Universiteit Leiden

Master Computer Science

Investigating Deep Learning of DFT Functionals for the 1D Hubbard Model in the NISQ Era

Name: Eric Prehn
Student ID: s2724731
Date: 13/07/2022

Specialisation: Data Science

1st supervisor: Dr. J. Tura Brugués
2nd supervisor: Dr. V. Dunjko

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Universiteit Leiden

Leiden University

Computer Science: Data Science M.S.

Thesis: Investigating Deep Learning of DFT Functionals for the 1D Hubbard Model in the NISQ Era

Author: Eric Prehn

Student ID: s2724731

1st Supervisor: Dr. J. Tura Brugués

2nd Supervisor Dr. V. Dunjko

July 25, 2022

Universiteit Leiden

Contents

1 Introduction

The techniques from Physics, Machine Learning and Quantum Computing can be combined to yield novel hybrid methods. In the following short paragraphs, the areas relevant for this project are introduced, followed by a proposed hybrid scheme.

In Physics a problem is called many-body when the quasi-particles involved (electrons, phonons, magnons, etc.) interact directly with each other and not through a mean potential [21]. Mean field potentials are approximations that replace all interactions to any one particle with an average interaction, which is not how particles interact. However, they offer a simplified analysis and a satisfactory qualitative explanation to many phenomena of interest. In the last century, solving the many-body problem for the electronic structure of molecules and solids has been one of the largest challenges in quantum mechanics. Classically, the computational time required, along with the typically adverse scaling with system sizes, renders exact analytical solutions infeasible.

An approximate many-body model for electron-electron interactions is the Hubbard model [18], named after John Hubbard, who first proposed it in 1963 to describe electrons in 3d transition metals. Later the one dimensional Hubbard model was solved using a Bethe Ansatz by E. Lieb and F. Wu in 1968 [19]. Since, the Hubbard model has been applied to the understanding of various systems, as it is the simplest model that captures the essence of strongly-correlated electrons in solids [30]. Also, it is used as a toy model, that is believed to capture the key behaviour explaining high-temperature superconductivity [7].

For smaller system sizes, the Hubbard model can be solved exactly via Exact Diagonalisation (ED), where solving refers to finding the energy spectrum and eigenstates. However, this is typically limited by the rapidly growing Hilbert Space dimension of the system in question and the computational cost of ED. Once the energy spectrum, and hence the eigenstates, are found, further properties are attainable. An example being the on-site occupation, which can act as the relevant density for a Density Functional approach.

Density Functional Theory (DFT) [17] is a powerful theory that is currently used in Physics, Chemistry and materials science to investigate electronic structure of many-body systems. As is described in the Background and Theory Section 3, complex many-body problems can be reformulated by defining an energy functional of the electron density, using DFT. In DFT, the exact form of the energy functional is not known and in practice approximations have to be chosen. Usually these approximations do not give one sufficient accuracy in highly correlated systems. It is therefore of great interest to find the exact functional, with

DFT currently being the most widely applicable and used approach for quantum chemistry. It has been shown that in the case of the one-dimensional Hubbard model, the form of the functional can be accurately learnt using Machine Learning (ML) [24].

Artificial Neural Networks (ANN's) are a tool in ML, which are inspired in part by the observation that biological learning systems are built of very complex webs of interconnected neurons [12]. Deep Learning involves ANN's with greater depth, meaning a larger number of layers of neurons. Deep Learning models require a large number of parameters to be tuned, typically requiring large data sets. Thanks to technological developments; the use of Deep learning has grown as the amount of available training data has increased. These models have grown in size over time, as their computer infrastructure (both hardware and software) have improved [12]. Today, ML provides an alternative route to researching physics, due to ML's ability to find solutions by analysing data sets. There exist a multitude of ANN's to choose from, many of which, in turn are trained using Deep Learning software. More details on ANN's , ML and Deep Learning are given in Sections 3.5 and 3.6. One set back of Deep Learning is the requirement that a large enough training set has to be generated. For example this could require exactly solving a quantum system, thousands of times.

Generally, exact simulation of quantum systems on classical computers remains infeasible due to the lack of computationally efficient methods. However the advances in quantum computing may lead to solutions for certain problems, in certain regimes, that classical computers cannot handle. This is because quantum computers can represent quantum systems natively [32]. Currently quantum computing is in the Noisy intermediate-scale quantum (NISQ) era. Here, noise in quantum gates limits the size of quantum circuits that can be executed reliably, and "intermediate scale" refers to the size of quantum computers (50-100 qubits). This rules out the implementation of famous quantum algorithms, such as Shor's factoring algorithm [25] and Quantum Phase Estimation (QPE) [25].

Due to its nature, the Hubbard model is seen as an early target for quantum simulation algorithms [3, 9, 32]. Firstly because it is useful for understanding technologically-relevant correlated systems. Secondly, its regular structure and relative simplicity, suggests it is easier to solve than, for example, a large unstructured molecule (especially in the NISQ era). Additionally it can be viewed as a benchmark for quantum algorithms, since classical solutions are restricted to small system sizes (17 fermions on 22 sites requiring over 7TB of memory and 13 TFlops on a 512-node supercomputer to diagonalise a 159-billion-dimensional matrix [35]).

This work is an attempt to combine the areas of DFT, ML and quantum computing. The goal is to investigate solving the 1D Hubbard model via a DFT approach, using Deep Learning, that trains on NISQ era data.

First, in Section 2 the problem statement is defined and in Section 3 background and theory are provided. In Section 4 the ED results of Sanvito [24] are reproduced, where a Deep Learning model learns an exact functional, for finite, periodic systems. Next, in Section 5 measurement noise is incorporated into the exact results, which emulates the process of measuring the ground state if it were prepared on a quantum computer. The effects of this noise on the Deep Learning method is investigated.
Lastly in Section 6, a quantum algorithm, the Variational Quantum Eigensolver (VQE), is used to generate training data for the Deep Learning method.

Overall this work aims at achieving a potential NISQ era application of the following proposed scheme:

1. A NISQ era quantum computer is used to generate a dataset of Hubbard model instances (of fixed size but with varying external potentials).

2. A Deep Learning model learns the exact form of the DFT functional, by training on this NISQ dataset. Ideally the Deep Learning model achieves some degree of noise mitigation, as it learns the functional and interpolates through noise.

3. Once the Deep Learning model is trained and performs/generalises well, it can be reused without the need for further quantum computing resources.

The advantages of using a classical Deep Learning model are that it is deterministic and has differentiable output. Additionally it can be robust to noise [31, 28] and perhaps even mitigate noise effects, which is of great importance for NISQ generated data. Importantly, in the NISQ era, the proposed scheme could be applied to larger Hubbard system sizes than classically solvable, and the DFT functional learned using limited quantum computing resources.

In summary, the Deep Learning model was applied to 3 types of data, including exact data (as in [24]), exact data with measurement noise and lastly, VQE generated data. The proposed scheme was successful for the measurement noise data, and achieved a degree of noise mitigation. In contrast, the Deep Learning model did not perform well on VQE generated data. Still, the VQE data generation method can be improved, yielding better data for the scheme. Hence, it remains to be confirmed how the scheme performs for higher-quality VQE data and this is the centre of attention of future work.

2 Problem Statement

The canonical problem to solve is to find the ground state energy of a quantum many-body Hamiltonian H. This work focuses specifically on the one-dimensional Hubbard model. Note that in general, producing the ground state of a quantum Hamiltonian is expected to be computationally hard for quantum computers [3, 11]. However this does not mean that solutions cannot be found. The ground state energy is the solution of the problem,

$$E_{GS} = \min_{|\Psi\rangle} \frac{\langle \Psi | H | \Psi \rangle}{\langle \Psi | \Psi \rangle}, \tag{1}$$

where the minimisation occurs over all the (unnormalised) states $\{|\Psi\rangle\}$ in the Hilbert space. In general the ground state is sought after because it contains much of the information for understanding the low-energy properties of a system. In this work, ground states are needed to learn the exact form of Hubbard model DFT Functionals. This thesis investigates how well the Deep Learning model of [24] learns the DFT Functionals, when trained on NISQ era data. The following three settings are investigated:

1. Exact Diagonalisation data.
2. Exact Diagonalisation data with measurement noise.
3. VQE generated data.

3 Background and Theory

3.1 The Hubbard Model

In 1963 John Hubbard introduced the Hubbard Hamiltonian in order to model electronic correlations in narrow energy bands, specifically for 3d transition metals. It is a tight-binding, many-body quantum model, based on nearest neighbour hopping and on-site interactions. The Hubbard Hamiltonian describes moving, interacting spin-$\frac{1}{2}$ electrons hopping on a set Λ of spatially localized orbitals.

$$\mathbf{H}_{hom} = -\sum_{i,j\in\Lambda}\sum_{\sigma} t_{ij}\hat{a}^{\dagger}_{i,\sigma}\hat{a}_{j,\sigma} + U\sum_{i\in\Lambda}\hat{n}_{i,\uparrow}\hat{n}_{i,\downarrow}, \tag{2}$$

where t_{ij} is usually restricted to nearest-neighbor sites and becomes a constant hopping parameter t, resulting from the overlap of the wavefunctions of adjacent atoms [33]. Every site can hold at most 2 electrons from the Pauli exclusion principle. Here $\sigma \in \{\uparrow,\downarrow\}$, $\hat{a}^{\dagger}$ and $\hat{a}$ are creation/annihilation operators, $\hat{n} = \hat{a}^{\dagger}\hat{a}$ is the number operator, and U is a Coulomb Potential term. The Hubbard Hamiltonian follows from second quantisation, where the quantum many-body states are represented in a Fock state basis. Details on second quantisation can be found in [20] and the canonical anti-commutation relations for the creation/annihilation operators are provided in Section 6.1. Below a simple example (with notation abuse) of the creation and annihilation operators acting on the possible states of orbital (i,σ) are shown:

$$\begin{aligned}
\hat{a}^{\dagger}_{i,\sigma}|0\rangle &= |1\rangle,\\
\hat{a}^{\dagger}_{i,\sigma}|1\rangle &= 0,\\
\hat{a}_{i,\sigma}|0\rangle &= 0,\\
\hat{a}_{i,\sigma}|1\rangle &= |0\rangle.
\end{aligned} \tag{3}$$

Where the notations, $|1\rangle$ represents an electron present at orbital (i,σ), and $|0\rangle$ represents none. The inhomogeneous 1D Hubbard Model is the focus of this work and is described by:

$$\mathbf{H} = \sum_{i\in L}\sum_{\sigma}\nu_i\hat{a}^{\dagger}_{i,\sigma}\hat{a}_{i,\sigma} - t\sum_{\langle i,j\rangle\in L}\sum_{\sigma}\hat{a}^{\dagger}_{i,\sigma}\hat{a}_{j,\sigma} + U\sum_{i\in L}\hat{n}_{i,\uparrow}\hat{n}_{i,\downarrow}, \tag{4}$$

Summation over nearest neighbour sites (i,j) is labeled by $\langle i,j\rangle$ and the chain of length L becomes a ring due to periodic boundary conditions. The presence of an external potential ν, with ν_i being site-dependent, leads to the inhomogeneous case. This corresponds to varying on-site energies, commonly referred to as disorder. The second term in Equation 4 creates and destroys electrons of neighbouring sites, representing electron transfer, or the

kinetic energy. Finally, the third term is composed of the product of 4 creation/annihilation operators, providing the interactions that make it a many-body problem. The hopping parameter can be set to $t = 1$ and without loss of generality, the remaining parameters U and ν_i correspondingly measured in units of t.

3.2 Density Functional Theory

The foundation of Density Functional Theory (DFT) comes from two simple theorems, known as the Hohenberg-Kohn theorems [17]. In principle, these allow one to circumvent the attention on the wavefunction and replace it with the electron density $n(\mathbf{r})$, when seeking ground state properties. Note the dimension of the wavefunction of a many-body problem typically scales very badly, whereas the electron density is at most a function of 3 spatial coordinates. This subsection briefly introduces the notion of functionals and functional derivatives, followed by the Hohenberg-Kohn theorems and their application to the Hubbard model.

3.2.1 Functionals and Functional Derivatives

Mathematically, functionals are linear maps from a vector space to its underlying scalar field. An example of a functional is the following:

$$E[|\Psi\rangle] = \frac{\langle\Psi|H|\Psi\rangle}{\langle\Psi|\Psi\rangle}, \tag{5}$$

where the energy is a function of another function, i.e. it is a function of the wavefunction. Typically a functional is a real (complex)-valued function on a vector (Hilbert) space $\mathcal{H}$. In Equation 5, $|\Psi\rangle$ is a wavefunction but in general one can define functionals of the electronic density, denoted as $F[n(\mathbf{r})]$. This is the case in DFT, where $n(\mathbf{r})$ is the electron density function, with $\mathbf{r} \in \mathbb{R}^3$. The functional derivative $\frac{\delta F}{\delta n}$ is defined as [26]:

$$\delta F = \int \frac{\delta F}{\delta n}(\mathbf{r})\delta n(\mathbf{r}) d^3\mathbf{r}, \tag{6}$$

where δn is an arbitrary function and $\delta F = F[n + \delta n] - F[n]$ is the variation of F.

3.2.2 Hohenberg-Kohn Theorems [26]

The electronic structure Hamiltonian in real space, under the non-relativistic Born-Oppenheimer approximation, for an isolated N-electron atomic or molecular system is given by:

$$H = \sum_{i=1}^{N}(-\frac{1}{2}\nabla_i^2) + \sum_{i}^{N}\nu(\mathbf{r}_i) + \sum_{i<j}^{N}\frac{1}{r_{ij}}, \tag{7}$$

where

$$\nu(\mathbf{r}_i) = -\sum_{\alpha} \frac{Z_\alpha}{r_{i\alpha}} \tag{8}$$

is an external potential which acts on electron i, typically due to nuclear charges Z_α. The ground state wavefunction and corresponding energy are determined, as in Equation 1, by the minimization of the energy functional:

$$E[|\Psi\rangle] = \frac{\langle\Psi|H|\Psi\rangle}{\langle\Psi|\Psi\rangle}. \tag{9}$$

Importantly it can be seen that N and ν determine all the properties for the ground state, because the external potential ν completely fixes the Hamiltonian [26]. Any desired property/observable X of the ground state $|\Psi_{GS}^{\nu,N}\rangle$ is then a functional of the external potential, so that $X = X[\nu]$. The first Hohenberg-Kohn theorem allows for replacing N and $\nu(\mathbf{r})$ with the electron density $n(\mathbf{r})$ as the basic variable.

Theorem 1: Existence of a universal functional *For any system of N interacting electrons moving under the influence of an external potential $v(\boldsymbol{r})$ (e.g. that of the nuclei), the external potential, and hence the total energy, is a unique functional of the electron density $n(\boldsymbol{r})$.*

The total energy of the system can be written as,

$$E[n(\mathbf{r})] = \int n(\mathbf{r})\nu(\mathbf{r})d\mathbf{r} + F[n(\mathbf{r})]. \tag{10}$$

Where $F[n(\mathbf{r})]$ is universal, namely it is independent of the system. A corollary of the first Hohenberg-Kohn theorem is that there is a one-to-one correspondence between the external potential $\nu(\mathbf{r})$ and the electron density $n(\mathbf{r})$, meaning that one is sufficient to determine the other.

