1023 independent real parameters, so the network was restricted to architectures with less parameters than this, as to not overparametrize the state (which would defeat the entire point of finding efficient state parametrization using NNs). Still, however, the network performs rather poorly on this state. The results in Tab. 3.3 show the best CNN that was tested, measured in terms of quantum fidelity. While the negativity happens to be slightly less than for plain LI, the CNN seems to combine the negative traits from LI and the Pauli-4 POVM into one: the resulting density matrix is negative, and approximates the target state the poorest out of all shown algorithms, according to all measures of quality. As a reference, Fig. 3.10 shows matrix plots of the target density matrix, as well as the resulting density matrices from three of the four analysis schemes, confirming the previous result.

On previous scenarios, it was seen that the network advantage is the greatest, when operating on small datasets. Therefore, one possible explanation for the poor performance might be that the full dataset is simply too big for the CNN to show any improvement compared to other methods. This is tested in the next section.

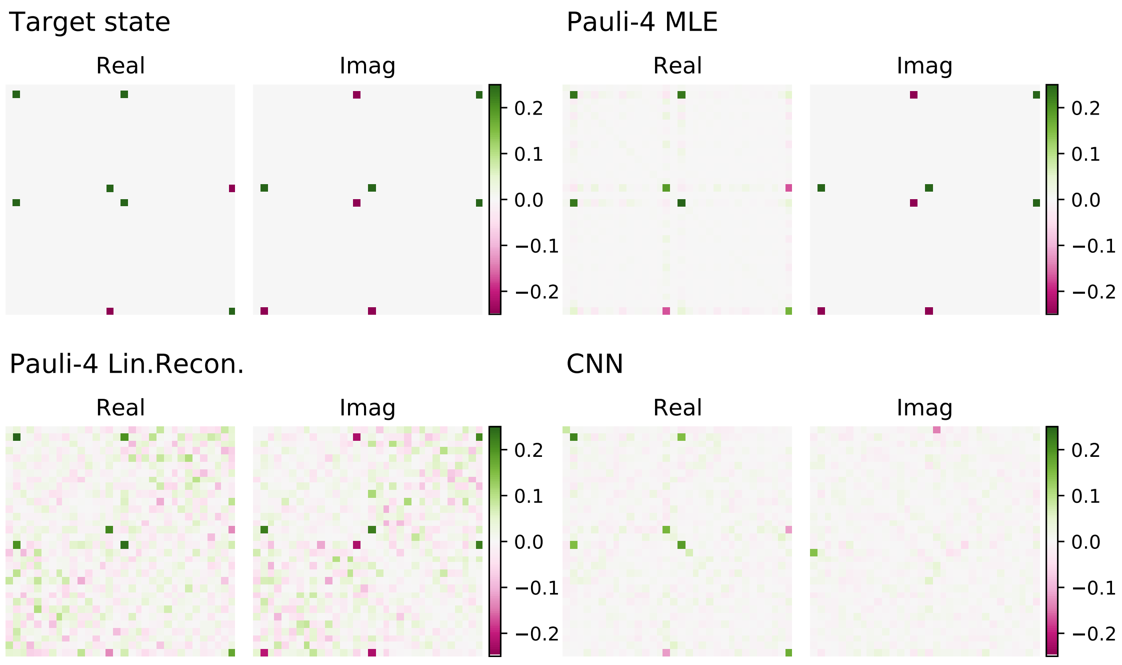

Figure 3.10: Matrix plots of the real and imaginary part of the density matrix, shown for target state (3.13) and the results of various reconstruction methods operating on the experimental dataset.

3.3.2 Analysis using resampling

To test the previous hypothesis, this section looks at smaller subsets of the full experimental dataset. This has the additional advantage that this resampling allows for estimation of statistical error bars. Figure 3.11 shows the CNN and MLE being applied to dataset sizes ranging from 500 up to 20 000 samples. Here, the MLE estimate using the full dataset and the Pauli-6 POVM is treated as a "ground truth", as it is the best available estimate of the experimentally prepared state. This allows to compute absolute errors of observables and to quote fidelities. The general trend is clear: with the exception of the zz correlator, all shown measures of accuracy indicate that the CNN is inferior to MLE, across all dataset sizes. This rules out the hypothesis of too large datasets.

This leaves just one explanation for the poor performance of the CNN: the cluster state at hand simply cannot be approximated well by the CNN. This hypothesis is reinforced upon replacing the experimental dataset by a synthetic one, as is shown in the last row of Tab. 3.3. The fact that the behaviour of the CNN on the experimental data is almost identical to the behaviour on synthetic data, rules out any further experimental factors that could have been unaccounted for.

For the creation of the cluster state, an entangling gate is applied to many *non-local* pairs of qubits, as described above. The CNN, however, was designed to deal with *locally* spreading correlations. This could also explain the failure of the CNN in this scenario and a non-local network might be the better option here.

An alternate explanation might be the following: The density matrix of a five-qubit state is still quite manageable in terms of the number of free parameters. The main idea behind neural network quantum states, is to find efficient representations of states that would be intractable otherwise, which is simply not the case for a five-qubit system. Often the great expressivity of neural networks only fully develops, after a certain network size is reached. Hence the term "deep" learning is often used. For small systems, it might therefore be possible to overparametrize a system, but still miss the physically interesting states, i.e. networks might need a certain "base complexity" before they become interesting in terms of representability. The conclusion may be drawn, that in general the advantage of this NN-QST method lies in larger systems, that are intractable for standard methods such as MLE.

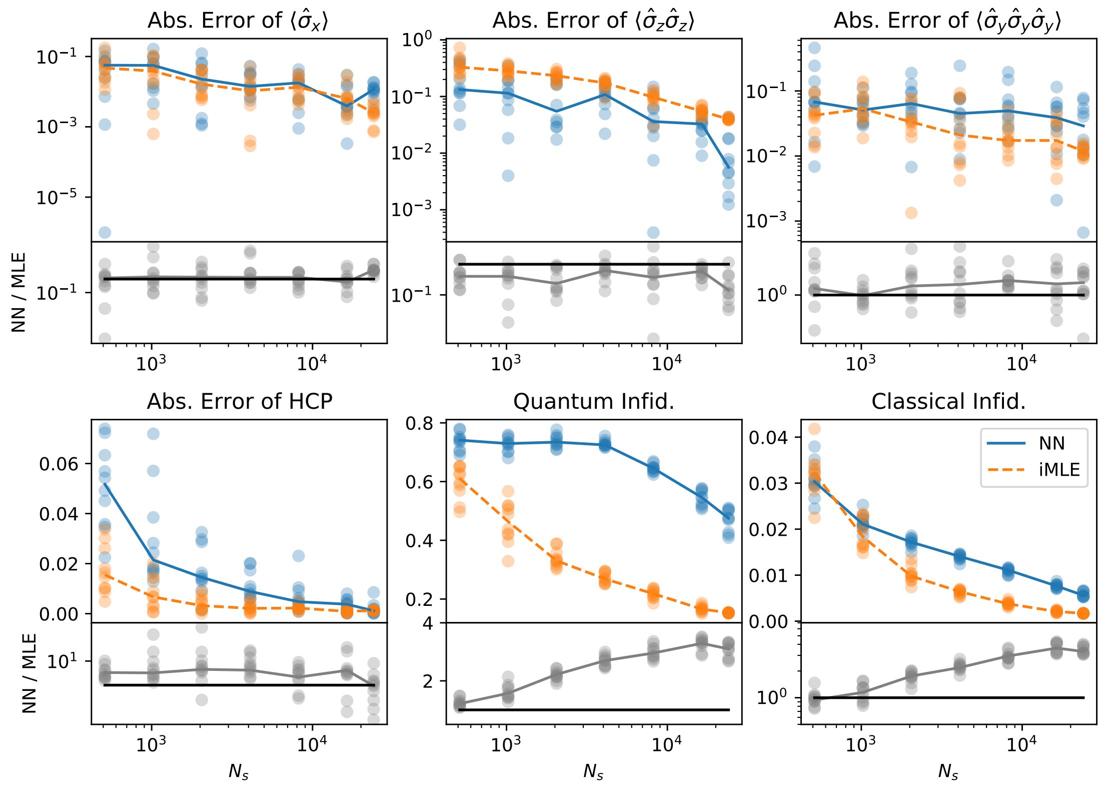

Figure 3.11: Taking a subset of size N_s of the full experimental dataset and training a network with $f = 10, k = 4$ and $l = 2$, and subsequently comparing observables to (Pauli-4) MLE-estimates obtained for same datasets (HCP = Half chain purity). With the exception of the two-point function, the network performs worse than MLE across all dataset sizes.

