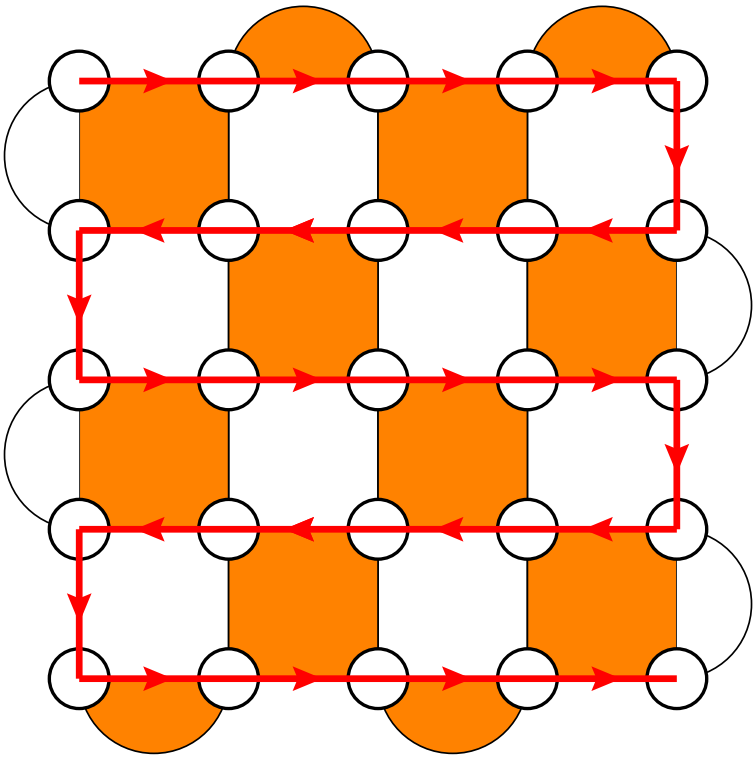

Quantum Error Correction using Variational Neural Annealing

Decoding surface code syndromes with Recurrent Neural Networks

Master's thesis in Complex Adaptive Systems

Axel Prebensen

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Quantum Error Correction using Variational Neural Annealing

Decoding surface code syndromes with Recurrent Neural Networks

AXEL PREBENSEN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2023

Quantum Error Correction using Variational Neural Annealing
Decoding Surface Code Syndromes with Recurrent Neural Networks
Axel Prebensen

Supervisor & Examiner: Mats Granath, Department of Physics

Master's Thesis 2023
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A surface code of size $d = 5$ with the path taken by the RNN visualized as a red arrow.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023 2023

Quantum Error Correction using Variational Neural Annealing
Decoding surface code syndromes with Recurrent Neural Networks

Axel Prebensen
Department of Physics
Chalmers University of Technology

Abstract

Ever since the theoretical possibility of quantum computers was discovered an effort to create the practical possibility has been ongoing. The challenge in creating a quantum computer lies mostly in the extremely sensitive and error prone quantum bits (qubits) making up the basis of the quantum computation. One of the most promising approaches to solve the issues of the sensitive qubits is quantum error correction, which aims to correct for errors rather than eliminating them altogether. One way to do error correction is to create a logical qubit consisting of multiple physical qubits, where errors on a physical qubit can be corrected to preserve the logical qubit from errors. The most common way to achieve this is by implementing a two-dimensional surface code, where a grid of qubits represent a single logical qubit.

In order to decode this surface code and correct for errors we need to measure error syndromes on the surface code and decide which qubit error is most likely to cause that syndrome. In this thesis a new and alternative solution to decode with an algorithm called Variational Neural Annealing (VNA) [Hibat-Allah et al, *Nature Machine Intelligence*, 3, 952 (2021)] is investigated. This an optimization technique that uses a Recurrent Neural Network (RNN). Its viability as a decoder is determined by its accuracy and runtime. Both a two-dimensional and a one-dimensional RNN structure is studied on a surface code with code distances 3 and 5. The results of the study show that its viability as a decoder is limited in all tested configurations, obtaining lower accuracy than comparable models. Alternative methods and approaches are presented that show more promising results, while still performing under par. The study is concluded by speculating on additional alternative approaches for further study on algorithms utilizing RNNs for quantum error correction.

Keywords: Quantum error correction, Variational Neural annealing, Recurrent neural networks, Surface Code.

Acknowledgements

First and foremost I would like to thank Mats Granath for the guidance and support during the course of the project. His input made this work possible and made the journey a great learning experience from start to finish. I would also like to thank Mohamed Hibat-Allah from the Vector Institute for answering questions about his work with variational neural annealing. Lastly I would like to thank my fellow students and friends for studying with me and making my last months at Chalmers some of the best.

Computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) and the Swedish National Infrastructure for Computing (SNIC) at Chalmers Centre for Computational Science and Engineering (C3SE), partially funded by the Swedish Research Council through grant agreements no. 2022-06725 and no. 2018-05973.

Axel Prebensen, Gothenburg, May 2023

Contents

List of Figures **xi**

List of Tables **1**

1 Introduction **3**
1.1 Thesis aim and goal 4

2 Theory **5**
2.1 Quantum computing/Quantum bits 5
2.1.1 Quantum error correction 7
2.1.2 Surface code 7
2.1.3 Syndrome measurement and decoding 8
2.1.4 Equivalence classes 9
2.2 Machine Learning and Artificial Neural Networks 9
2.2.1 Simulated annealing 10
2.2.2 Artificial neural networks 10
2.2.2.1 Recurrent neural networks 10
2.2.3 Variational neural annealing 11

3 Methods **15**
3.1 The environment for the RNN 15
3.1.1 Qubit and syndrome matrix 15
3.1.2 The variational free energy function 16
3.1.3 Optimizers 16
3.1.4 Gpu cluster and environment 16
3.2 The RNN 17
3.2.1 Structure 2D model 17
3.2.2 Structure 1D dilated model 18
3.2.3 Extra Inputs 18
3.3 Reference decoder 18

4 Results and discussion **21**
4.1 VNA for any general syndrome 21
4.2 VNA for a specific syndrome 22
4.2.1 Performance compared to reference decoder 23
4.2.2 Runtime 25

5 Conclusions **27**
5.1 Decoding performance 27
5.2 Runtime advantages 27
5.3 Alternatives for further research 28
5.3.1 Energy function 28
5.3.2 2D dilated RNN 28
5.3.3 Other methods 28

6 Ethical discussion **31**

Bibliography **33**

List of Figures

2.1 An illustration of a Bloch Sphere where the qubit state can be seen as a unit vector. A Pauli operator performed on that vector can be seen as a half-rotation about the axes $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$. The image is sourced from [19] 6
2.2 A 5×5 surface code, where we have two types of stabilisers. The blank faces consists of Z-type stabilisers measuring the parity of X-errors, signalling an error if the parity is odd. The orange faces in a similar fashion consists of X-type stabilisers measuring the parity of Z-errors. 8
2.3 Three different possible error chains that all result in the same syndrome measurement 8
2.4 The logical X operator X_L to the left and the logical Z operator Z_L to the right, any error chain that anti-commutes with either these will cause a logical error. 9
2.5 RNN presented in rolled for on the left and unrolled form on the right 11

3.1 A d=5 surface code with the path of the RNN visualized as a red line with arrows. 17
3.2 A d=5 surface code with the path of the RNN visualized as a red line with arrows, together with an example of which syndromes are used as an input to the first state of the RNN. All four surrounding syndrome measurements are used as an input at every step. Grey squares indicate dormant syndromes that are always 0. 19

4.1 Training progress for a 2D RNN on a $d = 3$ surface code measured in both sample error rate, p_s, and in energy while training on a single syndrome. 22
4.2 Training progress while training on a multiple (randomly sampled) syndromes 23
4.3 Performance of the two RNN models compared to the EWD decoder on a $d = 3$ surface code. 24
4.4 Performance of the two RNN models compared to the EWD decoder on a $d = 5$ surface code. 24

List of Tables

4.1 Table of runtime for the different methods, both on $d = 3$ and $d = 5$ with 200 samples. Note that the runtimes were tested once and not averaged due to the long runtime of some of the methods. 25

1
Introduction

Quantum computing has been a dream for physicists and computer scientists for decades [1], despite the mountain of challenges to overcome in order to create a functioning quantum computer. The dream has been thought to be more and more possible in later years with multiple companies and universities actively building quantum computers. The goal for all these institutions is to create a quantum computer running calculations on quantum bits (qubits). Qubits are analogous to bits in a classical computer but made to exist in a superposition. Despite these efforts the main challenge in building a quantum computer remains, making it fault tolerant. This is a challenge since the currently used technologies made to represent qubits all have something in common, which is that they are very prone to error.