Theorem 2: Variational Principle *The universal functional is minimised at the ground state density $n(\boldsymbol{r})_{GS}$, and the corresponding energy is the groundstate energy, E_{GS}. This means that,*

$$E_{GS}^{v} = E_v[n(\mathbf{r})_{GS}] \leq E[n(\mathbf{r})]. \tag{11}$$

With these theorems the functional given by Equation 10 can be minimised with respect to the normalisation constraint:

$$N = \int n(\mathbf{r})d^3\mathbf{r}, \tag{12}$$

such that,

$$\delta\left(E_v[n(\mathbf{r})] - \mu\left(\int n(\mathbf{r})d^3\mathbf{r} - N\right)\right) = 0. \tag{13}$$

Here μ is a Lagrange multiplier and together with Equation 10, this becomes:

$$\int \delta n(\mathbf{r}) v_{ext}(\mathbf{r}) d^3\mathbf{r} + \delta F[n(\mathbf{r})] = \mu \int \delta n(\mathbf{r}) d^3\mathbf{r}. \tag{14}$$

Finally applying Equation 6 one gets the following Equation to solve for n:

$$\begin{gathered} \int \delta n(\mathbf{r}) v_{ext}(\mathbf{r}) d^3\mathbf{r} + \int \frac{\delta F}{\delta n}(\mathbf{r}) \delta n(\mathbf{r}) d^3\mathbf{r} = \mu \int \delta n(\mathbf{r}) d^3\mathbf{r}, \\ v_{ext}(\mathbf{r}) + \frac{\delta F}{\delta n}(\mathbf{r}) = \mu. \end{gathered} \tag{15}$$

In practice, the universal functional F is unknown and therefore needs to be approximated. Note that typically these approximations do not give you sufficient accuracy in highly correlated systems. Still, DFT has been widely applied to various quantum mechanical systems and there exist several well-established approximations for performing DFT calculations [13].

In summary, the strength of DFT lies in needing to solve for the electron density rather than the wavefunction, but the accuracy of DFT methods is limited by approximations of the functional F.

3.3 DFT Application to the 1-D Hubbard Model

The two Hohenberg-Kohn theorems apply also for the case of interacting lattice models [29]. DFT on lattice models is sometimes referred to as SOFT (Site-Occupation Functional Theory). The functional has a global minimum at the ground-state density that corresponds to the ground-state energy. Note the Hohenberg-Kohn theorems originally applied to non-degenerate ground states, however in practise density functionals can be restored [4] for degenerate cases. The relevant densities for lattice models depend on the model being studied. The relevant density for the Hubbard Model is the on-site occupation,

$$\{n_{i\sigma}\} = \{n_{1\uparrow}, n_{1\downarrow}, n_{2\uparrow}, n_{2\downarrow}, \ldots, n_{L\uparrow}, n_{L\downarrow}\}, \tag{16}$$

where $n_{i\sigma} \in [0, 1]$ is the occupation of the orbital (i, σ), when the system is in state $|\Psi\rangle$. Note that the electron density $n(\mathbf{r})$ becomes $\{n_{i\sigma}\}$ when going from DFT to SOFT. Applying the Hohenberg-Kohn Theorem to the 1D Hubbard Model gives [24],

$$E = E[\{n_{i\sigma}\}] = \sum_{i,\sigma} \nu_i n_{i\sigma} + F[\{n_{i\sigma}\}], \tag{17}$$

$$F[\{n_{i\sigma}\}] = \langle\psi|(-t \sum_{\langle i,j\rangle}^{L} \sum_{\sigma} \hat{a}^{\dagger}_{i,\sigma} \hat{a}_{j,\sigma} + U \sum_{i}^{L} \hat{n}_{i,\uparrow} \hat{n}_{i,\downarrow})|\psi\rangle = \langle\psi|(\hat{\mathbf{T}} + \hat{\mathbf{U}})|\psi\rangle. \tag{18}$$

With $\hat{\mathbf{T}}$ and $\hat{\mathbf{U}}$ representing the kinetic and potential terms, respectively. Note that for this lattice model $F[\{n_{i\sigma}\}]$ is universal only for a given $\hat{\mathbf{T}}$, $\hat{\mathbf{U}}$ and N. Various approaches to finding this functional exist, one such being Machine Learning [24].

3.4 Machine Learning

In this section, Machine Learning theory is provided for Supervised Learning, Neural Networks and Convolutional Neural Networks. A reader familiar with these areas is advised to view Section 3.4.3 and then skip to Section 3.7. Broadly speaking, a Machine Learning algorithm is an algorithm that is able to learn from data [12]. The algorithms used in ML apply numerical techniques (mathematical/statistical models) to perform a specific task. Normally, one makes the classification of ML into Supervised, Unsupervised and Reinforcement Learning, each being more general than the previous. In this thesis the type of ML used is Supervised Learning.

3.4.1 Supervised Learning Algorithms

A Supervised learning algorithm attempts to learn a mapping or function from a data set consisting of pairs of inputs and matching outputs:

$$f_{ML} : \mathbf{x}_i \mapsto y_i, \tag{19}$$

where each input of the data set, $\mathbf{x}_i$, called an example, (whose individual components are called features) is mapped to the corresponding output, y_i, called a target. The algorithm attempts to create a one-to-one correspondence between inputs and outputs. Each pair of input and output are indexed by i in Equation 19 and typically $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \mathbb{R}$, with m being the number of features. Note the form of inputs and outputs can vary, for example the target may also be a vector or tensor, rather than a scalar as in Equation 19.

Supervised learning algorithms are typically applied to regression or classification problems. A famous classification example involves the iris data set [2], where ML algorithms can successfully (up to high accuracy) classify the type of iris plant (target), from its features. Each individual plant corresponds to one example, and the features are measurements such as petal width or length. Given a data set to train on or "learn", supervised learning algorithms can then predict the (e.g. iris species) for a new unseen example.

To evaluate how accurate the function f_{ML} is, the entire dataset can be split into 3 sets, the training data, the validation data, and finally, the 'test data. The ML algorithm is applied to the training data and a form of f_{ML} is found. This form of f_{ML} is validated on validation

data. If the accuracy is not sufficient then the ML algorithm can be altered and the process is repeated. The remaining test data is unseen data, which the algorithm has not used during training.

3.4.2 Measuring Accuracy of Supervised Learning Algorithms

The accuracy of a given model is obtained by acting on the "test data" with f_{ML}. For classification problems the accuracy is simply the percentage of correctly predicted examples from the "test data". For regression problems, where each output (e.g scalar y_i) is a continuous variable, the following measures exist:

The **Mean Absolute Error** is an average measure (over M test examples and their corresponding features),

$$\text{MAE} = \frac{1}{M}\sum_{i}^{M}|y_i - \hat{y}_i|, \tag{20}$$

where $\hat{y}_i$ is the ML prediction for the i'th data point and y_i is the actual value (y_i could aslo be a vector $\vec{\hat{y}_i}$, in which case the Euclidean norm is used).
The **Mean Squared Error** (MSE) is given by,

$$\text{MSE} = \frac{1}{M}\sum_{i}^{M}(y_i - \hat{y}_i)^2. \tag{21}$$

The accuracy of a model will be low if the model is underfitting, where underfitting means the training error is too large. Overfitting can also occur, in this case the training error is low, however the test error is not sufficiently low and the model fails to generalise. It is therefore important to monitor the losses of all the data sets.

With these measures, the accuracy of an ML approach to the Hubbard model, specifically if the desired functional is being learned, can be measured.

3.4.3 The Learning Goal: The Hubbard Model Density Functional

Following equations 16 and 18, the function sought after is the true DFT functional,

$$f_{\text{DFT}} : \{n_{i\sigma}^{GS}\} \mapsto F[\{n_{i\sigma}^{GS}\}] = E_{GS} - \sum_i \nu_i n_i^{GS} = \langle\psi_{GS}|(\hat{\mathbf{T}} + \hat{\mathbf{U}})|\psi_{GS}\rangle, \tag{22}$$

where $\{n_{i\sigma}^{GS}\}$ is the on-site occupation for the ground state energy E_{GS}, which acts as the input to the ML algorithm.

3.5 Neural Networks

ANN's provide a general, practical method for learning functions from examples. A Neural Network consists of an input layer, hidden layer(s) and a final output layer. The layers consist of many units, which connect to all units in adjacent layers, as shown in Figure 1. Data enters the input layer, each unit in this layer corresponds to a feature of the example. Every unit receives inputs adjusted by connection weights and processes these to form outputs. This output is activated or sent to the next layer's units, if a weighted sum (Σ) satisfies a condition specified by the activation function.

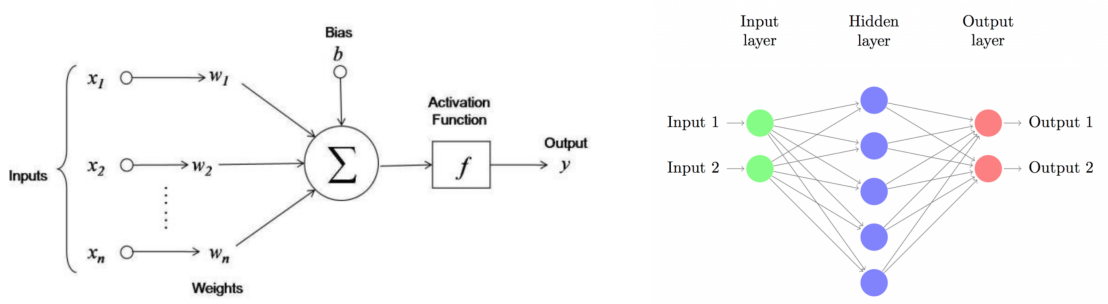

Figure 1: (a) An example of a Neural Network with no hidden layers and a single output. (b) An overview of a Neural Network with one hidden layer and two outputs [12].

There exist many forms of activation functions, the most common being the Rectified Linear Unit (ReLU) function,

$$g(s) = max[0, s]. \tag{23}$$

This activation function is non-linear, helping complex non-linear relationships in the data to be learned. Also, its piecewise linear property and simple derivatives (derivative at discontinuity $s = 0$ is evaluated as 0 during training) are convenient for training (backpropagation). The reader is referred to [12] for details on backpropagation, stochastic gradient descent and activation functions. For a general neural network with K hidden layers, with input vector $\mathbf{x}$ (layer 0), the first and second layer are expressed as:

$$\mathbf{h}_1 = w_{ij}^{0,1}\mathbf{x} + \mathbf{b}_0, \tag{24}$$

$$\mathbf{h}_2 = w_{ij}^{1,2}\mathbf{h_1} + \mathbf{b}_1. \tag{25}$$

The weights between unit i in the previous layer and unit j in the current layer are labelled w_{ij}. The bias vector ensures that a constant term is present in the layers, which may be required for successfull ML (by shifting the activation function). The final output is then given by,

$$\hat{\mathbf{y}} = w_{ij}^{K(K-1)}\mathbf{h}_K + \mathbf{b}_K. \tag{26}$$

Thus for such a fully-connected Neural Network the number of parameters to learn grows rapidly with input size and number of layers.

3.6 Convolutional Neural Networks

CNN's are a specialised form of neural networks for processing data with a grid-like topology. Examples of such data are images (pixels form a grid) or time-series data (1-D grid taking samples at equally spaced time intervals). CNN's employ a form of convolution in one or more layers rather than matrix multiplication as in equations 24, 25 and 26. 1-D real valued convolution of a general function *f(t)* and a weighted function *w(t)* is mathematically defined as

$$(f \circledast w)(t) = \int_{-\infty}^{\infty} f(\tau)w(t-\tau)d\tau. \tag{27}$$

In convolutional network terminology, the first argument to the convolution is referred to as the input, and the second as the kernel. The output is referred to as the feature map [12]. For time-series data, time is discretised and the convolution can be expressed by,

$$(f \circledast w)(t) = \sum_{\tau=-\infty}^{\tau=\infty} f(\tau)w(t-\tau). \tag{28}$$

For ML purposes the inputs are usually a multidimensional array of data, and the kernel a multidimensional array of parameters (weights) that are adapted by the learning algorithm. The infinite sum is then replaced by finite summation over array elements. In related work [24], 2-D CNN kernel filters are used, therefore it makes sense to discuss the 2-D case. As an example, if the input is a 2-D image (array of pixels labelled by $I(i,j)$), the kernel can be 2-D array $W(m,n)$, resulting in an output array S, satisfying,

$$S(i,j) = (I \circledast W)(i,j) = \sum_m \sum_n I(m,n)W(i-m, j-n). \tag{29}$$

In contrast to typical ANN layers, where every unit interacts with every unit in adjacent layers via matrix multiplication, CNN's can have fewer interactions or connections. This is accomplished by making the kernel size smaller than the input. CNN's can therefore be more efficient as fewer parameters have to be stored. Convolution is an extremely efficient way of describing transformations that apply the same linear transformation of a small local region across the entire input [12]. A key property of CNN's is that their convolutions are equivariant to translation. For detailed theory on this group equivariance the reader is re-

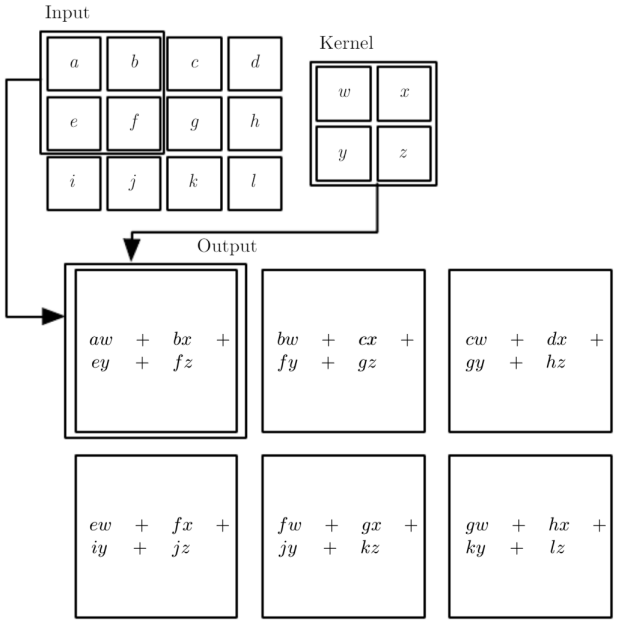

Figure 2: Illustrative example, taken from [12], of 2D convolution on pixels with "valid padding". This involves restricting the kernel window to positions where it fits entirely in the image. The kernel window "slides" through the input image. Due to this, CNN's can identify features or traits in data sets [12] .

ferred to [6], a less detailed description is provided here:
For CNN's the resulting convolutional matrices $S(w)$, in Equation 29, are circulant matrices (each row vector is rotated one element to the right relative to the preceding row vector). Importantly all convolutional matrices $S(w)$ commute with the translation operators or shift operators (which are also circulant matrices). This means circulant matrices enable translation equivariance to convolutions. In other words, when changing the location of the input, the outputs of the CNN layer will be the same, only shifted. Mathematically this stems from the fact that all circulant matrices have the same eigenspace (Fourier eigenspace) and Equation 29 applies the same filter W to every part of the image. Therefore, if the networks learns to recognize a certain feature, e.g., circles, in one part of the image, then it will be able to do so in any other part as well.

Multi-layer networks (Deep Learning networks) can be constructed from combinations of CNN's layers and fully-connected layers. Further layers can be added to avoid over-fitting such as the Dropout layer which randomly removes a fraction of the units of a layer (helping the model to not become too specialised to the training data).

The aim of these algorithms is to construct the desired function or mapping by minimizing corresponding loss functions (e.g. the MSE). The optimisation of the loss functions can be

achieved via a number of methods, many of which are based on or similar to the method of stochastic gradient descent.