4 Conclusion and Outlook

4.1 Summary

In this work, the task of quantum state tomography was translated to a task of density estimation, by mapping a state's representation from a density matrix to its POVM probability distribution. The density estimation task was then solved by employing a convolutional neural network as an expressive function approximator to learn the distribution underlying a discrete, limited experimental dataset.

In this setting, a variational approximation to the target state is trained, and encoded in the parameters of the CNN. This was motivated by recent theoretical developments, framing CNNs as a more expressive generalization to similar approaches like MPS, at favourably scaling numbers of parameters.

The resulting tomography scheme was tested in three different settings, using two different CNN architectures, and on a wide variety of quantum states.

Initial studies on the representational power of the standard CNN were performed by training state representations under the assumption of infinite data. Investigations on random states showed, that in general purer states are harder to approximate using the CNN. The transverse field Ising model served as a benchmarking platform, allowing to test states ranging from product states, over critical states to maximally entangled states. Here it was seen that Ising ground states at smaller coupling are generally easier to represent, only requiring a constant number of network parameters. States at critical and strong coupling were found to be efficiently representable using a network size, that scales only cubically in the system size. This polynomial scaling indicates that property ② from Section 2.1.1 is fulfilled.

A systematic analysis of the inner workings of the circular CNN as well as the autoregressive CNN allowed for a deeper understanding of the propagation of correlations within the networks, resulting in physical bounds on correlation lengths of network-represented states.

This insight led the way to testing the networks in a tomography setting, where only a limited (synthetic) dataset is available. To make results interpretable, comparisons to maximum likelihood estimation were made and the errors of observables were used as a quantifier for the quality of approximations. For systems sizes where MLE is feasible, the CNN showed a measurable advantage in terms of state fidelity when compared to MLE, on 1D Ising model ground states. This shows that the proposed method can give an advantage w.r.t. to property ①.

For system sizes of 16 qubits, an advantage in terms of RMS errors of observables was seen for 2D Ising ground states, long-range interacting noisy ion chain ground states as well as for steady states of dissipative Ising models. For the ion chains, the ARCNN enabled the estimation of correlation functions three orders higher than when directly computed from measured datasets, again giving an improvement w.r.t property ①. For the steady states, an advantage in measuring the order parameter was shown, under the consideration of a caveat: A small bias was found to manifest itself only when computing sums over all two-qubit correlators, resulting in a performance penalty for the network. This implies that the tomography scheme does not fulfil the observable universality property ③.

As a final setting, the tomography scheme was tested on experimental measurement data, provided by Christian Roos. Unfortunately, here the network was found to perform subpar for analysing the full dataset, compared to all tested tomography schemes. Even for smaller datasets, the network was inferior to simple maximum likelihood estimation, due to the target state not belonging to the class of states that is efficiently representable using the CNN. This demonstrates that the tomography scheme does not have the state universality property ④, which it (being a variational approach) was never designed to have.

This tomography scheme thus may be characterized as violating observable universality ③ and state universality ④, but at the same time improving upon the experimental ① and numerical resource requirements ②, matching the originally stated goals. It therefore offers an interesting addition to the class of variational tomography schemes, and can extend their range of applicability to a broader spectrum of states.

4.2 Outlook

During a thesis like this, one learns a lot about the topic at hand, and some optimizations to the method itself or the analysis scheme only become apparent in hindsight. A few of these points are laid out here.

4.2.1 Improvements to the tomography scheme

Learning Pauli-6 instead of Pauli-4

The Pauli-6 POVM is often the easiest POVM to measure experimentally. However, it is not invertible, preventing the direct computation of observables from its distribution by inversion of the T matrix. This problem was solved by grouping POVM outcomes and turning the Pauli-6 POVM into the Pauli-4 POVM which was then approximated using the neural network. This regrouping of outcomes inherently discards data.

However, invertibility of the POVM is not required for training the neural network (which only requires samples from a POVM distribution and the ability of the network to approximate the latter). Thus also the Pauli-6 POVM could in principle be trained, and subsequently samples drawn. This training would therefore make use of the entire experimental dataset and the grouping of outcomes for the computation of observables could be performed *after* sampling from the network.

Since the choice of POVM can make a difference for MLE as shown in Section 3.3, a similar difference could be expected from the neural network approach, possibly resulting in an improved performance.

Huge networks

In this thesis, the network architecture was always chosen according to the premise "the smaller the network the better (as long as the state is still representable)". This follows the thinking behind the so-called "bias-variance trade-off": a model that is too small for a given task, will give biased results, while a model that is too large will not generalize due to a high variance [93]. However, recently it was repeatedly shown that overparameterizing, or using vastly larger networks might be beneficial [94, 95], indicating that the Bias-Variance trade-off might not exist for neural networks. Therefore, it could be interesting to apply significantly larger networks to this problem.

2D Autoregressive CNN

The simple modifications to the standard CNN shown in Sec. 2.3.1 rendered the CNN autoregressive, granting it the ability to generate exact samples and to be exactly normalized. This made training on 1D systems much more stable and resulted in a much broader range of applications. Unfortunately, this autoregressive extension of the CNN is not straight forward in anything but 1D scenarios, as the 2D convolutions always result in some sites being uncorrelated (see Appendix A.1). Recently, solutions to this problem have been proposed in [44]. If these have a similar impact on the performance of 2D CNNs as the autoregressive modification has for 1D CNNs, this would be a huge step forward.

4.2.2 Improvements to the analysis scheme

Comparisons not only for product observables

For systems sizes where MLE is infeasible, local observables were used as a quantifier for the performance of the network. However, here only Pauli strings (or small sums thereof) were used as observables. If a given experimental system can collect Pauli-6 measurements, it can most certainly also measure these Pauli string observables directly, making a tomography unnecessary. These Pauli string observables therefore only serve as a benchmark of the quality of approximation, and not as a real-world use case of the method.

Had one chosen more general observables that are not simple products of Pauli-operators and therefore harder to measure experimentally, a performance advantage of the network could have made a strong argument for the measurement of those observables using the scheme presented here.

Comparison to shadow tomography

Finally, a comparison to shadow tomography [14, 15] would make a lot of sense. Shadow tomography is very similar to this tomography scheme, as it also shifts the emphasis away from the density matrix itself and towards observables that might be estimated from the data. This allows shadow tomography to scale much better than e.g. MLE, being subexponential in both computational complexity and required amount of data. In particular, shadow tomography claims superiority over some of the NN-QST schemes cited in this work, making a comparison even more intriguing.

4.3 Concluding remarks

One would now like to give a concrete recommendation to experimentalists on where and when to apply this method. Yet, this remains rather unsatisfying: Where this thesis showed success of the method using synthetic data, a direct application to an experiment should be straightforward. This includes many Ising type models, that are either 1D, 2D up to a certain coupling constant, or dissipative. Empirically, states with a lower purity or larger system sizes show higher success rates. In these situations, one can expect a considerably reduced experimental workload, as the method can give more precise estimates of target states than competing options, at similar dataset sizes. Conversely, smaller experimental datasets are required in order to achieve similar error bars. However, the inevitable possibility of experiencing a bias, makes it necessary to validate this method for every possible new experiment individually, if not already done so in this thesis.

Still, the main benefit of this method remains: by translating the task of tomography to one of density estimation, one can employ *any* variational function approximator as a solution to this task. That includes the vast zoo of neural network architectures, that keeps growing on a daily basis. Each of these architectures may have their strengths and benefits for different classes of states, thus building the foundation for a very general and ever-growing area of application for tomography schemes like the one presented here.

Acknowledgements

I would like to thank Martin Gärttner for giving me the opportunity of writing my thesis is his group. Furthermore, I would like to thank Moritz Reh for support with MCWF simulations and Moritz Reh, Juan Carrasquilla and Markus Schmitt for helpful discussions. I acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant no INST 40/575-1 FUGG (JUSTUS 2 cluster). I gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time through the John von Neumann Institute for Computing (NIC) on the GCS Supercomputer JUWELS [83] at Jülich Supercomputing Centre (JSC). Plots were created using Matplotlib [96] and the Numpy [97] library came to excessive use throughout this thesis.

References

[1] E. Altman et al. "Quantum Simulators: Architectures and Opportunities". In: *PRX Quantum* 2.1 (Feb. 24, 2021), p. 017003.

