# Machine Learning for Unconventional Superconductivity

Pinja Hirvinen

**School of Science**

Bachelor's thesis
Espoo 23.5.2024

**Supervisor**

Assoc. Prof. Fabricio Oliveira

**Advisors**

Asst. Prof. Jose Lado

D.Sc. (Tech.) Rouven Koch

Aalto University
School of Science

**Aalto University**
**School of Science**

**Aalto University, P.O. BOX 11000, 00076 AALTO**
**www.aalto.fi**
**Abstract of the bachelor's thesis**

| | |
|---|---|
| **Author** Pinja Hirvinen | |
| **Title** Machine Learning for Unconventional Superconductivity | |
| **Degree programme** Bachelor's Programme in Science and Technology | |
| **Major** Mathematics and Systems Sciences | **Code of major** SCI3029 |
| **Teacher in charge** Assoc. Prof. Fabricio Oliveira | |
| **Advisors** Asst. Prof. Jose Lado, D.Sc. (Tech.) Rouven Koch | |
| **Date** 23.5.2024 | **Number of pages** 23+9 | **Language** English |

**Abstract**
Superconductivity is a quantum mechanical phenomenon where the resistance of a material disappears and it starts to reject magnetic fields below a certain critical temperature. In conventional superconductors, instead of repulsion, two electrons interact attractively via positive ions to form so-called Cooper pairs, which require energy to be decoupled to restore resistance. In unconventional, topological superconductivity, exotic quasiparticles, so-called Majorana fermions, are found at the edges or defects of the superconductor. These quasiparticles are highly stable due to their topological nature, making topological superconductors potential candidates for fault-tolerant quantum computing.

A superconducting material can be identified using a scanning tunneling microscope, which can take atomic-level images of a material and measure differential conductivity, which in turn gives the density of states of the material, i.e. the number of available quantum states at a given energy. Mathematically superconductors are modelled by Hamiltonian functions that describe the total energy of the system. Since superconductors with different properties have both a unique density of states and a Hamiltonian function describing them, the density of states can be used to determine the parameters of the corresponding Hamiltonian using machine learning.

The goal of this thesis is to train a supervised neural network model to identify Hamiltonian parameters of unconventional, potentially topological superconductors using simulated density of states values. For this purpose, two superconducting systems of different sizes are simulated, and two neural network models are trained for each of these systems: one that uses as input the density of states of the whole system and another that uses the components obtained using principal component analysis that describe the main features of the density of states.

In this thesis, training a neural network using principal components was found to be significantly faster, and to also give better predictions of Hamiltonian parameters. A larger amount of training data was also found to improve the accuracy of the results. The best performing neural network model was trained using data on the larger system's principal components, and its prediction errors were one order of magnitude smaller than the other models.

**Keywords** superconductivity, unconventional superconductivity, topological superconductivity, machine learning, neural networks, principal component analysis

**Aalto University**
**School of Science**

**Aalto-yliopisto, PL 11000, 00076 AALTO**
**www.aalto.fi**
**Tekniikan kandidaatintyön tiivistelmä**

| | | | |
|---|---|---|---|
| **Tekijä** Pinja Hirvinen | | | |
| **Työn nimi** Epätavanomaisen suprajohtavuuden tunnistaminen koneoppimisen avulla | | | |
| **Koulutusohjelma** Teknistieteellinen kandidaattiohjelma | | | |
| **Pääaine** Matematiikka ja systeemitieteet | | **Pääaineen koodi** SCI3029 | |
| **Vastuuopettaja** Apulaisprofessori Fabricio Oliveira | | | |
| **Työn ohjaajat** Apulaisprofessori Jose Lado, TkT Rouven Koch | | | |
| **Päivämäärä** 23.5.2024 | **Sivumäärä** 23+9 | | **Kieli** Englanti |

**Tiivistelmä**

Suprajohtavuus on kvanttimekaaninen ilmiö, jossa aineen resistanssi katoaa ja aine alkaa hylkimään magneettikenttiä tietyn kriittisen lämpötilan alapuolella. Tavanomaisissa suprajohteissa kahden elektronin välille syntyy hylkivän sijaan vetovoimainen vuorovaikutus positiivisten ionien välityksellä, ja ne muodostavat ns. Cooperin pareja, joiden hajoamiseen tarvitaan energiaa resistanssin palauttamiseksi. Epätavanomaisessa, topologisessa suprajohtavuudessa suprajohteen reunoilla tai vikakohdissa tavataan lisäksi eksoottisia kvasihiukkasia, ns. Majorana-fermioneja, jotka ovat topologisen luonteensa ansiosta erittäin stabiileja, ja tekevät tällaisista suprajohteista soveltuvia esimerkiksi vikasietoiseen kvanttilaskentaan.

Suprajohtava materiaali voidaan tunnistaa tunnelointimikroskoopilla, jolla voidaan ottaa atomitason kuvia tutkittavasta materiaalista, ja mitata differentiaalijohtavuutta, josta puolestaan saadaan selville materiaalin tilatiheys, eli käytettävissä olevien kvanttitilojen määrä tietyssä energiassa. Matemaattisesti suprajohteita mallinnetaan Hamiltonian-funktioilla, jotka kuvaavat systeemin kokonaisenergiaa. Koska eri ominaisuudet omaavilla suprajohteilla on niitä kuvaava tilatiheys sekä Hamiltonian-funktio, voi tilatiheyden arvoista määrittää vastaavan Hamiltonian-funktion parametreja erilaisten koneoppimismenetelmien avulla.

Tämän työn tavoitteena on kouluttaa ohjattu neuroverkkomalli tunnistamaan epätavanomaisten, potentiaalisesti topologisten suprajohteiden Hamiltonian-parametreja simuloitujen tilatiheysarvojen avulla. Tätä varten on simuloitu kahta eri kokoista suprajohtavaa systeemiä, joille kummallekin on koulutettu kaksi neuroverkkomallia: yksi, joka käyttää syötteenä koko systeemin tilatiheyksiä ja toinen, joka käyttää pääkomponenttianalyysin tuloksena saatuja komponentteja jotka kuvaavat tilatiheysarvojen keskeisimpiä piirteitä.

Tässä työssä havaittiin pääkomponenttien avulla koulutetun neuroverkon koulutuksen olevan huomattavasti nopeampaa, ja antavan myös parempia ennusteita Hamiltonian-parametreille. Myös suuremman opetusdatamäärän havaittiin parantavan tulosten tarkkuutta. Parhaiten toimiva neuroverkkomalli koulutettiin suuremman systeemin pääkomponenttien avulla, ja sen ennusteiden virheet olivat yhden kertaluokan pienempiä kuin muiden mallien.

**Avainsanat** suprajohtavuus, epätavanomainen suprajohtavuus, topologinen suprajohtavuus, koneoppiminen, neuroverkot, pääkomponenttianalyysi

# Contents

# 1 Introduction

Superconductivity is a quantum mechanical phenomenon observed in certain materials, where the electrical resistance of the material drops to zero and magnetic fields are expelled below a critical temperature. At the microscopic level the transition of a material to its superconducting state causes its electronic structure to change as described by the Bardeen, Cooper, and Schrieffer (BCS) theory, where electrons are treated as waves rather than independent particles. BCS theory explains conventional superconductivity through so-called Cooper pairs, which form when two electrons in a lattice attract each other via positive ions instead of Coulomb repulsion. (Keller et al., 1993)

Superconductors have long been used in magnetic resonance imaging (MRI), and are now increasingly being used in quantum computer applications. A type of unconventional superconductivity called topological superconductivity is an especially potential candidate for topological quantum computing due to exotic quasiparticles called Majorana-fermions existing at the edges or defects of a topological superconductor. These quasiparticles are highly stable and therefore robust against local perturbations, which are common in quantum computer applications. A conventional superconductor can be turned into a topological superconductor using a combination of magnetic exchange fields and so-called Rashba spin-orbit coupling which couples electron spins with their momenta. (Kezilebieke et al., 2020; Sato & Ando, 2017)

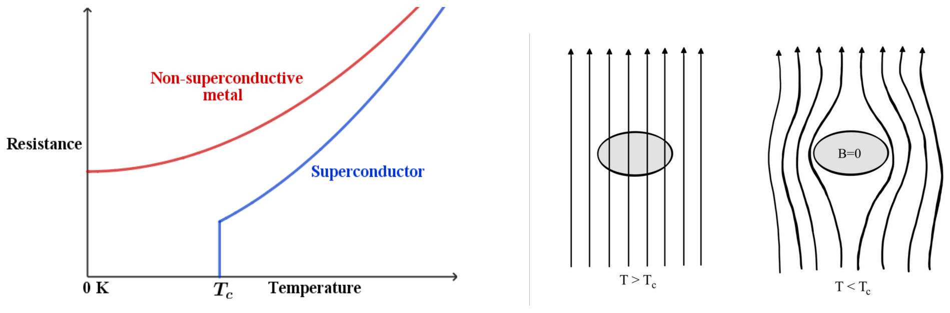
Superconductivity in a material can be recognised by using a method called scanning tunneling microscopy, which measures the differential conductance of a material and reveals its density of states, or number of available quantum states at a certain energy (Zhu, 2016). A topological superconductor can be recognised from its density of states spectra due to a clear conductance peak at zero energy caused by Majorana-fermions (Sato & Ando, 2017). Quantum systems such as superconductors are described mathematically using Hamiltonian functions, which parameterise the total energy of a system (Logan, 2005).