3.7 Quantum Simulation

The Schrödinger Equation for time evolution of quantum mechanical systems is given by:

$$i\hbar\frac{d}{d\mathrm{t}}|\Psi\rangle = H|\Psi\rangle, \tag{30}$$

whose solution for a time independent Hamiltonian H is given by:

$$|\Psi(\mathrm{t})\rangle = e^{iH\mathrm{t}}|\Psi(0)\rangle. \tag{31}$$

The exponentiation of the evolution matrix is simplified when the Hamiltonian can be written as a sum of local Hamiltonian's H_k, since they only act on small subsystems. Generally H_k and H_j don't commute and $e^{-iH\mathrm{t}} \neq \prod_k e^{-iH_k\mathrm{t}}$. The Trotter formula for Hermitian operators A and B:

$$\lim_{r\longrightarrow\infty}(e^{\frac{iA\mathrm{t}}{r}}e^{\frac{iB\mathrm{t}}{r}})^r = e^{i(A+B)\mathrm{t}}, \tag{32}$$

helps with non-commuting local Hamiltonians at the cost of errors determined by the size of time intervals $\Delta\mathrm{t} = \frac{\mathrm{t}}{r}$ [25]:

$$e^{i(A+B)\Delta t} = e^{iA\Delta t}e^{iB\Delta t} + \mathcal{O}(\Delta t^2). \tag{33}$$

Note that it is possible to construct higher order approximations. The following unitary can repetitively be applied to the initial state to evolve the system to $|\Psi(\mathrm{t})\rangle$, up to error corrections.

$$e^{-i\sum_{k=1}^{m}H_k\mathrm{t}} \approx (\prod_{k=1}^{m} e^{-iH_k\frac{\mathrm{t}}{r}})^r. \tag{34}$$

3.8 Variational Quantum Eigensolver (VQE)

Given a Hermitian matrix H with an unknown minimum eigenvalue $\lambda_{\min}$ and associated eigenstate $|\psi_{\min}\rangle$, VQE provides a bound on λ_θ:

$$\lambda_{min} \leq \lambda_\theta \equiv \langle\psi(\theta)|H|\psi(\theta)\rangle. \tag{35}$$

Where $|\psi(\theta)\rangle = U(\theta)|\psi\rangle$ is a parameterised state resulting from acting on an initial state via a parameterised unitary. Applying this to a Hamiltonian H yields:

$$\lambda_{min} \leq \langle H\rangle_\psi = \langle\psi|H|\psi\rangle = \sum_{i=1}^{n}\lambda_i|\langle\psi_i|\psi\rangle|^2. \tag{36}$$

Where in Equation 36, H has been substituted with its weighted eigenvector representation, $\sum_{i=1}^{n} \lambda_i |\Psi_i\rangle\langle\Psi_i|$. A parameterised quantum circuit can be optimised classically (outer loop) using the measurements of the observables (i.e the energy) as a loss function. The parameters θ of this hybrid algorithm can then be tuned and the minimum eigenvalue sought after. The algorithm typically has the following structure:

1. An initial state $|\Psi_0\rangle$ is prepared on the quantum computer. Ideally this state has a large overlap with the desired ground state.

2. Next, a parameterised quantum circuit $U(\theta)$ is applied, yielding the variational ansatz state $|\Psi(\theta)\rangle = U(\theta)|\Psi_0\rangle$.

3. The initial variational paramters θ_0 are chosen to prepare $|\Psi(\theta_0)\rangle$. Then the corresponding energy $E(\theta_0) = \langle\Psi(\theta_0)|H|\Psi(\theta_0)\rangle$ is measured.

4. Based on this energy, the classical optimiser adjusts the parameters θ until convergence to a minimum energy.

The VQE method can obtain results with relatively short circuits, making it well suited to NISQ devices. The success of the method is sensitive to the choice of the parameterised circuit and the problem in question. Important elements to take into account when selecting the type of parameterised circuit are: the ability to produce the ground state with high fidelity, and the implementability of the circuit on NISQ hardware. These two considerations are closely connected, due to the limited connectivity, length and qubit number that current quantum circuits can have, but also the number of variational parameters a classical optimizer can deal with.

4 Exact Diagonalisation

In this section, the ED method of Sanvito et al. [24] is implemented for the one-dimensional Hubbard model.

4.1 Solving the 1D Hubbard Model Exactly

The model can be solved by writing the Hamiltonian over a convenient representation.
The basis states are the number of possible configurations of the system, composed of $2L$ orbitals and N electrons. The ordering of basis states can be chosen to be spin-up followed by spin-down orbitals, such that: A basis state representing two electrons present at the first site with $L = 2$ is,

$$|1, 0 : 1, 0\rangle = |1, 0, 1, 0\rangle. \tag{37}$$

With $L = 2$, $N = 2$, $N_\uparrow = 1 = N_\downarrow$ one has 4 orbitals and 2 electrons, the wavefunction of the system can be written as a linear combinations of the basis vectors:

$$\mathbf{Basis}_{L=2,N=2,N_\uparrow=N_\downarrow=1} =$$

$$\{|1, 0, 1, 0\rangle, |1, 0, 0, 1\rangle, |0, 1, 1, 0\rangle, |0, 1, 0, 1\rangle\}. \tag{38}$$

Each basis state being represented by a vector $|\alpha_1, \alpha_2, \alpha_3, \alpha_4\rangle$, where α_i corresponds to the number of electrons in orbital i. Note the wavefunctions are automatically antisymmetrised through the basis definition, in terms of fermionic creation operators. This follows from the anti-commutation relations given in Equation 64, in Section 6.1. Once the basis states are generated, the eigenvalues and eigenstates of the problem are computed by representing the Hamiltonian over these states.

4.1.1 Hilbert Space Dimension

For the general 1D Hubbard Hamiltonian given by Equation 4, the dimension of the Hilbert space is

$$n = 2^{2L}. \tag{39}$$

However by restricting to the case where the number of electrons N is fixed, and the number of spin-up electrons is equal to the number of spin-down electrons ($N_\uparrow = N_\downarrow = \frac{N}{2}$), the dimension becomes:

$$n = \binom{L}{N_\uparrow}\binom{L}{N_\downarrow} = \binom{L}{\frac{N}{2}}^2. \tag{40}$$

The basis vectors satisfying $N_\uparrow = N_\downarrow = \frac{N}{2}$ live in this subspace and the Hubbard Hamiltonian can be solved over these states (the Hamiltonian is electron-number preserving and furthermore it preserves $N_\downarrow = N_\uparrow$).

4.1.2 Exact Diagonalisation

The matrix elements of the Hamiltonian, i.e. between basis states
$|s\rangle = |\alpha_1^s, \alpha_2^s, \alpha_2^s, \ldots, \alpha_{2L-1}^s, \alpha_{2L}^s\rangle$ and $|t\rangle = |\alpha_1^t, \alpha_2^t, \alpha_2^t, \ldots, \alpha_{2L-1}^t, \alpha_{2L}^t\rangle$ are computed via,

$$(H)_{s,t} = \langle s|\mathbf{H}|t\rangle. \tag{41}$$

The diagonalisation of the matrix $(H)_{s,t}$ will provide the eigenvalues, E_m, (total energies of the system) and their corresponding eigenvectors $\psi^m = (\beta_1^m, \beta_2^m, \ldots, \beta_{n-1}^m, \beta_n^m)^T$. The corresponding eigenstate for E_m is the wavefunction,

$$|\psi^m\rangle = \sum_i^n \beta_i^m |\alpha_1^i, \alpha_2^i, \alpha_3^i \cdots \alpha_{2L-1}^i, \alpha_{2L}^i\rangle. \tag{42}$$

Equations 4, 38 and 41 yield the Hamiltonian Matrix corresponding to $L = 2$, $N = 2$, $N_\uparrow = N_\downarrow = 1$:

$$H_{L=2,N=2,N_\uparrow=N_\downarrow=1} = \begin{pmatrix} U + 2\nu_1 & -(t+t) & -(t+t) & 0 \\ -(t+t) & \nu_1 + \nu_2 & 0 & -(t+t) \\ -(t+t) & 0 & \nu_1 + \nu_2 & -(t+t) \\ 0 & -(t+t) & -(t+t) & U + 2\nu_2 \end{pmatrix}. \tag{43}$$

Red font denotes terms resulting from periodic boundary conditions. For example, the action of $\mathbf{H}$ on the basis state $|1\rangle = |1, 0, 1, 0\rangle$ for the ring is,

$$\mathbf{H}|1, 0, 1, 0\rangle = (U + 2\nu_1)|1, 0, 1, 0\rangle - (t+t)|0, 1, 1, 0\rangle - (t+t)|1, 0, 0, 1\rangle. \tag{44}$$

By ED the matrix form of the Schrödinger Equation, $H|\psi\rangle = E|\psi\rangle$, can be solved. Note that in the worst case, matrix diagonalisation is a procedure that requires a computational time scaling as $\mathcal{O}(n^3)$, where n is the dimension of the matrix to diagonalise. This dimension n is given by Equation 40, the dimension of the Hilbert Space, resulting in a significant scaling problem, e.g. for even $N_\uparrow = N_\downarrow$, one has $n = \binom{L}{\frac{N}{2}}^2$: $\binom{4}{1}^2 = 16$, $\binom{8}{2}^2 = 784$, $\binom{16}{4}^2 = 3,312,400$.
As a result of this scaling problem, the ED method is limited to very small systems.

4.1.3 On-site Occupation

Following ED, the on-site occupations of the system in a given state, (e.g the ground-state) can be found. The on-site occupation for orbital (i, σ), for a given energy level E_m is given by,

$$n^m_{i,\sigma} = \langle \psi^m | \hat{n}_{i,\sigma} | \psi^m \rangle. \tag{45}$$

where $|\psi^m\rangle$ is given by Equation 42. The set of on-site occupations, for each eigenstate, form the electron density.

4.2 ML Data Set Generation and Implementation

An overview of the data set generation is visualised in a flowchart in Figure 3.

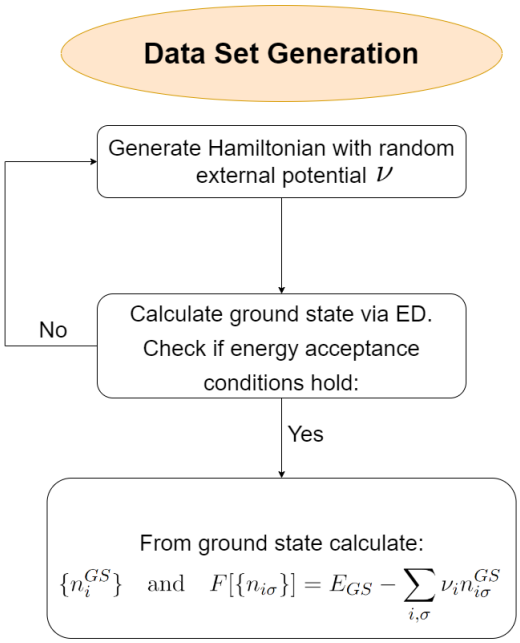

Figure 3: Data Set Generation for the ED method. Each entry in the data set consists of an electron density (example) and a corresponding Functional energy (target).

Data sets for ML were generated for the following configurations:

1. Quarter-filling with $L = 8, N = 4, N_\uparrow = 2 = N_\downarrow$ and larger allowed energy differences than in [24]. The data set size was $S = 6300$, which was increased to $S = 10800$ using the symmetries (described in Section 4.2.1) of the 1D Hubbard model.

2. Quarter-filling with $L = 8, N = 4, N_\uparrow = 2 = N_\downarrow$. The data set size was $S = 6300$, which was increased to $S = 10800$ using the symmetries of the 1D Hubbard model.

3. Quarter-filling with $L = 4, N = 2, N_\uparrow = 1 = N_\downarrow$. The data set size was $S = 6300$, which was increased to $S = 50400$ using the symmetries of the 1D Hubbard model.

4. Half-filling with $L = 4, N = 4, N_\uparrow = 2 = N_\downarrow$. The data set size was $S = 6300$, which was increased to $S = 50400$ using the symmetries of the 1D Hubbard model.

For these data sets, each *example* is the ground state electron density $\{n_{i\sigma}^{GS}\}$ resulting from the Hamiltonian with randomly generated on-site energies $\{\nu_i\}$. The corresponding *target* is the DFT functional. The Coulomb term U was set to 10 and periodic boundary conditions were kept throughout, resulting in a ring configuration. A Python script that generates and solves the Hubbard Hamiltonian matrix exactly (ED) was written, which can be found at [27]. This script returns the relevant ground-state density $\{n_{i\sigma}^{GS}\}$ with the corresponding DFT functional $F[n_{i\sigma}^{GS}] = E_{GS} - \sum_{i,\sigma} n_{i\sigma}^{GS}\nu_i$.

In order to stay in line with related work [24], specific conditions were implemented. Following the procedure of [24], the random external potential is generated from uniform distribution with $\nu_i \in [-W, +W]$, with W varying between $0.005t$ and $2.5t$. As in [24], external potentials $\{\nu_i\}$ yielding ground state energies greater than $0.15t$ different in magnitude than that of the homogeneous case, are discarded, in order to prevent the dataset from having very large fluctuations in the total energy. Data Set 2 has these specific conditions. In this way, the authors of [24] allow for random external potentials with large magnitudes, but only actually include the data point if its corresponding ground state E_{GS} is close to the homogeneous ground state energy E_0.

Data Set 1 was chosen to be a more general case than in [24], allowing for larger fluctuations. For it, the acceptance condition for inclusion is changed from the original condition

$$|E_{GS} - E_0| < 0.15, \tag{46}$$

to the condition

$$|E_{GS} - E_0| < |0.15E_0|\,. \tag{47}$$

This new acceptance condition for Data Set 2 was inspired by the fact that $E_0 \approx -5.6$ for the case $L = 8, N = 4, N_\uparrow = 2 = N_\downarrow$.
All energies are considered in units of t (=1), throughout the project these units are implied, although not explicitly given. For example, a result $E_{GS} = -5.6$ corresponds to $E_{GS} = -5.6t$.

The *openfermion* [23] Python library of Google Quantum AI was used to generate instances of the Hubbard Model. The *openfermion.hamiltonians.fermi_hubbard* method provides a

fermionic operator that can easily be converted to a sparse operator and then diagonalised. The projection into the correct subspace (e.g . $N = \frac{L}{2}, N_\uparrow = \frac{L}{4} = N_\downarrow$) was applied, reducing the cost of ED. Psuedocode for the ED data set generation is provided in Algorithm 1.

Algorithm 1 ED Data Set Generation

Inputs: S, L, N
Outputs: densities, Functionals
size=0
declare densities array
declare Functionals array
Generate basis_states (e.g for quarter filling $N = \frac{L}{2}, N_\uparrow = \frac{L}{4} = N_\downarrow$).
while size $< S$ **do**
    Draw random uniform external potential $\nu_i \in [-W, +W]$, with $W \in [0.005t, 2.5t]$.
    Create **H**: Hubbard model instance with added $\sum_i^L \sum_\sigma \nu_i \hat{a}^\dagger_{i,\sigma} \hat{a}_{i,\sigma}$ terms.
    Project **H** into subspace using basis_states.
    Diagonalise **H** to get ground state $|\Psi_{GS}\rangle$ and E_{GS}.
    if $|E_{GS} - E_0| < 0.15$ **then**
        Continue
    else
        Calculate $\{n_i^{GS}\}$.
        Calculate $F[\{n_{i\sigma}\}] = E_{GS} - \sum_{i,\sigma} \nu_i n_{i\sigma}^{GS}$.
        densities[size]= $\{n_{i\sigma}^{GS}\}$
        Functionals[size]=$F[\{n_{i\sigma}\}]$
        size = size +1

Calculating $n_{j,\sigma}^{GS} = \langle \psi^{GS} | \hat{n}_{j,\sigma} | \psi^{GS} \rangle$ involves the ground state

$$|\psi^{GS}\rangle = \sum_i^n \beta_i^{GS} |\alpha_1^i, \alpha_2^i, \alpha_3^i \cdots \alpha_{2L-1}^i, \alpha_{2L}^i\rangle, \tag{48}$$

specifically the probability amplitudes $|\beta_i^{GS}|^2$ for each basis state i. The value of $n_{j,\sigma}^{GS}$ can be found by summing over all the basis states $\{i\}$ and adding $|\beta_i^{GS}|^2$ to $n_{j,\sigma}^{GS}$ if the basis state i has an electron present in position $(j, \sigma) = \alpha^i_{j+L\delta_{\sigma\downarrow}}$. Note $j \in \{1, 2, \cdots, L-1, L\}$ and the dirac delta term $\delta_{\sigma\downarrow}$ is due to the labelling of orbitals in the basis states, in Equation 48.