One of the main reasons quantum computing is coming closer to reality is the field of quantum error correction. Using quantum error correction one could ideally reduce the amount of noise that individual qubits are prone to. One way to do this is by having multiple qubits entangled to realise one logical qubit. This means that errors on the individual qubits can be corrected, giving the logical qubit a lower error rate. One way to implement a logical qubit is by a 2D grid of qubits called a surface code [2], where errors can be detected.

When trying to correct errors in this way one has to decode the surface code by deciding which error is most likely. This task can be really complicated, and grows more and more complex when the surface code grows larger and larger. To decode can thus be computationally expensive, and it is an area where machine learning algorithms are utilized to make the process more efficient. Machine learning methods involving neural networks have shown promising results in decoding surface codes [3, 4, 5, 6, 7], and it is an active area of research in the field of quantum error correction.

The reason for trying to create a quantum computer in the first place is since there exists quantum algorithms that would greatly outperform any classical computer, the most famous being Shor's Algorithm [13]. Beside Shor's Algorithm quantum computing would allow for significant advances in several fields, including Chemistry [14], Cryptography [15] and Machine Learning in general [16]. It would also be a great advancement that hopefully opens up many doors to solutions and algorithms to be used in areas currently not discovered.

1.1 Thesis aim and goal

The aim of the thesis is to investigate the viability of using variational neural annealing as described by [8] as a decoder for syndromes on a error correcting surface code. This new type of decoder would hopefully provide and interesting alternative to other near-optimal decoders [9, 10, 11, 12]. The practical viability will be determined by its accuracy and runtime as compared to a reference decoder. The goal is that the recurrent neural network will pose a significant speed up and could be a faster alternative to the reference decoder. However if accuracy is sacrificed for a faster run time its viability will be limited.

2

Theory

In this chapter we will explore the topics of quantum computing and quantum error correction, providing an overview of the relevant theory and concepts. Additionally we will delve into artificial neural networks and machine learning, and lastly variational neural annealing as a learning model for error correction, shedding light on its potential role as a decoder.

2.1 Quantum computing/Quantum bits

In a classical computer, the basic unit of information is the bit, which can take on one of two states, 0 or 1. This is typically implemented using a transistor that can be in one of two states, open or closed. All calculations in a classical computer are performed using these binary bits. In a quantum computer on the other hand, the basic unit of information is the qubit, which can exist in a superposition of two states. This means that a qubit can be in both the 0 and 1 states simultaneously. The state of a qubit can be represented the wave function:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.1}$$

where α and β are complex numbers with the constraint $|\alpha|^2 + |\beta|^2 = 1$. This superposition state is what lies behind the theoretical superior computing power of the quantum computer, since it allows multiple qubits to be entangled together. This entanglement means that two qubits can be dependent on each other which will increase the number of possible states, unlike the classical bits which are completely independent. An example of of a case where this provides superior computing power is the quantum Fourier transform, which can perform a discrete Fourier transform on $\mathcal{O}(n^2)$ gates. The classical counterpart, the Fast Fourier transform (FFT), uses $\mathcal{O}(n2^n)$ gates [18]. The superposition of the qubits are often visualised using the Bloch sphere, which provides a geometric representation of the qubit state. An illustration of a Bloch sphere can be seen in figure 2.1.

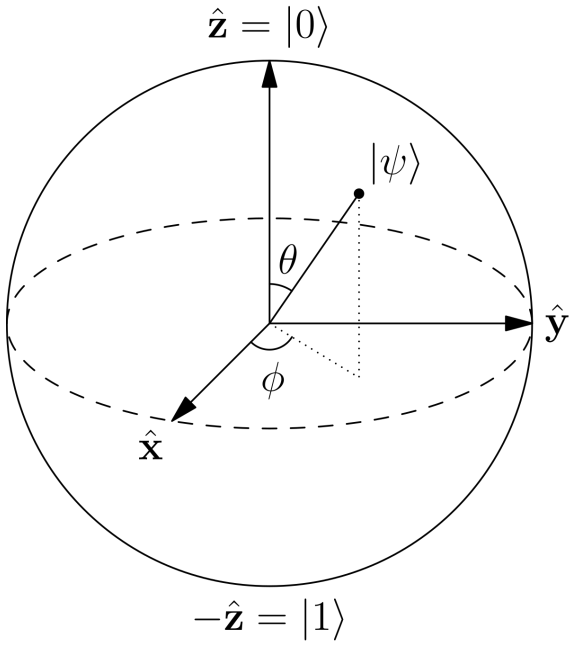

Figure 2.1: An illustration of a Bloch Sphere where the qubit state can be seen as a unit vector. A Pauli operator performed on that vector can be seen as a half-rotation about the axes $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$. The image is sourced from [19]

Central to quantum computing are the Pauli operators, which are a set of three operations that can be viewed as half-rotations around each of the three Cartesian axes (x, y, and z). The identity operator denoted by σ_I is also central and will be considered a fourth operator. In matrix form, the Pauli operators are:

$$\sigma_X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \sigma_Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad \sigma_Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad \sigma_I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.2}$$

From here the Pauli matrices will be referred to as simply X, Y, Z and I. Applied to our qubit state the X operator will perform a bit flip:

$$X(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle + \beta|0\rangle \qquad \text{or} \qquad X(|0\rangle) = |1\rangle, X(|1\rangle) = |0\rangle \tag{2.3}$$

And the Z operator will perform a phase shift:

$$Z(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle - \beta|1\rangle \qquad \text{or} \qquad Z(|0\rangle) = |0\rangle, Z(|1\rangle) = -|1\rangle \tag{2.4}$$

The Y operator performed on our qubit state will perform both operations ($Y = iXZ$) and thus the Y operator will not be central to this work since the error correction can be performed without it. The X operator is the error one could observe, very rarely, in a classical computer. In that case it would be a simple flip from 0 to 1 or from 1 to 0, and it happens with a probability on the order of $p = 10^{-19}$ or lower due to robust error correcting mechanisms [20]. In the best current quantum computers however, error still occur with a probability around $p = 10^{-2}$ more specifically around 2.9% for Google AI [3].

All Pauli operators are Hermitian (meaning they are equal to their own complex conjugate transpose), unitary (meaning they preserve the length and orthogonality of vectors), and either commute or anti-commute with each other. They are also self-inverse (meaning $X^2 = Y^2 = Z^2 = I$) and have $det = -1$ and $tr = 0$. These properties make the Pauli operators useful for manipulating qubits.

2.1.1 Quantum error correction

Since quantum bits are very fragile and are extremely sensitive to noise the physical quantum computers existing today operate in temperatures close to absolute zero. Despite this errors and decoherence occur frequently, which is the reason for trying to correct for errors rather than trying to eliminate them altogether. The simplest form of a quantum error correcting code would be the 3 qubit bit flip code, where 3 qubits together create a logical qubit:

$$|\psi_L\rangle = \alpha|000\rangle + \beta|111\rangle \tag{2.5}$$

The logical qubit here would be able to handle a bit flip error by checking the parity of two qubits in the first and second term. To do this we need to add a ancillary qubit doing a Z measurement:

$$(Z \otimes Z)|00\rangle = |00\rangle \quad (Z \otimes Z)|11\rangle = |11\rangle \tag{2.6}$$
$$(Z \otimes Z)|01\rangle = -|01\rangle \quad (Z \otimes Z)|10\rangle = -|10\rangle \tag{2.7}$$

Here we can see that if we detect a odd parity (eigenvalue -1) we have found an error. This is the way we can notice any errors at all, without revealing information about the qubit and collapse the superposition. Doing this measurement on 2 qubits in each of the terms we can identify which error has occurred at correct for it. We can call this measurement with a ancillary qubit the most rudimentary stabilizer code possible. A corresponding error correcting code for Z (phase flip) errors can be constructed in a similar way. Notice that this simple 3 qubit correcting code causes a logical error as soon as two or more errors are present at the same time. We therefor say that a general N qubit bit flip or phase flip code protects up to $(N-1)/2$ errors of that specific type.