The focus of this thesis is to build a neural network model capable of recognising parameters of a Hamiltonian describing unconventional, potentially topological superconductors. This is done by simulating Hamiltonians and density of states values of unconventional superconductors with two different system sizes. The density of states is used as input for the neural network, and the trained network is used to predict the corresponding Hamiltonian parameters. To make training faster and potentially decrease prediction errors, principal component analysis is used to reduce dimensionality of inputs for the neural network.

This thesis is structured as follows: first, the theory behind superconductivity and specifically topological superconductivity, as well as scanning tunneling microscopy and its relation to these phenomena will be introduced in section 2. In this section also some previous research on the topic of Hamiltonian parameter estimation will be discussed. Section 3 covers the methods used in this thesis to simulate superconductors and estimate Hamiltonian parameters. Section 4 presents the results obtained, and section 5 concludes the thesis.

## 2 Theoretical Background

Superconductivity is a state where the electrical resistance of a conductor suddenly drops to zero at a certain critical temperature $T_c$ instead of gradually decreasing as temperature lowers, as depicted in figure 1a. This state is usually achieved at a very low temperature, unique to each superconducting material. Above the critical temperature $T_c$, the material is in normal phase, and its electrons occupy energy levels up to the Fermi level, which is the amount of thermodynamic work required to add a new electron to the material. In a superconducting material a new energy gap starts forming in the band structure at temperature $T_c$. The band structures of the normal phase and superconducting phase of a metal are shown in figure 2, where holes are sites with no electrons. (Sidebottom, 2012)



(a) Superconductive vs. non-superconductive material resistance

(b) The Meissner effect

Figure 1: Properties of superconductors (Adapted from Keller et al., 1993)

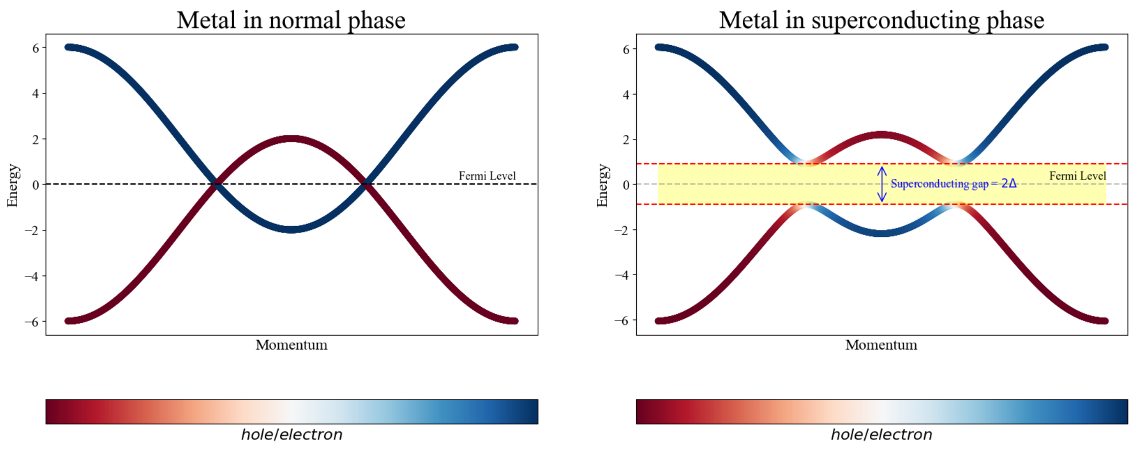


Figure 2: Formation of the superconducting gap in a metal

Another property of superconductors is their effect on magnetic fields. When a superconducting material is cooled below $T_c$ in the presence of a magnetic field, the field lines are completely expelled from the inside of the material as shown in figure 1b. This phenomenon is called the Meissner effect, which can essentially cause a superconducting specimen to levitate as it is cooled below its critical temperature.

The magnetic field also has a critical value $H_c$ which is dependent on temperature, above which the superconducting state is destroyed and the material returns back to its non-superconducting state. (Keller et al., 1993; Sidebottom, 2012)

## 2.1 Microscopic Theory of Superconductivity

The microscopic theory of superconductivity is known as the Bardeen-Cooper-Schrieffer (BCS) theory which states that superconductivity happens when electrons with equal but opposite momenta and opposite spin attract each other through lattice vibrations, forming a so-called Cooper pair as depicted in figure 3. More specifically, an electron passing through the lattice attracts ions by Coulomb interaction, which distorts the lattice structure slightly and creates a so called phonon. Phonons are quasi-particles that represent the quantisation of vibrations within the lattice in a similar way as photons represent quantised light waves. These phonons create an attractive interaction between the electrons that exceeds Coulomb repulsion, lowering the energy of the now formed Cooper pair to be lower than the energies of the two separate electrons combined. In the BCS theory these coupled electrons enter a new ground state called the BCS ground state. (Bardeen et al., 1957; Keller et al., 1993; Sidebottom, 2012)

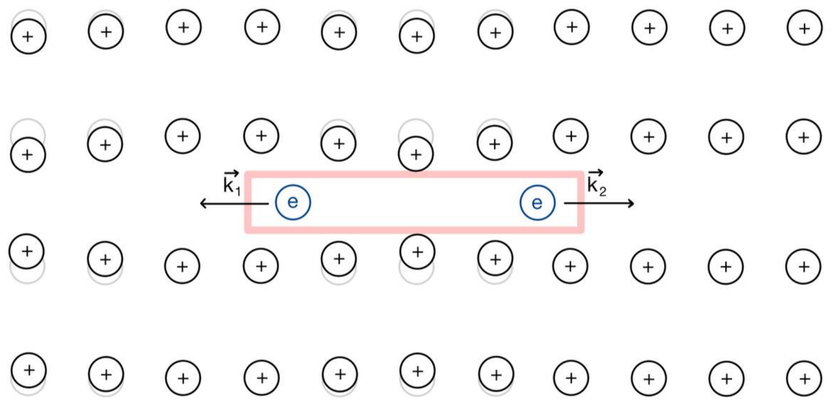


Figure 3: Formation of a Cooper pair in the lattice of a superconducting material (Adapted from Sidebottom, 2012)

Breaking a Cooper pair, i.e., a bound state of electrons, requires energy which is equal to the energy gap $2\Delta$ present in the superconducting state as depicted in figure 2. The energy gap is the difference between the uncoupled electron upper energy level and the BCS ground state. The lower the temperature, the more Cooper pairs are formed, causing the material to transfer to a superconducting state. (Bardeen et al., 1957; Sidebottom, 2012)

Superconductors can be divided into conventional and unconventional superconductors on the basis of the origin of the attractive interactions. The standard BCS theory described above explains conventional superconductivity, but for unconventional superconductivity the attraction between electrons can also come from other quasi-particles than phonons. All fluctuations of a material are some kinds of quasi-particles, and they can all mediate superconductivity. (Sigrist, 2005)

## 2.2 Conventional superconductivity

The physical properties of quantum systems can be represented by Hamiltonians, or total energy operators, that are written using so-called second quantisation, in which all operators can be expressed using creation and annihilation operators $c_i^\dagger$ and $c_i$ (Logan, 2005).