4.2.1 Data Set Extension: Symmetries

As in [24], the symmetries of the 1D Hubbard model are used to extend the size of the data set. The 1D Hubbard model is mirror-symmetric, this means that for any external potential $\nu_i \longrightarrow \nu_{L+1-i}$ yields a mirror-symmetric electron density with the same total energy.
Similarly there exists a translational symmetry, such that $\nu_i \longrightarrow \nu_{i+1}$ yields a shifted electron density with the same total energy. With these two symmetries the dataset is increased by a total factor of $2L$.

4.3 ML Model

For each data set, a Deep Learning model was built to learn the functional. The data sets were split into training(72%), validation(18%) and test(10%) sets.

4.3.1 Standardisation

Feature vector components, $n_{i\sigma}$ for sites i over all the examples, and targets were normalised as follows,

$$z \mapsto \frac{(z - \mu_{train})}{\sigma_{train}}. \tag{49}$$

All the data sets are standardised using the training data mean and standard deviation. Where μ_{train} is the mean and σ_{train} is the standard deviation, of the training data. Standardisation generally improves the likelihood of convergence of ML models [12], especially deep neural networks.

4.3.2 Data Preprocessing for CNN Layers

The 1D Hubbard model has an $SU(2)$ symmetry which combined with the condition $N_\uparrow = N_\downarrow$ guarantees that the local site occupation remains spin unpolarised, namely that $n_{i\uparrow} = n_{i\downarrow}$. This means that in the special case ($N = \frac{L}{2}, N_\uparrow = \frac{L}{4} = N_\downarrow$), the functional depends only on one of the two spin electron densities, for instance on $\{n_{i\downarrow}\}$ [24]. Additionally to account for periodic boundary conditions and to ensure each component of the electron density is acted on by the kernel the same number of times, $\{n_i\}$ was extended. This resulted in new "wrapped" electron densities. For example in the 8-site setting with the chosen kernel size of (3,1),

$$\{2n_{i\downarrow}\} = \{2n_{i\uparrow}\} = \{n_i\} =$$

$$\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8\} \longrightarrow \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_1, n_2, n_3\}. \tag{50}$$

To act as an input into a 2D CNN layer, these new examples were reshaped into a 2-D array of dimensions (L+3,1). The chosen kernel size of (3,1) could then "sweep" through the examples, allowing the interaction between the sites at the end and the sites at the beginning to be encoded.

4.3.3 Network Architecture

The ML algorithm were implemented using the *Keras* and *TensorFlow* Python packages [5, 1]. For benchmarking purposes the same network design as in [24] is kept. The convolution neural network used, had one CNN layer with 8 convolutional filters, followed by 2 fully connected layers each with 128 units/nodes and finally an output layer. The ReLU

activation function, given by Equation 23, was used for the fully connected layers.

A maximum of 50 loops/epochs over the entire training data were run, optimising the MSE loss function. Using the "Modelcheckpoint" callback in *Keras* the history of the model was saved after each epoch. After each epoch the weights of the model were updated if the validation loss improved. This ensured the model was not overfitting. The "EarlyStopping" method was used, which stops training once the validation loss stops improving. Once this occurred the best model (generalises best on validation data) is kept.

The architecture of [24] was kept and remained fixed throughout experiments. However to ensure that results did not depend on the training/validation/test set splits, 5-fold cross validation was performed. In k-fold cross validation, the original data set is randomly partitioned into k equal sized subsets. For each of the k subsets, it is retained as the test data and the remaining $k-1$ subsamples are used as training data. After this has been repeated, the results for the k different splits is averaged. Additionally, hyperparameter optimisation was performed for the learning rate, $\alpha \in \{0.1, 0.01, 0.001, 0.0001, 0.00001\}$ of the Adam optimiser [16].

4.4 ED: Results

For each data set, the experimental results of the ED method are outlined in this subsection.

4.4.1 Data Set 1: $L = 8, N = 4, N_\uparrow = 2 = N_\downarrow$

For this data set the acceptance condition for energy differences was loosened such that it became:

$$|E_{GS} - E_0| < |0.15 E_0| \,. \tag{51}$$

This allowed for much larger variation in total energies and therefore fluctuations within the data set. As an example, for a given split, test set predictions versus the exact test set are plotted in Figure 4 for the case with Adam learning rate $\alpha = 0.001$. Additionally in Figure 5 the corresponding model loss is plotted for each epoch, showing that the desired Functional is learnt by the CNN network. In Table 1 the full test set MSE results are provided, for each cross validation fold and learning rate parameter α. Following hyperparameter optimisation, the average MSE over the test sets is 0.000151. For the learning rates $\alpha \in \{0.1, 0.01\}$ the performance was significantly lower, and therefore in what follows, hyperparameter optimisation was restricted to the subset $\alpha \in \{0.001, 0.0001, 0.00001\}$. Overall the desired functional is learned by the model, however the accuracy can be improved by keeping the same acceptance conditions as in [24].

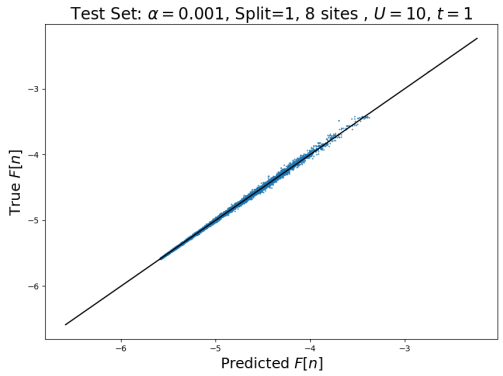

Figure 4: Example of test set Functional $F[n]$ predictions vs $F[n]$ true values.

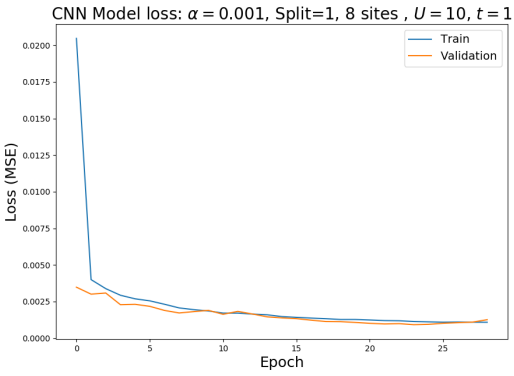

Figure 5: Corresponding Model loss over epochs.

MSE	Split 1	Split 2	Split 3	Split 4	Split 5	Average
α=0.1	0.157	0.152	0.156	0.157	0.154	0.155
α=0.01	0.000474	0.000493	0.000536	0.000750	0.000438	0.000538
α=0.001	0.000178	0.000264	0.000171	0.000201	0.000330	0.000229
α=0.0001	0.000135	0.000134	0.000197	0.000133	0.000154	**0.000151**
α=0.00001	0.000362	0.000336	0.000360	0.000402	0.000328	0.000357

Table 1: Data Set 1: Cross Validation and Hyperparameter Tuning for the case $L = 8, N = 4, N_\uparrow = 2 = N_\downarrow$, with more general energy acceptance conditions. MSE values up to 3 significant figures.

4.4.2 Data Set 2: $L = 8, N = 4, N_\uparrow = 2 = N_\downarrow$

The original energy acceptance condition:

$$|E_{GS} - E_0| < 0.15, \tag{52}$$

was kept and the model results are displayed in Table 2. Overall the performance improves and following hyperparameter optimisation, the average MSE over the test sets is 1.899×10^{-5}. This accuracy confirms that the functional has been learned by the ML model.

4.4.3 Data Set 3: $L = 4, N = 2, N_\uparrow = 1 = N_\downarrow$

Here the quarter-case is considered with $L = 4$ and again the same acceptance condition as in Equation 52 is kept. The results following hyperparameter optimisation are displayed in Table 3. The average MSE over the test sets is 0.619×10^{-5}, again showing that the desired

MSE $\cdot 10^{-5}$	Split 1	Split 2	Split 3	Split 4	Split 5	Average
α=0.001	6.535	12.467	10.453	4.339	2.981	7.355
α= 0.0001	1.603	1.938	2.0673	1.917	1.967	**1.899**
α=0.00001	4.263	3.933	4.115	3.899	3.7412	3.991

Table 2: Data Set 2: Cross Validation and Hyperparameter Tuning for the case $L = 8, N = 4, N_\uparrow = 2 = N_\downarrow$, as in [24]. MSE values up to 3 significant figures.

DFT functional has been learned.

MSE $\cdot 10^{-5}$	Split 1	Split 2	Split 3	Split 4	Split 5	Average
α=0.001	5.154	2.699	4.503	1.501	1.871	3.146
α= 0.0001	0.763	0.460	0.689	0.385	0.796	**0.619**
α=0.00001	2.344	1.518	1.878	0.984	1.237	1.592

Table 3: Data Set 3: Cross Validation and Hyperparameter Tuning for the configuration $L = 4, N = 2, N_\uparrow = 1 = N_\downarrow$. MSE values up to 3 significant figures.

4.4.4 Data Set 4: $L = 4, N = 4, N_\uparrow = 2 = N_\downarrow$

Lastly the half-filling case with $L = 4$ was considered, again with acceptance conditions given by Equation 52. The average test set MSE is remarkably low, with a value of 4.78×10^{-8}. In this case the ML model functional is almost indistinguishable from the desired exact functional. This impressive performance is due to the chosen configuration and the structure of data set. Next the effect of the data set structure is investigated.

MSE $\cdot 10^{-7}$	Split 1	Split 2	Split 3	Split 4	Split 5	Average
α=0.001	2.261	2.276	3.194	1.038	1.444	2.043
α= 0.0001	0.627	0.318	0.465	0.423	0.559	**0.478**
α=0.00001	0.957	1.287	1.145	0.987	1.034	1.087

Table 4: Data Set 4: Cross Validation and Hyperparameter Tuning for the configuration $L = 4, N = 4, N_\uparrow = 2 = N_\downarrow$ MSE values up to 3 significant figures.

4.4.5 Data Set Structure and Discussion

The varying accuracy of the ML model is a direct consequence of the data set in question. The more complex the data set (higher fluctuations in Functional Energies) the lower the

accuracy. From Figure 6 it is clear that Data Set 1 has far more fluctuation in F than Data Set 2, due to the energy acceptance conditions. For Data Set 2 and Data Set 3, the difference in fluctuations of F is even more pronounced. The performance of the ML model is negatively affected by the increase in fluctuations. This is an important factor to consider for the general approach of applying ML models to ED or VQE quantum data sets. Especially for quantum data generated by a VQE, where there may be less knowledge/control over which data points to admit into a training set. Despite this consideration, one can always increase the data set sizes to improve ML performance.

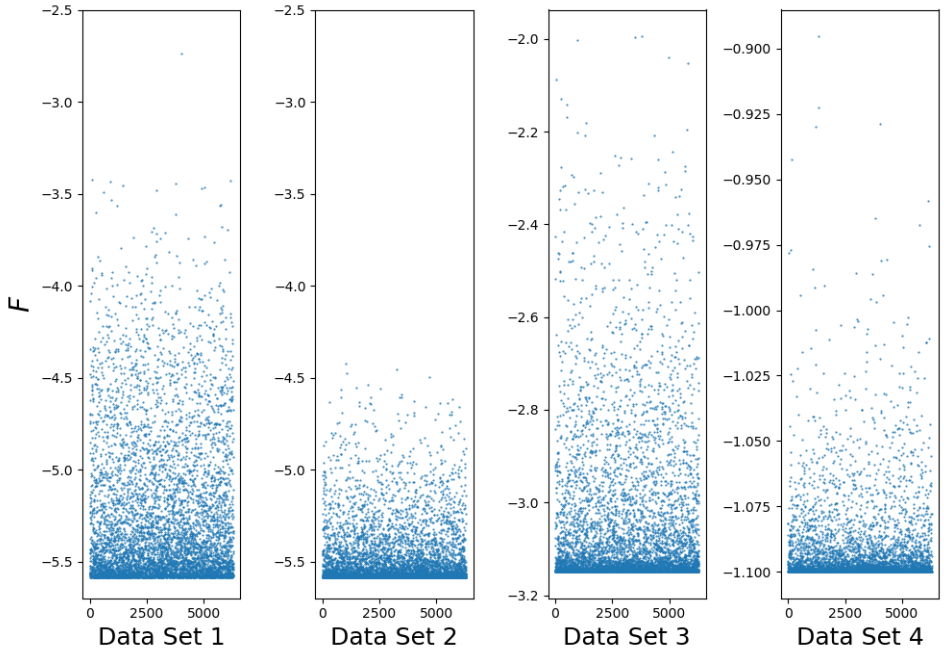

Figure 6: Functional values over the 6300 data points in each data set. The ML model average test set MSE is: 1.51×10^{-4} for Data Set 1, 1.899×10^{-5} for Data Set 2, 6.19×10^{-6} for Data Set 3 and 4.78×10^{-8} for Data Set 4.

In conclusion, the ML model learns the desired functional well across all ED data sets. On a NISQ quantum computer exact results are not accessible and therefore in the following Section, the impact of incorporating measurement/shot noise is investigated.

5 Measurement Noise

Given the ED results, the exact ground-state density $\{n_{i\sigma}^{GS}\}$ with the corresponding DFT functional, $F[n_{i\sigma}^{GS}] = E_{GS} - \sum_{i\sigma} n_{i\sigma}^{GS}\nu_i$, could easily be found. Conversely, on a quantum computer repetitive measurements in the relevant bases are typically required. In this section the ground state wavefunctions from the ED method are taken and noisy measurements are emulated. Throughout, the assumption is made that the quantum computer has yielded an accurate ground state $|\Psi_{GS}\rangle$, from which the properties,

1. $|\Psi^{GS}\rangle \sim \{n_i^{GS}\}$,
2. $\langle\Psi^{GS}|\mathbf{H}|\Psi^{GS}\rangle = \langle\Psi^{GS}|\left(\sum_{i,\sigma}^{L}\nu_i\hat{a}_{i,\sigma}^{\dagger}\hat{a}_{i,\sigma} - t\sum_{\langle i,j\rangle,\sigma}^{L}\hat{a}_{i,\sigma}^{\dagger}\hat{a}_{j,\sigma} + U\sum_{i}^{L}\hat{n}_{i,\downarrow}\hat{n}_{i,\uparrow}\right)|\Psi^{GS}\rangle = \sum_k\langle\Psi^{GS}|H_k|\Psi^{GS}\rangle = E_{GS}$,

are sought after. Note the desired DFT functional (target) requires measuring the kinetic and Coulomb terms $\langle\Psi^{GS}|\left(-t\sum_{\langle i,j\rangle,\sigma}^{L}\hat{a}_{i,\sigma}^{\dagger}\hat{a}_{j,\sigma} + U\sum_{i}^{L}\hat{n}_{i,\downarrow}\hat{n}_{i,\uparrow}\right)|\Psi^{GS}\rangle$. For the electron densities, since every qubit represents an orbital, one can repetitively measure $|\Psi^{GS}\rangle$ in the computational basis, hence the notation $|\Psi^{GS}\rangle \sim \{n_i^{GS}\}$.