[2] A Fraenkel. "Complexity of protein folding". In: *Bulletin of Mathematical Biology* 55.6 (Nov. 1993), pp. 1199–1210.

[3] E. Callaway. "'It will change everything': DeepMind's AI makes gigantic leap in solving protein structures". In: *Nature* 588.7837 (Dec. 10, 2020), pp. 203–204.

[4] J. Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* (July 15, 2021).

[5] T. B. Brown et al. "Language Models are Few-Shot Learners". In: *arXiv:2005.14165 [cs]* (July 22, 2020).

[6] P. Zhang, H. Shen, and H. Zhai. "Machine Learning Topological Invariants with Neural Networks". In: *Physical Review Letters* 120.6 (Feb. 6, 2018), p. 066401.

[7] A. Tanaka and A. Tomiya. "Detection of Phase Transition via Convolutional Neural Networks". In: *Journal of the Physical Society of Japan* 86.6 (June 15, 2017), p. 063001.

[8] G. Carleo and M. Troyer. "Solving the quantum many-body problem with artificial neural networks". In: *Science* 355.6325 (Feb. 10, 2017), pp. 602–606.

[9] Y. Levine et al. "Quantum Entanglement in Deep Learning Architectures". In: *Physical Review Letters* 122.6 (Feb. 12, 2019), p. 065301.

[10] O. Sharir, A. Shashua, and G. Carleo. "Neural tensor contractions and the expressive power of deep neural quantum states". In: *arXiv:2103.10293 [quant-ph]* (Mar. 18, 2021).

[11] J. Carrasquilla et al. "Reconstructing quantum states with generative models". In: *Nature Machine Intelligence* 1.3 (Mar. 2019), pp. 155–161.

[12] D. J. Griffiths. *Introduction to Quantum Mechanics.* Prentice Hall, 1994. ISBN: 0-13-124405-1.

[13] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information.* 7th ed. Cambridge University Press, 2010. ISBN: 978-1-107-00217-3.

[14] S. Aaronson. "Shadow tomography of quantum states". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing.* STOC '18: Symposium on Theory of Computing. Los Angeles CA USA: ACM, June 20, 2018, pp. 325–338. ISBN: 978-1-4503-5559-9.

[15] H.-Y. Huang, R. Kueng, and J. Preskill. "Predicting many properties of a quantum system from very few measurements". In: *Nature Physics* 16.10 (Oct. 2020), pp. 1050–1057.

[16] A. I. Lvovsky. "Iterative maximum-likelihood reconstruction in quantum homodyne tomography". In: *Journal of Optics B: Quantum and Semiclassical Optics* 6.6 (June 1, 2004), S556–S559.

[17] S. Ahmed et al. "Quantum State Tomography with Conditional Generative Adversarial Networks". In: *arXiv:2008.03240 [quant-ph]* (Dec. 4, 2020).

[18] B. P. Lanyon et al. "Measurement-Based Quantum Computation with Trapped Ions". In: *Physical Review Letters* 111.21 (Nov. 19, 2013), p. 210501.

[19] R. Blume-Kohout. "Optimal, reliable estimation of quantum states". In: *New Journal of Physics* 12.4 (Apr. 20, 2010), p. 043034.

[20] S. Aaronson. "The learnability of quantum states". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 463.2088 (Dec. 8, 2007), pp. 3089–3114.

[21] G. Struchalin et al. "Experimental Estimation of Quantum State Properties from Classical Shadows". In: *PRX Quantum* 2.1 (Jan. 13, 2021), p. 010307.

[22] T. Baumgratz et al. "Scalable Reconstruction of Density Matrices". In: *Physical Review Letters* 111.2 (July 11, 2013), p. 020401.

[23] M. Cramer et al. "Efficient quantum state tomography". In: *Nature Communications* 1.1 (Dec. 2010), p. 149.

[24] B. P. Lanyon et al. "Efficient tomography of a quantum many-body system". In: *Nature Physics* 13.12 (Dec. 2017), pp. 1158–1162.

[25] R. Orus. "A Practical Introduction to Tensor Networks: Matrix Product States and Projected Entangled Pair States". In: *Annals of Physics* 349 (Oct. 2014), pp. 117–158.

[26] D. Gross et al. "Quantum State Tomography via Compressed Sensing". In: *Physical Review Letters* 105.15 (Oct. 4, 2010), p. 150401.

[27] C. A. Riofrío et al. "Experimental quantum compressed sensing for a seven-qubit system". In: *Nature Communications* 8.1 (Aug. 2017), p. 15305.

[28] G. Tóth et al. "Permutationally Invariant Quantum Tomography". In: *Physical Review Letters* 105.25 (Dec. 16, 2010), p. 250403.

[29] T. Moroder et al. "Permutationally invariant state reconstruction". In: *New Journal of Physics* 14.10 (Oct. 1, 2012), p. 105001.

[30] C. A. Fuchs and R. Schack. "Quantum-Bayesian coherence". In: *Reviews of Modern Physics* 85.4 (Dec. 27, 2013), pp. 1693–1715.

[31] S. Weinzierl. "Introduction to Monte Carlo methods". In: *arXiv:hep-ph/0006269* (June 23, 2000).

[32] S. Brooks et al. *Handbook of Markov Chain Monte Carlo*. 1st ed. Chapman and Hall/CRC, 2011. 624 pp. ISBN: 978-1-4200-7941-8.

[33] S. Khan et al. "A Guide to Convolutional Neural Networks for Computer Vision". In: *Synthesis Lectures on Computer Vision* 8.1 (Feb. 13, 2018), pp. 1–207.

[34] M. Kawato et al. "Hierarchical neural network model for voluntary movement with application to robotics". In: *IEEE Control Systems Magazine* 8.2 (Apr. 1988), pp. 8–15.

[35] W. Baxt. "Application of artificial neural networks to clinical medicine". In: *The Lancet* 346.8983 (Oct. 1995), pp. 1135–1138.

[36] P. R. Burrell and B. O. Folarin. “The impact of neural networks in finance”. In: *Neural Computing & Applications* 6.4 (Dec. 1997), pp. 193–200.

[37] F. Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain.” In: *Psychological Review* 65.6 (1958), pp. 386–408.

[38] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals, and Systems* 2.4 (Dec. 1989), pp. 303–314.

[39] C. C. Aggarwal. *Neural Networks and Deep Learning*. Springer Nature, 2018. ISBN: 978-3-319-94462-3.

[40] R. G. Melko et al. “Restricted Boltzmann machines in quantum physics”. In: *Nature Physics* 15.9 (Sept. 2019), pp. 887–892.

[41] A. Fischer and C. Igel. “An Introduction to Restricted Boltzmann Machines”. In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Ed. by L. Alvarez et al. Red. by D. Hutchison et al. Vol. 7441. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 14–36. ISBN: 978-3-642-33274-6 978-3-642-33275-3.

[42] M. Hibat-Allah et al. “Recurrent Neural Network Wave Functions”. In: *Physical Review Research* 2.2 (June 17, 2020), p. 023358.

[43] X. Liang et al. “Solving frustrated quantum many-particle models with convolutional neural networks”. In: *Physical Review B* 98.10 (Sept. 24, 2018), p. 104426.

[44] O. Sharir et al. “Deep Autoregressive Models for the Efficient Variational Simulation of Many-Body Quantum Systems”. In: *Physical Review Letters* 124.2 (Jan. 16, 2020), p. 020503.

[45] A. Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[46] J. Carrasquilla et al. “Probabilistic Simulation of Quantum Circuits with the Transformer”. In: *arXiv:1912.11052 [cond-mat, physics:quant-ph]* (Dec. 23, 2019).

[47] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1, 1997), pp. 1735–1780.

[48] S.-H. Li and L. Wang. “Neural Network Renormalization Group”. In: *Physical Review Letters* 121.26 (Dec. 26, 2018), p. 260601.

[49] D. J. C. MacKey. *Information Theory, Inference, and Learning Algorithms*. 7.2. Cambridge University Press, 2005. ISBN: 978-0-521-64298-9.

[50] S. Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv:1609.04747 [cs]* (June 15, 2017).

[51] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *arXiv:1412.6980 [cs]* (Jan. 29, 2017).

[52] G. Torlai et al. "Neural-network quantum state tomography". In: *Nature Physics* 14.5 (May 2018), pp. 447–450.

[53] A. W. R. Smith, J. Gray, and M. S. Kim. "Efficient Quantum State Sample Tomography with Basis-dependent Neural-networks". In: *PRX Quantum* 2.2 (June 28, 2021), p. 020348.