2.1.2 Surface code

A surface code is a quantum error-correcting code that instead of using multiple qubits in a row, uses a two-dimensional grid to create a logical qubit protected against both X and Z errors. The surface code is a powerful tool for correcting errors and is currently one of the most promising approaches to realising large-scale quantum computing [21, 2]. Central to the surface code are the ancillary qubits measuring the parity among 4 qubits in the grid, encoding information in a subspace of the Hilbert space spanned by multiple qubits. The specific surface code shown in 2.2 is called a rotated surface code, which rather than having qubits on the edges have them on the corners of the grid, which is a technique to reduce the amount of physical qubits needed to create a logical qubit.

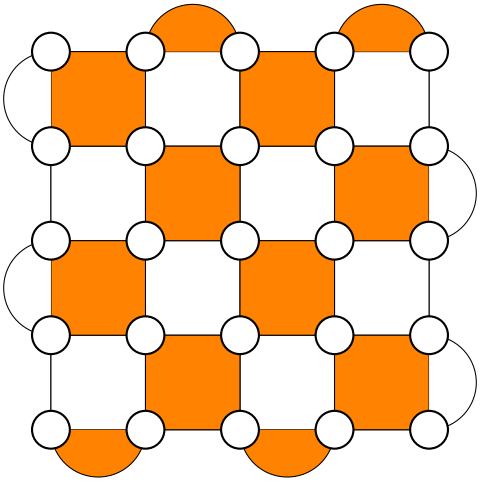

Figure 2.2: A 5×5 surface code, where we have two types of stabilisers. The blank faces consists of Z-type stabilisers measuring the parity of X-errors, signalling an error if the parity is odd. The orange faces in a similar fashion consists of X-type stabilisers measuring the parity of Z-errors.

2.1.3 Syndrome measurement and decoding

As mentioned earlier, while looking for errors on the surface code one needs to use stabilisers. The reason is due to the fact that checking the qubit mid calculation would collapse its state into either a 0 or 1, ruining the calculation. It is also not possible to duplicate or clone the qubit due to the no-cloning theorem. Therefore the best one can do to find an error is to use the mentioned approach of measuring stabilizers. The issue with this is that the syndrome measurement we get from the different stabilisers can come from many different configurations of qubit-errors, called error chains.

Although one single error chain causes the same type of syndrome measurement every time, any syndrome measurement can come from a large variety of error configurations. Meaning that the difficulty with decoding the surface code lies in trying to figure out which error chain is most likely to produce the current syndrome.

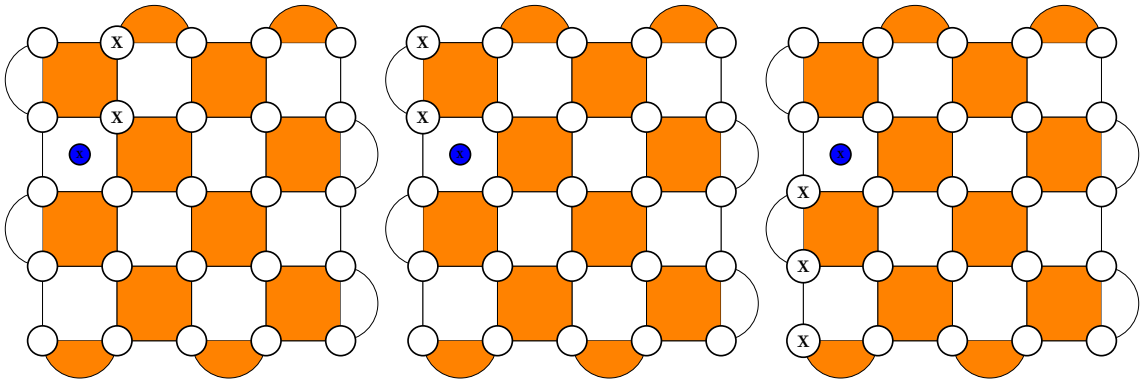

Figure 2.3: Three different possible error chains that all result in the same syndrome measurement

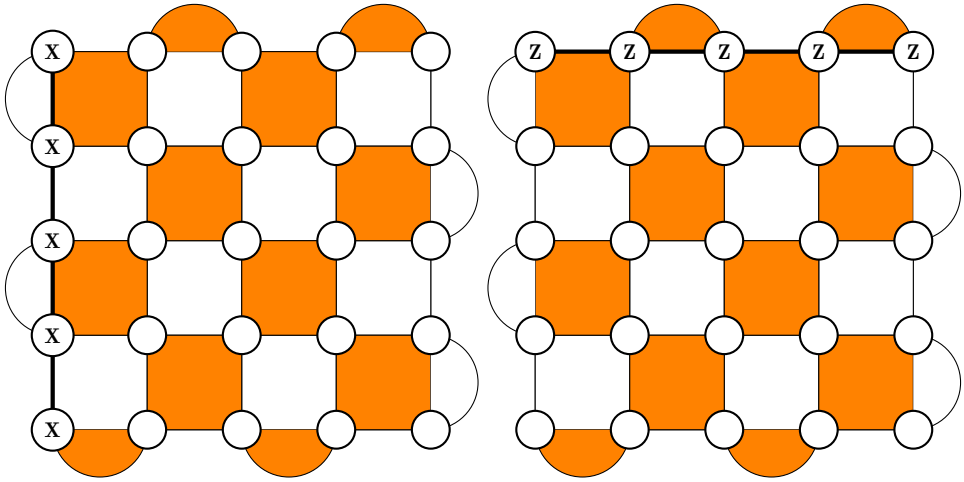

Figure 2.4: The logical X operator X_L to the left and the logical Z operator Z_L to the right, any error chain that anti-commutes with either these will cause a logical error.

2.1.4 Equivalence classes

All error chains on a surface code can be divided up into four equivalence classes. These classes will depend on how you specify the logical operators on the surface code. In this work the leftmost edge qubits will be used to represent the logical X operator X_L and the top edge will be used to represent the logical Z operator Z_L (see figure 2.4). The equivalence classes of any error chain can be determined solely by whether or not they commute with the logical operators.

The reason why we classify the errors into the equivalence classes chain is due to the fact that if the chosen correction is in the same equivalence class as the actual error the decoding is considered successful. Correcting for a error chain that is in a different equivalence class than the actual error chain causes a logical error, meaning the decoding is considered unsuccessful. As an example this means that in figure 2.3 the leftmost error corrected as if it was the rightmost error would cause a logical error. Therefor the basis of the results will be how accurate our algorithm is choosing error chains in the correct equivalence class.

2.2 Machine Learning and Artificial Neural Networks

The field of machine learning and artificial neural networks has been a hot topic for over a decade, leading to many advancements in many fields that involves data analysis [22] and pattern recognition [23]. Machine learning focuses on developing algorithms and models that enable computers to learn from data and improve their performance without being explicitly programmed to do so. By extracting meaningful patterns and insights from data, machine learning has revolutionised various industries, including finance, healthcare, and most notably various algorithms used

in social media platforms.

2.2.1 Simulated annealing

While not specifically in and of itself being a machine learning technique, simulated annealing is a optimisation technique that has been widely used to find the global optima of complex functions [24]. It was first introduced in a seminal paper in 1983 [25], where the concept of annealing in metallurgy was applied to other optimisation problems where the goal is to find the lowest energy state. Simulated annealing is based on the idea of gradually cooling a system from a high temperature to a low temperature, allowing it to settle into a state of minimum free energy.

At each temperature, the algorithm explores the search space by making small random changes to the current solution, accepting moves that improve the objective function and occasionally accepting moves that worsen it. This allows simulated annealing to escape local minima and converge towards the global optimum. Since its inception, simulated annealing has been applied to a wide range of optimisation problems, including scheduling, routing, machine learning, and computer vision, among others [26, 27].

2.2.2 Artificial neural networks

Artificial neural networks are computer codes inspired by the way that biological brains work, using an idealised version of a neuron [28]. The neurons are interconnected in layers to create the network, having an input layer, often one or multiple hidden layers, and then an output layer. This idea of trying to mimic neural behaviour was introduced by Warren McCulloch and Walter Pitts all the way back in 1943 [29], and since then the idealised neuron is often referred to as a McCulloch-Pitts neuron. An example of a simple neuron that is connected to n number of neurons in a previous layer can have the output:

$$y = F(\sum_{i=1}^{N} w_i x_i + b) \tag{2.8}$$

Where w_i denotes the weights of the connections between the neurons, x_i is the input from the neuron in the previous layer, b denotes the bias/threshold of the neuron and $F(...)$ denotes the activation function. An activation function is often non linear in order to model complex relationships and capture non-linear patterns in data but can also be linear.