5.1 Measuring Observbles on a Quantum Computer

Since the wavefunction on a quantum computer cannot be directly accessed, one has to repetitively measure observables. Many observables $\hat{X}$ can be written as a linear combination of polynomially (in the number of qubits) many Pauli strings $\mathcal{P} \in \{\mathcal{I}, \mathcal{X}, \mathcal{Y}, \mathcal{Z}\}$ with real coefficients h_j,

$$\hat{X} = \sum_j h_j P_j. \tag{53}$$

In order to estimate the value of an obervable $\hat{X}$, one has to take M samples, denoted $\{X_1, X_2, \cdots, X_M\}$, and then compute the expectation value: $\frac{1}{M}\sum_m^M X_m$. The standard Chebyshev's inequality,

$$\mathbb{P}\left(\left|\frac{\sum_m^M X_m}{M} - \mu_X\right| \geq \epsilon\right) \leq \frac{\sigma^2}{M\epsilon^2}, \tag{54}$$

shows that one needs to take a number of measurements M that scales as

$$M \sim \frac{1}{\epsilon^2}, \tag{55}$$

to get with high probability, an estimate of μ_X, up to error ϵ. Note, in general better scaling may be possible (such as a Chernoff bound), because often X_m's are bounded. Next the noisy electron density, Coulomb and Kinetic measurements are discussed.

5.2 Noisy Electron Densities

For the electron densities, since every qubit represents an orbital, one can repetitively measure $|\Psi^{GS}\rangle$ in the computational basis. The computational basis measurements were generated according to Algorithm 2. Here the random measurements are simulated by taking Bernoulli trials with probabilities equal to the relevant probability amplitudes $|\beta_i^{GS}|^2$, given by Equation 42.

Algorithm 2 Generate Noisy Electron Densities

Inputs: $|\Psi_{GS}\rangle$,basis_states, M, L
Output: densities
Declare densities array (zero values)
for m in range(M) **do**
    for i in basis_states **do**
        for j in range(2L) **do**
            if Electron present in basis_state position: i[j]='1' **then**
                densities[j]+ = $bernoulli(p = |\beta_i^{GS}|^2)$
            end if
        end for
    end for
end for
divide densities by M
Return: densities
=0

5.3 Noisy Coulomb Energy Measurements

The Coulomb terms $\langle\Psi^{GS}|\sum_j^L \hat{n}_{j,\downarrow}\hat{n}_{j,\uparrow}|\Psi^{GS}\rangle$ contribute to the energy when two electrons are present on the same site. The random measurements were performed in similar fashion to Algorithm 2 via Bernoulli trials.
Given the ground state,

$$|\psi^{GS}\rangle = |\sum_i^n(\beta_i^{GS}|\alpha_1^i, \alpha_2^i, \alpha_3^i \cdots \alpha_{2L-1}^i, \alpha_{2L}^i\rangle, \tag{56}$$

for each basis state the occurrence of doubly-occupied sites was found and used to create a probability matrix. As a simple example, consider the i'th basis state

$$|1,0,0,0,0,0,1,0:1,0,0,0,1,0,1,0\rangle = |1,0,0,0,0,0,1,0,1,0,0,0,1,0,1,0\rangle, \tag{57}$$

with probability amplitude $|\beta_i^{GS}|^2$. The i'th row of the probability matrix is then equal to

$$\begin{pmatrix} |\beta_i^{GS}|^2 & 0 & 0 & 0 & 0 & 0 & |\beta_i^{GS}|^2 & 0 \end{pmatrix}, \tag{58}$$

since the first and seventh sites are doubly-occupied. A probability matrix of shape $(n \times L)$ was created, where each column corresponds to a probability vector for one of the L double-occupation $\hat{n}_{j,\downarrow}\hat{n}_{j,\uparrow}$ terms. The contribution of $\hat{n}_{j,\downarrow}\hat{n}_{j,\uparrow}$ was estimated by taking n Bernoulli trials with probability equal to the n'th column vector entry. The sampling was repeated M times, yielding an estimate.

5.4 Noisy Kinetic Energy Measurements

Simultaneously measuring all qubits in the computational basis (eigenvectors of the Pauli-Z operator), the electron densities and double occupancy can be determined on a quantum computer. Thus the measurement noise can simply be emulated using Bernoulli trials. For the kinetic energy observable, rotating into the basis in which the kinetic energy operator is diagonal, is required on a quantum computer. The rotation into the diagonal basis is possible on a quantum computer by Bogoliubov transformations, which can be implemented through a Givens rotations circuit [14].
In order to simulate this shot noise, the Kinetic operator $\hat{\mathbf{T}}$ was first diagonalised:

$$\hat{\mathbf{T}} = V^{\dagger} D_{\lambda} V, \tag{59}$$

where the matrix representation of $\hat{\mathbf{T}}$ is the projected Hamiltonian $\hat{\mathbf{H}}$ with all diagonal entries set to zero. Using the basis change matrix $V^{\dagger}$ the ground state is rotated,

$$\Psi^{GS} \mapsto V^{\dagger} \cdot \Psi^{GS} = \begin{pmatrix} \beta_1' \\ \beta_2' \\ \vdots \\ \beta_{n-1}' \\ \beta_n' \end{pmatrix}, \tag{60}$$

followed by conversion to probabilities,

$$\begin{pmatrix} \beta_1' \\ \beta_2' \\ \vdots \\ \beta_{n-1}' \\ \beta_n' \end{pmatrix} \mapsto \begin{pmatrix} |\beta_1'|^2 \\ |\beta_2'|^2 \\ \vdots \\ |\beta_{n-1}'|^2 \\ |\beta_n'|^2 \end{pmatrix}. \tag{61}$$

The eigenvalues of the kinetic operator are randomly sampled according to the probabilities given by Equation 61, where eigenvalue j is chosen with probability $|\beta'_j|^2$. This was repeated M times and averaged to yield the estimate.

5.5 Noisy ML Models

The ED data sets from the previous section were used to generate 250 noisy versions for each data set, with varying number of measurements M, where the values of $M \in [10^2, 10^{6.3}]$ were log-scaled. This range for M was chosen for all data sets, after it was observed that the noisy model performance converged with the exact model at $M \approx 10^6$ for Data Set 1. Then the ML model was applied to each of the these data sets, where the same process was applied as in Section 4.3. However due to the added noise, Equation 50 does not hold and the wrapped electron densities were set to:

$$\{n_i\} = \{n_{i\downarrow} + n_{i\uparrow}\} =$$

$$\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8\} \longrightarrow \{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_1, n_2, n_3\}. \tag{62}$$

5.6 Nosiy Measurements Results

The Deep Learning network was run on all measurement noise data sets and the following measures recorded for test set MSE:

1. $f_{Noise}(X_{Noise})$ vs $F_{Noise} \longrightarrow$ Pure noise measure.
2. $f_{Noise}(X_{Noise})$ vs $F_{Exact} \longrightarrow$ Noise model against exact Functionals.
3. $f_{Noise}(X_{exact})$ vs $F_{Exact} \longrightarrow$ Exact inputs into noisy model.
4. $f_{Exact}(X_{exact})$ vs $F_{Exact} \longrightarrow$ Exact noiseless model for comparisons.

Where f_{Noise} is a model trained on noisy training data and X_{Noise} and F_{noise} are noisy electron densities (examples) and noisy Functionals (targets), respectively. Note that in the proposed hybrid scheme, the Deep Learning model only sees the pure noise measure during training. Then once it is trained it can take X_{exact} as inputs and correspondingly the third measure is of importance. With these four measures the performance of the model can be evaluated for each data set.

5.6.1 Data Set 1: $L = 8, N = 4, N_\uparrow = 2 = N_\downarrow$

The red line in Figure 7 is simply the MSE between the noisy and exact Functionals, with varying M, over the entire data set. The MSE involves squared terms, which from Equation 55, results in an expected slope of -1 in a log-scaled plot. This is seen in Figure 7, meaning

the method chosen to emulate shot noise succeeded. The black line indicates the exact noiseless model result from the previous section and is independent of M. The ML model is always trained on the noisy data and its $f_{Noise}(X_{Noise})$ vs F_{Noise} MSE is shown in blue. As expected, this value asymptotically reaches the exact model performance as M increases. The positioning of the orange and green line reveal insights into the ML models ability to learn the desired Functional. The orange line represents $f_{Noise}(X_{Noise})$ vs F_{exact} and is lower than the blue line. Additionally the green line, representing $f_{Noise}(X_{Exact})$ vs F_{exact}, is lower than the orange line. Therefore it is clear that the ML model, f_{Noise}, achieves a degree of noise mitigation, for low M. Whereby, the model learns the Functional more so than the noise within the data set.

As expected, for low M, the noisy model MSE is not near the desired the exact model MSE. Then as M increases, the noisy data converges towards the exact data and correspondingly all the model measures converge.

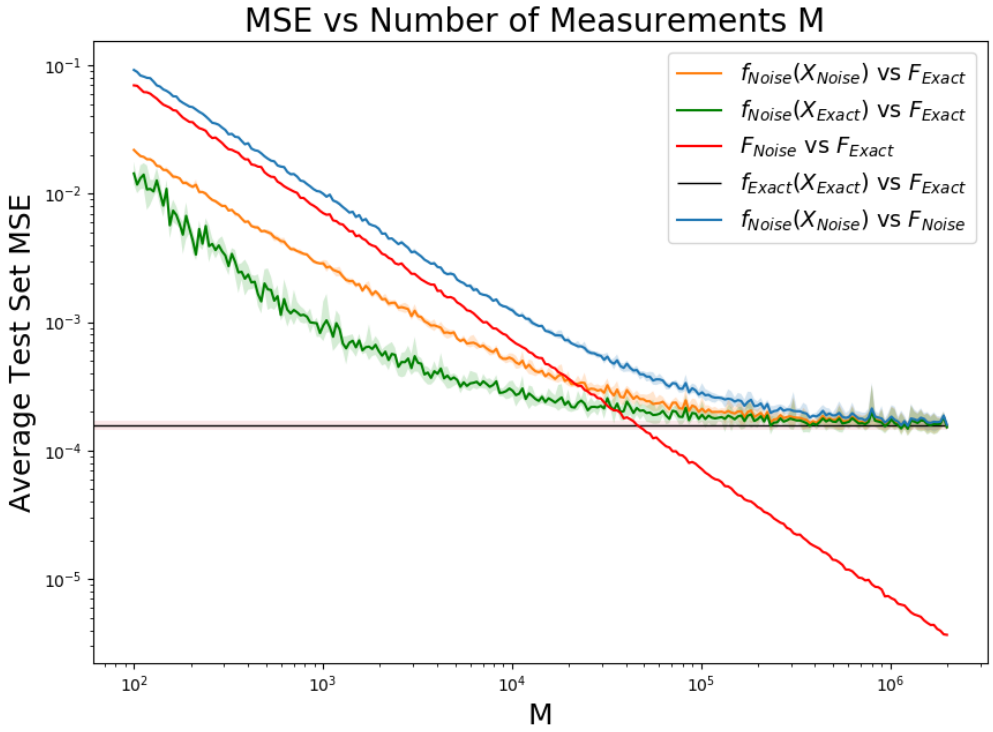

Figure 7: Test Set MSE plot for Data Set 1, for the case $\alpha = 0.0001$. MSE are averaged across the 5 Cross Validation folds and shaded regions denote minimum and maximum MSE values across folds.

5.6.2 Data Set 2: $L = 8, N = 4, N_\uparrow = 2 = N_\downarrow$

The results are visualised in a plot of MSE versus the number of shots M, in 8. For this data set, the effect of measurement noise on the performance of the ML model is similar in nature to that of Data Set 1. The one noticeable difference is that for very low M the $f_{Noise}(X_{Noise})$ vs F_{exact} MSE is lower than the $f_{Noise}(X_{Exact})$ vs F_{exact} MSE, as there is a lot of noise in the data. However as M approaches 10^3 the ML model can again learn to focus more on the Functional rather than the shot noise in the data, and the green line descends below the orange line.

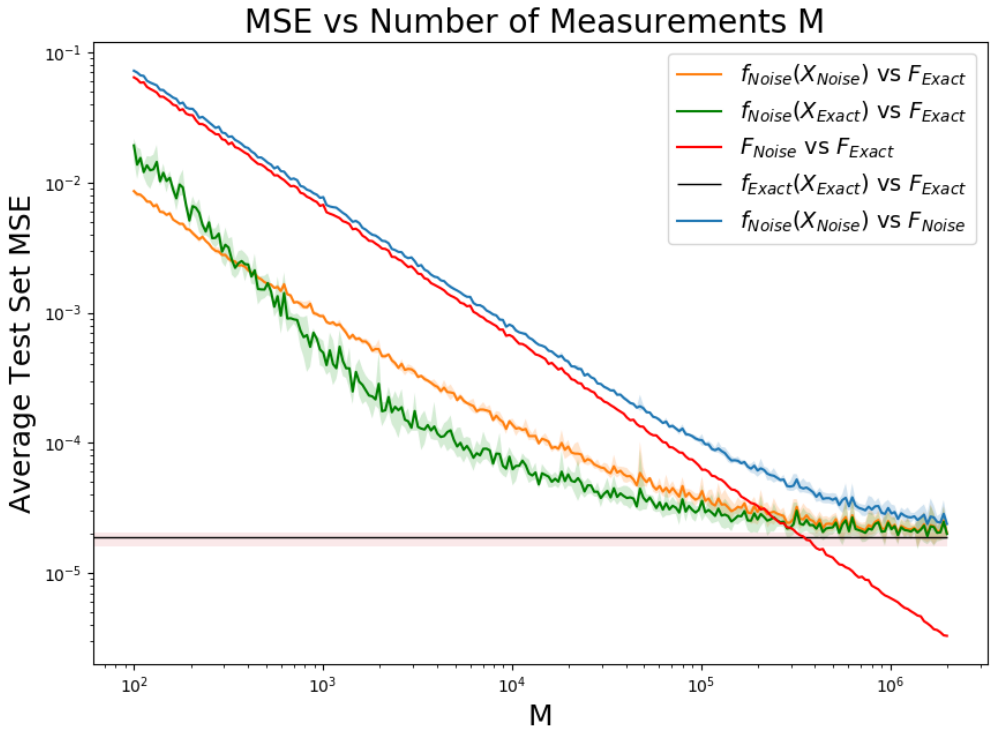

Figure 8: Test Set MSE plot for Data Set 2, for the case $\alpha = 0.0001$. MSE are averaged across the 5 Cross Validation folds and shaded regions denote minimum and maximum MSE values.

5.6.3 Data Set 3: $L = 4, N = 2, N_\uparrow = 1 = N_\downarrow$

The results for the quarter-filling case with $L = 4$ are shown in Figure 15 which displays the same overall behaviour as Data Set 1. For brevity Figure 15 is provided in the Appendix Section 8.1.

5.6.4 Data Set 4: $L = 4, N = 4, N_\uparrow = 2 = N_\downarrow$

The Data Set 4 results are displayed in Figure 9. The achieved MSE for the exact ML model is orders of magnitude lower than that of the other data sets. Also, when adding shot noise, the ratio of noise to the fluctuations within the data set is far larger, resulting in the behaviour seen in Figure 9. As with Data Set 2, the green line is not always below the orange line, due to the noise within the data. An error was made in choosing the same range for M as in preliminary experiments. In future work the range of M should be increased so that the convergence of the model measures can be fully observed. However, one takeaway from these results, is that if the data set in question has low fluctuations in Functional energies, and one is satisfied with a level of MSE, such as $\sim 10^{-6}$, then the ML model is effective at shot noise mitigation. Visually this can be interpreted as the green line being lower than the red line for a given MSE, such as $\sim 10^{-6}$.