[54] G. Torlai and R. G. Melko. "Latent Space Purification via Neural Density Operators". In: *Physical Review Letters* 120.24 (June 15, 2018), p. 240503.

[55] N. Yoshioka and R. Hamazaki. "Constructing neural stationary states for open quantum many-body systems". In: *Physical Review B* 99.21 (June 28, 2019), p. 214306.

[56] A. Nagy and V. Savona. "Variational Quantum Monte Carlo Method with a Neural-Network Ansatz for Open Quantum Systems". In: *Physical Review Letters* 122.25 (June 28, 2019), p. 250501.

[57] M. J. Hartmann and G. Carleo. "Neural-Network Approach to Dissipative Quantum Many-Body Dynamics". In: *Physical Review Letters* 122.25 (June 28, 2019), p. 250502.

[58] F. Vicentini et al. "Variational Neural-Network Ansatz for Steady States in Open Quantum Systems". In: *Physical Review Letters* 122.25 (June 28, 2019), p. 250503.

[59] M. Neugebauer et al. "Neural network quantum state tomography in a two-qubit experiment". In: *Physical Review A* 102.4 (Oct. 9, 2020), p. 042604.

[60] S. Lohani et al. "Machine learning assisted quantum state estimation". In: *Machine Learning: Science and Technology* 1.3 (July 27, 2020), p. 035007.

[61] Y. Quek, S. Fort, and H. K. Ng. "Adaptive quantum state tomography with neural networks". In: *npj Quantum Information* 7.1 (Dec. 2021), p. 105.

[62] A. Hallam et al. "Compact Neural Networks based on the Multiscale Entanglement Renormalization Ansatz". In: *arXiv:1711.03357 [quant-ph]* (Dec. 12, 2018).

[63] A. Valenti et al. "Correlation-Enhanced Neural Networks as Interpretable Variational Quantum States". In: *arXiv:2103.05017 [cond-mat, physics:quant-ph]* (Mar. 8, 2021).

[64] C. Miles et al. "Correlator Convolutional Neural Networks: An Interpretable Architecture for Image-like Quantum Matter Data". In: *arXiv:2011.03474 [cond-mat, physics:physics]* (Nov. 6, 2020).

[65] M. Medvidovic and G. Carleo. "Classical variational simulation of the Quantum Approximate Optimization Algorithm". In: *arXiv:2009.01760 [cond-mat, physics:quant-ph]* (Oct. 29, 2020).

[66] M. Schmitt and M. Heyl. "Quantum many-body dynamics in two dimensions with artificial neural networks". In: *Physical Review Letters* 125.10 (Sept. 2, 2020), p. 100503.

[67] S.-H. Lin and F. Pollmann. "Scaling of neural-network quantum states for time evolution". In: *arXiv:2104.10696 [cond-mat, physics:quant-ph]* (Apr. 21, 2021).

[68] M. Reh, M. Schmitt, and M. Gärttner. "Time-dependent variational principle for open quantum systems with artificial neural networks". In: *arXiv:2104.00013 [cond-mat, physics:physics, physics:quant-ph]* (Apr. 27, 2021).

[69] C. Harney et al. "Entanglement classification via neural network quantum states". In: *New Journal of Physics* 22.4 (Apr. 3, 2020), p. 045001.

[70] C. Harney, M. Paternostro, and S. Pirandola. "Mixed State Entanglement Classification using Artificial Neural Networks". In: *arXiv:2102.06053 [cond-mat, physics:quant-ph]* (Feb. 11, 2021).

[71] G. Torlai et al. "Precise measurement of quantum observables with neural-network estimators". In: *Physical Review Research* 2.2 (June 17, 2020), p. 022060.

[72] P. Cha et al. "Attention-based Quantum Tomography". In: *arXiv:2006.12469 [quant-ph]* (June 22, 2020).

[73] A. Melkani, C. Gneiting, and F. Nori. "Eigenstate extraction with neural-network tomography". In: *Physical Review A* 102.2 (Aug. 19, 2020), p. 022412.

[74] H. Huang and H. Situ. "Investigating reconstruction of quantum state distributions with neural networks". In: *The European Physical Journal Plus* 136.2 (Feb. 2021), p. 204.

[75] G. Torlai et al. "Integrating Neural Networks with a Quantum Simulator for State Reconstruction". In: *Physical Review Letters* 123.23 (Dec. 6, 2019), p. 230504.

[76] E. S. Tiunov et al. "Experimental quantum homodyne tomography via machine learning". In: *Optica* 7.5 (May 20, 2020), p. 448.

[77] A. M. Palmieri et al. "Experimental neural network enhanced quantum tomography". In: *npj Quantum Information* 6.1 (Dec. 2020), p. 20.

[78] J. Chen et al. "Equivalence of restricted Boltzmann machines and tensor network states". In: *Physical Review B* 97.8 (Feb. 2, 2018), p. 085104.

[79] G. Torlai and R. G. Melko. "Machine-Learning Quantum States in the NISQ Era". In: *Annual Review of Condensed Matter Physics* 11.1 (Mar. 10, 2020), pp. 325–344.

[80] J. Carrasquilla and G. Torlai. "Neural networks in quantum many-body physics: a hands-on tutorial". In: *arXiv:2101.11099 [cond-mat, quant-ph]* (Jan. 26, 2021).

[81] J. Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.2.12. 2018.

[82] J. Heek et al. *Flax: A neural network library and ecosystem for JAX*. Version 0.3.3. 2020.

[83] Supercomputing Support. "JUWELS: Modular Tier-0/1 Supercomputer at Jülich Supercomputing Centre". In: *Journal of large-scale research facilities JLSRF* 5 (Feb. 6, 2019), A135.

[84] G. B. Mbeng, A. Russomanno, and G. E. Santoro. "The quantum Ising chain for beginners". In: *arXiv:2009.09208 [cond-mat, physics:quant-ph]* (Sept. 19, 2020).

[85] C. Lanczos. "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators". In: *Journal of Research of the National Bureau of Standards* 45.4 (Oct. 1950), p. 255.

[86] P. Virtanen et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python". In: *Nature Methods* 17.3 (Mar. 2, 2020), pp. 261–272.

[87] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. Society for Industrial and Applied Mathematics, Jan. 1998. ISBN: 978-0-89871-407-4 978-0-89871-962-8.

[88] F. Mezzadri. "How to generate random matrices from the classical compact groups". In: *arXiv:math-ph/0609050* (Feb. 27, 2007).

[89] J. A. Miszczak. "Generating and using truly random quantum states in Mathematica". In: *Computer Physics Communications* 183.1 (Jan. 2012), pp. 118–124.

[90] W. L. Tan et al. "Observation of Domain Wall Confinement and Dynamics in a Quantum Simulator". In: *Nature Physics* (Mar. 18, 2021).

[91] J. Jin et al. “Phase diagram of the dissipative quantum Ising model on a square lattice”. In: *Physical Review B* 98.24 (Dec. 14, 2018), p. 241108.

[92] J. Johansson, P. Nation, and F. Nori. “QuTiP 2: A Python framework for the dynamics of open quantum systems”. In: *Computer Physics Communications* 184.4 (Apr. 2013), pp. 1234–1240.

[93] S. Geman, E. Bienenstock, and R. Doursat. “Neural Networks and the Bias-Variance Dilemma”. In: *Neural Computation* 4.1 (Jan. 1992), pp. 1–58.

[94] B. Neal et al. “A Modern Take on the Bias-Variance Tradeoff in Neural Networks”. In: *arXiv:1810.08591 [cs, stat]* (Dec. 18, 2019).

[95] Z. Yang et al. “Rethinking Bias-Variance Trade-off for Generalization of Neural Networks”. In: *arXiv:2002.11328 [cs, stat]* (Dec. 7, 2020).

[96] J. D. Hunter. “Matplotlib: A 2D Graphics Environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[97] C. R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 17, 2020), pp. 357–362.

[98] G. N. M. Tabia. “Experimental scheme for qubit and qutrit symmetric informationally complete positive operator-valued measurements using multiport devices”. In: *Physical Review A* 86.6 (Dec. 14, 2012), p. 062107.

A Additional details

A.1 Why does the ARCNN not work in 2D?