2.2.2.1 Recurrent neural networks

Recurrent neural networks (RNNs) has become a popular type of neural network for processing sequential data, such as time series or natural language. RNNs are capable of capturing the temporal dependencies in data by maintaining a hidden state that evolves over time. The hidden state h_t for each site n can be calculated as:

$$h_t = F_h(W_h x_t + U_h h_{t-1} + b_h) \tag{2.9}$$

With an output:

$$y_t = F_y(W_y h_t + b_t) \tag{2.10}$$

Where U_h,W_h and W_t are weights, x_t is an input vector, $F_h(..)$ and $F_y(..)$ are two activation functions. Here we can spot the advantage of the RNN, the dependency of the previous state h_{t-1} in the calculation of the current state h_t. This is what gives RNNs the ability to predict time series and learn sequential data. Equation 2.9 can also be modified to take input from two previous states, making it more suitable to a 2-dimensional grid. We can visualise the RNN as either rolled or unrolled, see figure 2.5.

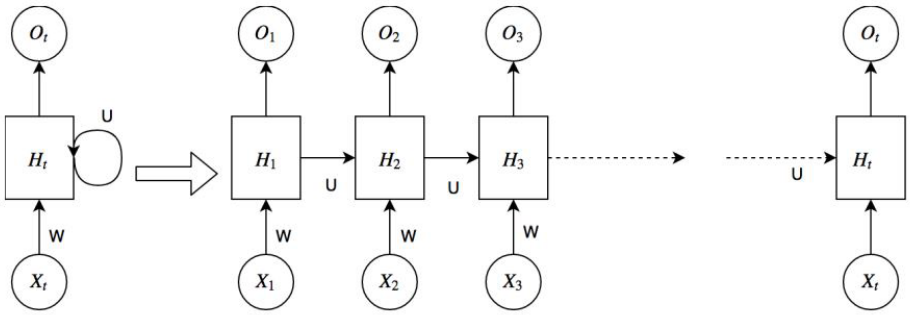

Figure 2.5: RNN presented in rolled for on the left and unrolled form on the right

In equation 2.9 all time steps are done consecutively which means each step depend on the previous hidden state, but this does not have to be the case. In a dilated RNN structure each time step can depend on a hidden state from many time steps ago, thereby expanding the effective receptive field of the network. This would mean h_{t-1} could change to another state, often depending on which layer of the network it is. It is a technique to capture more long term dependencies in the data, which means long-range interactions play a more important role.

2.2.3 Variational neural annealing

Central to this work is the concept of Variational Neural Annealing (VNA) as described in [8]. Variational neural annealing is a technique for solving optimization problems using a combination of RNNs and simulated annealing. The main idea is to solve the problem of the prohibitively slow sampling dynamics in a glassy or rough landscape that simulated annealing has. So the technique uses recurrent neural networks to provide a near ideal parameterization.

Using a RNN gives us the added benefit of being able to provide samples without slow dynamics, even though the landscape is rough. The problem being solved is trying to find the lowest energy configuration of a random Ising Hamiltonian (with

N nodes) of the form:

$$H_{Target} = -\sum_{i<j} J_{ij}\sigma_i\sigma_j - \sum_{i=1}^{N} h_i\sigma_i \tag{2.11}$$

where σ_i are variables that are defined on the nodes of a graph. The couplings J_{ij} and the fields h_i together with the topology of the graph uniquely encode the specific optimisation problem. To find the optimal set $\{\sigma_i\}$ that minimise H_{Target} the model slowly anneals the variational free energy of the system, which is defined by:

$$F_\lambda(t) = \langle H_{Target}\rangle - T(t)S(p_\lambda) \tag{2.12}$$

Where λ denotes the variational parameters, p_λ (or rather $p_\lambda(\sigma)$) denotes the probability distribution of σ, T(t) denotes the temperature at time t being lowered at each time step, the braket notation here ($\langle..\rangle$) denotes ensemble averages, and $S(p_\lambda)$ is the Von Neumann entropy given by:

$$S(p_\lambda) = -\sum_{\sigma} p_\lambda(\sigma) log(p_\lambda(\sigma)) \tag{2.13}$$

Here the sum runs over all possible configurations $\{\sigma\}$, calculating the probabilities out the outputs dependant upon the samples from the RNN. To summarise, the algorithm works as following:

Variational neural annealing;
Initialize variables and RNN
while T>0 **do**
  for each point on the grid **do**
    $h_t \leftarrow h_{t-1}$
  end for
  Sample x states from the RNN together with their probabilities
  Calculate the energy H_{Target} for every sample
  Calculate $F_\lambda(t)$ by taking average of H_{Target} - T$S(p_\lambda)$
  Perform a gradient descent step using an optimizer
  $T \leftarrow T - dt$

While the presented algorithm is simplified it captures the basic principals we want to utilize to fit our problem. A mayor advantage with this algorithm that we want to utilize is that the sampling of many outputs from the RNN is a computationally fast process. This means if our network can be trained correctly, even though the training might take a long time, the trained network could be saved to quickly provide error chains for a given syndrome. Adapting this algorithm to the task of quantum error correction necessitates several modifications, including the creation of a new energy function, additional inputs to the RNN and exploration of alternative RNN structures, among other adjustments.

3
Methods

The code for the implementations described in this section is available online at `https://github.com/AxelPre/VNA-Decoder`. This section is going to go into details how the VNA algorithm was implemented to fit the problem of quantum error correction. First by describing the environment that the algorithm is implemented in, and then describing the RNN itself and the inputs it gets.

3.1 The environment for the RNN

To begin using the VNA algorithm, the original problem and environment set up in [8] was examined. In this environment, the algorithm was set up to find the minimum energy for a 2D Edwards-Anderson model, whose Hamiltonian is given by:

$$H = -\sum_{\langle i,j \rangle} J_{ij}\sigma_i\sigma_j \tag{3.1}$$

Here the sum runs over all nearest neighbours, $\sigma_i \pm 1$ and $J_{ij} \in [-1, 1)$ are random couplings set to a fixed value. This section will describe how a new environment, energy function and new inputs is used to use the VNA algorithm to decode a surface code.

3.1.1 Qubit and syndrome matrix

In broad terms the decoding task consists of generating a random possible syndrome and from that generate a large number of corresponding error chains with that specific syndrome. These generated error chains can then be used to determine which is the most likely error chain that caused the error, which we can later use to actually quantum error correct. To do this we implement a qubit matrix and a corresponding syndrome matrix. The qubit matrix is of size $d \times d$ and the syndrome matrix is of size $(d+1) \times (d+1)$ in order to fit the syndrome measurements on the edges. Note that some of the squares on the edges are dormant and will always be 0, see figure 3.2. The elements of the qubit matrix can take on the numbers (0,1,2,3) representing (I,X,Y,Z) errors (or rather no error in case of 0), and the elements of the syndrome matrix can only take on 0 or 1 representing if a syndrome is detected or not.

In order to make to VNA algorithm fit the problem of quantum error correction, the spin variables is substituted for the qubit matrix. This means our samples are considered our qubit matrix, meaning the problem could be closely translated if we

only had X errors. So the first natural step is to run the algorithm with only X errors present.

3.1.2 The variational free energy function

Since the goal for the VNA algorithm is to find the lowest energy configuration we need to setup a new energy function suitable for the surface code. The desired lowest energy would correspond to the RNN only providing samples of the most probable error chain corresponding to the current syndrome. Meaning we need to provide a high energy to error chains not corresponding to our desired syndrome. For that reason the energy function used is set to be:

$$H_{Sample} = C \sum_{i,j}^{N} |\mathbf{S}_{ij} - \mathbf{S'}_{ij}| \tag{3.2}$$

Where $\mathbf{S}$ denotes our current syndrome matrix and $\mathbf{S'}_{ij}$ denotes one of our samples solutions from the RNN, and C is an arbitrary constant. This energy function would entail that when the temperature goes towards zero, the energy also would. It would mean that the RNN has produced samples, error chains, that all have the same syndrome as the current syndrome matrix. Since we use the temperature as a way to find this state, all along the way to $T = 0$ the RNN will have a smaller and smaller probability to choose anything else rather the lowest energy state.