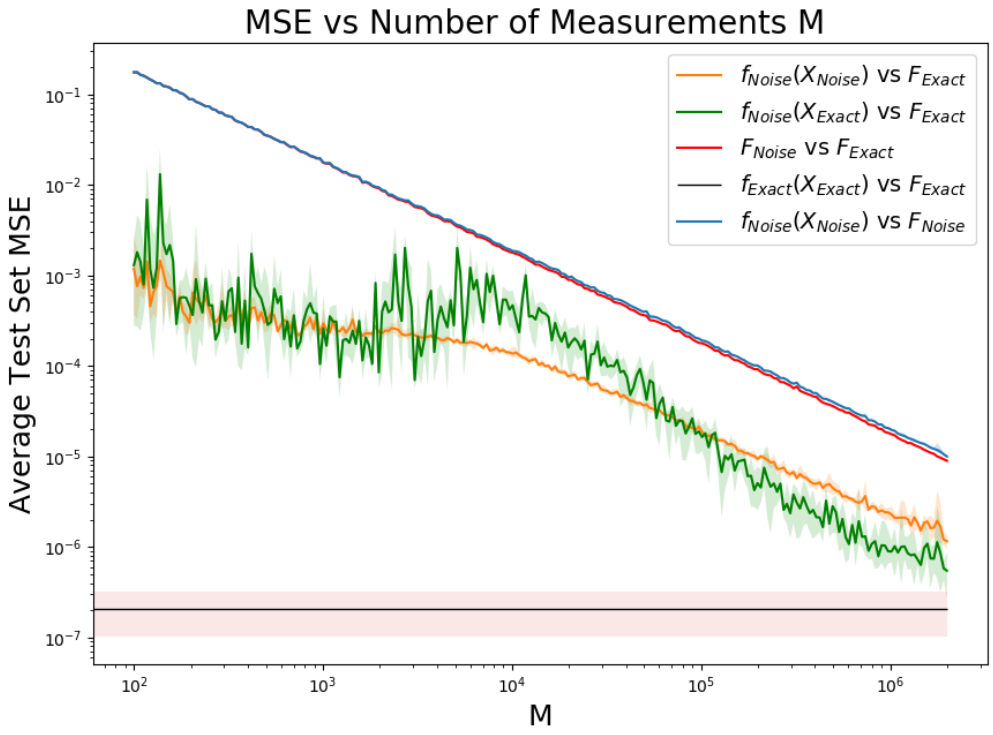

Figure 9: Test Set MSE plot for Data Set 4, the case $\alpha = 0.0001$. MSE are averaged across the 5 Cross Validation folds and shaded regions denote minimum and maximum MSE values.

5.7 Discussion

The effect of shot noise on the ML model was investigated for the data sets from Section 4. The emulated shot noise showed the correct error scaling given by Equation 55. In the previous section, the ML model trains on exact data, and learns the Functional from exact

data. In this section, the ML model sees the exact data with added shot noise and manages to mitigate some of the noise effects. As expected, the noisy ML model measures defined in Section 5.6, all converge asymptotically with M to the exact ML model. Overall, the potential for noise mitigation is a positive result, however its usefulness depends on the level of accuracy one wishes to achieve.

So far it has been assumed that the exact ground state can be prepared on a quantum computer, which is not NISQ applicable. Therefore in the following section the VQE method is implemented for generating the data sets.

6 VQE Method

A promising class of methods for finding the ground state of quantum many-body problems are variational methods. The variational quantum eigensolver (VQE) [22] is a hybrid quantum-classical approach to produce a ground state of a quantum Hamiltonian **H**. In this section the VQE method is simulated on a classical computer, where exact energy measurements $\langle\Psi|\mathbf{H}|\Psi\rangle$ are used as inputs to the classical optimiser. Throughout this section, measurement/shot noise is not considered and all the noise present in the data stems from imperfect VQE optimisation. In order to implement the quantum simulation of the fermionic system the qubits need to be mapped via the Jordan-Wigner Representation.

6.1 Anti-Commutation relations and Jordan-Wigner Representation

In the computational basis the operators in **H** acting on a qubit can be expressed as:

$$\hat{a}^\dagger = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \frac{X - iY}{2}, \quad \hat{a} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = \frac{X + iY}{2}, \quad \hat{n} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \frac{\mathbb{I} - Z}{2}. \tag{63}$$

For fermions, these operators acting on a Fock state need to satisfy the anti-commutation relations:

$$\{\hat{a_i}^\dagger, \hat{a_j}^\dagger\} = \{\hat{a}_i, \hat{a}_j\} = 0, \quad \{\hat{a}_i, \hat{a}_j^\dagger\} = \mathbb{I}\delta_{ij}. \tag{64}$$

Note in this subsection the indices i, j cover all orbitals, both spin-up and spin-down, such that $i \in [0, 1, \cdots, 2L-1, 2L]$. The required Fock Space anti-commutation relations don't hold, for example, $\hat{a_i}^\dagger \hat{a_j}^\dagger = \hat{a_j}^\dagger \hat{a_i}^\dagger$. The Jordan-Wigner Representation overcomes this by interspersing Z operators into the construction as follows:

$$\begin{aligned} \hat{a}_i^\dagger &\longrightarrow Z_{:i}\hat{a}_i^\dagger, \\ \hat{a}_i &\longrightarrow Z_{:i}\hat{a}_i \end{aligned} \tag{65}$$

Where $Z_{:i} = \prod_{j=0}^{i-1} Z_j$. The on-site nuclear terms, on-site Coulomb repulsion terms and hopping terms become ($j > i$):

$$\hat{a}_i^\dagger \hat{a}_i = \hat{n}_i \longrightarrow \left(\frac{\mathbb{I} - Z_i}{2}\right), \tag{66}$$

$$n_i n_j \longrightarrow \frac{1}{4}(\mathbb{I} - Z_i)(\mathbb{I} - Z_j), \tag{67}$$

$$\hat{a}_i^\dagger \hat{a}_j + \hat{a}_j^\dagger \hat{a}_i \longrightarrow \frac{1}{2}(X_i X_j + Y_i Y_j) Z_{i:j}. \tag{68}$$

Importantly when two qubits are adjacent in the qubit ordering then the corresponding hopping term has 2-qubit locality. The 1D Hubbard model therefore requires few Jordan Wigner strings.

6.2 Hamiltonian Variational Ansatz

The 1D Hubbard Hamiltonian can be separated into two parts. One being the non-interacting Hamiltonian $\mathbf{H}_0 = \mathbf{h}_h + \mathbf{h}_\nu$:

$$\begin{aligned} \mathbf{h}_\nu &= \sum_{i,\sigma}^{L} \nu_i \hat{a}^\dagger_{i,\sigma} \hat{a}_{i,\sigma}, \\ \mathbf{h}_h &= -t \sum_{\langle i,j\rangle,\sigma}^{L} \hat{a}^\dagger_{i,\sigma} \hat{a}_{j,\sigma}. \end{aligned} \tag{69}$$

Here $\mathbf{h}_h$ and $\mathbf{h}_\nu$ represent the hopping and on-site terms respectively. The full Hamiltonian $\mathbf{H}$ is then the sum of $\mathbf{H}_0 + \mathbf{h}_u$, where $\mathbf{h}_u$ is the interaction:

$$\mathbf{h}_u = U \sum_{i}^{L} \hat{n}_{i,\downarrow} \hat{n}_{i,\uparrow}. \tag{70}$$

Let $|\Psi_0\rangle$ be the ground state of the non-interacting Hamiltonian and $|\Psi_T\rangle$ be a trial state of the full Hamiltonian. If the state can $|\Psi_0\rangle$ be prepared on a quantum circuit, then assuming no gap closes, it is possible to adiabatically evolve from $|\Psi_0\rangle$ to the ground state of the full interacting Hamiltonian [8]. This can be achieved by breaking the annealing into short time steps dt and evolving for a total time T , this annealing is a sequence of (T/dt) different unitary rotations by Hamiltonians interpolating between $\mathbf{H}_0$ and $\mathbf{H}$ [8]. This could be implemented on a quantum computer using a Trotter-Suzuki method which further decomposes this sequence into a sequence of unitary rotations by individual terms in the Hamiltonian.

In this work the Hamiltonian Variational Ansatz of Wecker, Hastings and Troyer [8] was used. This method considers arbitrary angles for the rotations in the sequence, rather than choosing them from a Trotterization of an annealing process, allowing for a shorter sequence (resulting in a shorter circuit depth). Once the state $|\Psi_0\rangle$ is prepared, a quantum circuit can yield the trial state:

$$|\Psi_T\rangle = \prod_{b=1}^{S} \left(U_u(\frac{\theta_u^b}{2}) U_h(\theta_h^b) U_\nu(\theta_\nu^b) U_u(\frac{\theta_u^b}{2}) \right) |\Psi_0\rangle. \tag{71}$$

This method involves a repeating pattern and performs S repetitions or steps. In each of these steps there are 3 variational parameters $\theta_h^b, \theta_u^b, \theta_\nu^b$, where $b = 1, \ldots, S$. The Unitaries

$U_\alpha(\theta)$ implement the exponential $e^{i\theta \mathbf{h}_\alpha}$ exactly, where $\alpha \in [u, h, \nu]$.

6.3 Hamiltonian Variational Ansatz Implementation

The parameterised circuits were implemented using *Cirq* [10] and *Openfermion* and the VQE implementation was based on related works [3, 9, 32]. The qubit layout was chosen to be a line of qubits, with each qubit representing a spin orbital. The ordering was kept as in Equation 37, resulting in low Jordan-Wigner string counts, where all spin-up qubits/orbitals appear before all spin-down. Next the initial state preparation procedures are detailed.

6.3.1 Initial State Preparation

The ground state of the non-interacting Hamiltonian ($U = 0$) can be efficiently prepared [15] and act as the initial state for the VQE. The Hamiltonian $\mathbf{H}_0$ is quadratic in fermionic creation and annihilation operators and is also particle number conserving. With *Openfermion* the circuit which prepares the initial state can be efficiently produced. This method is described in detail in [14], where the algorithm prepares a Slater determinant using Givens rotations (Equation 72). The initial prepared state $|\Psi_0\rangle$ can then be acted on as in Equation 71.

The overlap between the initial prepared state $|\Psi_0\rangle$ and the desired ground state of $\mathbf{H}$ is important for the VQE's succcess. In the scenario with $U = 10$, i.e. strong interaction between the fermions, the overlap may not be sufficiently large. Therefore an alternative initial state preparation method was also experimented with. Here parameterised Givens Rotations were implemented, which are particle-conserving Unitaries. These operations are rotations in the plane spanned by two coordinates axes. In quantum computational chemistry the Givens rotation is typically defined as:

$$G(\theta) = e^{\frac{-i\theta(YX-XY)}{2}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{72}$$

acting on the subspace spanned by $|10\rangle$ and $|01\rangle$. For example in the case of 4 sites at half-filling, the parameterised Givens rotations acted as displayed in Figure 10. This alternative initial state preparation can help with providing a reasonable overlap, at the cost of introducing additional parameters for the classical optimiser. The two initial state preparations were experimented with and the best performing method kept for data generation. Next the

ansatz gates used for the hopping, Coulomb and on-site terms are detailed.

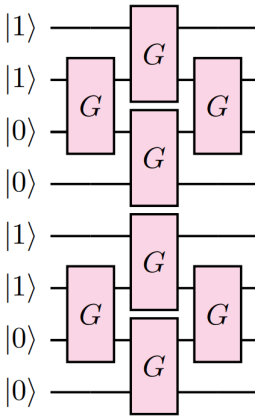

Figure 10: Givens rotations layout as in [32], for the case $L = 4, N = 4, N_\uparrow = 2 = N_\downarrow$. Note that each gate is parameterised, however the upper (spin-up) gates share the same parameters as the corresponding lower (spin-down) gates. Image taken from [32].

6.3.2 Ansatz: Unitary Gates

The Number Preserving (NP) unitary [3] is a general number-preserving, 2 qubit gate of the following form,

$$U_{NP}(\theta_h, \theta_u) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_h & i\sin\theta_h & 0 \\ 0 & i\sin\theta_h & \cos\theta_h & 0 \\ 0 & 0 & 0 & e^{i\theta_u} \end{pmatrix}. \tag{73}$$

This parameterised unitary accounts for all kinetic and Coulomb terms in $\mathbf{H}$. The Coulomb terms are accounted for by the $e^{i\theta_u}$ term which acts on the $|11\rangle$ state. The unitary is thus applied between Coloumb interacting qubit pairs, with $(\theta_h = 0, \theta_u = \frac{U}{2}\theta_u)$. Similarly the hopping terms are accounted for via $(\theta_h = -t\theta_h, \theta_u = 0)$, which correspond to $e^{\frac{-it\theta_h(XX+YY)}{2}}$. The external potential terms are accounted for by single phased qubit gates,

$$T(\theta_\nu) = e^{i\theta_\nu} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta_\nu} \end{pmatrix}. \tag{74}$$

Note each qubit corresponding to a fermionic mode at site j is acted on with $\theta_\nu = \nu_j\theta_\nu$, where ν_j is the random external potential at site j. The same parameter $\theta_\nu\nu_j$ is used for both the spin-up and spin-down modes at site j. The ansatz is implemented as given by Equation 71, where different layers can have different parameters. Note the the Hamiltonian

Variational ansatz is a special case of the NP ansatz [3]. In order to preserve spin and remain in the correct number-preserving subspace many parameters must be fixed to be identical or set to 0. In total, the number of variational parameters per ansatz layer is 3. With these unitary gates, the VQE method was implemented using *Cirq* and *Openfermion* to generate data sets for the ML method.

6.4 VQE Method: Data Generation

As in Section 4 and 5, energy acceptance conditions were set for including data points into data sets. In general this is not a realistic possibility for larger system sizes, due to the infeasible cost of ED. However for comparing the ML model performance, between VQE and ED data sets, the same acceptance conditions were kept, whereby exact ground state energies were found via ED. If the acceptance conditions were satisfied, the VQE method was run for the data point in question.
The Cobyla and L-BFGS optimisers were tried using *Scipy* [34] and it was found that Cobyla performing the best and was used throughout. The reason for Cobyla performing better than L-BFGS with the chosen ansatz is to be investigated in future work, it could be due to difficult optimisation landscapes for the inhomogeneous Hubbard cases being studied. This choice of non-gradient based optimiser acted on exact energy measurements $\langle\Psi(\theta)|\mathbf{H}|\Psi(\theta)\rangle$. After the VQE optimisation procedure is complete, the quantum circuit yields a state $|\Psi(\theta)\rangle$ given by Equation 71, corresponding to the lowest energy configuration found. Note that this state is not guaranteed to be the desired ground state and may have low fidelity with it.

In order to generate the data set, for each data point, the state $|\Psi(\theta)\rangle$ is directly accessed and electron densities $\{n_{i\sigma}^{VQE}\}$ are found as in Section 4. The corresponding Functional Energy or target for ML is then calculated as:

$$F[n_{i\sigma}^{VQE}] = E_{VQE} - \sum_{i,\sigma} n_{i\sigma}^{VQE}\nu_i, \tag{75}$$

where $E_{VQE} = \langle\Psi(\theta)|\mathbf{H}|\Psi(\theta)\rangle$ for the Hamiltonian $\mathbf{H}$ resulting from external potential ν. As in Section 5, when the VQE data is generated, the corresponding ED data is also saved, resulting in an exact data set and its corresponding noisy VQE data set.