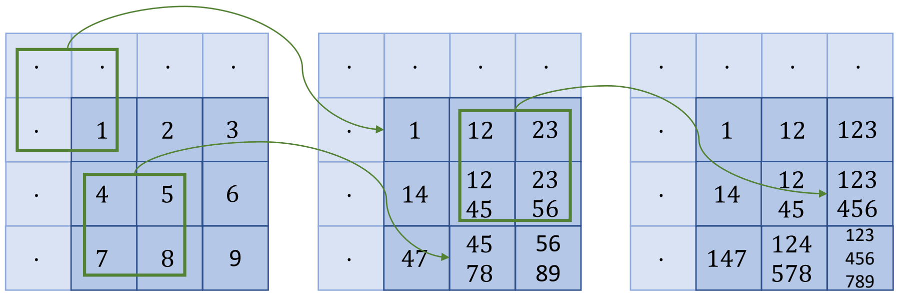

Figure A.1: Schematic representation of the naive autoregressive CNN in two dimensions for a 3×3 system.

Figure A.1 shows a schematic overview of the autoregressive CNN in two dimensions. Instead of POVM outcomes $a_1, ..., a_9$, only the indices are shown. The 3×3 image is again padded with ones, ensuring the size of the image remains constant. Using a 2×2 kernel, only 2 convolutions are required to establish all dependencies, i.e. further convolutions will not create further dependencies between cells.

The dependencies in the first row are fine: one can sample the first site outcome, make a forward pass and the second site will only depend on this first site outcome. This works until the third outcome has been sampled. However, site 4 will never depend on the outcome of site 3, meaning that sites 3 and 4 are never directly correlated. Clearly, the 2D nature of the correlation spread in the CNN and the linear causal dependence required for autoregressive sampling are at conflict with each other. This issue is known as the ”blind spot problem” [44] and to resolve it, some more clever approaches such as those presented in [44] have to be used.

Another possible solution is the following: Any function $P_a(a_1, ..., a_N)$, that maps N discrete inputs to a vector of normalized probabilities P_a ($a = 1, 2, 3, 4$ for most of this thesis), i.e.

$$\sum_{a=1}^{4} P_a(a_1, a_2, ..., a_N) = 1 \quad \forall a_i \in \{1, 2, 3, 4\},$$

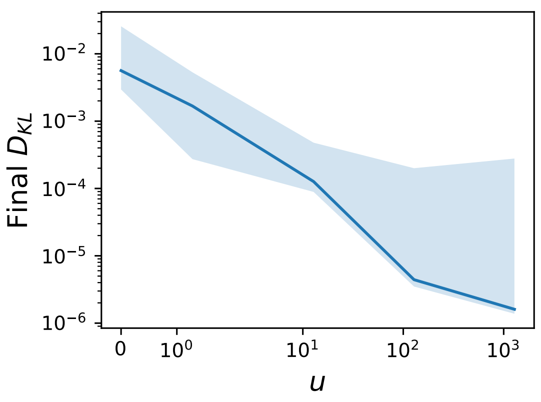

Figure A.2: Simulating dephased targets by adding $u \cdot \mathbb{1}/2^N$ to the target density matrix (and normalizing) for a seven qubit Ising ground state at critical coupling. Trained with CNN in infinite data limit. Stronger dephasing implies a better approximation. (Median, Min, Max shown).

can be made autoregressive in the following way: for c being any constant, $P_a(c, ..., c)$ returns the probabilities of the first site, from which a_1 may be sampled. Plugging this back in yields $P_a(a_1, c, ..., c)$, i.e. four probabilities depending on a_1, from which a_2 may be sampled, etc. The resulting probability distribution is exactly normalized and sampleable, at the cost of requiring N evaluations of P in order to generate a sample or to compute a single N-site probability. This architecture may be seen as a recurrent neural network, that replaces its hidden state with a memory of all previous local outcomes. While this architecture should also work with CNNs, this remains untested in this thesis.

A.2 The influence of purity for Ising states

To further strengthen the result from Sec. 3.1.1, stating that purity negatively correlates with representability, one can simulate dephasing noise by adding a unit matrix to a given target density matrix, i.e. training states of the form

$$\hat{\rho} \propto |\psi\rangle\langle\psi| + \frac{u}{2^N}\mathbb{1}. \tag{A.1}$$

For $u = 0$, this state is pure, for $u = 1$ it is an equal mixture of a pure state and the fully mixed state, and for the limit $u \to \infty$, the state is fully dephased. Figure A.2 shows that the representability monotonously decreases with u, for $|\psi\rangle$ corresponding the 7 qubit Ising ground state at critical coupling.

A.3 Ising scaling and encoding types

Figure A.3 shows the scaling of the representability of the Ising model in a CNN with an N^2 scaling of the number of parameters. The final D_{KL} increases with particle number, showing that this particular scaling of the network is not sufficient

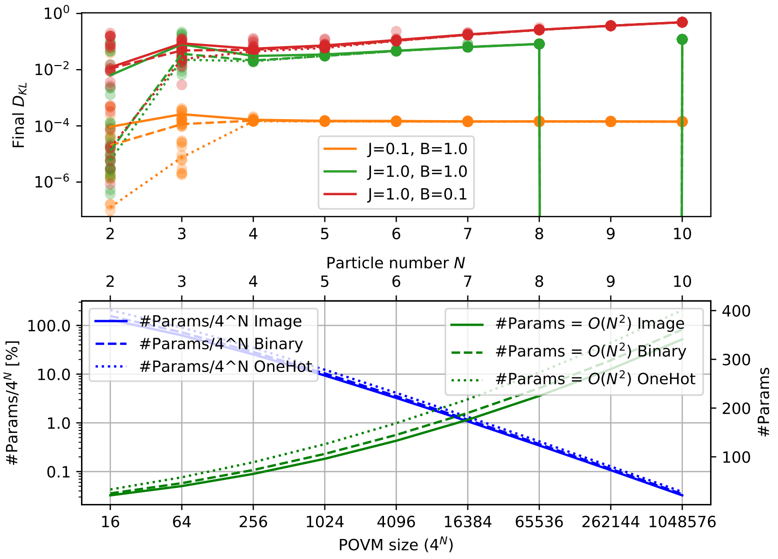

Figure A.3: Training three typical states from the Ising phase transition for different system sizes, using a network scaling of $f = N, k = 2$ and $L = 2$, resulting in an $\mathcal{O}(N^2)$ scaling in hyperparameters. This scaling is not sufficient for Ising ground states. 3 encoding types are shown. The 9 qubit critical state happens to have an exact zero (up to numerical precision) in its POVM distribution, causing an numerical issue in the D_{KL} loss function, hence this datapoint is missing.

to represent the Ising critical and strong coupling states. Section 3.1.2 stated, that one cannot expect to represent a state with correlation length d if the $d_{\max}$ quantity of the network is less than d. As explained previously, this result does not apply here, as a dense output layer was used. Therefore this result is interesting.

The state with small coupling does, however, seem to be representable with this scaling, as its error of representation is bounded. In Fig. A.4, one can see that even a CCNN of constant size, is able to represent this state with constant error for up to 10 qubits. Due to this state being close to a product state and having a short correlation length, this result is not very surprising. Figure A.4 shows that the product-like state can even be represented with a constant number of parameters, for the tested numbers of qubits.

Figure A.3 also serves as an example to show, that the one-hot encoding from Sec. 2.4.2 is superior to the other encoding types, even tough the differences disappear for large particle numbers. Using none of the encoding types from Sec. 2.4.2 is referred to as 'image encoding', as the local POVM outcomes $0-3$ are directly treated as pixel values.

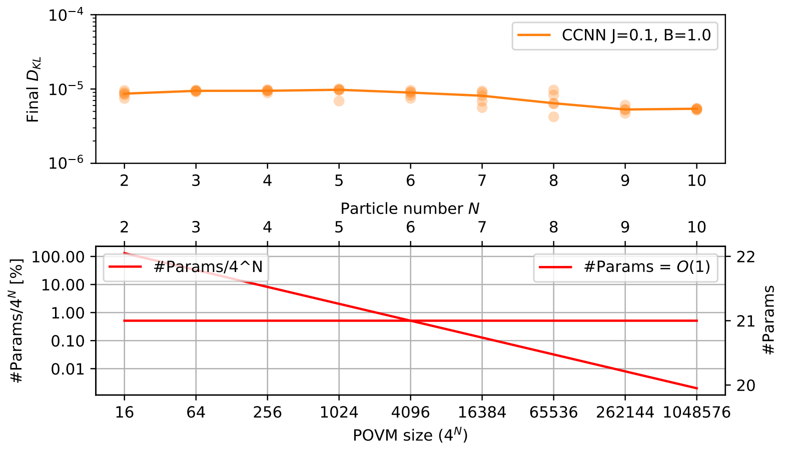

Figure A.4: Training the Ising ground state with small coupling for different system sizes, using the CCNN with of $f = 2, k = 2$ and $L = 1$, resulting in an $\mathcal{O}(1)$ scaling in hyperparameters. Even this constant number of parameters suffices to represent the given state with constant error, due to the very short correlation length of this state.