The problem with equation 3.2 is that there can exist many lowest energy configuration, namely all corresponding error chains. Since we are more interested in finding the most probable, longer error chains need to have a slightly higher energy. The new equation becomes:

$$H_{Sample} = C_1 \sum_{i,j}^{N} |\mathbf{S}_{ij} - \mathbf{S'}_{ij}| + C_2 N_E \tag{3.3}$$

Where N_e represents the number of total errors on the qubit grid, C_1 and C_2 are two constants. This alternative energy function can be used to favour shorter error chains, which is reasonable since they are more likely than longer error chains. Note that we need to have $C_1 >> C_2$ in order for the network to give non-zero solutions.

3.1.3 Optimizers

Multiple optimizers were tested but the Adam optimizer was chosen, one of the most commonly used. The Adam optimizer is an extension of stochastic gradient descent that in addition to using the gradient also uses the second order moments of the gradients. It also uses adaptive learning rates on each parameter, which makes for a quicker convergence compared to stochastic gradient descent [30].

3.1.4 Gpu cluster and environment

To speed up run time for longer training sessions the code was run on the Alvis cluster provided by C3SE [31] via the National Academic Infrastructure for Supercomputing in Sweden (NAISS) and the Swedish National Infrastructure for Computing

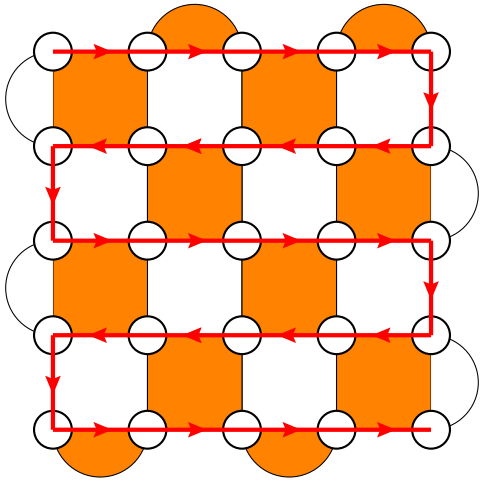

Figure 3.1: A d=5 surface code with the path of the RNN visualized as a red line with arrows.

(SNIC). This GPU cluster is provided with the intent of aiding machine learning research. The code was run partly in their Alvis on demand JupyterLab server to run notebooks for testing, and partly on the interactive desktop app for producing the results.

3.2 The RNN

Two different structures of RNNs are tested and work in two fundamentally different ways and will be explained more in detail in this section, as well as the extra inputs used to train the RNN. The RNNs were constructed in Python using the Tensorflow package. An explanation of the reference decoder concludes the section.

3.2.1 Structure 2D model

In the basic 1D model of the RNN the hidden state is calculated as:

$$\mathbf{h}_n = F(W_n[\mathbf{h}_{n-1}; \boldsymbol{\sigma}_{n-1}] + \mathbf{b}_n) \tag{3.4}$$

Where the notation $[\mathbf{h}_{n-1}; \boldsymbol{\sigma}_{n-1}]$ means a vector concatenation of $\mathbf{h}_{n-1}$ and a one-hot vector $\boldsymbol{\sigma}_{n-1}$ of the number σ_{n-1}. Extended to two dimensions this becomes:

$$\mathbf{h}_{i,j} = F(W^{(h)}_{i,j}[\mathbf{h}_{i-1,j}; \boldsymbol{\sigma}_{i-1,j}] + W^{(v)}_{i,j}[\mathbf{h}_{i,j-1}; \boldsymbol{\sigma}_{i,j-1}] + \mathbf{b}_n) \tag{3.5}$$

Where we now have two different sets of weights $W^{(h)}_{i,j}$ and $W^{(v)}_{i,j}$ to be updated to learn inputs both from the state to the left and from above, with periodic boundary conditions for the edges. This way of updating the hidden state is the main approach, together with the extra inputs discussed in the section 3.2.3. The RNN ran through the surface code in a zigzag path as shown in figure 3.1

3.2.2 Structure 1D dilated model

In addition to the 2D model a 1D dilated RNN model is also tested, for the dilated RNN equation 3.4 is changed to be:

$$\mathbf{h}_n^{(l)} = F(W_n^{(l)}[\mathbf{h}_{max(0,n-1-l)}^{(l)}; \mathbf{h}_n^{(l-1)}] + \mathbf{b}_n^{(l)}) \tag{3.6}$$

Where $\mathbf{h}_n^{(0)} = \boldsymbol{\sigma}_{n-1}$ and l is the layer of the neural network, meaning the update of the hidden state depends on the layer of the network. The supposed advantage of this is the ability to capture long-term dependencies better, which might be useful is this case where qubit number 10 in order (see figure 3.1) depends on the value of qubit number 1. In the same way qubit number 2 depends on qubit number 9, meaning the hope is to capture these dependencies using in this case 9 layers. In a general case it would mean the approach is to use $2d - 1$ layers.

3.2.3 Extra Inputs

In order for the RNN to learn the concept of the syndromes, new inputs need to be implemented. If we only would have one specific syndrome the two inputs from previous hidden state would be enough, but in order for the network to learn the difference between two different syndrome we need a way for the network to differentiate the two. To do this we add on four extra inputs to equation 3.5 corresponding to the four potential syndrome around that point in the surface code. For the qubits on the edges 1 or 2 of these input will be dormant at all times. This can be seen in figure 3.2. The extra inputs are equipped with weights in a similar way to the inputs from the previous hidden states, updating our equations 3.5 and 3.6 with four extra inputs.

3.3 Reference decoder

In order to investigate the viability and accuracy of our decoder a reference decoder must be used, and in this work the reference decoder will be an Effective Weight Decoder (EWD) decoder from [12]. The EWD decoder is based on Metropolis-based Monte Carlo sampling to determine the likelihood of different error chains and is considered very accurate but is relatively slow. The version of the EWD decoder used is the EWD decoder with alpha-noise and with $only\_shortest = True$ meaning the model only saves the shortest error chains. In order to find the error chains the model works by applying stabilizers randomly to a qubit matrix and saving new error chains created, only saving the shortest ones. This is advantageous compared to our model since the error chain created always corresponds to the correct syndrome.

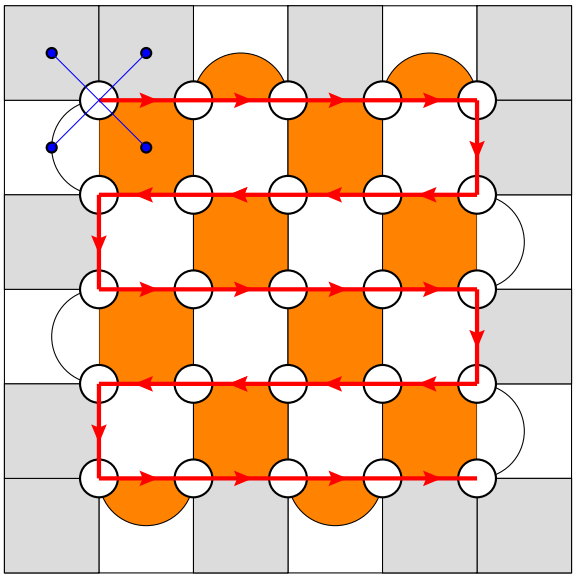

Figure 3.2: A d=5 surface code with the path of the RNN visualized as a red line with arrows, together with an example of which syndromes are used as an input to the first state of the RNN. All four surrounding syndrome measurements are used as an input at every step. Grey squares indicate dormant syndromes that are always 0.

4
Results and discussion

This chapter will present and discuss the results of the project, which will be divided into two parts to represent two different approaches to the problem. In order to test the models we need to have a metric to evaluate how well the decoder works. To do that we will use the logical error rate as a function of physical error rate, and compare our models to a Metropolis based Monte-Carlo decoder from [12]. The logical error rate is calculated as:

$$\text{Logical error rate} = p_f = \frac{\#\text{Incorrect predictions}}{\#\text{Predictions}} \tag{4.1}$$

Which is inevitably going to increase with an increasing physical error rate introduced. Another important metric used is dubbed sample error rate, which provides the share of solutions that corresponds to the specific syndrome:

$$\text{Sample error rate} = p_s = \frac{\#\text{Samples with incorrect syndrome}}{\#\text{Samples}} \tag{4.2}$$

Which is very analogous to accuracy and gives an indication of how well-trained our RNN is, where a high sample error rate indicates that the RNN is frequent in providing faulty error chains that do not correspond to the syndrome investigated. It is the inverse of accuracy and the reason for not using accuracy is since accuracy is used in a different context in this thesis as well, describing the accuracy of the RNN in terms of predicting correct equivalence classes. This metric can be useful since only looking at the energy can be somewhat misleading, since the energy can decrease while the sample error rate can remain relatively high for some syndromes.