The standard BCS theory Hamiltonian describing conventional superconductivity, or so called s-wave superconductivity, is shown in (1), where $c_{ns}^\dagger$ creates and $c_{ns}$ annihilates an electron in site $n$ with spin $s$. The term $t_{ij}$ is the hopping amplitude between sites $i$ and $j$ and $\langle ij \rangle$ runs over nearest neighbors of a lattice, $\mu$ is the on-site chemical potential, and $U$ the on-site interaction, with $U < 0$ corresponding to attractive interaction needed for superconductivity. All the terms are vectors or matrices. The first and second term of the Hamiltonian thus represent the kinetic and chemical potential energy of the system, respectively, and the last term is the (attractive) two-particle interaction between electrons. (Madhuparna, 2020)

$$H = \sum_{\langle ij \rangle, s} t_{ij} c_{is}^\dagger c_{js} + \mu \sum_{i,s} c_{is}^\dagger c_{is} + \sum_i U c_{i\uparrow}^\dagger c_{i\uparrow} c_{i\downarrow}^\dagger c_{i\downarrow} \tag{1}$$

This Hamiltonian includes quartic terms in the interaction part, which means the Hamiltonian is not (exactly) solvable. To make the system quadratic, i.e., exactly solvable, a method called mean-field approximation (2) can be used.

$$\begin{aligned} U c_{i\uparrow}^\dagger c_{i\uparrow} c_{i\downarrow}^\dagger c_{i\downarrow} &\approx U \langle c_{i\uparrow}^\dagger c_{i\downarrow}^\dagger \rangle c_{i\uparrow} c_{i\downarrow} + h.c. \\ &\approx \Delta c_{i\uparrow} c_{i\downarrow} + h.c. \end{aligned} \tag{2}$$

Here $\Delta \sim \langle c_{i\uparrow}^\dagger c_{i\downarrow}^\dagger \rangle$ is the superconducting order and $h.c.$ the hermitian conjugate $\Delta^* c_{i\uparrow}^\dagger c_{i\downarrow}^\dagger$. The mean-field Hamiltonian for an s-wave superconductor can now be written as follows:

$$H = \sum_{\langle ij \rangle, s} t_{ij} c_{is}^\dagger c_{js} + \mu \sum_{i,s} c_{is}^\dagger c_{is} + \sum_i \Delta c_{i\uparrow} c_{i\downarrow} + h.c. \tag{3}$$

Since the mean-field Hamiltonian defined by (3) is quadratic, it can be solved by diagonalisation. This process is known as Bogoliubov-de-Gennes transformation, for which we define a so called Nambu spinor (4), and rewrite the Hamiltonian in this new basis, which allows the Hamiltonian to be written in diagonal form (5), where $k$ is the electron momentum and $\epsilon_\alpha$ are the Nambu eigenvalues. (Röntynen & Ojanen, 2015; Zhu, 2016)

$$\Psi_n = \begin{pmatrix} c_{n\uparrow} \\ c_{n\downarrow} \\ c_{n_\downarrow}^\dagger \\ -c_{n\uparrow}^\dagger \end{pmatrix} \tag{4}$$

$$H = \sum_k \epsilon_k \Psi_k^\dagger \Psi_k \tag{5}$$

The diagonalisation described above is also the how Hamiltonians are essentially defined when simulating them in this thesis.

## 2.3 Topological superconductivity

While conventional s-wave superconductors discussed so far exhibit a simple electron pairing through phonon mediated interactions, the interactions in topological superconductors are more complex. Topological superconductors are unconventional superconductors with unique quantum properties that give rise to the emergence of so called Majorana fermions. These Majorana fermions are their own antiparticles, and they are predicted to exist at the edges of topological superconductors. (Sato & Ando, 2017)

Due to the topological nature of materials where Majorana fermions can exist, these particles are very stable, which provides robustness and allows them to carry information without losing it due to surrounding noise. This property makes them very promising candidates for robust quantum computing applications. (Kezilebieke et al., 2020; Sato & Ando, 2017)

A conventional s-wave superconductor can be turned into a topological superconductor by introducing a magnetic exchange field and so called Rashba spin-orbit coupling (SOC). The magnitude of the exchange field essentially controls the transition between trivial and topological states so, that in a weak exchange field the superconductor remains in a trivial non-topological superconducting state, but as the strength of the field increases, it eventually causes the superconducting gap to close and then reopen in a topological state. Rashba SOC on the other hand changes how electrons behave in the material by coupling their spin with their momentum, which allows for mixed pairing states. Under the right conditions, the combination of these two effects can lead to topological superconductivity, which is especially promising in the field of topological quantum computation. (Kezilebieke et al., 2020; Khosravian & Lado, 2022; Sato & Ando, 2017)

The Hamiltonian describing an exchange field in the z-direction is defined as

$$H_J = J_z \sum_{i,s,s'} \sigma_z^{s,s'} c_{is}^\dagger c_{is'} \tag{6}$$

where $J_z$ is the exchange coupling term and $\sigma_z^{s,s'}$ is the Pauli matrix representing the spin component along the z-axis as defined in (7).

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{7}$$

The Hamiltonian for Rashba SOC can similarly be defined as

$$H_R = i\lambda_R \sum_{\langle ij \rangle, ss'} d_{ij} \cdot \sigma^{s,s'} c_{is}^\dagger c_{js'} \tag{8}$$

where the term $\lambda_R$ is the spin-orbit coupling constant, $\sigma^{s,s'}$ denote the spin Pauli matrices as defined by (7), and $d_{ij}$ determines the relative position between neighboring sites $i$ and $j$. In both (6) and (8) the terms $c^\dagger_{ns}$ and $c_{ns}$ again create or annihilate an electron with spin $s$ in site $n$. (Khosravian & Lado, 2022)

## 2.4 Scanning Tunneling Microscopy

Superconductivity of a material can be recognised by using a method called scanning tunneling microscopy (STM). STM can be used to obtain atomic-scale images of surfaces (Chen & Smith, 1994), and it is specifically used to measure the differential conductance $\frac{dI}{dV}$ of a material, which is proportional to the quasi-particle density of states (DOS) (Zhu, 2016). Density of states, parameterised by (9), is the number of quantum states available to be occupied by particles, in this case electrons, at each given energy level (Zasadzinski, 2003). In (9) $\omega$ are the energies the DOS is evaluated for, $\epsilon_k$ are the eigenenergies of the corresponding Hamiltonian and $\delta$ is the Dirac delta function which ensures only states with energy equal to $\omega$ contribute to the integral.

$$D(\omega) = \int \delta(\omega - \epsilon_k) dk \tag{9}$$

From STM measurements it is possible to gain information on the magnitude of the superconducting gap $\Delta$ as well as pairing symmetry (Zasadzinski, 2003). In addition to this, the DOS measurements for the edge sites of topological superconductors show a spike at zero energy, signaling the existence of Majorana fermions (Sato & Ando, 2017).

## 2.5 Previous research

Traditionally Hamiltonian models have been fitted to experimental data (Fujita et al., 2018), and many tools have already been developed in condensed matter physics to obtain the state of a system from the Hamiltonian describing it (Bairey et al., 2019). The development of quantum devices however requires the opposite, namely recovering the Hamiltonian of a system from measurable observables of its physical state (Bairey et al., 2019). Machine learning has been used in many quantum physics related problems, and has been shown to work well also in estimating Hamiltonians (Che et al., 2021).

Bairey et al. (2019) have shown that obtaining the Hamiltonian of a specific region from measurements is possible in the case of short-range interactions by using measurements on local observables. Che et al. (2021) on the other hand showed that it is possible to learn Hamiltonian parameters from expectation values of single-qubit measurements using recurrent neural networks (RNNs), whereas a gradient descent algorithm was used to construct Hamiltonians from energy- and entanglement spectra by Fujita et al. (2018).

A more similar approach to the one taken in this thesis was taken by Khosravian et al. (2024), where Hamiltonian parameters of unconventional superconductors were successfully extracted using fully connected neural networks with local density of states (LDOS) measurements as inputs. Here both the superconducting order as well as exchange coupling were extracted, and extracting the superconducting order was found to be a harder task.

## 3 Data and Methods

This section will focus on describing how a neural network model able to recognise unconventional (topological) superconductivity is built. In short, this is done by first simulating data on superconductors, compressing it for faster computation, and building and training neural network models with both non-compressed and compressed data. This section begins by describing how a topological superconductor can be simulated, after which principal component analysis as a dimensionality reduction technique will briefly be discussed. Lastly the theory behind neural networks and how they are used in the scope of this thesis are explained.

All the data creation, preprocessing as well as machine learning model training and testing done for this thesis is done using Python. The Python code used for building, training and testing the machine learning models defined in this section can be found in appendix A.

### 3.1 Simulating a Topological Superconductor

In this thesis the Hamiltonian of interest is defined by combining exchange coupling, Rashba spin-orbit coupling and superconductivity, which gives rise to a topological superconducting state as described in section 2.3. The Python-library Pyqula (Lado, 2021) is used to construct a Hamiltonian of the form

$$H = H_{kin} + H_R + H_J + H_{SC} \tag{10}$$

where $H_{kin} = t \sum_{\langle ij \rangle, s} c^{\dagger}_{i,s} c_{j,s} + \mu \sum_i c^{\dagger}_{i,s} c_{i,s}$ accounts for kinetic energy and chemical potential and $H_{SC} = \sum_i \Delta c_{i\uparrow} c_{i\downarrow} + h.c.$ for superconductivity as in the mean-field Hamiltonian for an s-wave superconductor defined by (3). $H_R$ and $H_J$ are the Hamiltonians for Rashba SOC and exchange coupling defined by (8) and (6), respectively.

The variables quantifying the nature of topological superconductivity are Rashba SOC $\lambda_R$, exchange coupling $J_z$ and the superconducting order $\Delta$, which is why the values of these are varied randomly to create Hamiltonians with different couplings. The variable $\Delta$ gets random values in the range [0, 0.5], whereas variables $J_z$ and $\lambda_R$ both get random values in the range [0, 1]. These values are the output that will be predicted by the machine learning model defined later in this section.

Pyqula can also be used to obtain the DOS values corresponding to a given Hamiltonian and site. DOS values are obtained for each site of each Hamiltonian in the output data, and these will serve as inputs of the machine learning model. It is worth noting that because the values for $\Delta$, $J_z$, and $\lambda_R$ are randomly determined, not all of the Hamiltonians in the data are in a topological state. The edge site DOS of a random Hamiltonian from data simulated for a 10 site system is shown in figure

4, where we can clearly see a spike at zero energy, meaning that this Hamiltonian is potentially one describing a topological superconductor.

Two datasets were simulated for the purposes of this thesis. The first dataset consists of 20 000 Hamiltonians as defined in (10) that describe systems with 10 sites, and the corresponding DOS values for each site. This dataset was used to confirm that unconventional superconductivity can be recognised from DOS measurements using machine learning. The second dataset consists of 40 000 Hamiltonians also of the form (10) and the corresponding DOS values, this time describing systems with 30 sites. As this data has more sites, it also has more inputs, which makes it a more interesting dataset for exploring dimensionality reduction techniques.

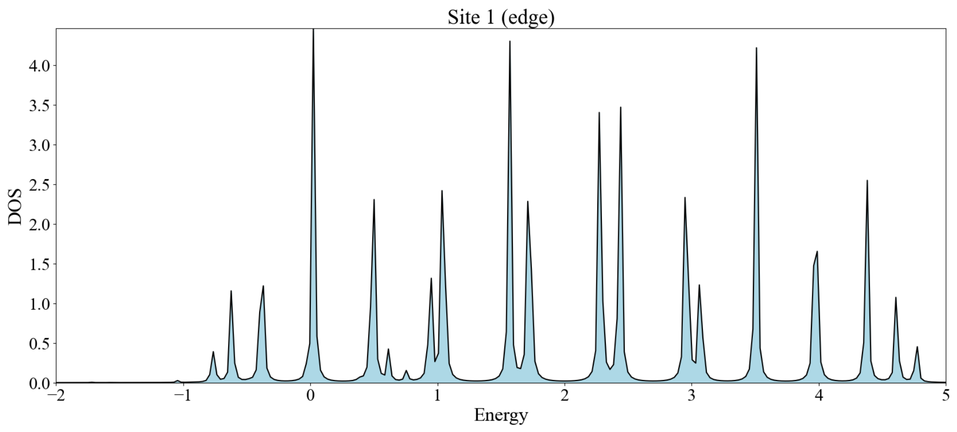


Figure 4: Edge site DOS of a Hamiltonian describing a system with 10 sites

## 3.2 Principal Components Analysis

Principal components analysis (PCA) is an orthogonal, linear transformation that projects the original data onto a new coordinate system corresponding to the directions of greatest variance. The new variable space consists of so-called principal components (PCs), where the first PC represents the linear combination of the variables in the original data that accounts for most of the variance, the second PC represents the largest proportion of variance once PC1 is removed, and so on. Due to using linear combinations of the original data as data points, this new representation has lower dimensionality than the original data, which makes PCA an effective method for dimensionality reduction. (Goodfellow et al., 2016)

In this thesis PCA is used to reduce the dimensionality of input data significantly before feeding it to the neural network model. This is done by transforming the DOS data into its principal components, and using as input the PCs explaining approximately 99.9 % of the variance in the full DOS. The results of the model trained with principal components will then be compared to those obtained using the full DOS.

## 3.3 Supervised Learning with Neural Networks

The learning process of all machine learning (ML) algorithms begins by specifying a model, i.e. the structure of a mathematical function that maps inputs to outputs. The model initially has random parameters called weights, and the objective of ML models is to optimise these weights during training by minimising a so-called loss function that essentially quantifies the difference between predictions and true values. Machine learning models can be roughly divided into supervised and unsupervised learning based on whether they are trained using labeled data or not. The focus of this thesis is on building supervised models, in which the model uses predetermined input-output pairs to learn how inputs are mapped to outputs and make predictions on unseen data. (Goodfellow et al., 2016; Greplova et al., 2020)

The models built in this thesis are based on a specific type of supervised learning, namely neural networks (NN), that work analogously to biological brains. A neural network is essentially a mathematical function consisting of elementary functions called (artificial) neurons that are organised in connected layers. The output from one layer of neurons is the input of the next layer of neurons, and information moves like this through the network, with the last layer giving the final output. NNs are extremely powerful algorithms and can technically learn to predict any smooth function as the number of neurons tends to infinity, but they generally require a large amount of data and parameters to learn effectively, making them very computationally heavy. (Greplova et al., 2020)

### 3.3.1 Mathematical Representation

The function corresponding to a single neuron in a NN is a composition of a linear function $q : \mathbb{R}^k \to \mathbb{R}$ and a non-linear activation function $g : \mathbb{R} \to \mathbb{R}$:

$$F(z_1, \dots, z_k) = g(q(z_1, .., z_k)) \tag{11}$$

where $z_1, .., z_k$ are the outputs of neurons from the previous layer. The linear function $q$ in (11) is parameterized by (12), where $w_j$ are weights that represent the importance of connections between a neuron and the neurons in the previous layer, and $b$ is a constant offset or bias, both of which are optimised during training.

$$q(z_1, \dots, z_k) = \sum_{j=1}^{k} w_j z_j + b \tag{12}$$

The activation function $g$ in (11) is chosen heuristically for one layer at a time, but it can vary by layer. In this thesis the activation function used in all hidden layers of the NN models is the so-called rectified linear unit (ReLu), which is defined by (13). ReLu gives a value 0 for negative inputs and is linear for positive ones. (Goodfellow et al., 2016)

$$g(q) = \max\{0, q\} \tag{13}$$

### 3.3.2 Parameter Optimisation and Training

In this thesis the outputs of the models are continuous, which is why mean squared error (MSE) is used as the loss function when training the model. MSE is defined by

$$L(\theta) = \frac{1}{m}\sum_{i=1}^{m} \|F(x_i) - y_i\|_2^2 \tag{14}$$

where $\|a\|_2 = \sqrt{\sum_i a_i^2}$ is the L2-norm, $F(x_i)$ are the predictions of the model on the data and $y_i$ are the true data output values (Goodfellow et al., 2016). As can be seen from (14), the loss decreases when the difference between the predictions and the true values decreases. Root mean square error (RMSE), which is the square-root of MSE, is used in this thesis alongside MSE for comparing the model predictions against true test values when testing the model on previously unseen data. RMSE quantifies the error in the same units as the target variables, so it shows the average absolute amount the predictions differ from true values.

In addition to choosing a loss function, an optimisation algorithm also needs to be chosen, with which the NN optimises the weights and biases. The optimiser used in this thesis is adaptive moment estimation (ADAM), which is based on gradient descent methods and is the most common optimiser used in NNs due to its ability to efficiently optimise models with less memory. (Greplova et al., 2020)

Network parameters are not updated after each training sample, but are instead divided into so-called batches. These batches, or groups of training data, are fed to the network at the same time and weights and biases are adjusted after each batch, which usually leads to faster optimisation. Each batch is processed several times, and one cycle through the full training data is called an epoch. If the number of epochs is too low, the model might not learn patterns in the data well, and conversely, if the amount of epochs is too high the model might start overfitting, reducing its ability to generalise on new, unseen data. (Greplova et al., 2020)

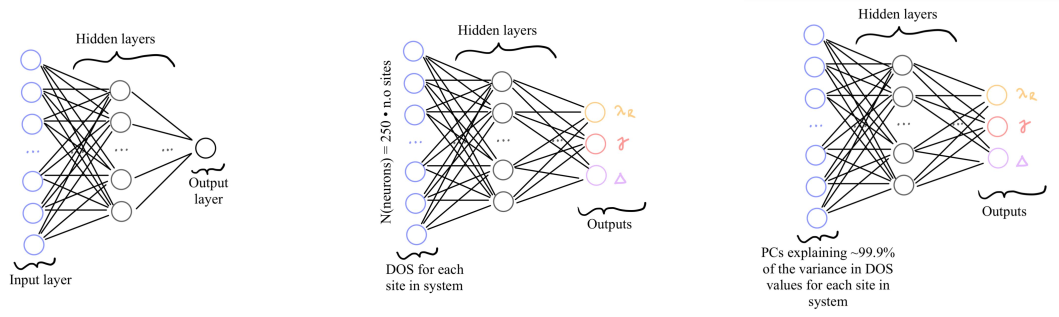
A method called early stopping can be used to prevent overfitting, which stops the training right before the model begins to overfit, meaning that not all the predetermined epochs will be completed. The training process of a NN involves so called validation, which also monitors and prevents overfitting. Validation evaluates the model's performance on a separate dataset (validation set) between epochs, and the loss values for training and validation data after each epoch can be compared to see how well the model performs on previously unseen data. (Greplova et al., 2020)

In this thesis training is done on all models with a batch size of 32 and maximum 200 epochs. Early stopping is also used in all models to prevent overfitting.

### 3.3.3 Network Architecture

The NN models built in this thesis are all so called fully connected or dense neural networks, meaning that each neuron in a layer takes as input the output of all neurons in the previous layer (Greplova et al., 2020). A general architecture of a NN is shown in figure 5a, where the input and output layers are visible layers that can be accessed directly, and the layers between them are so called hidden layers.

In this thesis two types of models will be built for both system sizes: models taking as input the full DOS of the system, and models taking as input the PCs explaining approximately 99.9 % of the variance in the full DOS. Figures 5b and 5c show the architectures of the models with DOS and PC inputs, respectively.


Hidden layers
Output layer
Input layer
Hidden layers
N(neurons) = 250 • n.o sites
DOS for each site in system
Outputs
Hidden layers
PCs explaining ~99.9% of the variance in DOS values for each site in system
Outputs

(a) A general neural network architecture (b) Architecture of NN with DOS as input (c) Architecture of NN with PCs as input

Figure 5: Neural network architectures

The amount of neurons in the input layer of the model using the full DOS is $250 \cdot \text{n.o.sites}$, where 250 is the amount of energies the DOS values are simulated for for each site of each Hamiltonian. The dataset describing systems with 10 sites has 2500 inputs, and the dataset describing systems with 30 sites has 7500 inputs per Hamiltonian. The original 20 000 and 40 000 Hamiltonian datasets are both split into 80 % training, and 20 % test data, and the models are trained using 10 % of the training data for validation.

# 4 Results

This section covers the results obtained from the neural network models built for both 10 and 30 site systems simulated as described in the previous section. First the results for the smaller dataset describing the more simple systems of 10 sites are presented and discussed, after which the same will be done for the more complicated dataset describing systems with 30 sites. Lastly a comparison between the systems and models is conducted, and some recommendations for further improvements discussed.

## 4.1 Predictions on 10 Site Systems

### 4.1.1 Full DOS as Training Data

The smaller system was trained on a model with five hidden layers using 80 % of the original data, after which it was tested using the remaining 20 %. Predictions on the test data output variables $\lambda_R$, $J_z$ and $\Delta$ against their true values are shown in figure 6 with the red dotted line showing where the points should ideally lie. Table 1 shows the errors in these predictions for each output variable. These results show that the model is predicting the three Hamiltonian parameters relatively well, with RMSE around 0.01 for all three variables. Predictions for the superconducting order are a bit less accurate than those of Rashba SOC, while the most accurate predictions are for exchange coupling with RMSE being approximately 0.001 less than for superconducting order and Rashba SOC. Figure 6 also shows that predictions for all parameters are less accurate near the boundaries of the parameter spaces.

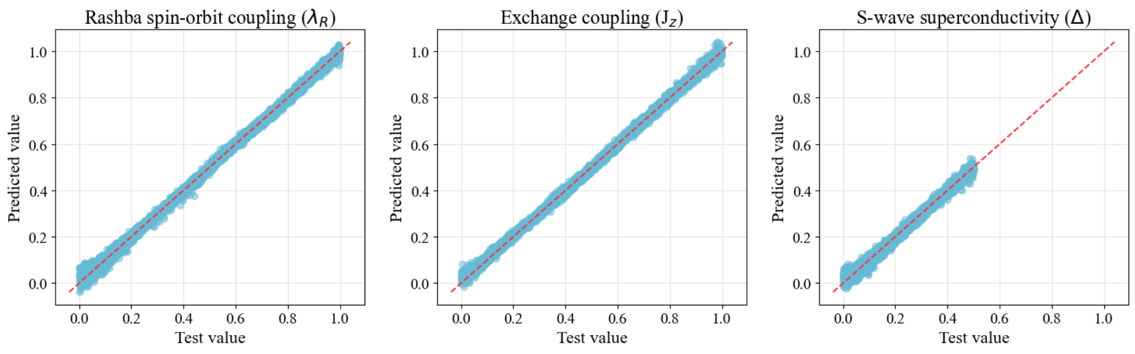


Figure 6: 10 site Hamiltonian test data vs. predicted values without data compression

Table 1: MSE and RMSE for predictions against true values

| **Variable** | **MSE** | **RMSE** |
|---|---|---|
| $\lambda_R$ | $1.70 \cdot 10^{-4}$ | 0.0130 |
| $J_z$ | $1.41 \cdot 10^{-4}$ | 0.0119 |
| $\Delta$ | $1.77 \cdot 10^{-4}$ | 0.0133 |

The loss values during training are shown in figure 7, which shows that early stopping was executed at 41 epochs. This model usually started to overfit around 35-45 epochs. The final values of both the training and validation loss end up being less than $3.00 \cdot 10^{-4}$, while the test loss is approximately $1.81 \cdot 10^{-4}$.

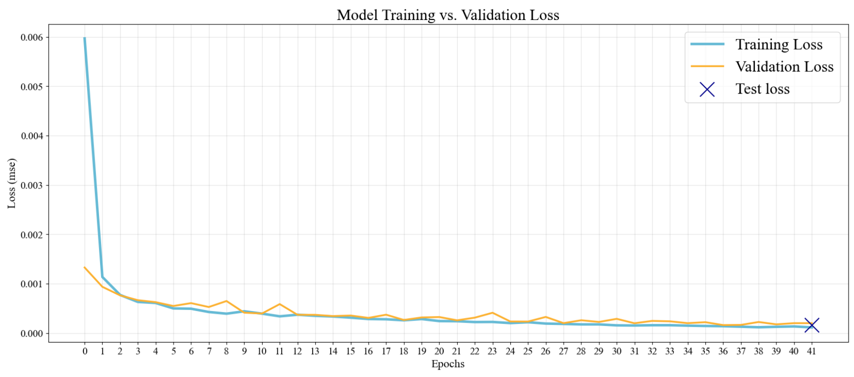


Figure 7: MSE loss for 10 site model without data compression

### 4.1.2 Training With PCA Transformed Data

To see if the model could be improved with dimensionality reduced data, PCA was performed. For the 10 site system 1000 PCs were found to explain 99.9 % of the variance in the data, so these were chosen as inputs instead of the full DOS. The model used here has five hidden layers as before, but the amount of neurons per layer is now smaller due to the input space being only slightly over a third of the original one.

Figure 8 shows the predictions against true values for each output and table 2 the errors in these predictions. There seem to be some outliers in the predictions for each variable, which is expected as the data does not include all the information from the original data. Overall the results are improved with RMSE for Rashba SOC improving by 0.001 and for superconducting order by almost 0.003. Interestingly the predictions for superconducting order $\Delta$ improve the most, with both MSE and RMSE values now being the smallest of the three outputs, whereas in the full DOS predictions they were the largest. However, PCA does not seem to perform any better on the boundaries of the parameter spaces.

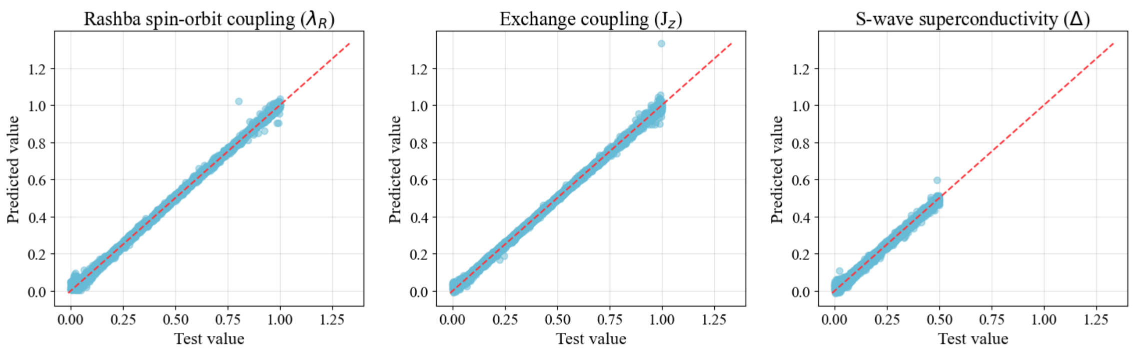


Figure 8: 10 site Hamiltonian test data vs. predicted values with PCA

Table 2: MSE and RMSE for predictions against true values

| Variable | MSE | RMSE |
|---|---|---|
| $\lambda_R$ | $1.41 \cdot 10^{-4}$ | 0.0119 |
| $J_z$ | $1.32 \cdot 10^{-4}$ | 0.0115 |
| $\Delta$ | $1.16 \cdot 10^{-4}$ | 0.0108 |

Figure 9 shows the loss values during training, which are similar to those of the non-compressed data. The training and validation loss are again below $3.0 \cdot 10^{-3}$, with a test loss of $1.82 \cdot 10^{-4}$. Early stopping is executed here at 44 epochs, and was executed around 35-45 epochs on each training similar to the full DOS model.

These results show evidence that PCA can be used to make NN model predictions more accurate. This is especially important as the system size, and consequently the input space, grows.

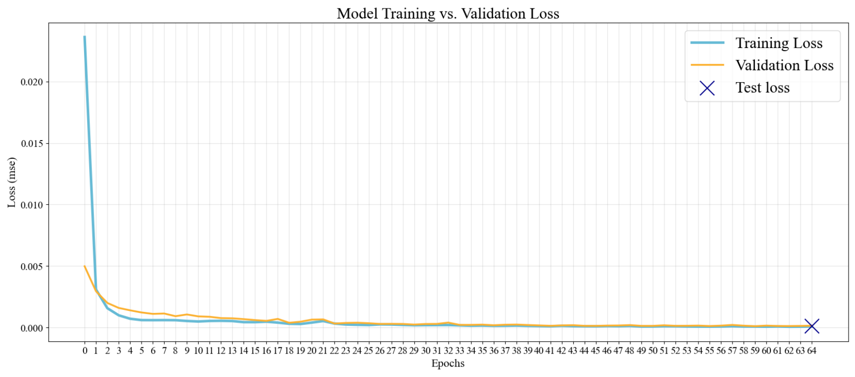


Figure 9: MSE loss for 10 site model using PCA

## 4.2 Predictions on 30 Site Systems

### 4.2.1 Full DOS as Training Data

A very similar model was built for the larger system, with five hidden layers but more neurons per layer than in the 10 site system full DOS model. Figure 10 shows the predictions for outputs and table 3 the prediction errors. This time the predictions for exchange coupling are the least accurate, whereas in the 10 site full DOS predictions they were the most accurate. Rashba SOC errors are nearly the same as before, but both MSE and RMSE for superconducting order predictions are better than for the 10 site model.

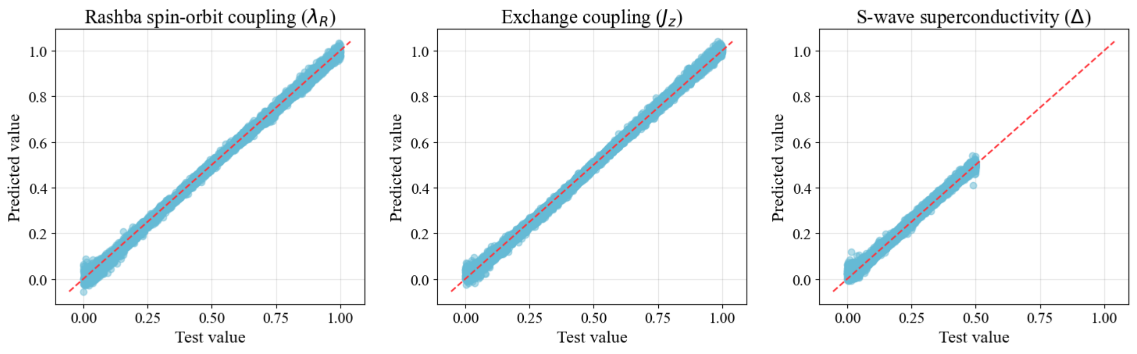


Figure 10: 30 site Hamiltonian test data vs. predicted values without data compression

Table 3: MSE and RMSE for predictions against true values

| Variable | MSE | RMSE |
|---|---|---|
| $\lambda_R$ | $1.73 \cdot 10^{-4}$ | 0.0131 |
| $J_z$ | $1.82 \cdot 10^{-4}$ | 0.0135 |
| $\Delta$ | $1.57 \cdot 10^{-4}$ | 0.0125 |

The loss values during training are shown in figure 11, which shows that early stopping was executed at 21 epochs. When training the model several times with new splits into training and testing sets, the amount of epochs usually ended up being around 20, so much earlier than for the 10 site system models. Both the training and validation loss end up being around $3.0 \cdot 10^{-4}$, with test loss being approximately $1.7 \cdot 10^{-4}$.

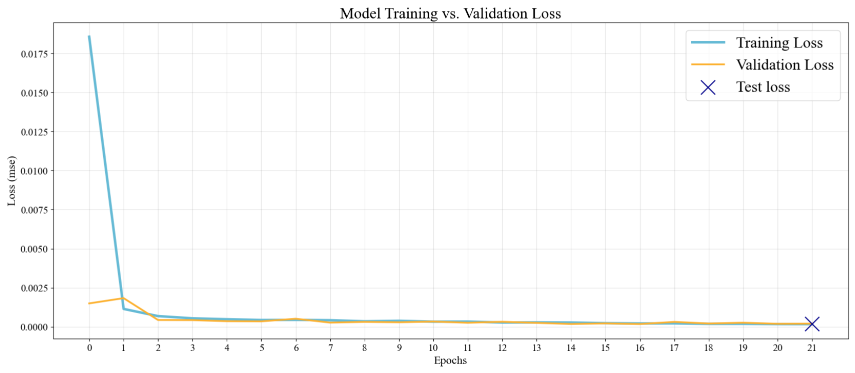


Figure 11: MSE loss for 30 site model without data compression

### 4.2.2 Training on PCA Transformed Data

As mentioned, dimensionality reduction is especially relevant for larger sized systems, as they require more inputs. 1500 PCs explain 99.9 % of the variance in the 30 site system dataset, so these were used as inputs instead of the full DOS in the last model. The prediction results of this model are shown in figure 12, and the errors in table 4. Both MSE and RMSE for all outputs are one order of magnitude smaller than in any of the previous models, meaning that this model is the most accurate one. However, the issue of larger prediction errors on the boundaries of parameter spaces still prevails in this model as well.

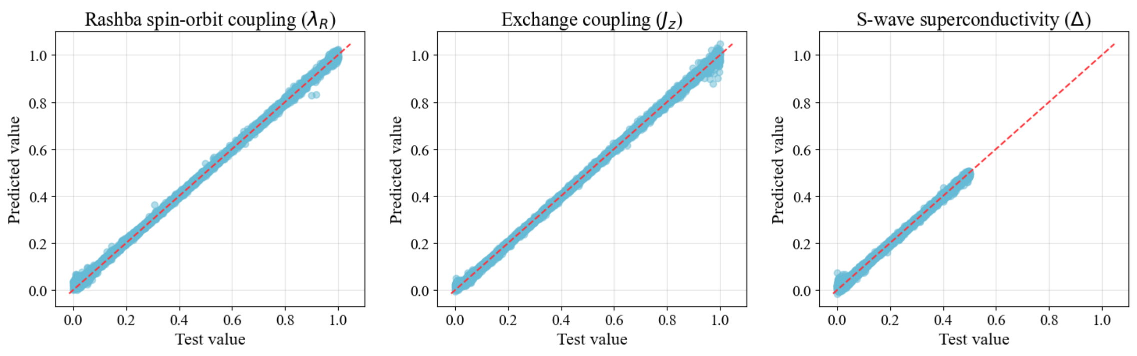


Figure 12: 30 site Hamiltonian test data vs. predicted values with PCA

The PCA loss is shown in figure 13, which shows that the training and validation loss both end up very close to 0, while the test loss here is $5.90 \cdot 10^{-5}$, further showing that this model is more accurate than the previous ones.

For the 30 site models PCA was found to decrease training time to only 1/3 of the training time when using the full DOS as input, whereas for the 10 site model

Table 4: MSE and RMSE for predictions against true values

| Variable | MSE | RMSE |
|---|---|---|
| $\lambda_R$ | $7.64 \cdot 10^{-5}$ | 0.0087 |
| $J_z$ | $6.50 \cdot 10^{-5}$ | 0.0081 |
| $\Delta$ | $4.11 \cdot 10^{-5}$ | 0.0064 |

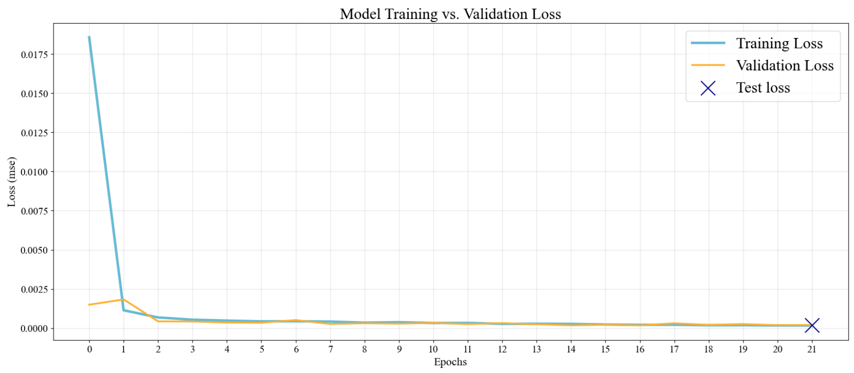


Figure 13: MSE loss for 30 site model using PCA

PCA did not significantly reduce training time. PCA did, however, make results more accurate for both models compared to using the full DOS as input.

## 5 Discussion and Summary

In this thesis four slightly different neural network models were trained on simulated Hamiltonians and DOS measurements of unconventional superconductors. Neural networks proved to be an effective way to recognise superconductivity from real space conductance measurements, and PCA was found to be effective not only for reducing the size of the input space and making training less computationally heavy, but also for making the models more accurate.

Out of the four models, the one with the lowest prediction error was one trained with PCA transformed data on a 30 site system. The RMSE values of this model were 0.0087 and 0.0081 for Rashba spin-orbit coupling and exchange coupling respectively, and only 0.0064 for superconducting order, which means this model gave predictions with errors one order of magnitude smaller than the three other models. Based on the results of all the models, it is clear that neural networks are able to recognise Hamiltonian parameters from real space (simulated) conductance measurements well.

As the results of the 30 site system models are better than for the 10 site one even before dimensionality reduction, it could be that simulating a larger dataset for the smaller model would make the results better, as this would also allow for more training examples on topological superconductors for which the DOS is different. The same applies in general, as more training examples help neural networks learn better. Another way to improve the results would be to have more DOS measurements per site. In this thesis 250 measurements per site were computed to get just enough detail in the DOS without making computation too heavy, but for even more accurate predictions the amount could be doubled to get a more detailed DOS that captures even smaller spikes in the DOS spectra.

## References

Bairey, E., Arad, I., & Lindner, N. (2019). Learning a Local Hamiltonian from Local Measurements. *Physical Review Letters*, *122*(2), 1–8. https://doi.org/10.1103/PhysRevLett.122.020504

Bardeen, J., Cooper, L., & Schrieffer, J. (1957). Theory of superconductivity. *Physical Review*, *108*(5), 1175–1204. https://journals.aps.org/pr/pdf/10.1103/PhysRev.108.1175

Che, L., Wei, C., Huang, Y., Zhao, D., Xue, S., Nie, X., Li, J., Lu, D., & Xin, T. (2021). Learning quantum Hamiltonians from single-qubit measurements. *Physical Review Research*, *3*(2), 1–10. https://journals.aps.org/prresearch/pdf/10.1103/PhysRevResearch.3.023246

Chen, C., & Smith, W. (1994). Introduction to Scanning Tunneling Microscopy. *62*(6), 573–574. https://doi.org/10.1119/1.17525

Fujita, H., Nakagawa, Y., Sugiura, S., & Oshikawa, M. (2018). Construction of Hamiltonians by supervised learning of energy and entanglement spectra. *Physical Review B*, *97*(7), 1–13. https://arxiv.org/pdf/1705.05372

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. http://www.deeplearningbook.org (Last accessed: 10.05.2024).

Greplova, E., Neupert, T., Fischer, M., Choo, K., & Denner, M. (2020). *Machine learning for scientists*. https://ml-lectures.org/docs/supervised_learning_w_NNs/ml_supervised_w_NNs.html (Last accessed: 15.05.2024).

Keller, F., Gettys, W., & Skove, M. (1993). *Physics: Classical and modern*. Mc Graw Hill.

Kezilebieke, S., Huda, M., Vaňo, V., Aapro, M., Ganguli, S., Silveira, O., Głodzik, S., Foster, A., Ojanen, T., & Liljeroth, P. (2020). Topological superconductivity in a van der Waals heterostructure. *Nature*, *588*(7838), 424–428. https://doi.org/10.1038/s41586-020-2989-y

Khosravian, M., Koch, R., & Lado, J. (2024). Hamiltonian learning with real-space impurity tomography in topological moiré superconductors. *Jounral of Physics: Materials*. https://iopscience.iop.org/article/10.1088/2515-7639/ad1c04/pdf

Khosravian, M., & Lado, J. (2022). Impurity-induced excitations in twisted topological van der Waals superconductors. *Physical Review Materials*, (500), 1–14. http://arxiv.org/abs/2202.11003

Lado, J. (2021). *Python library to compute different properties of quantum tight binding models in a lattice*. https://github.com/joselado/pyqula (Last accessed: 16.5.2024).

Logan, D. (2005). Many-Body Quantum Theory in Condensed Matter Physics—An Introduction. *Journal of Physics A: Mathematical and General*, *38*(8), 1829–1830. https://doi.org/10.1088/0305-4470/38/8/b01

Madhuparna, K. (2020). Imbalanced fermi systems and exotic superconducting phases. *A Comprehensive Guide to Superconductivity*, 129–201. https://arxiv.org/pdf/2008.11740

Röntynen, J., & Ojanen, T. (2015). Topological Superconductivity and High Chern Numbers in 2D Ferromagnetic Shiba Lattices. *Physical Review Letters*, *114*(23). https://doi.org/10.1103/PhysRevLett.114.236803

Sato, M., & Ando, Y. (2017). Topological superconductors: A review. *Reports on Progress in Physics*, *80*(7), 1–45. https://doi.org/10.1088/1361-6633/aa6ac7

Sidebottom, D. (2012). *Fundamentals of Condensed Matter and Crystalline Physics*. Cambridge University Press. https://doi.org/10.1017/cbo9781139062077

Sigrist, M. (2005). Introduction to unconventional superconductivity. *AIP Conference Proceedings*, *789*, 165–243. https://boulderschool.yale.edu/sites/default/files/files/Introduction-to-Unconventional-Superconductivity.pdf

Zasadzinski, J. (2003). Tunneling Spectroscopy of Conventional and Unconventional Superconductors. *The Physics of Superconductors*, *1*, 591–646. https://doi.org/10.1007/978-3-642-55675-3_8

Zhu, J. (2016). *Bogliubov-de gennes equations for superconductors in the continuum model* (Vol. 924). https://doi.org/10.1007/978-3-319-31314-6_1

# A Appendix

## Python Code for Neural Network Models

This notebook consists of the following sections:
**(I) 10 site models:**
*Model 1:* No data compression
*Model 2:* PCA
**(II) 30 site models:**
*Model 3:* no data compression
*Model 4:* PCA
**(III) Testing models and plotting results**

Each model *1-4* is trained in its separate section, after which it can be evaluated and tested in section **(III)**

**Libraries**

```
# ML model libraries
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import  Dense
from tensorflow.keras.metrics import R2Score
from tensorflow.keras.callbacks import EarlyStopping

# sklearn libraries
import sklearn.model_selection as sk
from sklearn.utils import shuffle
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA  # to apply PCA
from sklearn.metrics import mean_absolute_error,
↪  mean_squared_error, r2_score

# other libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

## 10 site models

**Opening data files**

```
# input
x = np.loadtxt('dos_10sites.txt', delimiter='\t', skiprows=1)

#output
y = np.loadtxt('delta_rashba_exchange_10sites.txt', delimiter='\t',
↪  skiprows=1)
```

### Model 1: No data compression

```
# split data into training and testing
x_train, x_test, y_train, y_test, = sk.train_test_split(x, y,
↪  test_size=0.2)
```

**Model architecture**

```
# define neural network structure
model = Sequential([
    Dense(512, input_shape=(2500,), activation='relu'),
    Dense(256, activation='relu'),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(3)
])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error',
↪  metrics=[R2Score()])

# Summarise model
model.summary()
```

**Training**

```
# train the model
hist = model.fit(x_train, y_train, epochs=200, batch_size=32,
↪  validation_split=0.1,
                 callbacks=[EarlyStopping(monitor='val_loss',
                 ↪  patience=5, restore_best_weights=True)])
```

### Model 2: PCA

```
# scale input data
scaler = StandardScaler()
scaled_data = pd.DataFrame(scaler.fit_transform(x))
```

```
# perform PCA
pca = PCA(n_components = 1000)
x_pca = pca.fit_transform(scaled_data)
```

```
# total variance explained by the chosen principal components
np.cumsum(pca.explained_variance_ratio_)[999]
```

```
# plot total variance
plt.plot(np.cumsum(pca.explained_variance_ratio_))
```

```
# split data into training and testing
x_train, x_test, y_train, y_test, = sk.train_test_split(x_pca, y,
↪   test_size=0.2)
```

**Model architecture**

```
# define neural network structure for PCA
model = Sequential([
    Dense(256, activation='relu', input_dim=900),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),
    Dense(3)
])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error',
↪   metrics=[R2Score()])

# Summarise model
model.summary()
```

**Training**

```
# train the PCA model
hist = model.fit(x_train, y_train, epochs=200, batch_size=32,
 ↪  validation_split=0.1,
                  callbacks=[EarlyStopping(monitor='val_loss',
                   ↪  patience=5, restore_best_weights=True)])
```

## 30 site models

**Opening data files**

```
# input
x = np.loadtxt('dos_30sites40k.txt', delimiter='\t', skiprows=1)

# output
y = np.loadtxt('delta_rashba_exchange_30sites40k.txt',
 ↪  delimiter='\t', skiprows=1)
```

### Model 3: No data compression

```
# split data into training and testing
x_train, x_test, y_train, y_test, = sk.train_test_split(x, y,
 ↪  test_size=0.2)
```

**Model architecture**

```
# define neural network structure
model = Sequential([
    Dense(512, input_shape=(7500,), activation='relu'),
    Dense(256, activation='relu'),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(3)
])

# Compile the model
model.compile(optimizer="adam", loss='mean_squared_error',
 ↪  metrics=[R2Score()])

# Summarise model
model.summary()
```

**Training**

```
# train the model
hist = model.fit(x_train, y_train, epochs=200, batch_size=32,
                 validation_split=0.1,
                 callbacks=[EarlyStopping(monitor='val_loss',
                 ↪  patience=5, restore_best_weights=True)])
```

### Model 4: PCA

```
# scale input data
scaler = StandardScaler()
scaled_data = pd.DataFrame(scaler.fit_transform(x))
```

```
# perform PCA
pca = PCA(n_components = 1500)
x_pca = pca.fit_transform(scaled_data)
```

```
# total variance explained by chosen principal components
np.cumsum(pca.explained_variance_ratio_)[1499]
```

```
# plot total variance
plt.plot(np.cumsum(pca.explained_variance_ratio_))
```

```
# split data into training and testing
x_train, x_test, y_train, y_test, = sk.train_test_split(x_pca, y,
↪  test_size=0.2)
```

**Model architecture**

```
# Define neural network structure for PCA
model = Sequential([
    Dense(256, activation='relu', input_dim=1500),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),
    Dense(3)
])

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error',
↪  metrics=[R2Score()])
```

```
# Summarise model
model.summary()
```

**Training**

```
# train the PCA model
hist = model.fit(x_train, y_train, epochs=200, batch_size=32,
 ↪  validation_split=0.1,
                 callbacks=[EarlyStopping(monitor='val_loss',
                  ↪  patience=5, restore_best_weights=True)])
```

## Testing models and plotting results

Run the following after training a model (No data compression / PCA)

**Evaluation**

```
# evaluate the model
test_loss, test_r2 = model.evaluate(x_test, y_test)
```

**Testing**

```
# test model
y_pred = model.predict(x_test)
```

```
# split predictions and test data into own lists for plotting

y_pred_1 = []; y_pred_2 = []; y_pred_3 = []
y_test_1 = []; y_test_2 = []; y_test_3 = []

for i in y_pred:
    y_pred_1.append(i[0])
    y_pred_2.append(i[1])
    y_pred_3.append(i[2])

for i in y_test:
    y_test_1.append(i[0])
    y_test_2.append(i[1])
    y_test_3.append(i[2])
```

**Prediction errors**

```
print(f"Superconductivity $\Delta$:")
mse1 = mean_squared_error(y_test_1, y_pred_1)
print("Mean Squared Error (MSE):", mse1)
rmse1 = np.sqrt(mse1)
print("Root Mean Squared Error (RMSE):", rmse1)

print(f"Rashba spin-orbit coupling $\lambda_R$:")
mse2 = mean_squared_error(y_test_2, y_pred_2)
print("Mean Squared Error (MSE):", mse2)
rmse2 = np.sqrt(mse2)
print("Root Mean Squared Error (RMSE):", rmse2)

print(f"Exchange coupling $\J_z$:")
mse3 = mean_squared_error(y_test_3, y_pred_3)
print("Mean Squared Error (MSE):", mse3)
rmse3 = np.sqrt(mse3)
print("Root Mean Squared Error (RMSE):", rmse3)
```

**Plotting actual data vs. predictions**

```
# plot actual test data vs. predicted values
plt.figure(figsize=(15, 5))

# Rashba SOC
plt.subplot(1, 3, 1)
plt.scatter(y_test_2, y_pred_2, alpha=0.5, c="#65BAD5", zorder=3)
plt.xlabel('Test value')
plt.ylabel('Predicted value')
plt.title('Rashba spin-orbit coupling ($\lambda_R$)', fontsize=18)

# diagonal line representing perfect predictions
max_val = max(np.max(y_pred), np.max(y_test))
min_val = min(np.min(y_pred), np.min(y_test))
plt.grid(color='grey', alpha=0.2, zorder=0)
plt.plot([min_val, max_val], [min_val, max_val], color='#FF343A',
 ↪  linestyle='--', zorder=4)

# exchange coupling
plt.subplot(1,3,2)
plt.scatter(y_test_3, y_pred_3, alpha=0.5, zorder=3,
 ↪  color="#65BAD5")
plt.xlabel('Test value')
plt.ylabel('Predicted value')
plt.title('Exchange coupling (J$_z$)', fontsize=18)

# diagonal line representing perfect predictions
max_val = max(np.max(y_pred), np.max(y_test))
min_val = min(np.min(y_pred), np.min(y_test))
plt.grid(color='grey', zorder=0, alpha=0.2)
plt.plot([min_val, max_val], [min_val, max_val], color='#FF343A',
↪  linestyle='--', zorder=4)
plt.tight_layout()

# Superconductivity
plt.subplot(1, 3, 3)
plt.scatter(y_test_1, y_pred_1, alpha=0.5, zorder=3, c="#65BAD5")
plt.xlabel('Test value')
plt.ylabel('Predicted value')
plt.title('S-wave superconductivity ($\Delta$)', fontsize=18)

# diagonal line representing perfect predictions
max_val = max(np.max(y_pred), np.max(y_test))
min_val = min(np.min(y_pred), np.min(y_test))
plt.grid(color='grey', alpha=0.2, zorder=0)
plt.plot([min_val, max_val], [min_val, max_val], color='#FF343A',
↪  linestyle='--', zorder=4)
```

**Plotting loss**

```
# plot loss
plt.figure()

# training and validation loss
plt.plot(hist.history['loss'], label='Training Loss', c='#65BAD5',
↪  linewidth=3.5)
plt.plot(hist.history['val_loss'], label='Validation Loss',
↪  c='#fcb130', linewidth=2.5)

# test loss at the last epoch
plt.grid(color='grey', zorder=0, alpha=0.2)
plt.scatter(len(hist.history['loss']) - 1, test_loss, c='darkblue',
↪  marker='x', s=400, label='Test loss', zorder=4)

plt.title('Model Training vs. Validation Loss', fontsize=22)
plt.xlabel('Epochs')
plt.ylabel('Loss (mse)')
```

```
epochs = len(hist.history['loss'])
plt.xticks(range(0, epochs))
plt.legend()
plt.show()
```