A.4 Training process

Figures A.5 and A.6 show the evolution of the network performance during the training procedure for the ARCNN and CCNN respectively. The target state is a 16 qubit Ising critical state with periodic boundaries, and the networks are set up with $f = 16, l = 3$ and $k = 5$ and trained using 32k samples. Each panel in the figures shows a different measure of accuracy or an observable. These quantities are explained in Table A.1. Several independent network runs are shown, and observables are also shown as computed directly from the dataset.

Notice that the loss function converges much nicer for the ARCNN than it does for the CCNN. This also reflects in the magnitude of the gradients. In fact, the Ising ground state at strong coupling cannot be trained using the CCNN in this setting, while the ARCNN shows no problems. A possible explanation is the fact that the normalization of the ARCNN does not have to be Monte-Carlo-estimated, as required for the (C)CNN. This is further studied in Appendix A.5.

Section 2.2.4 explains why the half chain purity estimates are so noisy.

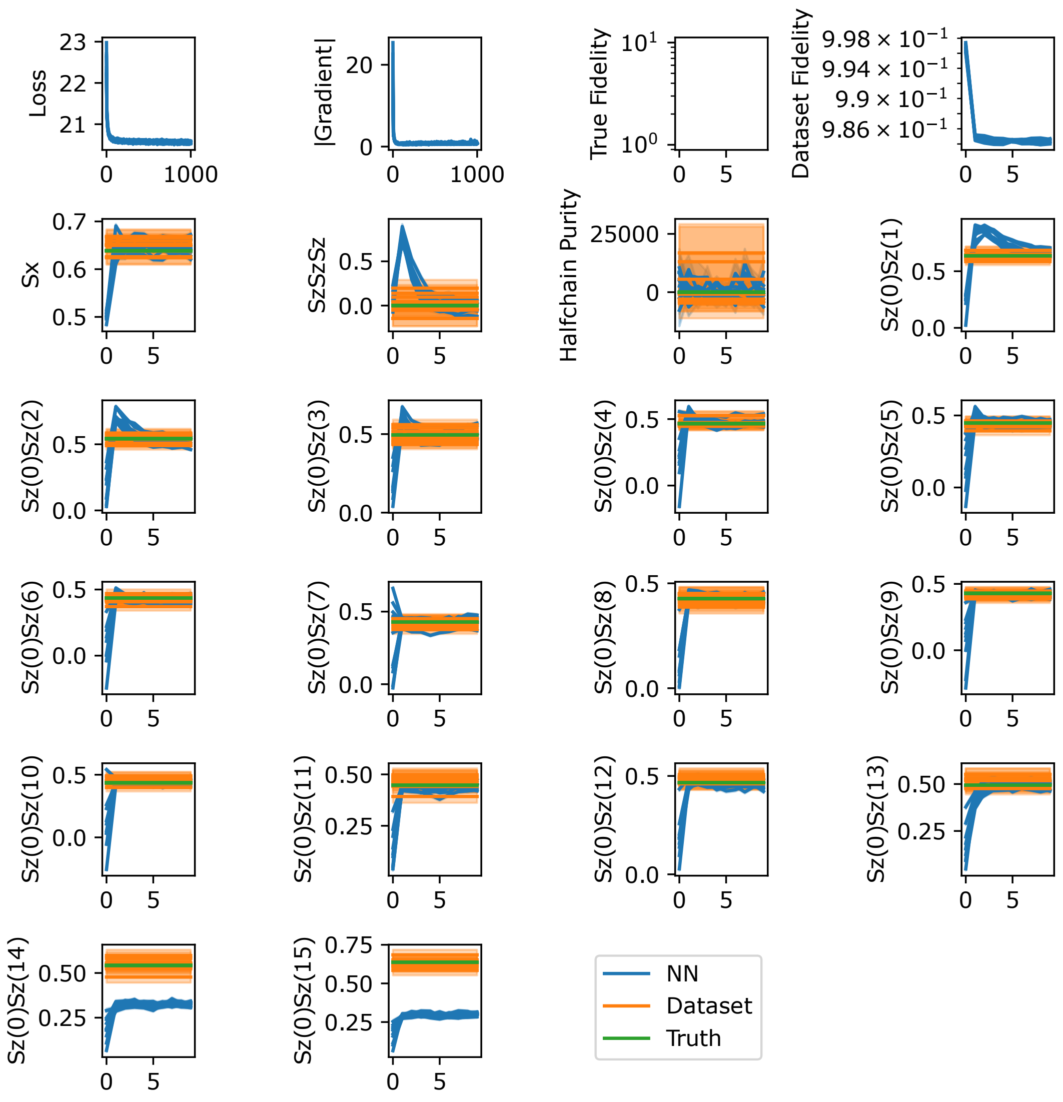

Figure A.5: Tracking the quantities from Tab. A.1 while training the ARCNN on a 16 qubit Ising critical state. Horizontal axis shows number of training steps (first two plots in units, further plots in 100's). The failure of both final 2-point functions to converge can be explained using the correlation bounds from Section 3.1.2 and the huge variance of the purity is explained in Sec. 2.2.4. Shaded error bars are computed using Eq. (2.20).

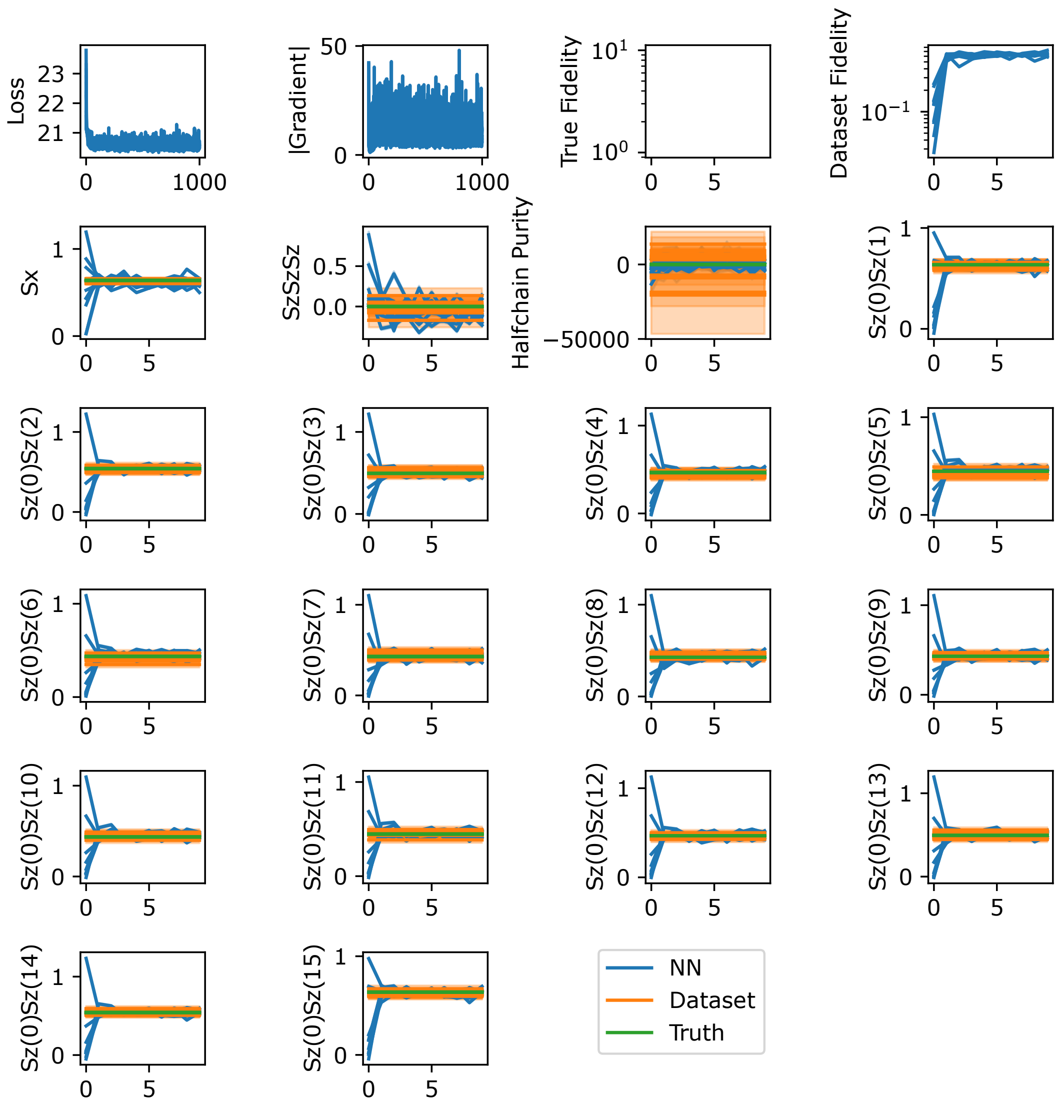

Figure A.6: Tracking the quantities from Tab. A.1 while training the CCNN on a 16 qubit Ising critical state. Horizontal axis shows number of training steps (first two plots in units, further plots in 100's). The huge variance of the purity is explained in Sec. 2.2.4. Shaded error bars are computed using Eq. (2.20).