The results are separated into two parts, the first part being the results from using variational neural annealing to train a RNN to be able to sample error chains from any general syndrome. The second part presents a second approach that instead lets the RNN run through and learn a single syndrome, then sample from it and save the percentages of error chains in each of the equivalence classes. An important note is that all errors introduced to produce the results are X errors, meaning $p_y = p_z = 0$. The reason will be explained in more detail later in this chapter.

4.1 VNA for any general syndrome

During the training of the RNN the sample error rate was monitored closely to give an indication of convergence, together with the mean energy of all samples. The

Figure 4.1: Training progress for a 2D RNN on a $d = 3$ surface code measured in both sample error rate, p_s, and in energy while training on a single syndrome.

first result comes from using variational annealing to train the RNN on one single syndrome measurement and the training progress can be seen in figure 4.1.

In figure 4.1 we can clearly see the energy and sample rate decrease with every episode. and plateauing around 0 after 1000 steps. The reason for specifically 1000 steps is since the temperature is set to decrease from 500 to 0 during 1000 steps and then take 100 steps at $T = 0$. What the annealing process does here is making the model converge quicker and quicker, and at $T = 0$ the model acts deterministic never allowing for solutions with higher energy. The advantage of this is that the model explores wildly in the start of the training process to net get stuck in a local optima, allowing the model to find the global optima as explain in section 2.2.1.

During the annealing process while training the RNN on multiple syndromes problems occur, and the RNN has a tendency to overfit to certain syndromes. This phenomena can be seen in figure 4.2 where we can see that as the temperature is lowered not all syndromes gets a lower sample rate. This phenomena occurred despite many efforts to avoid it, changing of hyperparameters, energy functions, RNN-structures and many more modifications. These results were done with only X errors and the RNN still could not converge, leading to the reason why only X errors were simulated. It also lead to the creation of the alternative approach presented in the next section.

4.2 VNA for a specific syndrome

In addition to the approach of using variational neural annealing to train a neural network, a different algorithm was produced as an alternative. The reason being the poor performance of variational neural annealing applied in the current way. Since

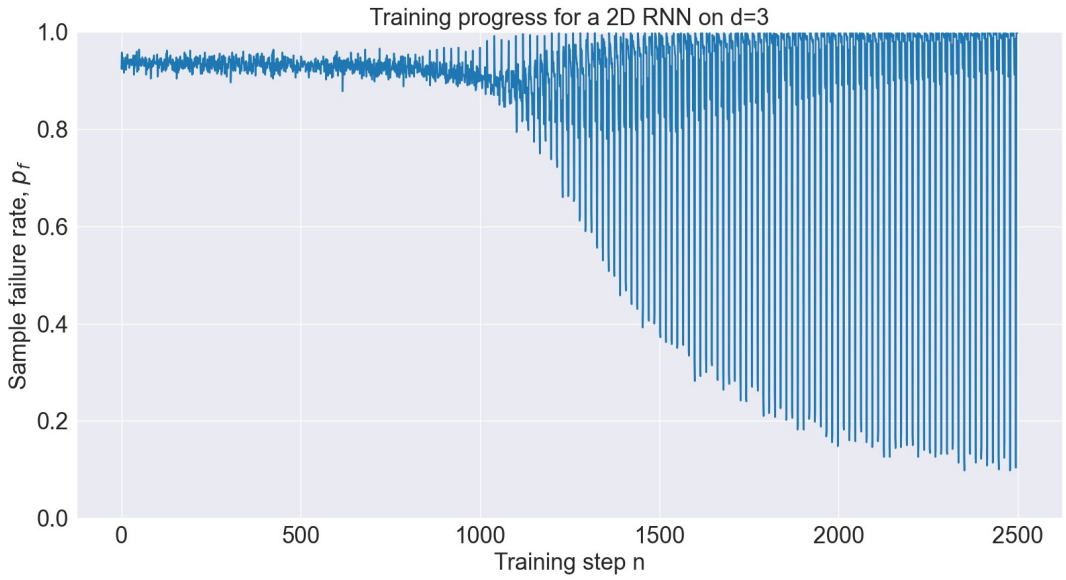

Figure 4.2: Training progress while training on a multiple (randomly sampled) syndromes

we have concluded that the RNN can be trained on a single syndrome and produce error chains corresponding to it the new approach will build upon that. Meaning the new approach is to run the network on one syndrome configuration and then sampling from it, repeating this multiple times for each physical error rate and then averaging. This also lets us investigate the accuracy of the error chains found, to see whether they are the most probable.

This approach strays from the original idea but is performed in order to provide an alternative for further exploration and to investigate how accurate the RNN is in decoding. The annealing process is slightly modified to fit this solution, with the difference being a higher starting temperature and the temperature being held at zero until a specific threshold for sample error rate is met. The process is also made shorter with a high learning rate, in order to speed up the process. A shorter process is reasonable since a lower diversity of solutions is much smaller now that the network only needs to provide samples from one syndrome measurement.

Results presented for this solution are produced with the energy function in equation 3.3. The reason for this being an attempt to make the RNN favour shorter solutions, which will be necessary now that the diversity of solutions is smaller. This approach still only investigates $d = 3$ and $d = 5$ surface codes as well as $p_y = p_z = 0$.

4.2.1 Performance compared to reference decoder

To test the logical error rate the 1D dilated model and the 2D model got to sample 200 error chains 200 times and averaged at a given physical error rate. This was done for 20 physical error rates $\in [0, 0.15]$ on both a $d = 3$ and a $d = 5$ surface code. The threshold was set to 0 in order to only sample error chains with correct

syndrome. The structure of the 2D RNN consisted of 2 layers of 20 neurons in the hidden layers, and for the 1D dilated RNN $2d-1$ layers of 20 neurons were used for the different code distances. The results can be seen in figures 4.3 and 4.4.

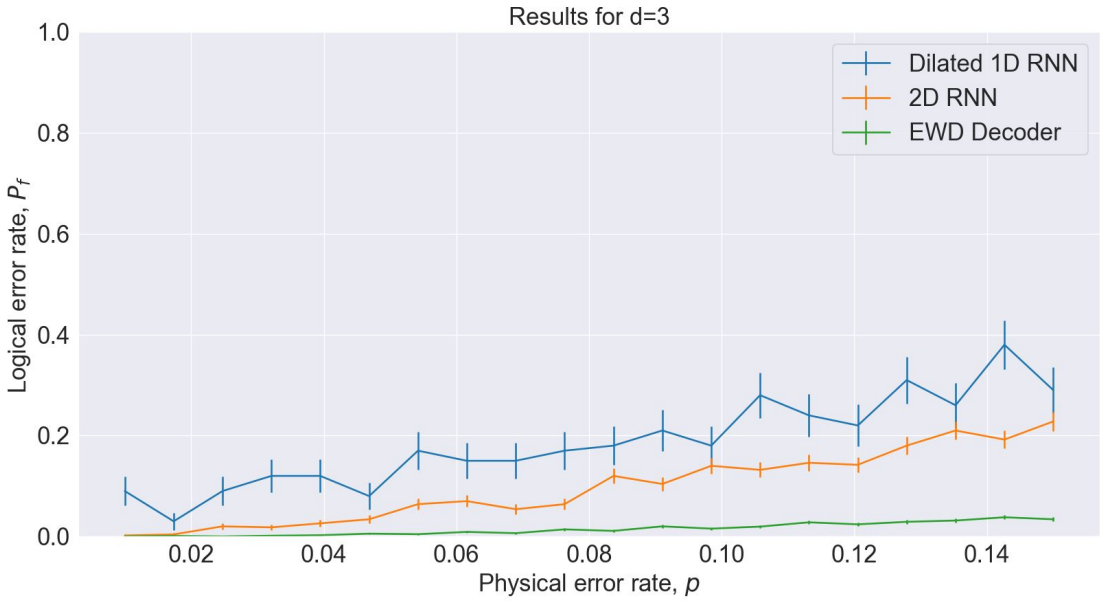

Figure 4.3: Performance of the two RNN models compared to the EWD decoder on a $d = 3$ surface code.