6.5 VQE Results

Overall the ML method is similar to Section 5 and the same measures from Section 5.6 are chosen:

1. $f_{VQE}(X_{VQE})$ vs $F_{VQE} \longrightarrow$ Pure VQE measure.

2. $f_{VQE}(X_{VQE})$ vs $F_{Exact} \longrightarrow$ VQE model against exact Functionals.

3. $f_{VQE}(X_{exact})$ vs $F_{Exact} \longrightarrow$ Exact inputs into VQE model.

4. $f_{Exact}(X_{exact})$ vs $F_{Exact} \longrightarrow$ Exact model for comparisons.

With these measures the performance of ML model can be evaluated and it can be determined whether noise mitigation is achieved as in Section 5. Next the varying success of the VQE optimisation for different cases with $L = 8$ and $L = 4$ is discussed.

6.5.1 $L = 8$

For the case with 8 sites or 16 qubits, the VQE optimisation performed poorly. Extensive experimentation with the VQE configuration was performed, including different initial state preparations, varying ansatz depths, freeing up extra variational parameters and different parameter initialisations. See Appendix Section 8 for further details on VQE attempts. Overall for the quarter-filling case, $L = 8, N = 2, N_\uparrow = 1 = N_\downarrow$, the resulting states produced were not accurate enough for ML purposes. Unfortunately this ruled out the ability to obtain comparative results for Data Set 1 and Data Set 2 in Sections 4 and 5. As an illustrative example of the large differences between ground state energies and VQE energies, see Figure 11. The red line denotes the energy w.r.t to the initial state and the green line denotes the desired ground state energy. The VQE method does not yield energies close to the desired ground state. Whilst this plot is only for a given data point with random external potential ν, this behaviour was observed for almost all data points. Next VQE results for the smaller system sizes with $L = 4$ are presented.

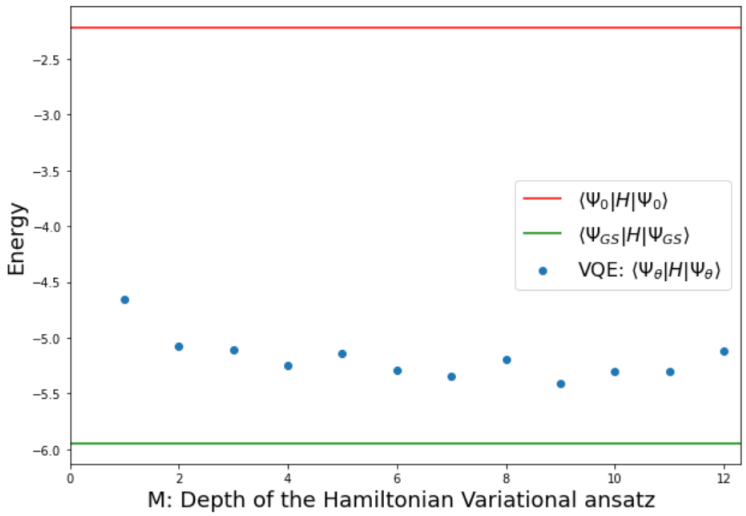

Figure 11: Example plot of VQE performance with varying ansatz depth. The initial state preparation method was that of the non-interacting Hamiltonian as detailed in Section 6.3, similar results are observed for the alternative Givens rotation method.

6.5.2 $L = 4$

Due to a smaller system size, the VQE method achieved better optimisation and data sets could be produced for the ML model. Following experimentation with the VQE configuration, two data sets were produced, each corresponding with Data Set 3 and Data Set 4 from Sections 4 and 5. The depth of the circuit was chosen to be 4, as the performance plateaued for larger depths. For the quarter-filling case, with $L = 4, N = 2, N_\uparrow = 1 = N_\downarrow$, the VQE yielded many states with high overlap with desired ground states. This can be seen in the left of Figure 12, where the fidelities $|\langle\Psi(\theta)|\Psi^{GS}\rangle|^2$ are plotted for each data point. However there exist many points with low overlap, which negatively affect the ML models ability to learn the desired Functional. This issue is more pronounced for the half-filling case with $L = 4, N = 4, N_\uparrow = 2 = N_\downarrow$ as seen in the right of Figure 12. The variability in the VQE's success is substantial and this may be related to the physics of the Hubbard model at half-filling, where it is a Mott insulator. The accumulation of points with fidelities close to 0 and 0.5 suggest that these points have optimisation landscapes that contain difficult regions such as barren plateaus. In spite of this, the ML model was still applied to both VQE data sets for completeness.

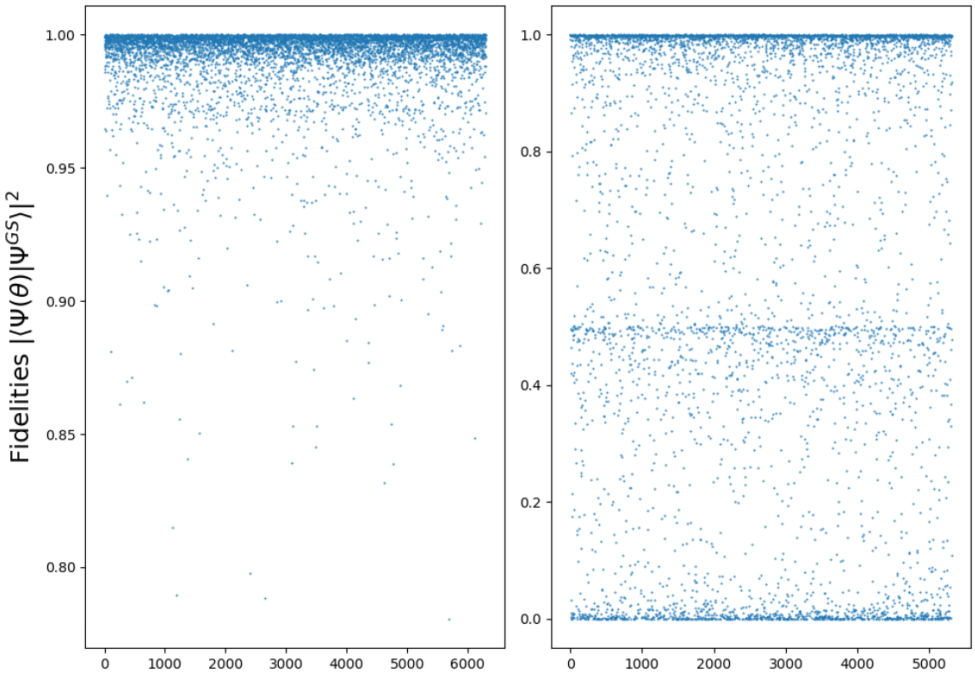

Figure 12: Fidelities $|\langle\Psi(\theta)|\Psi^{GS}\rangle|^2$ of data points generated by the VQE method for the quarter-filling case, $L = 4, N = 2, N_\uparrow = 1 = N_\downarrow$ on the left and the half-filling case $L = 4, N = 4, N_\uparrow = 4 = N_\downarrow$ on the right.

6.5.3 VQE Results: ML Model

Given the VQE data sets for $L = 4$, the ML Model was implemented as in Sections 4 and 5. Previously there were 250 data sets to compare against the exact results, for varying levels of shot noise. Now there is a single ML model trained on the VQE data set, along with an exact model for comparison. Therefore to illustrate the ML models performance, the MSE over test sets for each cross validation split are plotted in Figures 13 and 16. As in previous sections, an Adam learning rate of $\alpha = 10^{-4}$ performed best. From the figures it can be seen that the ML model fails to learn the Functional when trained on VQE generated data. Additionally the noise mitigation seen in previous sections is not present, whereby the blue line is now lower than the orange and green lines.

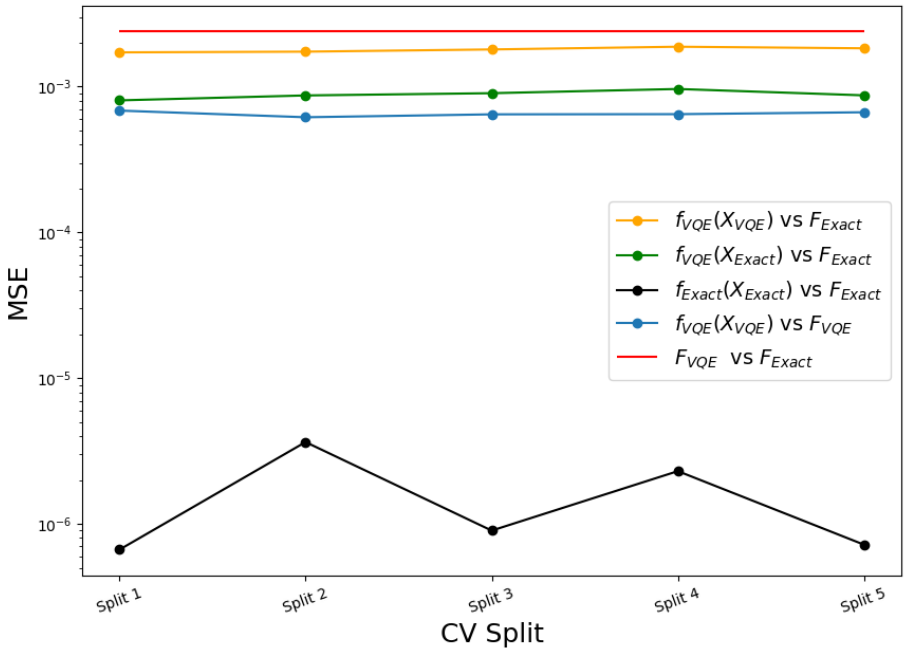

Figure 13: Test Set MSE plot for the VQE results in the quarter-filling case, $L = 4, N = 2, N_\uparrow = 1 = N_\downarrow$, with $(\alpha = 10^{-4})$.

6.6 VQE: Discussion

The ML model attempts to learn the DFT Functional, which maps

$$f_{\text{DFT}} : \{n_{i\sigma}^{GS}\} \mapsto F[\{n_{i\sigma}\}] = E_{GS} - \sum_i \nu_i n_i^{GS} = \langle \psi_{GS} | (\hat{\mathbf{T}} + \hat{\mathbf{U}}) | \psi_{GS} \rangle. \tag{76}$$

Unfortunately, the VQE method yielded numerous states that have low fidelity with the desired ground state. This had a negative impact on the accuracy of the electron densi-

ties $\{n_{i\sigma}^{VQE}\}$, which ultimately acted as misleading inputs into the ML model. In order to investigate this, the differences in electron densities, $X_{Exact} - X_{VQE}$ are compared to $X_{Exact} - X_{Noise}$ from Section 5. The X_{Noise} data set for comparison is taken as the one with the closest F_{Noise} vs F_{Exact} MSE to that of the VQE data set. E.g, for quarter-filling this corresponds to Data Set 3 with $M = 2053$ measurement shots. In other words, we investigate the error in electron density (inputs), for data sets with the same MSE in Functional energies (outputs). This shines light on the fact that even if the VQE yields a state with energy close to that of the exact ground state, the electron density can be far from the actual ground state density. In contrast, in Section 5, the exact ground state is found and gaussian/shot noise added, resulting in two very different forms of noise/error. In Figure 14 this is clearly visualised. These large differences (outliers) in the resulting VQE densities prevent the ML Model from effectively learning the desired Functional. However, if the VQE method is improved and more accurate ground states produced, the ML method performance will increase, which is the focus of future work.

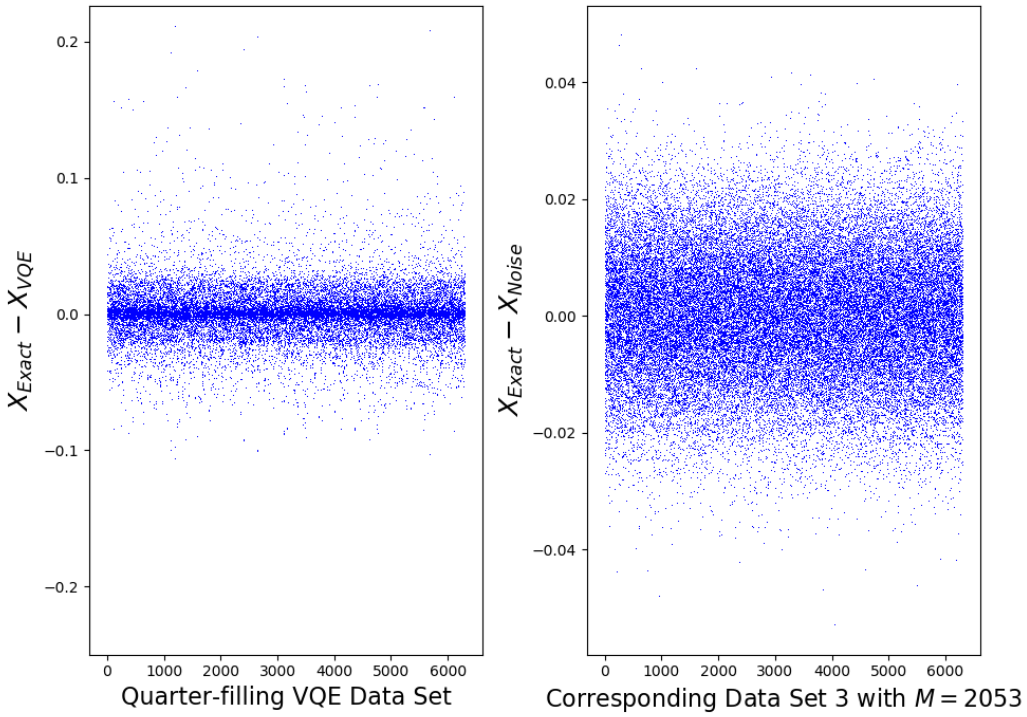

Figure 14: Differences in electron densities for the quarter-filling case, $L = 4, N = 2, N_\uparrow = 1 = N_\downarrow$. Both data sets have MSE of $\sim 2 \times 10^{-3}$. The VQE data is given on the left and the shot noise data on the right. Each marker represents a density $\Delta n_{i\sigma}$ of which there are $2L = 8$ (shown vertically) for each data point.

7 Conclusion and Future Work

In this thesis a potential hybrid scheme was investigated, which combined the techniques and problems from Physics, Machine Learning and Quantum Computing. The focus of the thesis was the Hubbard model, which can be solved exactly via ED for finite system sizes. For larger system sizes this is not generally the case and thus the Hubbard model is a prominent target for quantum computers, where future NISQ devices are aiming to solve larger Hubbard model instances than classically feasible. DFT is currently one of the most widely used approaches for quantum chemistry, and in this thesis the existing work of Sanvito [24] was extended to the field of Quantum Computing. The existing work shows that the exact form of the DFT functional for the 1D Hubbard model can be learnt using ML. Here, this work investigated using NISQ-like data for training the ML model.

Throughout, 4 different configurations of the Hubbard model were considered, including system sizes with $L = 4$ and $L = 8$. Two $L = 8$ configurations were based on [24] and for $L = 4$, half-filling and quarter-filling cases were considered. Firstly, the ML model was reproduced for the ED data and the desired DFT Functionals were accurately learnt. Next, shot noise was introduced to account for measurement noise on a quantum computer. The ML model was applied to data sets with varying levels of shot noise and learned the desired Functional, as well as achieving a degree of noise mitigation. This noise mitigation is a result of the ML model's ability to learn the functional more so than the measurement noise. Lastly the VQE method was investigated and overall the VQE method was not very successful. For $L = 8$, the optimisation procedure was difficult and secondly the ML model did not train well on the simpler ($L = 4$) VQE data. The ansatz of [3] was implemented, however the important generality of the NP Unitary gate could not fully utilised. This is because the original ML model paper [24], required spin preservation and the need to remain in the correct subspace. In future work the VQE method will be further experimented with for the $L = 8$ case. Additionally the Coulomb term $U = 10$ should be decreased, as in hindsight this value should have been set lower. A lower value of U will result in easier to prepare ground states, as there is a larger overlap between the initial non-interacting state and the full Hamiltonian. This also alleviates the need for additional Givens rotation parameters for the VQE.

Overall, the important distinction between forms of noise in NISQ era data has been highlighted in this project. If a quantum computer prepares ground states perfectly and only shot noise exists, then this application or proposed scheme is sound. Realistically for NISQ devices, the perfect ground state will not be produced. The VQE noise present in this work is inherently a mismatched form of noise to this model. However, with different

ansätze, a different ML model and training, the results could improve. Hence, it is not clear yet how the method will perform with better VQE data and this is the focus of future work.

Acknowledgements

I would like to express my gratitude to the Applied Quantum Algorithm group for making this research possible, being extremely welcoming and even providing me with a workspace. I am indebted to my project supervisors Dr. Jordi Tura, Stefano Polla and Emiel Koridon for their feedback and guidance during the thesis. Throughout, help has always and continues to be available. I am also grateful for the opportunity to partake in AQA seminars, events and journal clubs, which have been a great experience. Lastly I would like to thank my supervisors for proof-reading the thesis and providing corrections, along with useful feedback.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Edgar Anderson. The species problem in iris. *Annals of the Missouri Botanical Garden*, 23(3):457–509, 1936.

[3] Chris Cade, Lana Mineh, Ashley Montanaro, and Stasja Stanisic. Strategies for solving the fermi-hubbard model on near-term quantum computers. *Phys. Rev. B*, 102:235122, Dec 2020.

[4] K. Capelle, C. A. Ullrich, and G. Vignale. Degenerate ground states and nonunique potentials: Breakdown and restoration of density functionals. *Physical Review A*, 76(1), jul 2007.

[5] Francois Chollet et al. Keras, 2015.

[6] Taco Cohen and Max Welling. Group equivariant convolutional networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR.

[7] Elbio Dagotto. Correlated electrons in high-temperature superconductors. *Rev. Mod. Phys.*, 66:763–840, Jul 1994.

[8] Matthew B. Hastings Dave Wecker and Matthias Troyer3. Towards Practical Quantum Variational Algorithms. *arXiv e-prints:1507.08969*, 2015.

[9] Nathan Wiebe Bryan K. Clark Chetan Nayak Dave Wecker, Matthew B. Hastings and Matthias Troyer. Solving strongly correlated electron models on a quantum computer. *arXiv e-prints:1506.05135*, 2015.

[10] Cirq Developers. Cirq, August 2021. See full list of authors on Github: https://github.com/quantumlib/Cirq/graphs/contributors.

[11] Sevag Gharibian, Yichen Huang, Zeph Landau, and Seung Woo Shin. Quantum hamiltonian complexity. *Foundations and Trends® in Theoretical Computer Science*, 10(3):159–282, 2015.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[13] Robin Haunschild, Andreas Barth, and Bernie French. A comprehensive analysis of the history of dft based on the bibliometric method rpys. *Journal of Cheminformatics*, 11(1):72, 2019.

[14] Zhang Jiang, Kevin J. Sung, Kostyantyn Kechedzhi, Vadim N. Smelyanskiy, and Sergio Boixo. Quantum algorithms to simulate many-body physics of correlated fermions. *Physical Review Applied*, 9(4), apr 2018.

[15] Zhang Jiang, Kevin J. Sung, Kostyantyn Kechedzhi, Vadim N. Smelyanskiy, and Sergio Boixo. Quantum algorithms to simulate many-body physics of correlated fermions. *Physical Review Applied*, 9(4), apr 2018.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[17] W. Kohn. Nobel lecture: Electronic structure of matter—wave functions and density functionals. *Rev. Mod. Phys.*, 71:1253–1266, Oct 1999.

[18] E. Lieb and F. Wu. Physical Review Letter 20, 1445 (1968).

[19] Elliott H. Lieb and F. Y. Wu. Absence of mott transition in an exact solution of the short-range, one-band model in one dimension. *Phys. Rev. Lett.*, 20:1445–1448, Jun 1968.

[20] G. D. Mahan. *Many Particle Physics, Third Edition*. Plenum, New York, 2000.

[21] G.D. Mahan. *Many-Particle Physics*. Physics of Solids and Liquids. Springer US, 2012.

[22] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, feb 2016.

[23] Jarrod R. McClean, Kevin J. Sung, Ian D. Kivlichan, Yudong Cao, Chengyu Dai, E. Schuyler Fried, Craig Gidney, Brendan Gimby, Pranav Gokhale, Thomas Häner, Tarini Hardikar, Vojtěch Havlíček, Oscar Higgott, Cupjin Huang, Josh Izaac, Zhang Jiang, Xinle Liu, Sam McArdle, Matthew Neeley, Thomas O'Brien, Bryan O'Gorman, Isil Ozfidan, Maxwell D. Radin, Jhonathan Romero, Nicholas Rubin, Nicolas P. D. Sawaya, Kanav Setia, Sukin Sim, Damian S. Steiger, Mark Steudtner, Qiming Sun, Wei

Sun, Daochen Wang, Fang Zhang, and Ryan Babbush. Openfermion: The electronic structure package for quantum computers, 2017.

[24] James Nelson, Rajarshi Tiwari, and Stefano Sanvito. Machine learning density functional theory for the hubbard model. *Phys. Rev. B*, 99:075132, Feb 2019.

[25] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[26] Robert G. Parr and Yang Weitao. *Density-Functional Theory of Atoms and Molecules*. Oxford University Press, jan 1995.

[27] Eric Prehn, Stefano Polla, and Emiel Koridon. Investigating deep learning of dft functionals in the nisq era. https://github.com/aQaLeiden/DFTQML, 2022.

[28] David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise, 2017.

[29] K. Schönhammer, O. Gunnarsson, and R. M. Noack. Density-functional theory on a lattice: Comparison with exact numerical results for a model with strongly correlated electrons. *Phys. Rev. B*, 52:2504–2510, Jul 1995.

[30] Medha Sharma and M.A.H. Ahsan. Organization of the hilbert space for exact diagonalization of hubbard model. *Computer Physics Communications*, 193:19–29, 2015.

[31] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey, 2020.

[32] Stasja Stanisic, Jan Lukas Bosse, Filippo Maria Gambetta, Raul A. Santos, Wojciech Mruczkiewicz, Thomas E. O'Brien, Eric Ostby, and Ashley Montanaro. Observing ground-state properties of the fermi-hubbard model using a scalable algorithm on a quantum computer, 2021.

[33] Adrian P Sutton. *Electronic structure of materials*. Clarendon Press, 1993.

[34] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[35] S. Yamada, T. Imamura, and M. Machida. 16.447 tflops and 159-billion-dimensional exact-diagonalization for trapped fermion-hubbard model on the earth simulator. In *SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, pages 44–44, 2005.

8 Appendix

8.1 Additional Figures

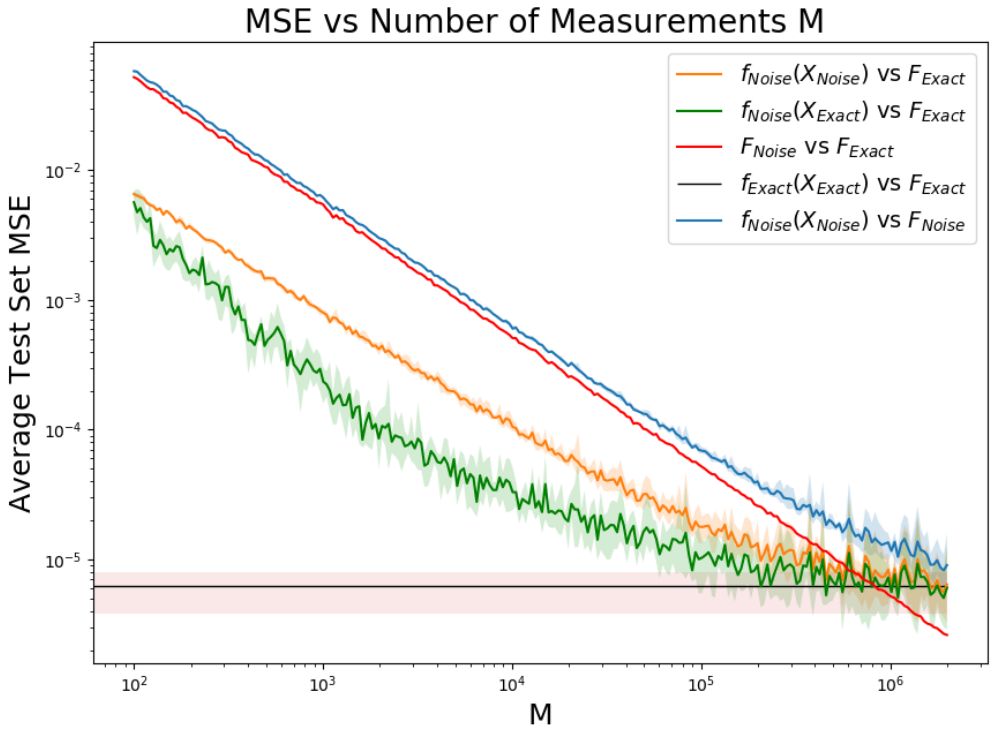

Figure 15: Test Set MSE plot Data Set 3, for the case $\alpha = 0.0001$. MSE are averaged across the 5 Cross Validation folds and shaded regions denote minimum and maximum MSE values.

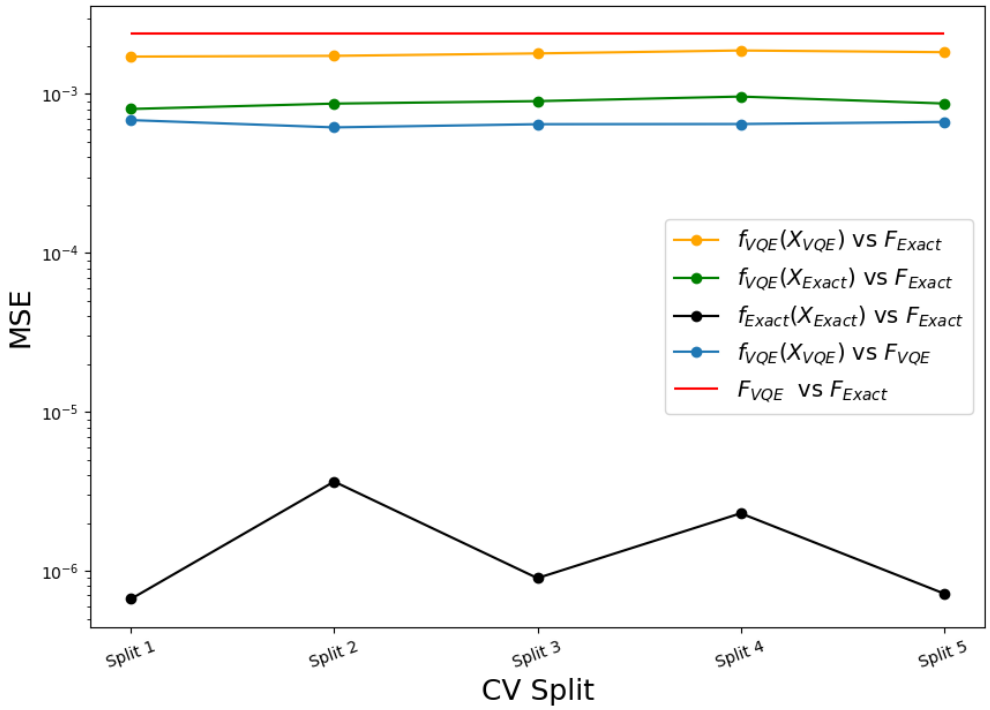

Figure 16: Test Set MSE plot for the VQE results in the half-filling case, $L = 4, N = 4, N_\uparrow = 2 = N_\downarrow$, with $(\alpha = 10^{-4})$.

8.2 VQE Attempts

The following various methods were experimented with when implementing the VQE in Section 6.

8.2.1 Different Initial State Preparations

The two methods are discussed in Section 6.3. For each VQE attempt, first the the ground state of the non-interacting Hamiltonian ($U = 0$) was prepared using existing *openfermion* methods. Depending on the overlap between this initial state and the desired final state, the Givens Rotation method was used as an alternative. Note that in reality on a quantum computer, knowing the exact overlap may not be possible. Hence, (especially for high values of U) the Givens Rotation initialisation is likely required.

8.2.2 Varying Ansatz Depths

Each VQE was run for varying ansatz depths, in order to find an appropriate depth. A sample of Hamiltonians (each resulting from different external potentials) were tested with. Between 20-40 samples were considered for each VQE, in order to select the depth. Note

that NISQ devices will preform far better with low depth circuits and therefore a conscious effort was made to choose lower depths (as long as performance was the same as higher depth circuits). Once a depth was chosen, the depth remained fixed throughout the entire data set generation.

8.2.3 Extra Variational Parameters

The VQE ansatz of depth M has the variational parameters (ignoring Givens Rotation parameters $\vec{\theta}_G$),

$$\theta = \{\theta_U^1, \cdots, \theta_U^{M-1}, \theta_U^M, \theta_h^1, \cdots, \theta_h^{M-1}, \theta_h^M, \theta_\nu^1, \cdots, \theta_\nu^{M-1}, \theta_\nu^M\} = \{\vec{\theta}_U, \vec{\theta}_h, \vec{\theta}_\nu\} \tag{77}$$

Where $\vec{\theta}_h, \vec{\theta}_U, \vec{\theta}_\nu$ account for hopping, Coulomb and on-site terms, respectively. In this set up the total number of variational parameters is $3M$. An ansatz method following a coarse and then fine-tuning approach was tested. Here, first the normal VQE was run using parameters as in Equation 77. Once it converges on a final set of parameters $\theta^\star$, a new VQE is run, with initial parameters $\vec{\theta}_h^\star$ and $\vec{\theta}_U^\star$, along with extra parameters for on-site terms. The number of $\vec{\theta}_\nu$ parameters is increased from M to LM, where now each site j has its own variational parameter within every layer. The initial values of $\theta_{\nu j}^m$ (m'th layer and j'th site) are set to θ_ν^m. This fine-tuning VQE was run following the original VQE and the performance remained similar. Note that the total number of parameters is increased from $3M$ to $2M + ML$, which suggests that on this may not be feasible for larger system sizes and circuit depths, due to NISQ circuit noise and optimisation difficulties. Nonetheless, future work will continue experimentation with freeing up additional parameters.

8.2.4 Parameter Initialisation

For all ansätze, the initialisation of parameters θ has to be chosen. Random initialisation within the range $[0, 0.5]$ were tested and performance was found to be variable and therefore this method was not chosen for data set generation. An attempt was made to mimic adiabatic evolution, whereby the initial parameters gradually increased in each layer of the ansatz. However, the final parameters, after optimisation, were far off from the adiabatic inspired initial parameters. A trend was seen, where the parameters in the first (few) layers of the ansatz obtained larger values, and ensuing layers had final parameters small in magnitude. In other words, the last few layers only slightly adjusted the larger evolution of the preceding layers. Following experimentation, all parameters were set to 10^{-5} for data set generation, representing a small initial evolution.