Name	Description
Loss	Likelihood function used as optimization target
|Gradient|	A measure for the magnitude of the gradient of the loss function w.r.t. the network parameters
True Fidelity	(not shown)
Dataset fidelity	Classical Fidelity (3.5) between NN distribution and dataset distribution
Sx	$\sum_{i=0}^{15} \langle \hat{\sigma}_i^x \rangle$
SzSzSz	$\langle \hat{\sigma}_0^z \hat{\sigma}_1^z \hat{\sigma}_2^z \rangle$
Halfchain Purity	Purity of the reduced density matrix of the first half of the chain
Sz(0)Sz(i)	$\langle \hat{\sigma}_0^z \hat{\sigma}_i^z \rangle$

Table A.1: Descriptions of the quantities shown in Figures A.5 and A.6.

A.5 Is normalization a problem?

If one is interested in applying the tomography scheme to a more general class of neural networks, that is not restricted to generative models, one has to ensure, that computing the normalization of the network-encoded probability distribution does not pose a fundamental scaling limitation. As stated in Sec. 2.4.2, the normalization then has to be estimated using a Monte Carlo estimate (2.27). Since having a normalized probability distribution is fundamental requirement for the optimization of the network parameters to converge, it is a priori not clear, whether replacing an exact normalization with an approximate one, hinders training performance.

To test this, a CNN architecture and target state is chosen as a test subject and the "batch size" from Eq. (2.27) is varied. Figure A.7 shows two physical states being examined, namely the $J/B = 0.1$ and $J/B = 1$ ground states of the 7-site 1D Ising model. The states are approximated for batches sized < 10, up to $\mathcal{O}(4^7)$. For the case where the sample size is equal to 4^7, the normalization is computed exactly, rather than sampled.

For the case of $J/B = 0.1$, the results might look surprising on first sight: the performance of the network does not seem to suffer at all from sampled normalizations: Using as little as 10 % of the POVM size, for normalizing results in the same performance as exact normalization. This may be explained by the fact, that this state has high overlap with a product state, resulting in a POVM distribution that is approximated very well, by a factorized distribution. Normalizing this

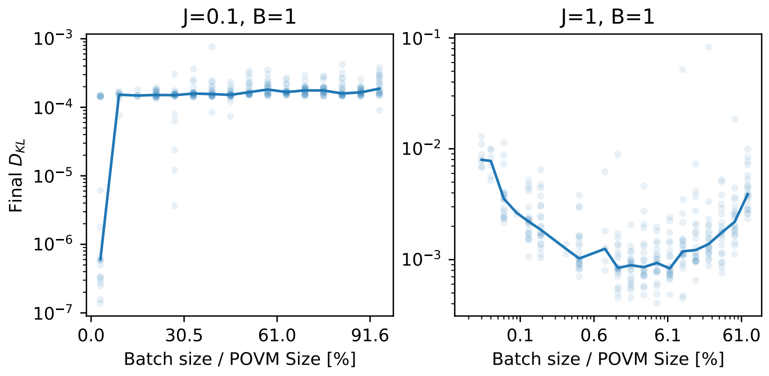

Figure A.7: Investigating the influence of approximate normalization by varying the "normalization batch size" for two Ising ground states for $N = 7$ spins, $f = 7, l = 4$ and $k = 3$.

factorized distribution simply amounts to normalizing one of the factors, which requires much less samples.

For the case of $J/B = 1$, one can see a penalty for a normalization batch that is too small. This effect is one possible explanation of the fact, that the ARCNN is the more stable and thus more powerful architecture, as it does not require the normalization to be estimated.

B Mathematical detours

B.1 Measuring the single-particle SIC-POVM

Any simple single particle measurement amounts to performing a number of unitary operations on a qubit, followed by a Pauli-Z measurement. The latter has two outcomes, which lie on opposing sides of the Bloch-sphere. The SIC-POVM has 4 outcomes, which follow a different geometry on the Block sphere. It is nevertheless possible to measure the SIC-POVM by coupling the target qubit to an ancillar, performing a joint unitary transform U and finally measuring both qubits in the Z-basis. The right choice of unitary transform will ensure, that every joint 2-particle projection is mapped to a corresponding SIC-POVM measurement on the target qubit. The unitary U can be found by solving the equation "SIC-POVM-Probabilities on target" = "Z-basis probabilites on joint system", which reads

$$\mathrm{Tr}\left[(\hat{M}_i \otimes \mathbb{1})\,(\hat{\rho} \otimes |\uparrow\rangle\langle\uparrow|)\right] = \mathrm{Tr}\left[P_i \hat{\rho}'\right] \equiv \mathrm{Tr}\left[P_i \hat{U}^\dagger \left(\hat{\rho} \otimes |\uparrow\rangle\langle\uparrow|\right) \hat{U}\right] \qquad \forall \hat{\rho}, i \tag{B.1}$$

where $P_i \in \{|00\rangle\langle 00|, |01\rangle\langle 01|, |10\rangle\langle 10|, |11\rangle\langle 11|\}$ are the 4 orthonormal projectors for the joint measurement, $\hat{M}_i$ are the single particle SIC-POVM operators and $\hat{\rho}$ is the single particle target density matrix. This can be rewritten as

$$0 = \mathrm{Tr}\left[\left(\hat{M}_i \otimes \mathbb{1} - \hat{U}^\dagger P_i \hat{U}\right)\left(\hat{\rho} \otimes |\uparrow\rangle\langle\uparrow|\right)\right] \qquad \forall \hat{\rho}, i \tag{B.2}$$

$$= \mathrm{Tr}\left[\begin{pmatrix} \hat{A}_i & \hat{B}_i \\ \hat{C}_i & \hat{D}_i \end{pmatrix} \cdot \begin{pmatrix} \hat{\rho} & 0 \\ 0 & 0 \end{pmatrix}\right] = \mathrm{Tr}\left[\hat{A}_i \hat{\rho}\right] \qquad \forall \hat{\rho}, i \tag{B.3}$$

Here, the matrix in the first parentheses in Eq. (B.2) is written using four 2×2 block-matrices $\hat{A}_i, \hat{B}_i, \hat{C}_i$ and $\hat{D}_i$. The final trace in Eq. (B.3) can only be zero for all 2×2 density matrices $\hat{\rho}$ if all $\hat{A}_i$, i.e. the upper left block of $\hat{M}_i \otimes \mathbb{1} - \hat{U}^\dagger P_i \hat{U}$ are the zero matrix. This condition leads to 4 matrix-valued equations, if one makes an ansatz for $\hat{U}$ such as

$$\hat{U} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} \tag{B.4}$$

and computes $\hat{U}^\dagger P_i \hat{U}$ explicitly, resulting in

$$\hat{M}_1 \equiv \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} |a|^2 & a\bar{e} \\ \bar{a}e & |e|^2 \end{pmatrix} \qquad \hat{M}_2 \equiv \frac{1}{6}\begin{pmatrix} 1 & \sqrt{2} \\ \sqrt{2} & 2 \end{pmatrix} = \begin{pmatrix} |b|^2 & b\bar{f} \\ \bar{b}f & |f|^2 \end{pmatrix}$$

$$\hat{M}_3 \equiv \frac{1}{6}\begin{pmatrix} 1 & \sqrt{2}x \\ \sqrt{2}\bar{x} & 2 \end{pmatrix} = \begin{pmatrix} |c|^2 & c\bar{g} \\ \bar{c}g & |g|^2 \end{pmatrix} \quad \hat{M}_4 \equiv \frac{1}{6}\begin{pmatrix} 1 & \sqrt{2}\bar{x} \\ \sqrt{2}x & 2 \end{pmatrix} = \begin{pmatrix} |d|^2 & d\bar{h} \\ \bar{d}h & |h|^2 \end{pmatrix}$$

using the definitions of the SIC-POVM operators and $x = e^{\frac{2\pi i}{3}}$. One possible solution of this system of equations can easily be found:

$$\hat{U} = \frac{1}{\sqrt{6}}\begin{pmatrix} \sqrt{3} & 1 & 1 & 1 \\ 0 & \sqrt{2} & \sqrt{2}x & \sqrt{2}\bar{x} \\ i & j & k & l \\ m & n & o & p \end{pmatrix}$$

Demanding that $\hat{U}$ is unitary $\hat{U}^\dagger\hat{U} = \mathbb{1}$ gives more conditions on the lower half of $\hat{U}$, to which an analytic solution may be found by hand:

$$\hat{U} = \frac{1}{\sqrt{6}}\begin{pmatrix} \sqrt{3} & 1 & 1 & 1 \\ 0 & \sqrt{2} & \sqrt{2}x & \sqrt{2}\bar{x} \\ -\sqrt{3} & 1 & 1 & 1 \\ 0 & \sqrt{2}\bar{x} & \sqrt{2}x & \sqrt{2} \end{pmatrix} \tag{B.5}$$

is unitary and maps the SIC-POVM for one qubit onto 4 orthonormal projectors on 2 qubits. An independent derivation of an equivalent matrix can be found in [98], where also a decomposition into standard gates is given.

B.2 Parametrizing positive POVM distributions

As said in the main text, the mapping between density matrices and probability distributions is not bijective: While every density matrix has a corresponding POVM distribution, not every probability distribution corresponds to a *positive* density matrix. If a distribution corresponds to a positive density matrix, one may call the distribution ”positive”. Fuchs and Schack show how to parametrize the positive POVM distributions in [30] for SIC-POVMs. However these results can easily be generalized to all POVMs as follows: Any positive, hermitian matrix $\hat{\rho}$ can be written in terms of its square root, which may be expanded using the POVM operators:

$$\hat{\rho} = \hat{B}^2 = (b_i\hat{M}_i)^2 = b_i b_j \hat{M}_i\hat{M}_j, \qquad \text{with } b_i \ 4^N \text{ real numbers.} \tag{B.6}$$

Computing the POVM probabilities is straight forward:

$$P_a = \mathrm{Tr}\left[\hat{M}_a\hat{\rho}\right] = b_i b_j \,\mathrm{Tr}\left[\hat{M}_a\hat{M}_i\hat{M}_j\right] = b_i b_j c_{aij} = \langle b| C_a |b\rangle \qquad \text{with} \tag{B.7}$$

$$(C_a)_{ij} = c_{aij} = \mathrm{Tr}\left[\hat{M}_a\hat{M}_i\hat{M}_j\right]. \tag{B.8}$$

Plugging any $|b\rangle \in \mathbb{R}^{4^N}$ into this expression results in a positive POVM distribution. The c_{aij} tensor factorizes into local tensors if the POVM is a product POVM, i.e. all indices are replaced by multiindices. So far the resulting POVM distribution Eq. (B.7) is not normalized, but this is easily taken into account:

$$1 = \sum_a P_a = b_i b_j \,\mathrm{Tr}\left[\sum_a \hat{M}_a\hat{M}_i\hat{M}_j\right] = b_i b_j \,\mathrm{Tr}\left[\hat{M}_i\hat{M}_j\right] = b_i b_j T_{ij} = \langle b| T |b\rangle\,. \tag{B.9}$$

For the product-Pauli and product-SIC POVM, T is an element-wise-positive, positive definite, symmetric matrix. This implies that the normalized $|b\rangle$ vectors lie on an ellipsoid. By splitting $T = \sqrt{T}\sqrt{T}$ into two factors, the normalization looks a bit nicer

$$1 = \langle b| T |b\rangle = \langle x|x\rangle\,, \qquad \text{with } |x\rangle = \sqrt{T}\,|b\rangle\,, \tag{B.10}$$

i.e. the $|x\rangle$ vectors lie on a hypersphere in a *real* Hilbert space of $2N$ qubits. One can verify that sparse, local observables remain sparse and local when translated to the $|x\rangle$ language.
Looking back at the parametrization of $\hat{\rho}$, it becomes

$$\hat{\rho} = x_i x_j \sqrt{T^{-1}}_{im}\sqrt{T^{-1}}_{jn}\hat{M}_m\hat{M}_n. \tag{B.11}$$

This is an expansion of the square root of the density matrix using the operators $\hat{M}'_i = \sqrt{T^{-1}}_{im}\hat{M}_m$ which form an orthonormal basis. This can be seen by

$$\mathrm{Tr}\left[\hat{M}'_i\hat{M}'_j\right] = \sqrt{T^{-1}}_{im}\sqrt{T^{-1}}_{jn}\,\mathrm{Tr}\left[\hat{M}_m\hat{M}_n\right] = \left(\sqrt{T^{-1}}T\sqrt{T^{-1}}\right)_{ij} = \delta_{ij},$$

since T is diagonalizable and $\sqrt{T^{-1}} = \left(\sqrt{T}\right)^{-1}$ commutes with T.
If the $\hat{M}'_i$ are an orthonormal basis, they cannot be a POVM [30]. Thus one could have started out by choosing any other arbitrary orthonormal basis for representing the square root of $\hat{\rho}$. Also, the parametrization Eq. (B.7) is not useful from a numerical standpoint, because computing a single probability requires the contraction $b_i b_j c_{aij}$, which in infeasible as c_{aij} is not sparse.

Figure		N	NN type	Num. Layers L	Kernel size k	Features/ layer f	Encoding	Num. Param.	Num. Samples	Learn rate	Batch size	Weight decay μ
2.2	CPU/GPU Speedup	10	CNN	5	4	N	1-Hot	1481	-	5e-4	10k	0
3.1	Random states	6	CNN	2	6	N	Image	301	-	2e-3	4k	0
3.2	TFIM phase trans.	4	CNN	2	4	N	Image	105	-	1e-3	2k	0
3.3	TFIM N^3 scaling	3-10	CNN	4	$\lfloor N/2 \rfloor$	N	Image	52-1691	-	2e-3	2k	0
3.4, 3.5	Correlation Propag.	16	see Fig.	see Fig.	see Fig.	N	1-Hot	97-3572	32k	5e-4	5k	0.01
3.6	Small 1D TFIM	3-8	CCNN	$\left[\sqrt{N}\right]$	$\left\lceil\sqrt{N}\right\rceil + 1$	N	1-Hot	73-673	see Fig.	1e-3	4k	0
3.7	Larger 2D TFIM	4×4	CCNN	2	2	N	1-Hot	1329	see Fig.	5e-4	5k	0.01
3.8	Ion Chain	16	ARCNN	3	6	N	1-Hot	3572	10k	5e-4	5k	0.01
3.9	Steady States	4×4	CNN	2	3	N	1-Hot	3169	1k/100k	5e-4	5k	0.01
3.10	Experim. matrix plots	5	CNN	2	4	10	1-Hot	631	24k	5e-4	2k	0
3.11	Experim. resampling	5	CNN	2	4	10	1-Hot	631	see Fig.	5e-4	4k	0
A.2	dephased TFIM	7	CNN	2	7	N	Image	456	-	2e-3	2k	0
A.3	TFIM N^2 scaling	2-10	CNN	2	2	N	see Fig.	21-401	-	1e-3	2k	0
A.4	TFIM const. scaling	2-10	CCNN	1	2	2	1-Hot	21	-	1e-3	2k	0
A.5	Training logs ARCNN	16	ARCNN	3	5	N	1-Hot	2996	32k	5e-4	5k	0.01
A.6	Training logs CCNN	16	CCNN	3	5	N	1-Hot	2945	32k	5e-4	5k	0.01
A.7	Scan batch size	7	CNN	4	3	N	Image	540	-	2e-3	see Fig.	0

Table B.1: Hyperparameters used for each figure. In all cases we used a tanh-activation function and the (C)CNN uses an exponential output function. $[x]$ denotes the nearest integer function, $\lceil x \rceil$ the ceiling function and $\lfloor x \rfloor$ the floor function. Parameters for the ADAM algorithm are kept at their default values $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

Erklärung:

Ich versichere, dass ich diese Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den 18.10.2021 [illegible]