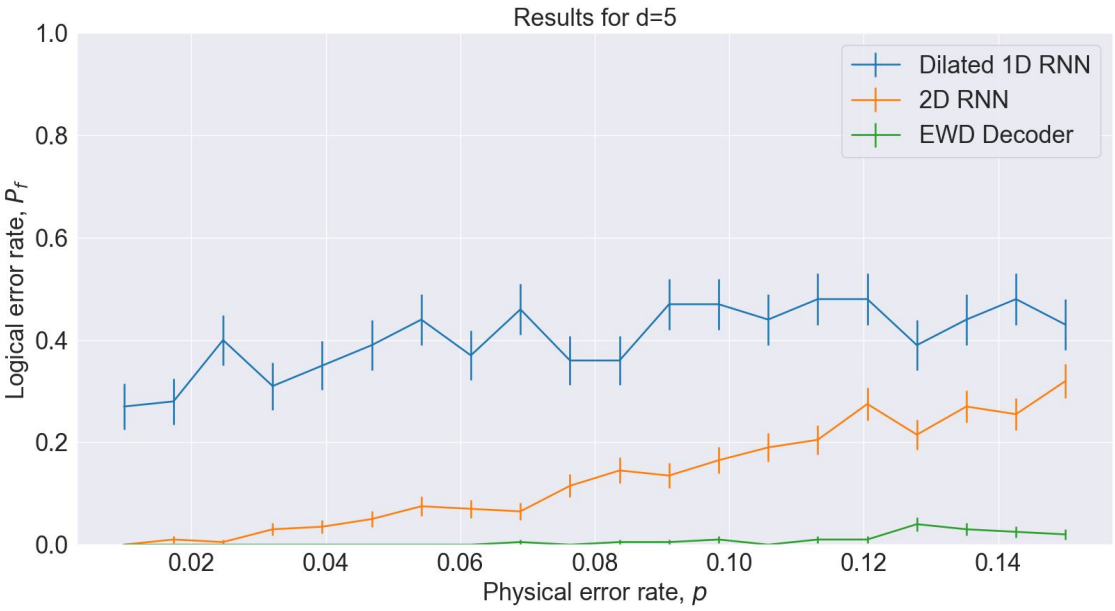

Figure 4.4: Performance of the two RNN models compared to the EWD decoder on a $d = 5$ surface code.

4.2.2 Runtime

Since the results show that the accuracy of our RNN is significantly lower than the reference decoder the discussion around runtime is less relevant. If the accuracy would have been comparable the runtime might decide whether or not our RNN model would be viable, but because of the poor performance is not viable regardless of runtime. The runtimes can be seen in table 4.1.

Method	d	Number of samples	Runtime
2D RNN	5	200	253 min ($\approx$ 4 h)
1D dilated RNN	5	200	2204 min ($\approx$ 36 h)
EWD decoder	5	200	40 min
2D RNN	3	200	123 min ($\approx$ 2 h)
1D dilated RNN	3	200	1932 min ($\approx$ 32 h)
EWD decoder	3	200	3 min

Table 4.1: Table of runtime for the different methods, both on $d = 3$ and $d = 5$ with 200 samples. Note that the runtimes were tested once and not averaged due to the long runtime of some of the methods.

Where we can unfortunately see that our RNN used in this alternative approach does not give us an advantage in term of runtime either. However as the theory would suggest after training the RNN the sampling is very quick, so if one would succeed is training a RNN to be able to handle any general syndrome the advantage would be significant. To sample just 200 samples takes under 1 second for a 2D RNN trained on a $d = 3$ surface code, and to sample 100000 error chains takes ≈ 1 minute.

5
Conclusions

While the results of the variational neural annealing model did not live up to the expectation some important lessons can be learned and there are many possible ways to explore for further research. This chapter will explore some of these ideas and how to move forward in exploring this field of RNN based decoders.

5.1 Decoding performance

From the results seen in figure 4.3 and 4.4 the accuracy of the RNN when trained on a single syndrome is not as accurate as was hoped for, with the accuracy being significantly lower than the reference decoder. We can also see that the 1D dilated RNN did not have the advantage that the theory would suggest and was clearly outperformed by the 2D RNN. The performance of the 2D model can at the very least be said to be accurate for very small amounts of physical errors, seemingly being able to keep up with the EWD decoder for the first few smallest physical error rates.

5.2 Runtime advantages

The original idea of variational neural annealing would pose a big advantage in terms of runtime since the network only would have to be trained once, and sampling from it would be computationally advantageous. This would be an advantage even though the training time might be multiple days, since it would only have to be done once. But for this to become reality some new way of training the RNN or some new RNN structure has to be found, one that is not prone to overfitting on one or multiple specific syndromes. Some alternatives for this is presented in the next section 5.3.

When it comes to the alternative approach of training on each and every syndrome the runtime is unfortunately slower (see table 4.1) but there could be many ways to improve that in the future. One could train the network even quicker than present by trying to find an alternative energy function that makes the RNN converge quicker, one that can tolerate an even higher learning rate for example. But in conclusion the alternative idea of training on each specific syndrome does not pose the computational advantage originally intended and did not deliver in terms of accuracy either. This means it can be seen more as a proof of concept and a test in using RNNs as a decoder on a surface code, leaving much to be desired in terms of runtime

and accuracy.

One definite conclusion can be made and that is that the most promising of the networks was the 2D RNN, having both better accuracy and better runtime. It might be appropriate to mention that the longer runtime almost certainly can be attributed to the larger network needed. It seems like the RNN needs information from two previous hidden states from two directions in order to perform better. Which means going forward 2D models seem to be the best alternative.

5.3 Alternatives for further research

There are many ways to improve on continue this work, both in terms of trying to train a RNN for any general syndrome and in terms of improving accuracy when trained on a single syndrome. This section presents some alternatives that could help improve the model.

5.3.1 Energy function

There are many ways to improve and continue to work on this model of RNN decoder or create other RNN based decoders. One mayor improvement would be to fin a way to optimize the constants $C1$ and $C2$ for any given code size, since having it too small or too large causes accuracy to drop. Some efforts were made to find such a relationship but no reliable way was found, and the best relation was $C1 \approx 50 \times C2$.

If no such relation would be found to improve this model includes one would need to find an alternative energy function, one that would capture the problem in a different way. The issue with the energy function currently is that it is always changing iteration to iteration, possibly creating a too complex landscape for the RNN to solve. At the same time trying to run the RNN multiple iterations on the same syndrome tends the RNN to overfit even quicker, which has been the main problem during the course of the work.

5.3.2 2D dilated RNN

Another improvement to the current model would be to create a 2D dilated RNN and see if we could get a benefit there, since the 1D dilated model did not seem to perform according to theory. The 2D dilated RNN would have inputs from almost every aspect of the surface code, maybe leading to some type of improvement.

5.3.3 Other methods

In one wants to go outside the scope of the VNA algorithm one could try to combine the method with similar neural network based decoders [37, 36]. The cited

works obtain a higher accuracy and their neural network techniques could be examined to investigate the possibility of merging methods. Other improvements could be done by using other RNN-based algorithms [33, 34, 35] entirely but integrating aspects of annealing to be able to find the probability distribution of the error chains.

One could also explore using LSTMs to try to capture long term dependencies rather than using the dilated RNN model, since LSTMs have been proven to be more successful at many machine learning tasks [38]. Another alternative method to explore is the use of transformers, which have been replacing RNNs and LSTMs in multiple areas of machine learning [39]. Transformers uses self-attention which helps neural networks identify more important parts of data. They are also more amendable to parallelization, allowing for greater memory.

6
Ethical discussion

When discussing the ethics quantum computing the most common concern surrounds the topic of encryption and the potential to break RSA encryption. This might be a concern since Shor's algorithm is an extremely effective way of factoring large prime numbers which is the basis of RSA encryption. This has sparked the field of quantum safe cryptography that holds up in an eventual post-quantum era [40]. So while some see quantum computing as potentially harmful one could argue that it can help produce even more powerful encryption, making for added IT-safety.

Now its important to note these concerns often make the assumption that quantum computing is both possible and right around the corner. Neither of these assumptions are facts and many experts in the field speculate that we have many decades to go before we have large scale fault-tolerat quantum computing, including Peter Shor himself [41]. While this discussion on the ethics of quantum computing is important this work does not pose a threat to RSA encryption and would or could never solve the giant challenge of quantum computing. It is rather a project exploring this field and an alternative to a current way of quantum error correction.

Working with machine learning it is easy to forget that there are a lot of ethical concerns and methods to avoid. Some of these include using data sets from unethical sources or unintended discriminatory results from using large data sets to train a neural network. This work however uses no data sets and only uses data from qubit states that gives results as qubit states, which can by no means be seen as unethical. Since the code is available at GitHub transparency is promoted and contributes to an open and ethical use of machine learning.

Bibliography

[1] J. Preskill, “Quantum Computing 40 years later,” *Feynman Lectures on Computation*, pp. 193–244, 2023. doi:10.1201/9781003358817-7

[2] A. Yu. Kitaev, "Fault-tolerant quantum computation by anyons", *Annals of Physics*, vol. 303, no. 1, pp. 2-30, Mar. 2003. doi: 10.1016/S0003-4916(02)00018-0.

[3] Google Quantum AI, "Suppressing quantum errors by scaling a surface code logical qubit," *Nature*, vol. 614, pp. 676–681, Feb. 2023. doi: 10.1038/s41586-022-05434-1.

[4] G. Torlai and R. G. Melko, "Neural Decoder for Topological Codes," *Physical Review Letters*, vol. 119, no. 3, p. 030501, Jul. 2017. doi:10.1103/PhysRevLett.119.030501.

[5] C. Chamberland and P. Ronagh, "Deep neural decoders for near term fault-tolerant experiments," *Quantum Science and Technology*, vol. 3, p. 044002, 2018. doi:10.1088/2058-9565/aad1f7.

[6] P. Andreasson, J. Johansson, S. Liljestrand, and M. Granath, "Quantum error correction for the toric code using deep reinforcement learning," *Quantum*, vol. 3, p. 183, 2019. doi:10.22331/q-2019-09-02-183.

[7] X. Ni, "Neural Network Decoders for Large-Distance 2D Toric Codes," *Quantum*, vol. 4, p. 310, Aug. 2020. doi:10.22331/q-202

[8] M. Hibat-Allah, E.M Inack, R. Wiersema, et al., "Variational neural annealing," *Nat Mach Intell*, vol 3, pp. 952–961, 2021. doi: https://doi.org/10.1038/s42256-021-00401-3

[9] J. R. Wootton and D. Loss, "High Threshold Error Correction for the Surface Code," *Physical Review Letters*, vol. 109, no. 16, pp. 160503, Oct. 2012. doi:10.1103/PhysRevLett.109.160503

[10] A. Hutter, J. R. Wootton, and D. Loss, "Efficient Markov chain Monte Carlo algorithm for the surface code," *Physical Review A*, vol. 89, no. 2, pp. 022326, Feb. 2014. doi:10.1103/PhysRevA.89.022326

[11] S. Bravyi, M. Suchara, and A. Vargo, "Efficient algorithms for maximum likelihood decoding in the surface code," *Physical Review A*, vol. 90, no. 3, pp. 032326, Sept. 2014. doi:10.1103/PhysRevA.90.032326

[12] K. Hammar, A. Orekhov, P. W. Hybelius, A. K. Wisakanto, B. Srivastava, A. F. Kockum, and M. Granath, "Error-rate-agnostic decoding of topological stabilizer codes," *Phys. Rev. A*, vol. 105, no. 4, pp. 042616, Apr. 2022. doi:10.1103/PhysRevA.105.042616

[13] P. W. Shor, “Algorithms for quantum computation: discrete logarithms and factoring,” 1994. [Online]. Available: https://ieeexplore.ieee.org/document/365700

[14] A. Peruzzo, J. McClean, P. Shadbolt, et al., "A variational eigenvalue solver on a photonic quantum processor," *Nature Communications*, vol. 5, no. 1, Jan. 2014. doi: 10.1038/ncomms5213.

[15] J. Suo, L. Wang, S. Yang, et al., "Quantum algorithms for typical hard problems: a perspective of cryptanalysis," *Quantum Inf Process 19*, 178 (2020). doi: https://doi.org/10.1007/s11128-020-02673-x

[16] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum algorithms for supervised and unsupervised machine learning," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 185-199, Sep. 2019. doi: https://doi.org/10.48550/arXiv.1307.0411

[17] S. Ding and Z. Jin, “Review on the study of entanglement in Quantum Computation Speedup,” *Chinese Science Bulletin*, vol. 52, no. 16, pp. 2161–2166, 2007. doi:10.1007/s11434-007-0324-8

[18] D. Coppersmith, “An approximate Fourier transform useful in quantum factoring,” *arXiv preprint quant-ph/0201067*, vol.1, 2002, [Online]. Available: http://arxiv.org/abs/quant-ph/0201067

[19] "Bloch Sphere.svg," *Wikimedia Commons*, Nov. 18, 2020. [Online]. Available: https://commons.wikimedia.org/wiki/File:Bloch_Sphere.svg. [Accessed: Apr. 29, 2023].

[20] S. Peisert, R. Gentz, "An Examination and Survey of Random Bit Flips and Scientific Computing," *Trusted CI Technical Report*, 2019, Available: http://hdl.handle.net/2022/24910

[21] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, “Surface codes: Towards practical large-scale quantum computation,” *Physical Review A*, vol. 86, no. 3, 2012. doi:10.1103/physreva.86.032324

[22] J. H. Friedman et al., "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1-22, 2010. doi: 10.18637/jss.v033.i01

[23] C. M. Bishop, "Pattern Recognition and Machine Learning," *Springer*, 2006. doi: 10.1007/978-0-387-31073-2

[24] W. L. Goffe, G. D. Ferrier, and J. Rogers, “Global optimization of statistical functions with simulated annealing,” *Journal of Econometrics*, vol. 60, no. 1, pp. 65–99, 1994.

[25] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[26] S. S. Das and A. Konar, "A Modified Simulated Annealing Algorithm for Optimal Feature Selection," *Applied Soft Computing*, vol. 8, no. 1, pp. 646-656, Jan. 2008.

[27] A. E. Akinlar and A. Topal, "Object Tracking in Videos with Simulated Annealing Optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 4, pp. 1068-1080, Aug. 2012.

[28] B. Mehlig, "Machine learning with neural networks an introduction for scientists and Engineers," *Cambridge University Press*, 2022.

[29] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115-133, Dec. 1943.

[30] J. Ba, D. Kingma, "Adam: A method for stochastic optimization.", *arXiv preprint arXiv:1412.6980*, 2014

[31] C3SE Chalmers University of Technology, "Alvis OnDemand Documentation," [Online]. Available: https://www.c3se.chalmers.se/documentation/alvis-ondemand/. [Accessed: Apr. 13, 2023].

[32] S. E. Hihi and Y. Bengio, "Hierarchical recurrent neural networks for long-term dependencies," *Advances in Neural Information Processing Systems*, Available: http://papers.nips.cc/paper/1102-hierarchical-recurrent-neural-networks-for-long-term-dependencies.pdf

[33] S. Bai, J. Z. Kolter, and V. Koltun, "Deep Equilibrium Models," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[34] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing Machines," *arXiv preprint arXiv:1410.5401*, 2014. [Online]. Available: https://arxiv.org/pdf/1410.5401.pdf

[35] M. Jiao, D. Wang, J. Qiu, "A GRU-RNN based momentum optimized algorithm for SOC estimation", *Journal of Power Sources*, Volume 459, 2020, DOI: https://doi.org/10.1016/j.jpowsour.2020.228051.

[36] M. Becker K.R Chiu Falck, "Deep Reinforcement Learning for Quantum Error Correction", *Chalmers University of Technology*, 2021, [Online] Available: https://hdl.handle.net/20.500.12380/302593

[37] P. Havström, O. Heuts, "Machine Learning Assisted Quantum Error Correction Using Scalable Neural Network Decoders", *Chalmers University of Technology*, 2023 , [Online] Available: http://hdl.handle.net/20.500.12380/305996

[38] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network", *Physica D: Nonlinear Phenomena*, vol. 404, 2020. doi:10.1016/j.physd.2019.132306.

[39] S. Karita et al., "A Comparative Study on Transformer vs RNN in Speech Applications," *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Singapore, 2019, pp. 449-456, doi: 10.1109/ASRU46091.2019.9003750.

[40] J. Wang, L. Liu, S. Lyu, et al, "Quantum-safe cryptography: crossroads of coding theory and cryptography", *Science China. Information Sciences*, vol. 65, 2022. https://doi.org/10.1007/s11432-021-3354-7

[41] P. Shor, Guest lecture, *Chalmers University of Technology*, March. 2023

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY