Elias Wachmann, BSc. BSc.

Machine Learning-Driven Initial Guess Prediction for SCF Procedures

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Technical Physics

submitted to

Graz University of Technology

Supervisor

Assoc.Prof. Mag.phil. Dipl.-Ing. Dr.phil. Dr.techn. Andreas Hauser

Institute of Experimental Physics

Co-Supervisor

Dipl.-Ing. Dr.techn. Ralf Meyer, BSc

Institute of Experimental Physics

Graz, September 2025

Affidavit

I declare that I have authored this thesis independently, that I have not used anything other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date

Signature

Abstract

This thesis presents a machine learning based approach to construct initial electron densities directly from overlap matrices to accelerate SCF convergence. We explore whether a graph neural network (GNN) can map atomic overlap integrals in an atom-centred basis to initial electron densities in that same basis, thereby significantly reducing SCF iteration count. Although graph neural networks excel at learning molecular descriptors and energies, their potential to reduce SCF iterations through tailored initialization remains largely unaddressed. We introduce a simple message passing GNN that predicts initial densities from atomic coordinates and overlap integrals. The GNN is trained and evaluated on three 500 molecule subsets of the QM9 dataset (isomers, isomers MD trajectories and a stratified sample). With a reduction in mean SCF iterations of 25% to 35% compared to a superposition of atomic densities (SAD) initialization on unseen data, we demonstrate the potential of GNN-based density initialization to dramatically lower computational costs in electronic structure simulations, aiding high-throughput materials discovery and molecular design.

Keywords: computational quantum chemistry · density functional theory · graph neural network initial guess prediction · machine learning · self-consistent field

Kurzfassung

Diese Arbeit behandelt auf Machine Learning basierende Ansätze zur direkten Konstruktion einer initialen Elektronendichte aus dem Elektronenüberlapp, um die Konvergenz von SCF Rechnungen zu beschleunigen. Es wird untersucht, ob ein Graph-basiertes neuronales Netzwerk (GNN) atomare Überlappungsintegrale in der atomzentrierten Basis auf initiale Elektronendichten in der gleichen Basis abbilden kann, um die Anzahl der SCF Iterationen signifikant zu reduzieren. Obwohl Graph-basierte neuronale Netzwerke vielversprechende Ergebnisse im Bereich von molekularen Deskriptoren und der Energievorhersage zeigen, wurde ihr Potenzial zur Reduktion von SCF Iterationen durch maßgeschneiderte Initialisierung bisher kaum erforscht. Wir stellen ein einfaches Message-Passing GNN vor, welches basierend auf Atomkoordinaten und Überlappungsintegralen, initiale Dichten vorhersagt. Trainiert wird auf drei, jeweils 500 Moleküle umfassenden Teilmengen des QM9 Datensatzes (Isomere, Isomere MD-Trajektorien und eine stratifizierte Stichprobe). Eine Reduktion der mittleren SCF Iterationen um 25% bis 35% im Vergleich zu einer Superposition of Atomic Densities (SAD) Initialisierung auf neuen Daten zeigt das Potenzial von GNN basierter Dichte-Initialisierung, den Rechenaufwand in der Elektronenstruktur-Simulation dramatisch zu senken.

Acknowledgements

I am profoundly thankful to Prof. Andreas W. Hauser and Ralf Meyer for letting me choose a topic dear to my heart and giving me the freedom to explore it in my own way. Their guidance and support were invaluable throughout this journey.

I also want to express my appreciation towards the rest of our group, especially Dominik, for the many insightful conversations we shared. A special thanks go to Nadja, for her thoughtful critiques and editorial support. To all my friends and colleagues, your support means the world to me and made stressful times much more bearable. My sincere gratitude goes to the Fürst Dietrichstein'sche Stiftung for their scholarship and the Institute of Experimental Physics for funding my master's thesis.

I would like to extend my heartfelt thanks to my parents, who always supported me unconditionally throughout my studies. Their encouragement and belief in me have been a constant source of motivation.

Contents

1 Introduction **1**

2 Theory & Methodological Background **3**

2.1 Self-consistent Field (SCF) Theory . . . 3
2.1.1 Foundations and Formalism of the Hartree-Fock Method . . . 3
2.1.2 Matrix formulation of the Hartree-Fock equations . . . 5
2.1.3 The Self-Consistent Field (SCF) Algorithm . . . 7
2.1.4 Guessing schemes . . . 8
2.1.5 Expectation values . . . 8
2.1.6 Basis sets . . . 9
2.1.7 Post-Hartree-Fock Methods . . . 11
2.1.8 Density Functional Theory (DFT) . . . 11
2.2 Machine Learning . . . 13
2.2.1 Classification & General Remarks . . . 13
2.2.2 Loss Functions . . . 14
2.2.3 Regression Models . . . 15
2.2.4 Neural Network (NN) concepts . . . 16
2.2.5 Feed forward Networks: Multilayer Perceptron (MLP) . . . 17
2.2.6 Graph Neural Network (GNN) / Message Passing Neural Nets (MPNN) . . . 18
2.2.7 Neural Network Training and Hyperparameter Tuning . . . 19
2.3 QM9 dataset . . . 20

3 Fock Matrix prediction **21**

3.1 Introduction . . . 21
3.2 $C_5H_4N_2O_2$-subset: A first trial . . . 22
3.3 Ridge Regressor model . . . 23
3.4 Kernel Ridge Regression . . . 25
3.5 Further trials using MLP . . . 26
3.6 Conclusion . . . 28

4 A GNN approach to the problem **31**

4.1 Input & Output Matrices . . . 31
4.2 Preprocessing . . . 32
4.2.1 Graph creation . . . 33
4.2.2 Data Augmentation . . . 33
4.2.3 Normalization & data split . . . 34
4.3 ML pipeline & GNN Design . . . 34
4.4 Training . . . 36
4.5 Benchmark model . . . 37

5 Application **39**

5.1 QM9 - $C_7H_{10}O_2$ Isomers . . . 39
5.1.1 Initial training . . . 40

5.1.2 Hyperparameter tuning . . . 41
5.1.3 Evaluation & Conclusion . . . 44
5.2 QM9 - $C_7H_{10}O_2$ Molecular Dynamics (MD) . . . 44
5.2.1 Zero-shot predictions . . . 45
5.2.2 Hyperparameter tuning . . . 45
5.2.3 Evaluation & Conclusion . . . 46
5.3 QM9 - Full Dataset . . . 47
5.3.1 Sampling the QM9 dataset . . . 47
5.3.2 Hyperparameter tuning . . . 48
5.3.3 Evaluation & Conclusion . . . 48

6 Conclusion & Outlook **51**

A Appendix **53**
A.1 Initial guessing methods in PySCF [34] . . . 53
A.2 Comments on second-order Newton solvers & iterations . . . 54
A.3 Source Code & Packages used . . . 55

Bibliography **57**

1. Introduction

Quantum-chemistry *ab initio* methods form the foundation for understanding the distribution and arrangement of atoms and their respective electron distributions in molecules. Regardless of the specific method, every calculation must be started with an initial estimate of the electron density. This serves as an initial reference for the self-consistent procedure that iteratively refines the density to minimize the system's total energy under the constraints of the Schrödinger equation. These self-consistent field (SCF[1]) methods scale very poorly with respect to the number of basis functions used to expand the multi-electron-wavefunction. Tremendous progress, both in theory and in algorithmic design and compute power, has been made in the last decades. Nevertheless, the scaling of SCF calculations remains of order $\mathcal{O}\left(K^4\right)$ - $\mathcal{O}\left(K^7\right)$ (depending on the specific method used) in terms of number of basis functions K. Especially for large and poorly converging systems, this still constitutes a bottleneck.

Improving the initial guess can drastically cut down the number of SCF iterations needed to reach self-consistency. Most available schemes, whether heuristic shortcuts or minimal-basis atomic projections, produce wildly different convergence behaviours. A high-quality guess not only speeds up convergence, but can mean the difference between a successful calculation and one that fails to converge. Because molecular simulations underpin R&D across pharmaceuticals, material science, and chemical engineering, faster SCF convergence translates directly into lower computational cost and accelerated development cycles. In these industries, where thousands of SCF-calculations may be launched, even millisecond-scale improvements per calculation can add up to enormous time savings, reducing both energy consumption and financial expenses.
This thesis employs statistical and machine learning (ML) techniques to predict an initial guess for the electron density. We explore the viability of various ML methodologies to derive the density straight from the overlap matrix or via an intermediate Fock matrix. All schemes will operate directly on the atomic basis functions, rather than projecting them into a chosen computational basis, as in several classical methods. To our knowledge, no such ML-based guessing scheme has been devised and benchmarked against established guessing methods on a large, diverse dataset. Given the rapid advances and broad applicability of ML models, we expect data-driven approaches to uncover subtleties in electronic correlations that traditional schemes, constrained by approximations or minimal basis expansions, cannot capture.

Chapter 2 reviews SCF theory and ML methodologies, then introduces the dataset we use. In Chapter 3, we develop density-guessing schemes based on the Fock matrix in a small basis set and extend them to a larger basis. A graph neural network (GNN) approach is devised in Chapter 4 and later applied to various datasets in Chapter 5. Finally, Chapter 6 offers closing remarks and an outlook.

[1] In this work, SCF does not refer exclusively to Hartree-Fock but is used as a general term encompassing both Hartree-Fock and Kohn-Sham methods.

2. Theory & Methodological Background

This chapter provides an overview of the theoretical background used in this thesis. While the focus of this thesis is set on the application of machine learning methods within a quantum chemistry context, a basic introduction to computational quantum chemistry methods, in particular self-consistent field (SCF) methods, is provided.

2.1. Self-consistent Field (SCF) Theory

Quantum chemistry has its roots far before the advent of the computer age. Based on the original theories on quantum mechanics formulated by Schrödinger and Heisenberg, the interest in the accurate description of matter via this new theory was sparked. After the introduction of the wavefunction by Schrödinger almost a century ago, Max Born's statistical interpretation enabled a direct calculation of the electron density. [1] Already a year later, Hartree coined a self-consistent method to solve the many-electron problem utilizing a mean-field approach. Slater and Fock independently adapted the method by adding the exchange term and consistency with the Pauli exclusion principle. This method was later named Hartree-Fock (HF) method. [2, 3, 4] From that point on many advancements have been made in this field. Most prominently density functional theory (DFT), coupled cluster methods (CC), and pertubative expansions (MP2) have been developed. Yet, the theory behind the HF method is still the foundation for nearly all of these methods.

The following introductory overview of the Hartree-Fock method is mostly based on the book by Szabo and Ostlund. [5]

2.1.1. Foundations and Formalism of the Hartree-Fock Method

The Born Oppenheimer approximation allows for a separate treatment of the nuclear and electronic problem of a system. This is motivated by the vastly different masses and hence different time scales of nuclei and electron movement. The total wavefunction of a system may be written as a product state of electronic and nuclear wavefunctions (using the convention of small letters for electrons and capital letters for nuclei coordinates):

$$\Psi_{\text{tot}}(\mathbf{r}_n, \mathbf{R}_m) = \Psi_{\text{elec}}(\mathbf{r}_n; \mathbf{R}_m)\Psi_{\text{nuc}}(\mathbf{R}_m) \tag{2.1}$$

Using our approximation, the nuclear wavefunction can be obtained rather easily by solving the nuclear Schrödinger equation for an effective potential which corresponds to the electronic energy written as a function of nuclear coordinates. Within the confines of this thesis, we will focus on the electronic wavefunction as a prerequisite, and therefore omit the subscript from here on. It

parametrically depends on the nuclear coordinates and is the solution to the time independent Schrödinger equation:

$$H\Psi(\mathbf{r}_n;\mathbf{R}_m) = E[\Psi](\mathbf{R}_m)\Psi(\mathbf{r}_n;\mathbf{R}_m) \tag{2.2}$$

The electronic energy E is a functional of the electronic wavefunction. We will minimize this energy using orthogonality constraints on the wavefunction, as will be explained later. Written in atomic units, the corresponding Hamiltonian operator H for N electrons and M nuclei reads

$$H = T + V_{\text{ne}} + V_{\text{ee}} = -\frac{1}{2}\sum_{i=1}^{N}\nabla_i^2 - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{|\mathbf{r}_i - \mathbf{R}_A|} + \sum_{i<j}^{N}\frac{1}{|\mathbf{r}_i - \mathbf{r}_j|}. \tag{2.3}$$

Note that we do not include kinetic or repulsion terms for the nuclei as we are only considering the electronic problem. The first term is the kinetic energy operator, the second term describes the interaction of the electrons with the nuclei, and the last term describes the electron-electron interaction.

With electrons being fermions, we have to take antisymmetrization into account. Mathematically, this can be achieved by writing the wavefunction as a determinant, which ensures parity change under exchange of two rows[1]. For N electrons this can be written as:

$$\Psi(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \frac{1}{\sqrt{N!}}\begin{vmatrix} \chi_1(\mathbf{x}_1) & \chi_2(\mathbf{x}_1) & \cdots & \chi_N(\mathbf{x}_1) \\ \chi_1(\mathbf{x}_2) & \chi_2(\mathbf{x}_2) & \cdots & \chi_N(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \chi_1(\mathbf{x}_N) & \chi_2(\mathbf{x}_N) & \cdots & \chi_N(\mathbf{x}_N) \end{vmatrix} \tag{2.4}$$

To write down this so-called Slater determinant we have introduced the notation $\mathbf{x}_i = (\mathbf{r}_i, \omega_i)$ with ω_i being the spin coordinate of electron i. The functions χ_i are called spin orbitals and are a product of spatial and spin functions.

$$\chi_i(\mathbf{x}) = \sum_\nu C_{\nu i}\,\chi_\nu(\mathbf{x}) = \sum_\nu C_{\nu i}\,\phi_\nu(\mathbf{r})\,\sigma(\omega), \quad \langle\chi_i|\chi_j\rangle = \delta_{ij}, \quad \sigma(\omega) = \begin{cases} \alpha(\omega) & \text{if } \omega = +\frac{1}{2} \\ \beta(\omega) & \text{if } \omega = -\frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

Here, spin functions use the common notation $\alpha(\omega)$ and $\beta(\omega)$ for spin up and down respectively. Using this expansion, we reduce the problem to the determination of the coefficients $C_{\nu i}$ for a given set of orthogonal basis functions $\phi_\nu(\mathbf{r})$. For a complete set of basis functions this expansion is exact. However, in practice, we have to limit the number of basis functions and hence the size of our Slater determinant to a computationally feasible size. For the subsequent considerations we introduce the shorthand ket for a slater determinant $|\Psi\rangle = |\chi_1, \chi_2, \ldots, \chi_N\rangle$ or $|\Psi\rangle = |1, 2, \ldots, N\rangle$ for the sake of brevity.

A slater determinant is one simple way of describing the state of a system as a linear combination of basis functions. From that we can derive the energy of the given systems state E using

$$E[\Psi] = \langle\Psi|\,H\,|\Psi\rangle\,. \tag{2.6}$$

For our considerations we will be interested in the ground state of the system. This means we need to find the minimum of our energy functional $E[\Psi]$ and its corresponding slater determinant

[1] Furthermore, if two electrons occupy the same orbital the determinant will vanish.

$|\Psi\rangle$, which marks our approximation to the ground state wavefunction. This is done via the variational principle, which yields an eigenvalue problem,

$$F\,|\Psi_i\rangle = \varepsilon_i\,|\Psi_i\rangle\,, \tag{2.7}$$

called the Hartree-Fock (HF) equations. Here ε_i corresponds to the energy of the i-th spin-orbital $|\Psi_i\rangle$. The operator F is called the Fock operator and is defined as an effective one electron operator acting on electron coordinate i:

$$F(i) = \underbrace{-\frac{1}{2}\nabla^2 - \sum_{A=1}^{M}\frac{Z_A}{|\mathbf{r}-\mathbf{R}_A|}}_{\text{core Hamiltonian } h(i)} + \underbrace{\sum_{j\neq i} J_j(i) - K_j(i).}_{\text{HF potential } v^{HF}(i)} \tag{2.8}$$

While the core Hamiltonian $h(i)$ contains the well-known kinetic and nuclear attraction terms, the Hartree-Fock potential $v^{HF}(i)$ condenses all electron-electron interaction into a mean-field effect on electron i using the Coulomb and exchange terms,

$$J_j(i)\chi_i(\mathbf{x}_1) = \left[\int d\mathbf{x}_2\,\frac{|\chi_j(\mathbf{x}_2)|^2}{r_{12}}\right]\chi_i(\mathbf{x}_1), \tag{2.9a}$$

$$K_j(i)\chi_i(\mathbf{x}_1) = \left[\int d\mathbf{x}_2\,\frac{\chi_j^*(\mathbf{x}_2)\chi_i(\mathbf{x}_2)}{r_{12}}\right]\chi_j(\mathbf{x}_1). \tag{2.9b}$$

Here, $J_j(i)$ is the Coulomb operator, representing the classical electrostatic repulsion, and $K_j(i)$ is the exchange operator, arising from the antisymmetry of the wavefunction.

While Equation 2.7 seems like an ordinary eigenvalue problem, the χ terms in J and K depend on our slater determinant wavefunction $|\Psi\rangle$. Thus, to get the ground state wavefunction, we need the corresponding Fock operator F, which in turn depends on the ground state wavefunction.

2.1.2. Matrix formulation of the Hartree-Fock equations

To solve the self-dependence of the Hartree-Fock equations, we need to iteratively update our wavefunction until convergence is reached. To achieve this we first need to express our eigenvalue problem in a way that can be solved numerically. Assuming each spin up ($\uparrow$) and spin down ($\downarrow$) electron pair occupies the same spatial orbital (referred to as restricted HF), we can write the HF-equations for electron 1 as follows:

$$F(\mathbf{x}_1)\chi_i(\mathbf{x}_1) = \varepsilon_i\chi_i(\mathbf{x}_1) \equiv F(\mathbf{x}_1)\psi_i(\mathbf{r}_1)\sigma_i(\sigma_1) = \varepsilon_i\psi_i(\mathbf{r}_1)\sigma_i(\sigma_1). \tag{2.10}$$

With the spins integrated out, we obtain the HF-equations for spatial orbitals,

$$F(\mathbf{r}_1)\psi_i(\mathbf{r}_1) = \varepsilon_i\psi_i(\mathbf{r}_1), \tag{2.11}$$

which differs from Equation 2.7 in the fact that we have integrated out the spin degrees of freedom and thus need to sum over half the electrons (i.e. $\uparrow$) and double the result, reflecting the closed-shell assumption that all spins are paired. With this, the Fock operator takes the form

$$F(\mathbf{r}_1) = h(\mathbf{r}_1) + 2\sum_{j\neq i}^{N/2}\left(J_j(\mathbf{r}_1) - K_j(\mathbf{r}_1)\right). \tag{2.12}$$

We need to represent the spatial orbitals in a finite, computer-readable way. While the spatial orbitals in Equation 2.5 are exact for a complete basis set, we will limit ourselves to a finite set of known basis functions, which will approximate the spatial orbitals by a linear combination.

$$\psi_i(\mathbf{r}) \approx \sum_{\nu}^{K} C_{\nu i}\phi_\nu(\mathbf{r}) \tag{2.13}$$

Possible ways to construct a finite set of basis functions $\phi_\nu(\mathbf{r})$ to represent the spatial orbitals $\psi_i(\mathbf{r})$ are introduced in Subsection 2.1.6.
Given this finite set of equations, we can now find an expression in matrix form to make it suitable for numerical solution. Substituting Equation 2.13 into Equation 2.11 yields

$$F(\mathbf{r_1})\sum_{\nu}^{K} C_{\nu i}\phi_\nu(\mathbf{r_1}) = \varepsilon_i \sum_{\nu}^{K} C_{\nu i}\phi_\nu(\mathbf{r_1}). \tag{2.14}$$

We multiply both sides by $\phi^*_\mu(\mathbf{r_1})$ and integrate over the spatial coordinate $\mathbf{r_1}$ to get

$$\sum_{\nu}^{K} C_{\nu i} \underbrace{\int d\mathbf{r_1}\phi^*_\mu(\mathbf{r_1})F(\mathbf{r_1})\phi_\nu(\mathbf{r_1})}_{F_{\mu\nu}} = \varepsilon_i \sum_{\nu}^{K} C_{\nu i} \underbrace{\int d\mathbf{r_1}\phi^*_\mu(\mathbf{r_1})\phi_\nu(\mathbf{r_1})}_{S_{\mu\nu}}. \tag{2.15}$$

Now we define the matrix elements for the Fock matrix $\mathbf{F}$ and the overlap matrix $\mathbf{S}$ using

$$F_{\mu\nu} = \int d\mathbf{r_1}\phi^*_\mu(\mathbf{r_1})F(\mathbf{r_1})\phi^*_\nu(\mathbf{r_1}), \tag{2.16a}$$

$$S_{\mu\nu} = \int d\mathbf{r_1}\phi^*_\mu(\mathbf{r_1})\phi^*_\nu(\mathbf{r_1}), \tag{2.16b}$$

so that the eigenvalue problem writes:

$$\sum_{\nu}^{K} F_{\mu\nu}C_{\nu i} = \varepsilon_i \sum_{\nu}^{K} S_{\mu\nu}C_{\nu i}, \tag{2.17a}$$

$$\mathbf{FC} = \mathbf{SC}\boldsymbol{\varepsilon}. \tag{2.17b}$$

In Equation 2.17b $\boldsymbol{\varepsilon}$ is a diagonal matrix with the orbital energies ε_i on the diagonal and $\mathbf{C}$ is the matrix of the expansion coefficients $C_{\nu i}$. This equation is called the Roothaan equations and can be solved iteratively.
Yet, we still are missing a representation of the density in terms of the expansion coefficients $C_{\nu i}$ and the basis functions $\phi_\nu(\mathbf{r})$. The probability to find an electron at a certain position $\mathbf{r}$ is given by the square of the wavefunction $\psi_i(\mathbf{r})$ and thus the electron density $\rho(\mathbf{r})$ can be written as

$$\rho(\mathbf{r}) = 2\sum_{i=1}^{N/2}|\psi_i(\mathbf{r})|^2 = 2\sum_{i=1}^{N/2}\sum_{\mu,\nu}^{K} C_{\mu i}C^*_{\nu i}\phi_\mu(\mathbf{r})\phi^*_\nu(\mathbf{r}) = \sum_{\mu,\nu}^{K} P_{\mu\nu}\phi_\mu(\mathbf{r})\phi^*_\nu(\mathbf{r}). \tag{2.18}$$

Here we have used Equation 2.13 to express the spatial orbitals in terms of the basis functions[2] and defined the density matrix $\mathbf{P}$ with matrix elements

$$P_{\mu\nu} = 2\sum_{i=1}^{N/2} C_{\mu i}C^*_{\nu i}. \tag{2.19}$$

[2]We still need to multiply by two in order to account for spin up and spin down.

As we have seen earlier, the Fock operator F and thus the Fock matrix $\mathbf{F}$ depend on the expansion coefficients $\mathbf{C}$ and thus the Fock matrix also depends on the density matrix $\mathbf{F}(\mathbf{P})$. If we insert Equation 2.12 into Equation 2.16a and write out Coulomb and exchange terms we have:

$$
\begin{aligned}
F_{\mu\nu} = & \underbrace{\int d\mathbf{r}_1 \, \phi_\mu^*(\mathbf{r}_1) \, h(\mathbf{r}_1) \, \phi_\nu(\mathbf{r}_1)}_{H_{\mu\nu}^{\text{core}}} \\
& + \sum_{\lambda\sigma} \mathbf{P}_{\lambda\sigma} \Bigg[\underbrace{\iint d\mathbf{r}_1 d\mathbf{r}_2 \, \phi_\mu^*(\mathbf{r}_1)\phi_\nu(\mathbf{r}_1) r_{12}^{-1} \phi_\sigma^*(\mathbf{r}_2)\phi_\lambda(\mathbf{r}_2)}_{\text{Coulomb-term}} \\
& - \frac{1}{2} \underbrace{\iint d\mathbf{r}_1 d\mathbf{r}_2 \, \phi_\mu^*(\mathbf{r}_1)\phi_\lambda(\mathbf{r}_1) r_{12}^{-1} \phi_\sigma^*(\mathbf{r}_2)\phi_\nu(\mathbf{r}_2)}_{\text{Exchange-term}} \Bigg]
\end{aligned}
\tag{2.20}
$$

One and two electron contributions are now split into the core Hamiltonian $H_{\mu\nu}^{\text{core}}$ term, which is independent of the density matrix and thus stays constant for a given set of basis functions, and the Coulomb and exchange terms that depend on the density matrix $\mathbf{P}$. The latter terms are often denoted as $G_{\mu\nu}$, the so-called two-electron integrals, hence $F_{\mu\nu} = H_{\mu\nu}^{\text{core}} + G_{\mu\nu}$.

2.1.3. The Self-Consistent Field (SCF) Algorithm

We now have all ingredients to solve Equation 2.17b iteratively. The self-consistent field (SCF) algorithm is a fixed-point iteration method: it starts from an initial density guess and then alternates between building the Fock matrix and updating the density until convergence. To generate the initial guess, we must supply the molecular geometry $\{\mathbf{R}_A, Z_A\}$, the total electron count N, and a chosen basis set $\{\phi_\nu\}$ to represent our orbitals.

Initially, we calculate the core Hamiltonian $H_{\mu\nu}^{\text{core}}$ and the overlap matrix $S_{\mu\nu}$ once (they will be reused in every step) and we guess an initial density. The guess of this initial density is crucial and has a big effect on the convergence of the algorithm. Then we start the loop:

1. Calculate $G_{\mu\nu}$ using the current $\mathbf{P}$.
2. Build a new $\mathbf{F}$ using $H_{\mu\nu}^{\text{core}}$ and $G_{\mu\nu}$.
3. Solve the generalized eigenvalue problem[3] to obtain a new $\mathbf{C}$ and $\boldsymbol{\varepsilon}$.
4. Obtain a new density matrix $\mathbf{P}$ from the new $\mathbf{C}$.
5. Total energy or $\mathbf{P}$ converged? If not, repeat.

Computationally, the calculation of the two-electron integrals $G_{\mu\nu}$ is the most expensive step which scales with the number of basis functions as $\mathcal{O}\left(K^4\right)$. Given this fact, a small enough basis set that still captures the essence of the physical system, should be chosen and the number of

[3] For numerical stability one usually converts to a standard eigenproblem using canonical orthonormalization: diagonalize $\mathbf{S} = \mathbf{U}\, s\, \mathbf{U}^T$, form $\mathbf{X} = \mathbf{U}\, s^{-1/2}$, solve the standard problem $\mathbf{X}^T \mathbf{F}\, \mathbf{X}\, \mathbf{C}' = \mathbf{C}'\, \boldsymbol{\varepsilon}$, and recover $\mathbf{C} = \mathbf{X}\, \mathbf{C}'$.

iterations shall be kept to a minimum by a good initial guess. Keeping our approximations of the spatial orbitals via Equation 2.13 in mind, we see that better approximations of the electronic wavefunction and its corresponding energy can be achieved with larger basis sets. Given the drawback of increased computational cost (mostly in the first step of the SCF algorithm), a reduction of iterations is even more crucial for larger systems and larger basis sets.

2.1.4. Guessing schemes

As eluded to earlier, our algorithm needs an initial density matrix **P** to start the SCF iteration. Over the years various schemes have been developed and implemented in quantum chemistry packages. It is instructive to introduce two commonly used schemes here. For sake of brevity, details regarding initial guess strategies in PySCF can be found in Appendix A.

Superposition of Atomic Densities (SAD) / Minimal Atomic Orbital (MINAO)
This scheme builds on atomic HF calculations, which are performed for each element appearing in a given molecule. With very little overhead, these calculations yield initial atomic densities, usually calculated in a minimal atomic basis, which are expanded into the molecular basis and used to build an initial block diagonal density matrix. After calculating the Fock matrix from this density, one diagonalizes it to obtain the molecular orbitals. Although the initial density matrix is strictly block-diagonal, diagonalizing the Fock matrix naturally mixes atomic orbitals and generates non-zero off-block elements. [6]

Generalized Wolfsberg-Helmholtz (GWH)
A semi-empirical approach is given by the generalized Wolfsberg-Helmholtz method. It approximates the off-diagonal elements of the core Hamiltonian from the diagonal values, weighting them according to orbital overlap,

$$F_{ij} \approx H_{ij}^{\text{core}} = \frac{K}{2}(H_{ii}^{\text{core}} + H_{jj}^{\text{core}})S_{ij}, \tag{2.21}$$

where K is a constant usually set to 1.75. From Equation 2.20 it is immediately evident that this approximation completely omits Coulomb and exchange terms, reducing the Fock matrix to core-Hamiltonian integrals evaluated with the same orbitals on both sides. [7, 8]

2.1.5. Expectation values

Several quantities can be derived from the solution of the Hartree-Fock equations. As eluded to in Equation 2.7, the eigenvalues of F are treated as orbital energies ε_i. The lowest $N/2$ eigenvalues correspond to the occupied orbitals, while the rest are so-called virtual orbitals [4]. One might be tempted to simply sum over the occupied orbital energies to obtain the total energy of the system. However, this leads to a double counting of the energy given by the exchange interaction for electron pairs. The correct electronic energy needs to be evaluated by calculating

$$E_0^{\text{HF}} = \left\langle \Psi_0^{\text{HF}} \middle| H \middle| \Psi_0^{\text{HF}} \right\rangle, \tag{2.22}$$

[4] We still assume closed shell systems with equal number of spin up and spin down electrons.

with Ψ_0^{HF} denoting the HF ground state wavefunction built from the $N/2$ lowest energy orbitals. Furthermore, the orbital energies can be used to estimate ionization energies in a frozen shell model, as stated by Koopmans' theorem. [9] Removing an electron from an occupied orbital χ_i will lead to an increase in energy of ε_i.

By inserting F for the Hamiltonian in Equation 2.6, one directly derives the total energy of the closed-shell ground state:

$$E_0^{\mathrm{HF}} = \left\langle \Psi_0^{\mathrm{HF}} \right| H \left| \Psi_0^{\mathrm{HF}} \right\rangle = 2\sum_{i=1}^{N/2} \langle \chi_i | \, h(i) \, | \chi_i \rangle + \frac{1}{2}\sum_{i=1}^{N/2}\sum_{j=1}^{N/2} \left[2J_j(i) - K_j(i)\right] \tag{2.23a}$$

$$\implies E_0^{\mathrm{HF}} = \sum_{i=1}^{N/2} \left[h(i) + \varepsilon_i\right] \text{ with } \varepsilon_i = F_i = h(i) + \sum_{j}^{N/2} \left[2J_j(i) - K_j(i)\right] \tag{2.23b}$$

$$\implies E_0^{\mathrm{HF}} = \frac{1}{2}\sum_{\nu,\mu} P_{\mu\nu}\left[H_{\mu\nu}^{\mathrm{core}} + F_{\mu\nu}\right] = \frac{1}{2}\mathrm{Tr}\left(\mathbf{P}\left[\mathbf{H}^{\mathrm{core}} + \mathbf{F}\right]\right) \tag{2.23c}$$

To get the total energy of the system, one adds the nuclear repulsion energy

$$E_{\mathrm{nuc}} = \sum_{A<B} \frac{Z_A Z_B}{|\mathbf{R}_A - \mathbf{R}_B|} \tag{2.24}$$

to the electronic energy E_0^{HF}, yielding the total energy of the system. Additionally, particularly from a chemical point of view, the number of electrons assigned to each atom is of interest, since partial charges can be derived from it. Given the fact that the density matrix represents spatial orbitals in the atom centred basis set, we can interpret $(\mathbf{PS})_{\mu\mu}$ as the population of electrons on orbital μ. This leads to a possible definition of a partial charge at atom A,

$$q_A = Z_A - \sum_{\mu \in A} (\mathbf{PS})_{\mu\mu}, \tag{2.25}$$

with index μ running over all basis functions located at atom A. This represents one of several approaches to population and charge analysis, commonly referred to as Mulliken population analysis. [10]

2.1.6. Basis sets

For computational treatment, our abstract basis functions $\phi_\nu(\mathbf{r})$ need to manifest in a concrete form. Motivated by the analytical solution of the Hydrogen atom, one can introduce so-called Slater-type orbitals (STOs) using an exponential radial term and spherical harmonics: [5]

$$\phi_{n,l,m}^{\mathrm{STO}}(r, \Omega, \zeta) = N Y_{l,m}(\Omega) r^{n-1} e^{-\zeta r} \tag{2.26}$$

Computationally, the integrals that occur by using linear combinations of STOs are expensive to calculate.

[5] For sake of clarity the spherical harmonics term is written with Ω instead of θ & φ.

Gaussian-Type Orbitals (GTOs)
To circumvent this problem, one approximates STOs using contractions of Gaussian-type orbitals (GTOs), which have a Gaussian $e^{-\zeta r^2}$ in place of the Slater exponential $e^{-\zeta r}$. A GTO is defined as:

$$\phi^{\mathrm{GTO}}_{n,l,m}(r,\Omega,\zeta) = NY_{l,m}(\Omega)r^{n-1}e^{-\zeta r^2} \tag{2.27}$$

A linear combination of N GTOs to approximate a STO is referred to as a contracted Gaussian function (CGF). We choose the contraction coefficients c_i and the exponents ζ_i such that the CGF approximates the STO as closely as possible.

$$\phi^{\mathrm{STO}}_{n,l,m}(r,\Omega,\zeta) \approx \phi^{\mathrm{CGF}}_{n,l,m}(r,\Omega,\zeta) = \sum_{i=1}^{N} c_i \phi^{\mathrm{GTO}_i}_{n,l,m}(r,\Omega,\zeta_i) \tag{2.28}$$

Many basis sets with varying number of CGFs have been developed over the years, some common ones from the STO-nG family include STO-3G, STO-6-31G or STO-6-31G(2df,p). The n in this nomenclature of Pople type basis sets [11] denotes the number of GTOs used in the contraction. While STO-3G uses three GTOs for each orbital, STO-6-31G uses six GTOs for the inner shell and two functions with three and one GTO respectively for the outer shell. Basis sets using more than one contracted function (31 in 6-31G) are so-called double-zeta basis sets; in this particular case it is a split-valence double zeta basis set because the core orbitals (6 in 6-31G) are modelled with only one contraction. The notation in parentheses denotes additional functions, e.g. STO-6-31G(2df,p)[6] uses two additional d-functions and one f-function on heavy atoms (C, O, N & F).

Segmented Contraction
Generally, the GTOs used in the contraction of Pople type basis sets are not reused and thus of the segmented contraction type. In contrast, general contraction type basis sets (such as cc-pVXZ [12]) reuse some or all primitives in multiple contractions. The latter can be converted to a computationally more efficient segmented form, which retains the initial accuracy. One such modern segmented basis family optimized for density functional theory are the pcseg-n basis sets. [13]

Accuracy & Hartree Limit
Expanding the spatial orbitals using basis functions introduces an error in the electron density distribution and subsequently the converged Energy E_{HF}. Given the limitation of a longer runtime-scaling with $\mathcal{O}\left(K^4\right)$ in the number of basis functions, the accuracy can be refined using a larger and more complete basis set. The so-called Hartree limit constitutes the lowest energy asymptotically attainable by any HF-calculation, achievable only with an infinitely large basis set. This can be estimated by the shrinking differences of energies obtained in a series of calculations with basis sets of increasing size. [14] Yet, it still lacks the correlation energy of electrons, and is therefore only of academic interest. We will discuss methods that build or improve on Hartree Fock next.

[6] also referred to as 6-31G(2df,p) in this thesis

2.1.7. Post-Hartree-Fock Methods

HF theory provides a solid foundation to solve the many-electron problem and is usually used as the starting point for following improvements. These are generally bundled under the umbrella term of post-Hartree-Fock methods.

Configuration Interaction (CI)
Configuration Interaction (CI) improves upon the Hartree-Fock approximation by expressing the many-electron wavefunction as a linear combination of multiple Slater determinants, including excited configurations. While HF provides the best single-determinant approximation to the ground state, it neglects electron correlation. CI systematically includes this by mixing determinants generated from excitations out of the HF reference, thus recovering the missing correlation energy. Combinatorially, for a system with $2K$ spin orbitals and N electrons, the number of possible determinants grows as $\binom{2K}{N}$, which makes CI calculations, using all possible determinants (full CI) intractable for all but the smallest molecular systems. The number of determinants is usually limited by calculating only single and double excitations (CISD), resulting in a time complexity of $\mathcal{O}\left(K^6\right)$.

Coupled Cluster (CC)
Truncated CI methods suffer from the lack of size consistency. This is repaired by Coupled Cluster (CC) theory, which uses an exponential ansatz of the form $|\Psi\rangle = e^T |\phi_0\rangle$, where T is a cluster operator typically truncated to single and double excitations: $T = T_1 + T_2$. Although only low-rank excitation operators are included, the exponential generates higher excitations implicitly as products of lower excitations. CCSD (Coupled Cluster Singles and Doubles) is the most common variant and scales with $\mathcal{O}\left(K^6\right)$ in computational cost. Extensions like CCSD(T), which include perturbative triples, increase the scaling to $\mathcal{O}\left(K^7\right)$.

Møller-Plesset Perturbation Theory (MP)
Møller-Plesset Perturbation Theory (MP) partitions the total energy into a zeroth-order Hartree-Fock contribution, with the Fock operator as H_0, and perturbative corrections based on the difference $H - H_0$. Within this formalism, HF energy appears as a first order correction, while the second-order term (MP2) provides the leading contribution and captures a good part of the electron correlation. MP2 scales with $\mathcal{O}\left(K^5\right)$.

2.1.8. Density Functional Theory (DFT)

Density funcitonal theory (DFT) takes an alternative approach to the many-electron problem. It uses a system of non-interacting electrons to approximate the actual many-electron interacting system. The formulation based on the Hohenberg-Kohn theorems [15] states that the ground state energy of a many-electron system is a functional of the electron density $\rho(\mathbf{r})$. Contrary to HF-theory, this functional $G[\rho] = T[\rho] + E_{\text{ex + corr}}[\rho]$ encompasses kinetic energy as well as both exchange and correlation energy. [16]
While the exact functional cannot be attained (with the exception of free-electron gas), functionals can be constructed via fitting methods to experimental data or higher level methods. In its most common implementation, Kohn-Sham DFT, one again solves for the atomic orbitals in an analogous way by minimizing the energy under the constraint of orthogonal orbitals in an iterative, self-consistent manner. In practice, one usually combines multiple different sources such

as fitted functionals, corrections and exact exchange contributions from HF. Coefficients for this combinations are fitted and can often be altered in computational chemistry packages. Common choices of DFT functionals grouped by level of accuracy on Perdew's famous Jacob's Ladder [17] include:

1. **LDA (Local Density Approximation)**
   Assumes that the exchange-correlation energy depends only on the local electron density, as in a uniform electron gas.
   Example: SVWN (Slater exchange + VWN correlation) [18, 19]

2. **GGA (Generalized Gradient Approximation)**
   Incorporates both the local density and its gradient.
   Example: PBE (Perdew-Burke-Ernzerhof) [20]

3. **Meta-GGA**
   Adds kinetic energy density or Laplacian of the density.
   Example: SCAN (Strongly Constrained and Appropriately Normed) [21]

4. **Hybrid-GGA**
   Mixes exact (Hartree-Fock) exchange with GGA exchange.
   Example: B3LYP (Becke3 + LYP correlation) [22, 23]

5. **Double Hybrids**
   Adds second-order perturbation theory (like MP2) to hybrid functionals; uses occupied and unoccupied orbitals.
   Example: B2PLYP (Becke2 + LYP + MP2 correction) [24]

DFT accuracy is limited by the choice of functional; standard Kohn-Sham DFT is bound by a $\mathcal{O}\left(K^3\right)$ scaling (rung 1-3) with the number of basis functions. In practice, commonly used functionals of rung four or five on Jacob's ladder are bound by a $\mathcal{O}\left(K^4\right)$-$\mathcal{O}\left(K^6\right)$ scaling due to the need for exchange-correlation integrals.

2.2. Machine Learning

Machine Learning (ML) constitutes a rich subfield of computer science developing algorithms capable of inferring patterns from data and making predictions without being explicitly programmed with deterministic instructions.
The history of ML closely coincides with the development of computers. In the 1950s, Arthur Lee Samuel coined the term "machine learning" while programming a computer to play checkers, eventually beating the 4th ranked player in the US at the time. [25] Since it's humble beginnings, ML has influenced many scientific fields as well as everyday life.

In this section, a brief overview will be given of the most important concepts in ML with special emphasis on techniques used in this thesis. The books written by Bishop [26] and Goodfellow [27] provide a more in-depth treatment of the various subjects covered in the following introduction.

2.2.1. Classification & General Remarks

Machine learning itself is a subfield of artificial intelligence (AI) research, which is the broader field of creating machines that can perform tasks that would normally require human-level intelligence. Central to ML is the concept of learning from data. This means that we need to define data given as input to predict a target output. More broadly speaking, we can either predict a discrete class label (classification) or a continuous value (regression) given input data. The focus will be set on the latter task, in particular on predicting a continuous value (Fock/density matrix entries) given a set of continuous input values of substantially different character (atomic coordinates, overlap matrix elements, …).
The various approaches can also be grouped by learning paradigms: supervised, unsupervised, or reinforcement. Supervised models learn a function that maps input data to labelled target values. Unsupervised models, on the other hand, seek patterns or structure in unlabelled data. Reinforcement learning refers to models that learn a policy through interaction with an environment by maximizing a reward signal over time.

For the purpose of this thesis, we will focus on supervised models applied to our regression task. For consistency in the following sections, we introduce the general notation for our model as follows:

$$\mathbf{y} = f(\mathbf{x}) + \Delta, \tag{2.29}$$

with the input vector denoted as $\mathbf{x} = (x_1, x_2, \dots)$ and the target or output vector $\mathbf{y} = (y_1, y_2, \dots)$, which is not necessarily of the same size as the input vector. The model f represents the learned mapping from $\mathbf{x}$ to $\mathbf{y}$ with a certain error Δ[7].

[7] This error includes both reducible components—such as discretization error, which may be minimized by refining the DFT grid—and irreducible components, which stem from fundamental uncertainty in quantum mechanical systems.

2.2.2. Loss Functions

The loss function, also referred to as the cost function, quantifies the quality of the model's predictions by measuring the discrepancy between the predicted and true values. It may be motivated by system specific attributes, as is the case for F-score or DIIS error, or provide a general-purpose, model-agnostic measure of prediction accuracy, such as the mean squared error (MSE) or its root (RMSE).

F-score
One way of measuring the closeness of a guessed density $\psi^{\sigma}_{\text{guess}}$ to the converged density $\psi^{\sigma}_{\text{SCF}}$ can be defined as the overlap of same spin (σ) orbitals in Equation 2.30,

$$f = \frac{\sum\limits_{\sigma} Q^{\sigma}}{\sum\limits_{\sigma} N^{\sigma}_{\text{occ}}}, \tag{2.30a}$$

$$\text{with} \quad Q^{\sigma} = \sum_{i,j=1}^{N^{\sigma}_{\text{occ}}} \left|\left\langle \psi^{\sigma}_{i,\text{guess}} \middle| \psi^{\sigma}_{j,\text{SCF}} \right\rangle\right|^2 = \text{Tr}\left(\mathbf{P}^{\sigma}_{\text{guess}}\, \mathbf{S}\, \mathbf{P}^{\sigma}_{\text{SCF}}\, \mathbf{S}\right), \tag{2.30b}$$

using the notation introduced in Section 2.1. The normalization in Equation 2.30a is introduced to translate the extensive measure of Equation 2.30b into an intensive quantity and thus attain comparability of differently sized systems. F-scores range from 0 to 1, with 1 indicating perfect overlap of the guessed and converged orbitals.

Mean Squared Error (MSE) / Root Mean Squared Error (RMSE)
For an abstract prediction matrix $\mathbf{Y}^{(\text{pred})}$, the MSE is simply defined by the arithmetic mean of the squared differences with the actual values $\mathbf{Y}^{(\text{truth})}$ as given in Equation 2.31a. F denotes the Frobenius norm in this case, and our matrices will usually be square matrices[8] $\rightarrow n \equiv m$.

$$\text{MSE} = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \left(Y_{ij}^{(\text{pred})} - Y_{ij}^{(\text{truth})}\right)^2 = \frac{1}{nm} \|\mathbf{Y}^{(\text{pred})} - \mathbf{Y}^{(\text{truth})}\|_F^2 \tag{2.31a}$$

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{2.31b}$$

Calculating the square root of the MSE gives the RMSE in Equation 2.31b. Besides the fact that RMSE offers the same unit as the predicted quantity, it offers no benefit and incurs additional computations. For this reason usage of MSE for model optimization is preferred.

DIIS error
A computationally more demanding score, the DIIS error, may also be used. In this thesis it will be defined as the root mean squared error of the commutator relation of Fock and density matrix. In the general case this writes:

$$\mathbf{E}_{\sigma} = \mathbf{F}_{\sigma}\mathbf{P}_{\sigma}\mathbf{S} - \mathbf{S}\mathbf{P}_{\sigma}\mathbf{F}_{\sigma}, \tag{2.32a}$$

$$\text{DIIS error} = \sqrt{\frac{\|\mathbf{E}_{\uparrow} + \mathbf{E}_{\downarrow}\|_F^2}{n_{\text{elements},\uparrow} + n_{\text{elements},\downarrow}}}, \tag{2.32b}$$

with a normalization factor $n_{\text{elements},\uparrow} + n_{\text{elements},\downarrow}$ to make different matrix sizes comparable. DIIS gives a measure of self-consistency with DIIS error $= 0$ (commuting $\mathbf{F}$&$\mathbf{P}$), denoting convergence.

[8] Only the GNN will use non-square sub-matrix blocks on the off-diagonals to evaluate its loss.

2.2.3. Regression Models

Regression models try to establish the mapping in Equation 2.29 directly by modelling the mapping using a linear combination of terms. In its simplest form, a linear regression in the multivariate multi-target case is given by

$$\begin{aligned} y_m &= w_{0,m} + \sum_{i=1}^{d} w_{i,m}\, x_i \quad \text{for } m = 1, \ldots, k, \quad \text{or} \\ \mathbf{y} &= \mathbf{W}^\top \mathbf{x} \quad \text{in matrix form.} \end{aligned} \tag{2.33}$$

Here, $w_{i,m}$ denotes the coefficients, also called weights in ML, to create a relation between feature x_i and target y_m. The zeroth weight, a linear offset, is referred to as 'bias' and is commonly introduced into the matrix notation by prepending a one in the input vector $\mathbf{x}$. Note that we have dropped the Δ term for compact notation. The weights are chosen to reduce the sum of squared errors, a procedure known as least squares (LS) fitting:

$$\text{Loss}_{\text{LS}}(\mathbf{W}) = \sum_{n=1}^{N} \|\mathbf{y}^{(n)} - \mathbf{W}^\top \mathbf{x}^{(n)}\|_2^2 \tag{2.34}$$

Weights represent the influence of a given feature x_i on the target y_m. Least squares fitting of these weights guarantees the minimization of the prediction error on the data the model was trained on; however, nothing can be said about 'unseen', i.e. new data. Especially for high dimensional data with correlations between input features this model is prone to overfitting, which may happen if the model is too flexible and captures noise in the data instead of actual patterns. We will introduce regularization into our problem to combat this issue.

Ridge Regression
There are multiple ways of introducing a regularization in order to keep weights small and limit overfitting of data. One simple way is to add a penalty term dependent on the size of the weight:

$$\text{Loss}_{\text{Ridge}}(\mathbf{W}) = \sum_{n=1}^{N} \|\mathbf{y}^{(n)} - \mathbf{W}^\top \mathbf{x}^{(n)}\|_2^2 + \lambda \|\mathbf{W}\|_2^2 \tag{2.35}$$

This loss function is used in Ridge Regression (RR) and penalizes large weights by adding a term proportional to the squared L_2 norm of the weights $\|\mathbf{W}\|_2^2$. Another common regularization technique is to use the L_1 norm of the weights without squaring them, which leads to Lasso Regression (LR).

Kernel Ridge Regression
Instead of a fixed weight matrix, Kernel Ridge Regression (KRR) uses a kernel function $k(\mathbf{x}, \mathbf{x}')$ to build the Gram matrix $\mathbf{K}_{ij} = k(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$. The dual coefficient matrix $\mathbf{A}$ is learned by minimizing

$$\text{Loss}_{\text{KRR}}(\mathbf{A}) = \|\mathbf{Y} - \mathbf{K}\mathbf{A}\|_F^2 + \lambda \,\text{Tr}(\mathbf{A}^\top \mathbf{K}\mathbf{A}). \tag{2.36}$$

We get $\mathbf{A} = (\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{Y}$ which is used to predict the target for a new input $\mathbf{x}^*$, via the similarity vector $\mathbf{k}(\mathbf{x}^*) = [\, k(\mathbf{x}^*, \mathbf{x}^{(1)}), \ldots, k(\mathbf{x}^*, \mathbf{x}^{(N)}) \,]^\top$,

$$f(\mathbf{x}^*) = \mathbf{k}(\mathbf{x}^*)^\top \mathbf{A}. \tag{2.37}$$

KRR thus extends ridge regression in a non-parametric way, with the regularization term $\lambda \mathbf{I}$ ensuring that $(\mathbf{K} + \lambda \mathbf{I})$ is invertible and numerically stable, while enabling the model to capture complex, nonlinear patterns at a computational cost of $\mathcal{O}\left(N^3\right)$ in the number of training samples N.

2.2.4. Neural Network (NN) concepts

Neural networks are based on a simplified model of biological neuron interactions. Rosenblatt's perceptron algorithm, depicted in Figure 2.1, lays the foundation for nearly all neural network architectures. [28] It introduced the nonlinear mapping from the features $\mathbf{x}$ to the targets $\mathbf{y}$ using a step function Θ as shown[9] in Equation 2.38.

$$y = \Theta(\mathbf{w} \cdot \mathbf{x}) \quad \text{with} \quad \Theta(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.38}$$

The function used to transform the weight-feature product is also commonly referred to as an 'activation' function. A step function gives a binary output and is thus most relevant in classification tasks. In the general case, Θ denotes an arbitrary activation function which may also be monotonic and continuous, contrary to the step function.

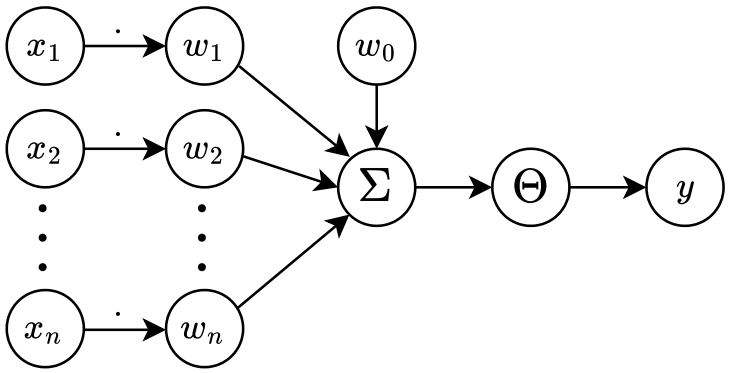

Figure 2.1.: Architecture of the perceptron algorithm. Features x_i are multiplied with their respective weights w_i summed together with the bias term w_0 and passed through an activation function Θ to produce the output y.

Functions mapping to values in the range $[0, 1]$, such as the step function or sigmoid functions, are used in classification tasks. Other (mostly) monotonic functions are used for activation in regression tasks. Due to their frequent invocation, computationally inexpensive functions like the Rectified Linear Unit (ReLU) are widely used in practice. The Gaussian Error Linear Unit (GeLU) converges to ReLU in the positive and negative limits, but smooths the transition around zero, ensuring differentiability over the entire domain. Some commonly used functions are depicted in Figure 2.2.

[9] Bias is again prepended to the weight vector

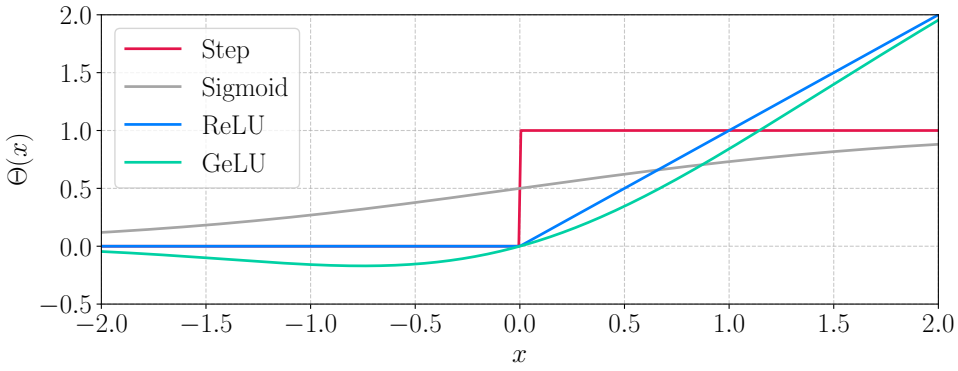

Figure 2.2.: Commonly used activation functions.

2.2.5. Feed forward Networks: Multilayer Perceptron (MLP)

Feed forward networks refer to a network architecture in which the outputs of the networks do not directly feed back into the inputs or any intermediate representations. Hence the input is exclusively fed forward through the network to produce the output. This way, there are no loops in the network's dataflow, allowing for training via the backpropagation algorithm. A Multilayer Perceptron (MLP) as shown in Figure 2.3 combines many perceptrons in layers. Data feeds into the input layer perceptrons which are each connected to every other perceptron in the first hidden layer. This connectivity is referred to as 'fully connected' or 'dense'. After several hidden layers, an output layer provides the output in the desired dimensionality and often without activation. Each connection is associated with a weight that must be learned to enable meaningful predictions on unseen data. This learning process relies on backpropagation, which applies the chain rule iteratively to compute the gradient of the loss with respect to each weight, propagating from the output layer backwards through the network. While data flows forward through the network during prediction, the resulting loss propagates backward, guiding how the weights are adjusted.

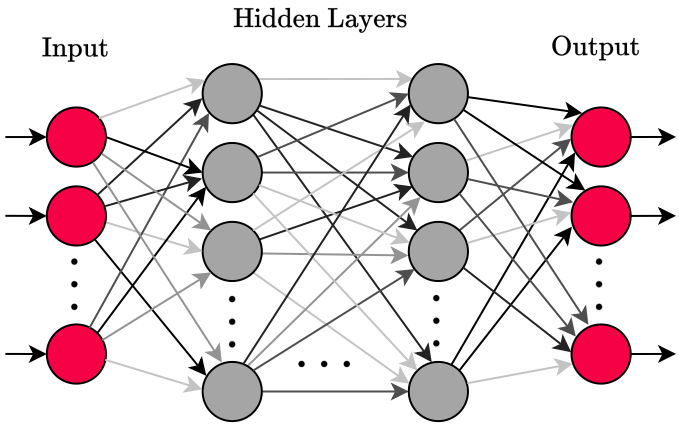

Figure 2.3.: Multilayer Perceptron with fully connected layers. Different shades of gray represent weights between perceptrons.

Similar to regression, regularization may also be added in various places. Besides L_1 or L_2 penalities, which can be easily added to the loss function, dropouts offer another, neural network-specific way of regularization. During training, it randomly sets outputs of perceptrons in the hidden layers to 0, thus deactivating the outgoing signal and reducing interdependence between perceptrons.

2.2.6. Graph Neural Network (GNN) / Message Passing Neural Nets (MPNN)

Many problems cannot be simply mapped to a flattened input vector without losing information that is important for accurate predictions. Within the context of atomistic theories, a natural representation e.g. for a molecule may be attained by using a graph consisting of atoms as the nodes and edges between these nodes to represent interaction (e.g. bonds) between atoms. Learning features on graphs using a Graph Neural Network (GNN) is exploited in various fields and across different scales, from large protein structurs in DeepMind's Alphafold [29] down to molecular features such as energy, polarization and charge density in computational quantum chemistry. For the latter, the term 'Message Passing Neural Nets' (MPNN) was coined by Glimer et al. in [30].

An abstract representation of a MPNN is depicted in Figure 2.4. Data, such as colour in the concrete example, is encoded in the nodes of a graph. For many use cases, separate encoder functions or even MLPs may be suitable for the encoding of various features into a feature vector for a given node or edge. For molecules, these might include atomic number, electronegativity or even overlap matrix elements. After initialization of the graph, the MPNN iteratively updates the nodes feature vector for several message passing rounds.

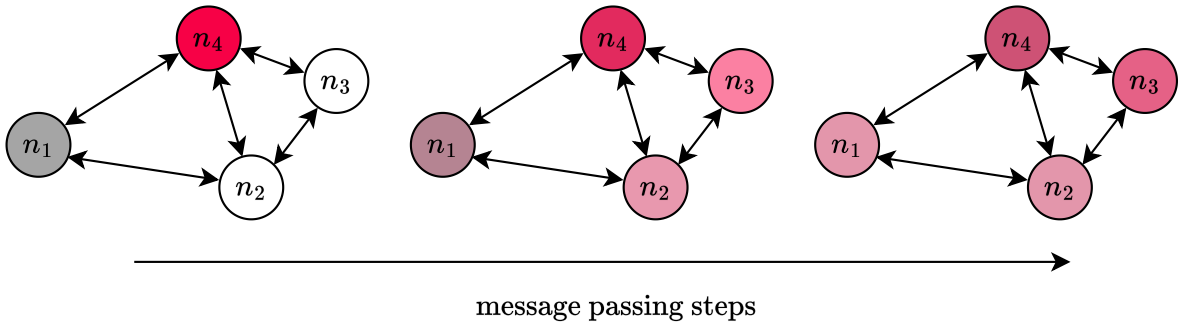

Figure 2.4.: Schematic representation of message passing in a bidirectionally connected graph. Node colours influence neighbours which aggregate neighbour colours and updates own colour via leaned update function each step.

For every round, all nodes send a message containing their feature vector (or a derived representation) to all their neighbours. The node n_1 in Figure 2.4 sends its colour to its neighbours n_2 and n_4. After all messages have been sent, every node aggregates the received message via a permutation invariant function. In the example, this is simply the average of the received colours. Finally, a learned update function changes the node's feature vector based on the aggregated message and the current feature vector. After several rounds of message passing, which allow information flow between nodes not directly connected to each other, the final graph representation can be used to predict various properties.

Decoders essentially invert the encoder's process to extract either node-level features (e.g. atomic energy, partial charge or node colour in Figure 2.4) or graph-level properties such as total energy, polarizability or electric dipole moment.

2.2.7. Neural Network Training and Hyperparameter Tuning

Training neural networks and tuning their hyperparameters are crucial steps to ensure models learn meaningful patterns and generalize to unseen data. This section outlines the standard training workflow and introduces two hyperparameter optimization methods employed in this thesis: Bayesian search and Hyperband.

Network Training
Training begins with the random initialization of network weights. Input data is processed in the network in batches (e.g. 16, 32 or 64 input samples). For each batch, the network performs a forward pass to generate predictions and compute a loss value using the loss function. Backpropagation then calculates gradients of the loss with respect to each weight in the network. The weights are updated using an optimization algorithm, which adjusts the weights in the direction that reduces the loss. This cycle (forward pass, loss computation, backpropagation) repeats for multiple epochs, where one epoch corresponds to one full pass through the training dataset.
Key hyperparameters influencing training dynamics and overall model performance include the learning rate, batch size, number of epochs, and network architecture: the number of layers, neurons per layer, and activation functions.

Hyperparameter Tuning
Hyperparameter tuning explores the space of settings to identify combinations that yield the best results. An exhaustive search over all possible combinations is typically infeasible and not possible for continuous-valued hyperparameters. Within this thesis, two approaches will be used: Bayesian search and Hyperband tuning.

Bayesian Search Tuning
Bayesian search constructs a surrogate probabilistic model mapping hyperparameters to performance. After evaluating a set of configurations, it updates this model to predict promising regions in hyperparameter space. By sampling where improvement is most likely, it outperforms random or grid search in efficiency and convergence speed.

Hyperband Tuning
Hyperband tuning methods combine the exploratory power of random search with early stopping to efficiently allocate resources only to promising hyperparameter configurations. It starts with many random configurations, trains each for a few epochs, and discards the worst performers. The top configurations receive additional resources in subsequent rounds. This process is repeated until a certain budget is reached or a maximum number of configurations have been evaluated. This optimization strategy can be fully parallelized, allowing independent evaluation of configurations and fast identification of strong candidates.

2.3. QM9 dataset

Dataset selection for training is dominated by two practical considerations:

1. Cost per sample: Time savings through faster convergence are especially relevant for larger systems where the number of SCF iterations and the number of integrals to be calculated are large. Stated differently, it is of little interest to optimize guessing methods for small systems that converge nearly instantly on conventional hardware.

2. Fixed I/O size: Most ML models are constrained to a constant input and output size.

The QM9 dataset [31, 32, 33] ticks both of these two boxes. It offers a variety of molecules from as little as three constituent atoms up to 29 atoms. Additionally, there are large enough chunks of constitutional isomers to train models on these subsets of same sized matrices. The distribution of molecules by atom count with the predominant constitutional isomers is shown in Figure 2.5.

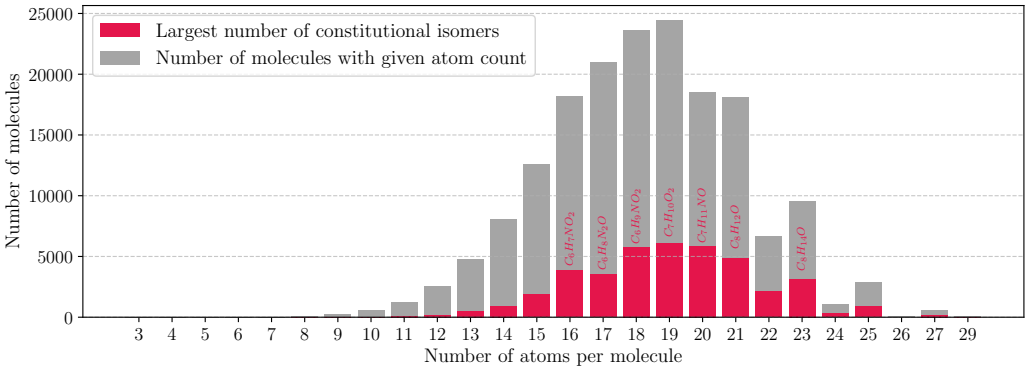

Figure 2.5.: Overview of the QM9 dataset. The dataset contains 134,000 molecules with up to nine second row elements (C, O, N, F). Large groups of constitutional isomers are present (largest depicted in red).

3. Fock Matrix prediction

SCF methods initially need a density matrix to start off their iterative calculations. Regardless of how the initial guess is actually generated, the computational effort of this step should be negligible compared to that of the SCF procedure.
In a first step, we introduce a method by which the density matrix is constructed from the Fock matrix. The latter is the target for various regression and MLP models which will be introduced in the subsequent sections.

3.1. Introduction

As explained in Subsection 2.1.2, the density matrix P is calculated from the coefficient matrix C, which is obtained from the eigenvalue problem of the Fock matrix F:

$$F(P)\,C = SC\varepsilon \rightarrow P = 2\sum_{i}^{N/2} C_{\mu i}\,C_{\nu i}^{*}. \tag{3.1}$$

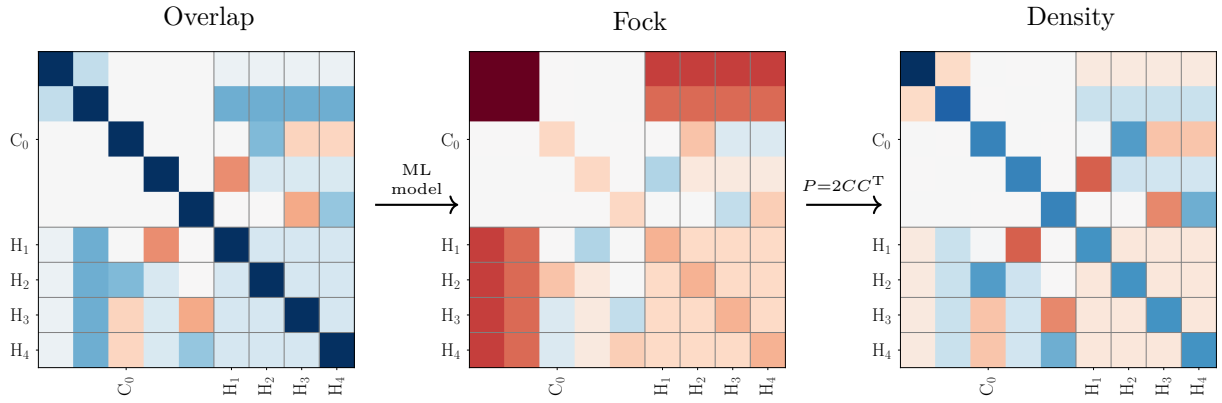

Figure 3.1.: Schematic overview of the data flow (using CH_4): overlap matrix to Fock matrix via ML model prediction, and construction of the density matrix from the Fock matrix.

Effectively, one performs a part of the SCF procedure, namely the guessing of the Fock matrix (step 2 in Subsection 2.1.3) using the ML model and then subsequently solves the eigenvalue problem to obtain the density matrix. This step takes $\mathcal{O}\left(K^3\right)$ time, which is asymptotically faster than a single SCF step scaling as $\mathcal{O}\left(K^4\right)$.

The schematic workflow for the generation of a density matrix from the overlap matrix and via the Fock matrix is shown in Figure 3.1. Overlap and density matrices appear visually similar,

which might lead to the conclusion that predicting the density matrix directly from the overlap matrix is easier than predicting the Fock matrix from the same. However, learning the Fock matrix involves fewer strict physical constraints. A Fock matrix primarily needs to be Hermitian, while a density matrix must be strictly positive semidefinite by construction, as well as normalized to the correct number of electrons. This makes direct density matrix learning more challenging, whereas learning the Fock matrix and then obtaining the density through diagonalization might yield better results. This hypothesis is tested in the following computer experiments.

Of further relevance is the fact that, reconstructing the density from the Fock matrix amplifies residual errors in the predicted Fock matrix, since the density arises from a self-consistent eigenproblem involving both F and S. Ideally, such errors should be small enough to not significantly affect the subsequent SCF iterations. In practice, a few iterations (2-5) are always needed, even in calculations with a guessed density constructed from the converged F & S matrices.

3.2. $C_5H_4N_2O_2$-subset: A first trial

As a proof of concept we will use 508 constitutional isomers[1] of $C_5H_4N_2O_2$ from the QM9 dataset. Single point RKS simulations in PySCF [34] were performed for these molecules using the B3LYP[2] functional and the STO-3G basis set. The resulting converged density, Fock, and overlap matrices were saved for future training purposes. For our very minimal basis, these matrices are of size 49×49 as can be seen for a converged density in Figure 3.2. Due to the symmetry of the matrices we can discard nearly half of the elements and are left with 1225 features to learn. A rule of thumb for training classical statistical models is to have at least ten samples per feature. [35] The given dataset of 508 samples is therefore far from this rule if we discard multiplicity in atoms in the samples. As a first step we shall limit our efforts to single models predicting the system based on the full overlap matrix. Nevertheless, the row/column order of our atoms in the overlap matrix has to be consistent for all the samples in the dataset (as seen in Figure 3.2). This is achieved by sorting the atoms in the overlap matrix according to their atomic number. Now we are ready to launch our endeavour with a ridge regression model as a first trial.

[1] From the 509 constitutional isomers of $C_5H_4N_2O_2$ 508 converged using the STO-3G basis set and the B3LYP functional.

[2] B3LYPG was used to be consistent with the Gaussian program package.

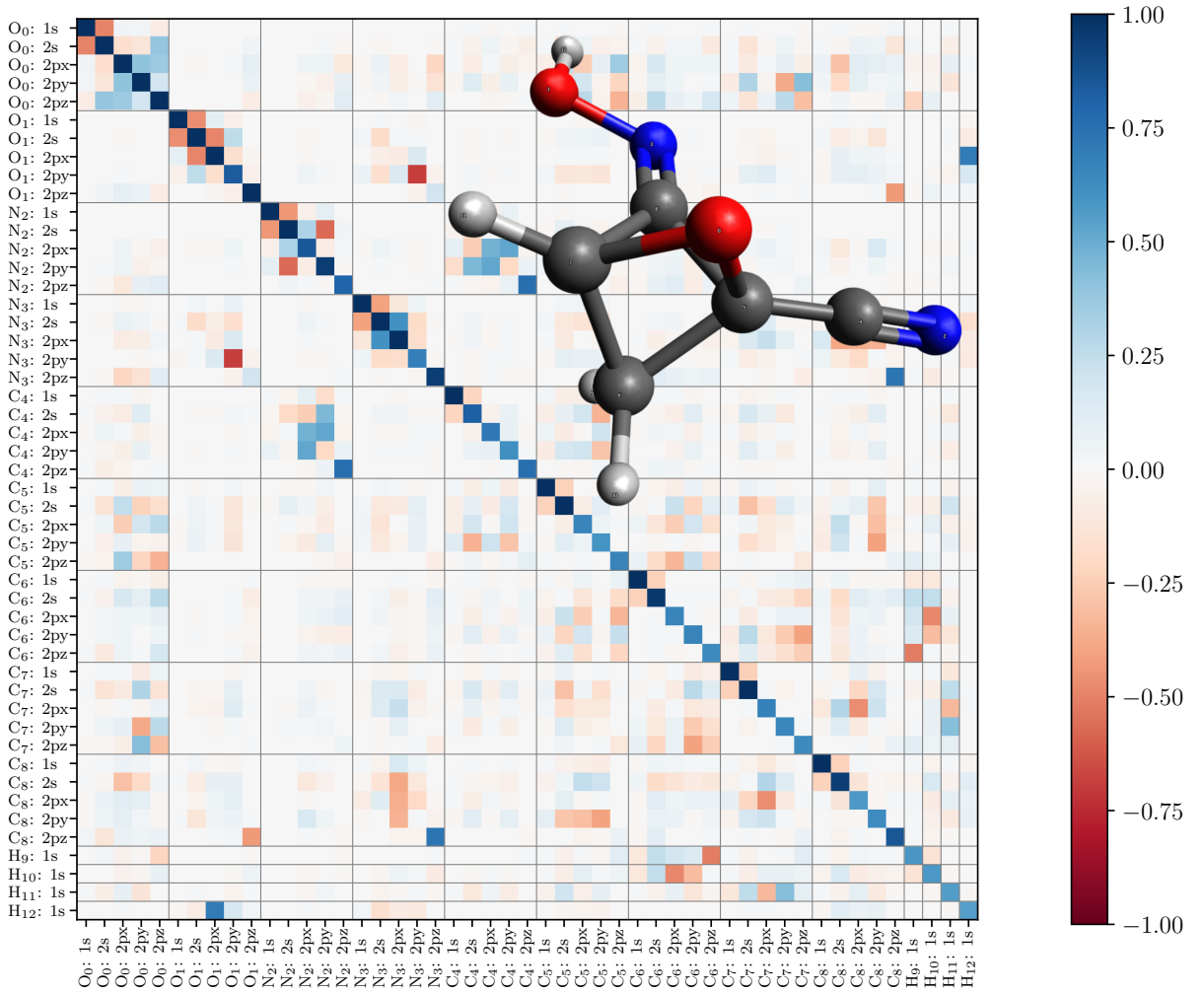

Figure 3.2.: Converged density of dsgdb9nsd_022700 in the STO-3G basis with theory level B3LYP. The density matrix is of size 49×49 and is symmetric. The diagonal elements in the shown AO basis correspond to Mulliken AO-populations.

3.3. Ridge Regressor model

The ridge Regressor (RR) is set up with a typical 80/20 split into training and testing data. Overlap (input) as well as Fock (output) matrices are flattened, and overlap (input) matrices are additionally rescaled using the SCIKIT-LEARN's default Standard Scaler. [36] Using a 5-fold cross validation the model, a Multi-Output-Regressor is trained[3] with ten equally $\log_{10}$-spaced L^2-regularization parameter α values ranging from 10^{-2} to 10^3. Subsequently, the model is re-trained with the arithmetic mean of the best performing α values ($\alpha_{\text{mean}} \approx 34$). This averaging reduces overfitting and yields a slightly lower RMSE on the test set compared to using individual α values per regressor. For the Fock matrix prediction the model yields a RMSE of 0.0057 on the training set and 0.032 on the test set which indicates overfitting of the model. We obtain the

[3] also using SCIKIT-LEARN's RIDGECV and MULTIOUTPUTREGRESSOR classes

results presented in Table 3.1 for the model and various guessing schemes implemented in PySCF (guessing schemes implemented in PySCF are listed in Section A.1).

Table 3.1.: Comparison of different guessing schemes for 102 (20%) test samples from the $C_5H_4N_2O_2$ subset from QM9. [32] The average F-score is calculated on the test set using the Fock matrix prediction from the ridge regression model and various guessing schemes implemented in PySCF. Averages are shown for the number of iterations until convergence, and the inference time in units of the inference time of the `minao` guess. The row 'Diverged' represents the percentage of samples without convergence within 50 iterations.

Method	`Ridge-model`	`minao`	`1e`	`atom`	`hückel`	`vsap`
F-score / 1	0.93(4)	0.899(2)	0.71(2)	0.8020(10)	0.840(10)	0.993(2)
Iterations / 1	14(3)	11.7(10)	23(4)	11.4(8)	22(6)	12.8(11)
Inference-speed / 1	0.7(3)	1.0	0.05(2)	0.5(4)	0.5(3)	3.0(13)
Diverged / %	4	0	31	0	27	0

In terms of iterations, ridge regression performs better than the `1e` and `hückel` guessing schemes, but worse than `minao`, `atom` and `vsap`. It should be noted that there seems to be no significant correlation of the F-score with the number of iterations until convergence. While `vsap` yields by far the highest F-score, it takes on average one iteration more to converge than the `minao` guessing scheme. This hints at the fact that some guessing strategies might offer a better guess in regions of the density matrix which are more relevant for convergence speed than others.
This prompted a deeper investigation of the guess structures. For each of the 102 test samples, we computed the element-wise difference between the initial guess and the converged density, normalized these differences, and then averaged them across the set. The resulting error patterns for `minao`, `vsap` and the ridge regression model are shown in Figure 3.3.

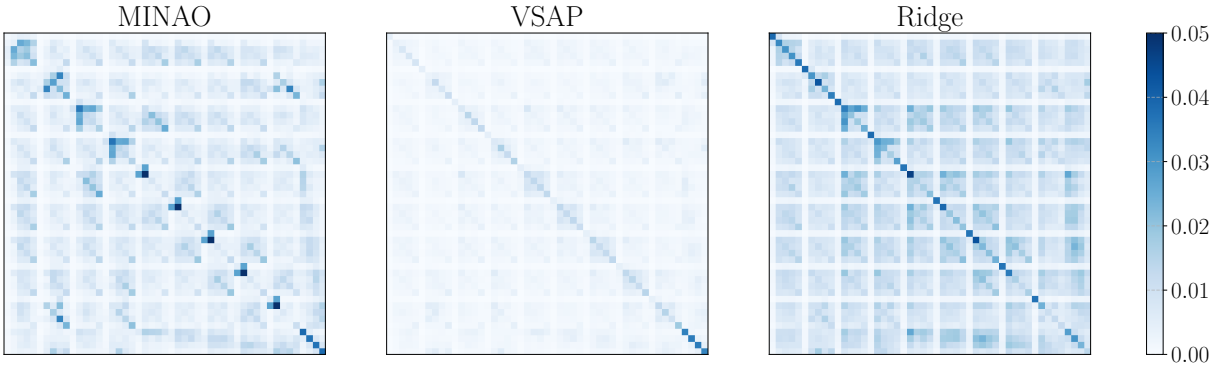

Figure 3.3.: Normalized mean absolute difference of α-electron density guesses to their converged density for `minao` & `vsap` guess and the ridge regression model evaluated on the test set: 102 (20%) test samples from the $C_5H_4N_2O_2$ subset from QM9. [32]

All three guessing schemes have spots on the main diagonal of the density matrix, where they are likely to differ from the reference solution. Interestingly, a Mulliken population analysis (see Subsection 2.1.5) [37] yields a mean value of 31.766 on the test set prediction for `minao`, while `vsap` yields 32 (only α-electrons), as expected. A small error in the initial guess does not slow

down the convergence since the self-consistent cycle corrects it right away. When comparing `minao` and `vsap` with our ridge model, the latter performs worse on the main diagonal, especially for p-orbitals. While `minao` exhibits more pronounced off-diagonal noise in comparison to `vsap`, this does not seem to affect convergence speed. At least for this small basis set, errors in the off-diagonal elements do not appear to hinder fast convergence.

3.4. Kernel Ridge Regression

Our baseline Ridge Regression (RR) model already introduced L^2-regularization. However, it only models linear relationships between our input and output features. Kernel Ridge Regression (KRR) extends this by using kernel functions to generate a mapping from the input space via a higher dimensional feature space to the output space. This allows nonlinear relationships to be learned.
Using the same train/test split as before, we fit a KRR model with three different kernels: RBF, a polynomial of degree 2 and a polynomial of degree 3. Rather than averaging per-target α values (like in the RidgeCV above), we use a joint Bayes Search with 5-fold cross validation and 30 iterations[4] to obtain the results in Table 3.2.

Table 3.2.: Comparison of different Kernel Ridge Regression (KRR) guessing schemes for 102 (20%) test samples from the $C_5H_4N_2O_2$ subset from QM9. [32] The F-score is calculated using the Fock matrix prediction from the Kernel Ridge Regression model and the `minao` guess. The number of iterations until convergence is shown, as well as the percentage samples not converging within 50 iterations and the inference time as a factor of the inference time of the `minao` guess.

Method	`KRR RBF`	`KRR quadratic poly`	`KRR cubic poly`	`minao`
F-score / 1	94(4)	94(4)	53.0(15)	0.899(2)
Iterations / 1	14(3)	14(3)	20(5)	11.7(10)
Inference-speed / 1	30	20	19	1.0
Not-Converged / %	4	4	12	0

Hyperparameters found by the Bayes Search are shown below in Table 3.3.

Table 3.3.: Hyperparameters found using Bayes Search for the Kernel Ridge Regression models with different kernels. Search space: $\alpha \sim \text{LogUniform}(10^{-8},10^{4})$, $\gamma \sim \text{LogUniform}(10^{-6},10^{3})$ and coef0 $\sim$ Uniform(0,1) for polynomial kernels; for the RBF kernel coef0 is ignored.

Hyperparameter	`KRR RBF`	`KRR quadratic poly`	`KRR cubic poly`
α	$2 \cdot 10^{-8}$	$1 \cdot 10^{-8}$	$5 \cdot 10^{-2}$
γ	$4 \cdot 10^{-5}$	$7 \cdot 10^{-5}$	$1 \cdot 10^{-6}$
coef0	-	$7.5 \cdot 10^{-1}$	$1.4 \cdot 10^{-1}$

[4] Refer to Table 3.3 for hyperparameters.

While RBF and quadratic kernels yield nearly comparable results (with the exception of inference speed) the cubic kernel performs significantly worse. This may be explained by the fact that the Bayes Search gave a relatively high α value of 0.05, leading to a strong regularization of the model and making it less flexible. Even with lower regularization, the cubic kernel, as well as the other two kernels, does not yield comparable performance to the `minao` guessing scheme. Interestingly, our models provide a significantly higher F-score (exception: cubic kernel) compared to `minao`, but take more iterations to converge. A pattern seen before with the ridge regression model emerges again. This, and the inefficient scaling of KRR with the number of samples suggests further exploration of alternative approaches in the next section.

3.5. Further trials using MLP

Previously, we tried to learn a mapping from the overlap to the Fock matrix solely via machine learning means. The study revealed that the quality for these guesses, even on the minimal STO-3G basis, is not on par with the established guessing schemes.
Predicting the main diagonal for a given system is an easier target, given the fact that the model predicts N features from the N^2 input features. One way of constructing the off diagonal elements is the generalized Wolfsberg-Helmholtz (GWH) construction [7] (see Equation 2.21). Utilizing this construction on our dataset, we obtain comparable results to our base ridge regression model in terms of iterations until convergence.
Choosing the larger 6-31G(2df,p) basis set, we see the limitations of our initial approach: Instead of roughly $49^2/2$ features we now have $254^2/2$, a nearly 27-fold increase in the number of features. Furthermore, we move to a species offering more samples for training, namely the 6095 $C_7H_{10}O_2$ structural isomers (this set of isomers will also be used for the GNN benchmarking in Chapter 5).

MLP Network
We implement an MLP model which is trained to map the overlap matrix to the main diagonal of the converged Fock matrix, from which density can be reconstructed via GWH. All samples are split into 80% training, 10% validation and 10% test sets. Batchwise data loading is implemented to facilitate training on memory-limited hardware. To gauge the performance of different configurations, a hyperparameter search is performed over the parameters in Table 3.4.

Table 3.4.: Hyperparameter & Parameters of the MLP model. Model with best parameters marked bold.

Hyperparameter	**Search Space**
`layers / 1`	$\{\mathbf{1}, 2, 3, 4\}$
`neurons / 1`	$\{256, 512, \mathbf{1024}\}$ separate for each layer
`dropout rate / %`	$\{0, 5, \mathbf{10}, 15, 20\}$ separate for each layer
Parameters	**Value**
`layer type`	dense
`activation`	gelu
`batch normalization`	after each hidden dense layer
`batch size`	32
`input dimension`	40470
`output dimension`	284

TENSORFLOW is used as backend with KERAS and the module KERAS_TUNER is employed for hyperparameter optimization. [38, 39, 40] All models in the hyperparameter optimization are trained using the Adam optimizer with an exponential decay (rate 0.96 every 100 batches). Using mean squared error as the optimization objective on the validation set, the Hyperband tuner (see Subsection 2.2.7) selected a single-layer MLP with 1024 neurons and a 10% dropout rate. Subsequently, this model was re-trained for 50 epochs for initial evaluation on the test set. A prediction performance (RMSE on test set) of 0.100(14) compared to 0.357(7) `minao`'s on the main diagonal looks promising. However, after reconstructing the full Fock matrix using Equation 2.21, the global prediction quality on the full matrix suffers drastically as seen in Figure 3.4.

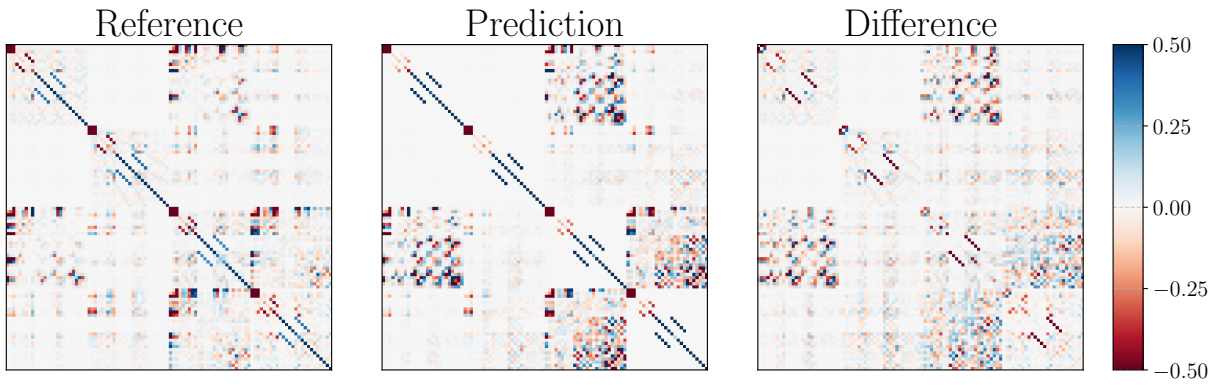

Figure 3.4.: Reference (fully converged Fock matrix) vs. predicted Fock matrix using the MLP model and the GWH reconstruction and element-wise difference for `dsgdb9nsd_082452`. Plots only show entries for two O (upper left) and two C (lower right) atoms and their respective off-diagonal elements.

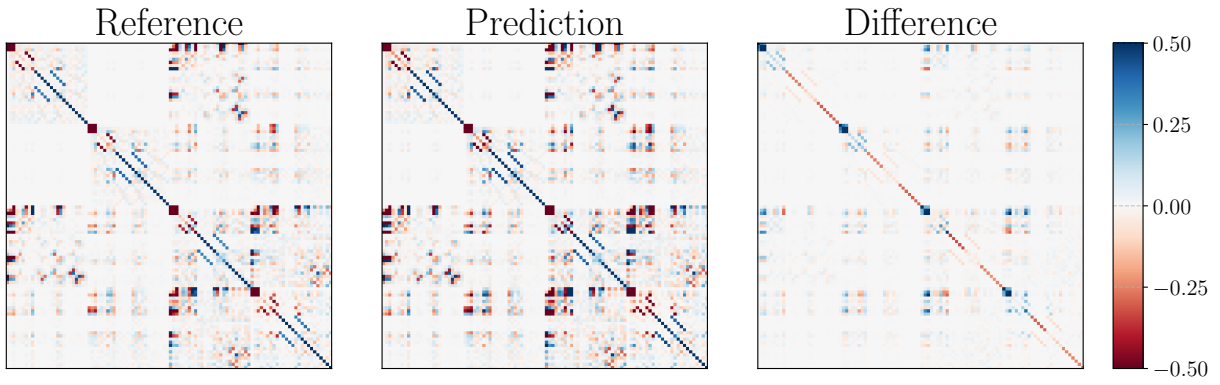

Figure 3.5.: Reference (fully converged Fock matrix) vs. Fock matrix constructed via `minao` guess and element-wise difference for `dsgdb9nsd_082452`. Plots only show entries for two O (upper left) and two C (lower right) atoms and their respective off-diagonal elements.

When comparing Figure 3.4 and Figure 3.5, a difference in error patterns is immediately apparent. While the MLP model performs well on the main diagonal, the GWH reconstruction cannot adequately capture the off-diagonal elements, thus resulting in a high RMSE of 0.085 compared to 0.029 for the `minao` guess. Given that many entries are zero—thereby artificially lowering the overall RMSE—the roughly three-fold difference further highlights the substantial underperfor-

mance of the GWH reconstruction compared to the `minao` guess.
To obtain the density from our Fock matrices we solve Equation 3.1 and gather the lowest n_{occ} eigenvectors to build the density matrix. Figure 3.6 shows the density obtained by this procedure and compares it to the density obtained from the `minao` guess.

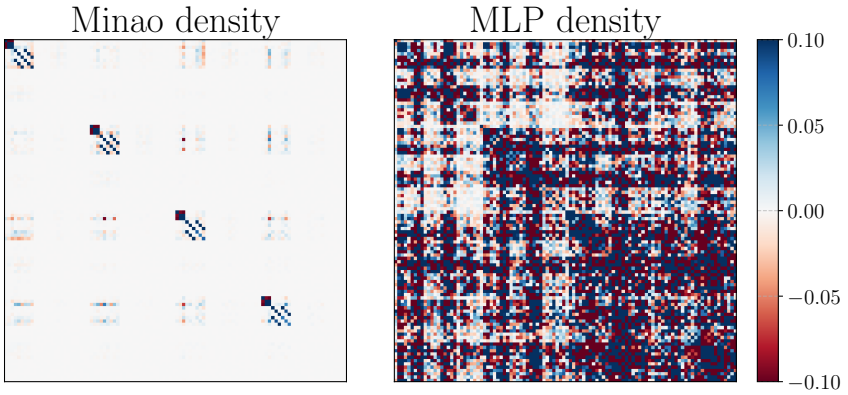

Figure 3.6.: `minao` density vs. density given by the diagonalization of the Fock prediction from the MLP model for `dsgdb9nsd_082452`. Plots only show entries for two O (upper left) and two C (lower right) atoms and their respective off-diagonal elements.

While `minao` provides a sparse density matrix, our MLP implementation yields a very dense result with many non-zero-off-diagonal elements. This stems from the fact that GWH does introduce large errors in the Fock matrix which propagate through the eigenvalue problem to the density matrix.
A qualitative benchmark of the iteration count until convergence offers a similar picture as before: Calculations using the `minao` guess converge in roughly eleven iterations, while the simulations using the MLP model with GWH reconstruction take around 20 iterations. The present model thus performs similar to other basic initialization schemes such as the `1e` guess. Interestingly, our model and the `1e` guess do not offer statistically relevant benefits in terms of iterations compared to a randomly initialized density matrix as seen in Table 3.5.

Table 3.5.: Iterations needed to convergence for different guessing schemes on the $C_7H_{10}O_2$ test subset. MLP_GWH uses the MLP prediction with GWH reconstruction of Fock off-diagonals and subsequent derivation of density matrix. The `random` column refers to a random density guess in the range $[-0.5, 0.5]$

Method	minao	1e	MLP_GWH	random
Iterations (#)	10.8(4)	19(3)	19.8(10)	19.8(12)

3.6. Conclusion

This chapter introduced an approach to predict the density matrix by first estimating the Fock matrix and then reconstructing the density via diagonalization. Initial experiments on the $C_5H_4N_2O_2$ subset of QM9 showed that linear models (Ridge Regression and Kernel Ridge Regression) can learn the Fock matrix reasonably well, but fail to match the accuracy of established

PySCF guessing schemes such as `minao` and `vsap` in terms of iterations until convergence. When scaling up to a larger 6-31G(2df,p) basis and the $C_7H_{10}O_2$ isomer set, a single-layer MLP captured the Fock diagonal with low RMSE, yet the generalized Wolfsberg-Helmholtz reconstruction of off-diagonals introduced large errors. Diagonalization amplifies these errors, producing a noisy density matrix and nearly doubling SCF iterations (≈ 20 vs. 11).

Overall, while the ML-based Fock prediction workflow can outperform simple Hückel or one-electron guesses in diagonal accuracy, it still underperforms in convergence speed and off-diagonal fidelity compared to physics-based schemes. This suggests that a one-shot prediction of the full Fock matrix and a subsequent density derivation are not feasible. Accounting for local interactions and directly predicting a sparse density matrix (compared to the Fock matrix) may perform better. We shall explore this approach in the next chapter by representing the molecular structure using a Graph Neural Network (GNN).

4. A GNN approach to the problem

So far we have experimented with various models trying to predict the Fock matrix as a whole. However, as we have seen in Chapter 3, this approach, which works on minimal basis sets, is not feasible for larger basis sets such as the 6-31G(2df,p) basis set. Recent advances using regression models have largely focused on separate models for each molecular species, which limits their applicability to the constitutional isomers we are interested in. [41, 42] We shall investigate the applicability of Graph Neural Networks (GNNs) to the problem of predicting the Fock or the density matrix. They have shown promising results in the field of quantum chemistry, especially for predicting molecular properties and structures. [43]

The implementation of the GNN in this chapter is based on the PYTORCH-GEOMETRIC framework. [44, 45]

4.1. Input & Output Matrices

Generally speaking, the input and the output of neural networks is fixed. We thus need to find a specific way to embed our input (overlap matrix) and output (Fock or density matrix) into a fixed structure. Given the nature of our problem, we can simply split our matrix representation into blocks representing the different atom sorts and their respective combinations. This yields three different types of matrix-blocks:

- **Self-blocks:** These blocks mix orbitals of the same atom type[1], e.g. O_0-O_0 or H_1-H_1. The self-blocks are diagonal blocks of the input and output matrices.
- **Homo-blocks:** These blocks mix only orbitals of the same atom type, e.g. H_1-H_2.
- **Hetero-blocks:** These blocks mix orbitals of different atom sorts, e.g. O_0-H_2.

Subsequently, we shall label self-blocks as node-blocks and homo- / hetero-blocks as interaction-blocks interchangeably.

A schematic depiction for the H_2O molecule using the 6-31G(2df,p) Basis is given in Figure 4.1. Due to the symmetry of our input and output matrices, we are only concerned with the upper triangular matrices for self-blocks and the homo- / hetero-blocks in the upper triangular part with respect to the full matrix. For the given example we thus have 351 entries for O and 15 for H self-blocks. Additionally, the H homo-blocks yield 25 entries each and we obtain 130 values per O-H hetero-block. In total 666 entries suffice to reconstruct the full 36×36 matrix.

[1] Subindices only enumerate elements and are not part of the chemical formula.

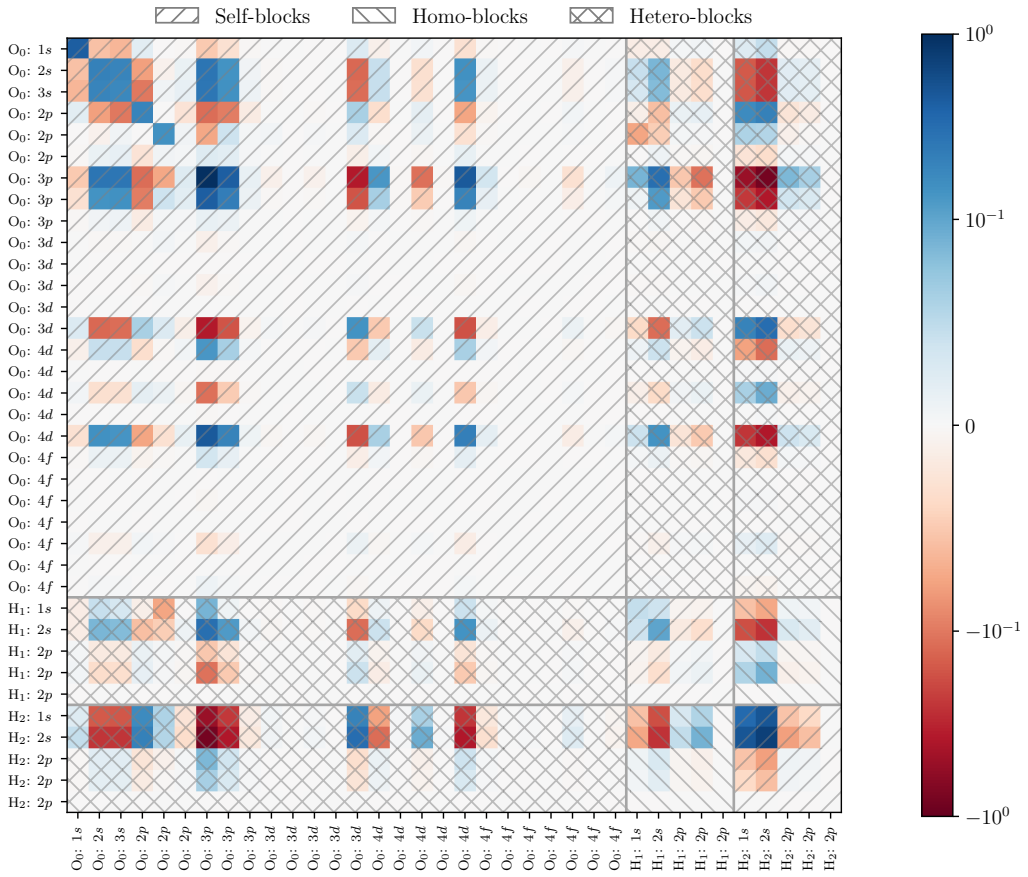

Figure 4.1.: Converged density of H_2O with matrix block types indicated.

Evidently, the different block types vary in size. This needs to be addressed before feeding the data into the network. There are multiple ways of addressing this problem. A straightforward solution includes a simple zero pad to attain the maximum block size and subsequent masking. Another approach is the compression or transformation to a fixed size latent space or some sort of pooling to reach the desired input dimension. In order to not lose relevant data we take yet another route and encode each block using separate encoders. By using this approach we mitigate the problem of different sized inputs as long as the basis set is fixed. For the example in Figure 4.1 we would thus have four differently sized encoders (two for the self-blocks and one for homo-/ and hetero-blocks respectively). These encoders will transfer the input features into a common hidden dimension to be used in the network. After the propagation through our GNN the decoder side will decode the output dimension of our network to the respective block dimensions from which the whole matrix can be reconstructed.

4.2. Preprocessing

Before feeding our data into the network we must bring it into a suitable form for learning with the PYTORCH-GEOMETRIC framework. [44, 45] After introducing the design of our data objects and graph creation we discuss a way to tackle the problem of sensitivity to spatial rotation before closing out with normalization and dataset creation remarks.

4.2.1. Graph creation

A graph representation of our data is required and will be populated using the paradigms described in Section 4.1. Nodes of our graph are essentially represented by embeddings of our overlap self-blocks. This leaves us with one node per atom in our molecule. To enable easier benchmarking and training processes the targets (Fock or density matrix-self blocks) are also stored in our graph object. Nodes are connected by directed edges which are given by an embedding of the homo- / hetero-block regions under a given restriction. The restriction on edge formation may be given by a distance cutoff (e.g. 3 Å), but not necessarily limited to that method. In particular, for bonds involving high-angular-momentum orbitals, the highly directed radial extension of the electronic wavefunction should also lead to an edge between these respective nodes. Edge formation can thus be controlled by a threshold with regard to the maximum value or another metric e.g. a matrix norm for a given interaction-block. A tensor of indices is used to represent the directed edges between nodes identified with said index. Every bond is represented by two directed edges to support bidirectional data exchange between nodes.

In addition to the node and edge embeddings of our inputs and targets, we store node atom-symbols and edge-symbols (e.g. C-O) to match encoder/decoder later on in our network. Furthermore, we store global index information for both node- and interaction-blocks to facilitate matrix reconstruction from our block representation.

4.2.2. Data Augmentation

The input (overlap) as well as the output (Fock/density matrix) are quantities which are not generally invariant under rotation of our molecular system. In essence, this means that we must find a way to learn differently rotated molecules and produce their respective Fock/density matrices to later deduce our initial density. While the idea of making the input invariant under rotation using a predetermined standard orientation was initially considered, problems arise with this approach. Defining a standard orientation for isomers/isomer-parts, such as $C_7H_{10}O_2$ or submolecules thereof, is far from trivial. Even if such a standard orientation is defined, one has to consider an additional pre- and post-processing step to rotate the input into the standard orientation and the output back to the original orientation.
Contrary to this, the model can learn differently rotated inputs to generate the corresponding outputs. This is achieved by augmenting the dataset with different rotations of the same molecules/submolecules. Rotating the input coordinates using a rotation matrix R is rather straightforward in principle. However, the corresponding overlap, density and Fock matrices also have to be transformed accordingly or recalculated. The later is computationally not feasible, hence we use the corresponding Wigner D-matrices to transform input and output matrices. A Wigner D-matrix is a unitary matrix with $2L + 1$ rows and columns, where L is the angular momentum. For a given rotation R the matrix elements of overlap, density and Fock matrix transform according to

$$O'_{ij} = \sum_{k,l} \mathcal{D}^{(L)}_{ik}(R) O_{kl} \mathcal{D}^{(L)*}_{lj}(R). \tag{4.1}$$

Naturally, the transformation only acts on spatial orbitals with no rotational symmetry along the axis of rotation (i.e. $L \neq 0$). Given the blocks defined in Section 4.1, we see that $\mathcal{D}^{(L)}$ will only transform elements of those blocks for which at least one orbital has $L \neq 0$.

For our purpose, the input data is augmented by sampling a random rotation axis, using three normally distributed values to obtain an axis and a random rotation angle $\theta \in [0, 2\pi]$. Given this axis and angle, the corresponding transformations are performed to the overlap, density and Fock matrices to obtain augmented samples. New graphs are created using these transformed matrices according to the procedure described in the precious section and added to the dataset.
Due to grid-spacing in DFT calculations, small deviations ($\approx 0.1\,\mathrm{m}E_\mathrm{h}$ in the Fock matrix) arise between the transformed matrices and newly computed ones from the rotated atom coordinates. These deviations only slightly affect the accuracy of the reconstructed density (see Equation 3.1) and are not relevant during training, as we evaluate the model against the provided target. In other words, if the model predicts the density via the Fock matrix, we compare it to a reference density that was also reconstructed from the (transformed) Fock, not from a directly transformed density matrix.

4.2.3. Normalization & data split

Means and standard deviations are calculated on a per block-type basis (e.g. for the O-H hetero-block) on all training data samples. This is done for the input (overlap matrix) and the output/targets (Fock or density matrix).
It should be noted that the inclusion of data augmentation samples does not change these values besides machine precision errors. The normalization is then performed by subtracting the mean and dividing by the standard deviation for each block-type. To avoid numerical instabilities, especially for very sparse blocks (std. dev $\to 0$), a value of 10^{-6} is used in case the standard deviation falls below this value for a given block.

The dataset is ultimately split into training, validation, and test sets, typically following a ratio of 80% / 10% / 10%, with respect to the total number of original, non-augmented samples.

When applying a data augmentation factor of 1.5, meaning that 50% more samples are generated synthetically, the training set is extended to contain 120% of the original data, while the validation and test sets remain unchanged at 10% each. This ensures that evaluation is performed exclusively on the original, unaltered data and avoids any data leakage through augmented samples. To ensure comparability of benchmarks in Chapter 5 the SCF_GUESS_DATASETS package was used to obtain samples. [46]

4.3. ML pipeline & GNN Design

The GNN's pipeline is designed to take coordinates from xyz-files, embed them into a graph form to facilitate training, and eventually obtain the full density matrix. A schematic overview of the pipeline is given in Figure 4.2.

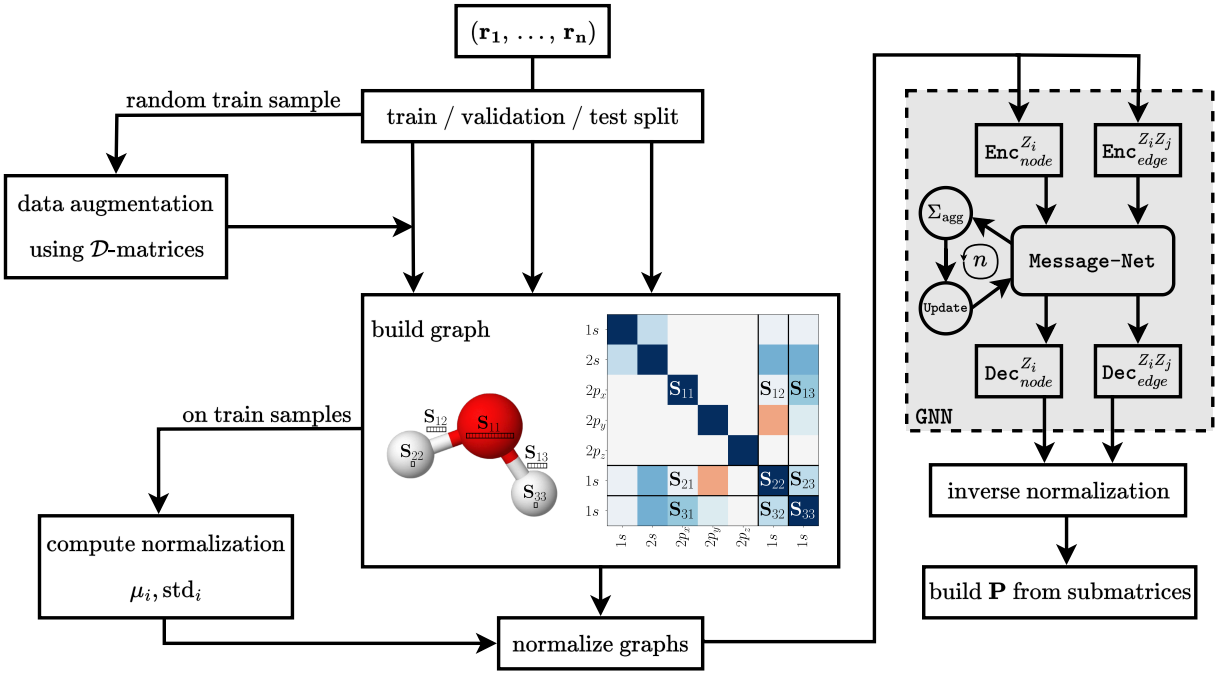

Figure 4.2.: Overview of the GNN pipeline: training data is augmented, converted to graphs, normalized, and passed through a message passing GNN. Predicted submatrices are denormalized and combined to reconstruct the full density matrix. H_2O overlap matrix is shown with exemplary embedding vectors for the STO-3G basis set.

Encoders

To obtain a common input dimension, differently sized blocks are encoded separately. Programmatically, this is enabled by a factory class that creates the required encoders for the different block types and sizes at runtime Given the H_2O example in Figure 4.1, we obtain two encoders for the self-blocks, one for O: $\texttt{Enc}^{8}_{node}$ and one for H: $\texttt{Enc}^{1}_{node}$. The example above uses a cutoff distance of 1 Å for the edge formation, which yields one edge encoder type for the O-H bonds: $\texttt{Enc}^{81}_{edge}$. Due to the arguably small cutoff distance – in practice this will be set to higher values to capture interactions between orbitals of higher angular momentum with longer radial extension – there is no H-H edge encoder.

Both node and edge encoders are implemented as a 2-layer network with a GeLU-activation between the layers. The first layer maps the input matrix to a fixed hidden dimension[2], while the second layer maps from a hidden dimension to a hidden dimension.

Message Net (Message Passing)

The Message Net is a simple feed forward network with a given linear layer depth GeLU activation and optional dropout, both fixed at initialization. It is designed to map an input of three times the hidden dimension, which includes the node embeddings and the edge embedding of two connected nodes, to an output the size of one hidden dimension. This happens for each source-target node connection in the graph in the example graph in Figure 4.1, we 'pass a message' from O to H and vice versa. Messages are only passed between nodes that are connected by an edge and never from one node to itself (i.e. there are no self-loops). For each target node, all incoming messages are added up; for example, O receives two messages from the H atoms and each H receives one from O. A concatenation of the aggregated sum and the node embedding is fed into a node updater

[2]For edge encoders, the interatomic distance is appended as an additional input feature.

network, which maps from twice the hidden dimension via a linear layer, GeLU activation and another linear layer back to one hidden dimension. When all node updates are completed, one round of message passing is finished. Multiple rounds of message passing are performed to enable information travel between more distant non-directly connected nodes.
Parameters mentioned above, such as the hidden dimension, number of message passing steps, dropout, etc. are all hyperparameters that can be tuned during training.

Decoder
Decoders are effectively reversing the transformation of the encoders to obtain the original dimensions of the matrix blocks. This preferably results in accurate target predictions. Node decoders, like $\texttt{Dec}^{8}_{node}$ or $\texttt{Dec}^{1}_{node}$, map back to the upper triangular matrix of self-blocks, while edge decoders, such as $\texttt{Dec}^{8\,1}_{edge}$, map back to the full interaction matrix. Both use one linear layer which directly maps from the hidden dimension to the respective output dimension.

The raw output of the GNN needs to be inversely normalized to yield meaningful predictions. Ultimately, the full matrix is rebuilt from self-blocks and interaction-blocks by reversing the block splitting procedure described in Section 4.1. If the target is the Fock matrix, the reconstruction of the density matrix is performed as a post-processing step.

4.4. Training

The GNN is trained using various hyperparameters and configurations to optimize its performance on different datasets. While details on the application of the network will be given in Chapter 5, the general training procedure will be discussed in this section.

Loss Function
The choice of the loss function can also be handled as a hyperparameter. While a physically motivated loss function which correlates well with the iteration count is preferable, it is far from trivial to derive such a metric. A variety of these metrics[3] were initially promising when scoring guesses against converged density, but turned out to be suboptimal in terms of their correlation with iteration count.
If not mentioned otherwise, in later subchapters the MSE (see Subsection 2.2.2) will be used as a loss function. While this function may not offer a direct physical interpretation besides the total global similarity of prediction and target matrices, it is well understood and established in a wide variety of learning problems. Additionally, the loss function, which is evaluated blockwise[4] on the target matrices, can be combined and/or weighted in various ways to obtain a single scalar loss value per sample.

Optimizer, learn rate & early stopping
Similar to the loss function, the optimizer may also be treated as a hyperparameter. However, due to computational and time constraints, it will be fixed for the scope of this thesis. A stochastic gradient descent optimizer, namely AdamW, is used in all training runs in combination with a learn rate scheduler. AdamW is preferred over the initial Adam implementation due to the decoupling of weight decay in AdamW. [47] All parameters in the learn rate scheduler (reduction factor, patience, threshold, etc.) and the initial learn rate are also treated as hyperparameters.

[3] E.g. F-score for different guessing schemes in Chapter 3
[4] Blockwise evaluation is computationally easier because the full density matrix is not reconstructed during training.

Training trials, during which the validation loss stagnates, will be cut short using the grace-epochs parameter. In most systems with high atom counts our model learns sufficiently fast. This is due to the fact that each sample contains multiple self- and interaction-blocks per atom sort and atom combination. Therefore, the number of grace epochs is set to a rather low 5 epochs in most runs.

Hyperparameter tuning
Hyperparameter tuning is performed using the Ray Tune [48] package. Given the the huge variety of hyperparameters, a grid search is computationally not feasible. The Asynchronous Successive Halving Algorithm (ASHA) is used as a scheduler to find suitable hyperparameter combinations faster. The authors claim that the algorithm finds good hyperparameters nearly an order of magnitude faster, compared to a random search. [49]
To further cut down on discrete combinations, ASHA samples hyperparameters by drawing values uniformly or log-uniformly from an interval or by randomly selecting from a predefined list; some hyperparameters remain fixed during this process. This constitutes a resource efficient way, which can be easily extended to larger search spaces and larger compute clusters if needed.

4.5. Benchmark model

Since loss and iteration count are not inherently correlated, we must validate our model by actually running SCF calculations using its predicted densities. This computationally rather expensive step can only be performed on selected models, which means that hyperparameter tuning cannot directly benefit from actual iteration counts obtained by new calculations.
Similar to the benchmarks in Chapter 3, we will benchmark the GNN against established PySCF guesses. Additionally, we introduce an element-wise average guess $\overline{P}$ defined as

$$\overline{P}_{\mu\nu} = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} P^{(i)}_{\mu\nu}. \tag{4.2}$$

For the sake of fairness, this average is only evaluated on the elements of the respective training set. The performance of $\overline{P}$ guess and various PySCF schemes on the $C_7H_{10}O_2$ isomer test subset is given in Figure 4.3.

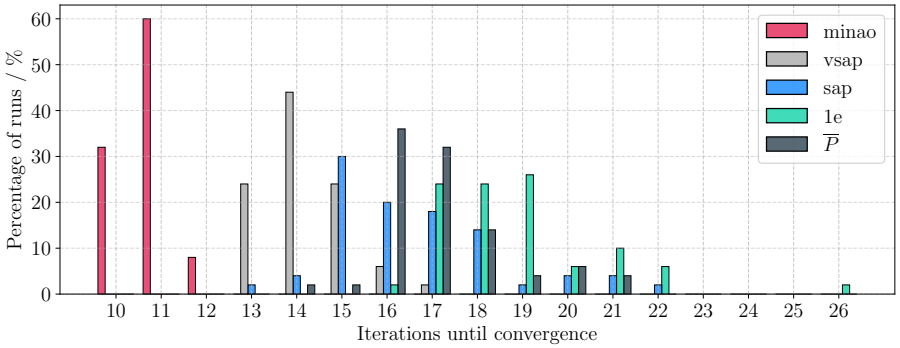

Figure 4.3.: Iterations until convergence distribution for QM9-isomers for PySCF and $\overline{P}$ guesses.

For further insights regarding convergence behaviour using a second order Newton solver, refer to Section A.2

5. Application

The GNN model introduced in Chapter 4 will be benchmarked on various datasets in the following sections. All reference data was calculated using the 6-31G(2df,p) basis at theory level B3LYP.

Models will be evaluated by their iteration count until convergence, by the energy difference from the converged solution the DIIS error (see Equation 2.32b), and by the RMSE. The inference time of all models, just a few milliseconds, is roughly three orders of magnitude shorter than that of a typical DFT calculation, and can therefore be regarded as negligible. This performance holds across all applications discussed below.

5.1. QM9 - $C_7H_{10}O_2$ Isomers

There are 6095 structural isomers of $C_7H_{10}O_2$ in the QM9 dataset (see Section 2.3). Analogous to the trials performed in Section 3.5, we will train and validate on a randomly drawn sample of 500 isomers[1]. This reduction is necessary to make training and hyperparameter-tuning feasible in the scope of the thesis. Contrary to the full matrix prediction schemes of Chapter 3, we employ sub-matrix predictions and reconstruct the full matrix, thus increasing the actual number of samples significantly. Per molecule, we obtain 7, 10 and 2 samples for C, H and O, respectively, yielding a total of 3500 C, 5000 H and 1000 O samples. This already provides a certain rotational variability, but additional rotations can be introduced through data augmentation during training to develop a model that is insensitive to rotation when predicting density.

[1] using SCF_GUESS_DATASETS (see Subsection 4.2.3)

5.1.1. Initial training

To evaluate the performance of the GNN devised in Chapter 4, several manual runs were initiated during development, with hyperparameters set to the values listed in Table 5.1.

Table 5.1.: Hyperparameters used for the initial MGNN training (manually selected)

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Hidden dimension	256	Msg. passing rounds	4
MsgNet layers	3	MsgNet dropout	15 %
Batch size	16	Grace period	10 epochs
Target	Density matrix	Loss function	MSE (blockwise)
Learn rate (initial)	2.68×10^{-3}	Weight decay	1.78×10^{-5}
Edge threshold	3 Å	Data augmentation	No
Learn rate factor	0.5	Learn rate patience	3 epochs
Learn rate threshold	10^{-3}	Learn rate cooldown	2 epochs
Learn rate min	10^{-6}	—	—

Note that these initial runs did not use data augmentation, so the training comprised 400 samples (80% of our data), leaving 10% for validation and 10% for testing. The grace period (time without improvement) was set to 10 epochs to leave the learning rate scheduler sufficient time to take effect.

Training and validation losses both monotonically decrease until around epoch 30. As can be seen in Figure 5.1, while the loss on the validation set flattens out rather early, training loss decreases throughout the entire training process.

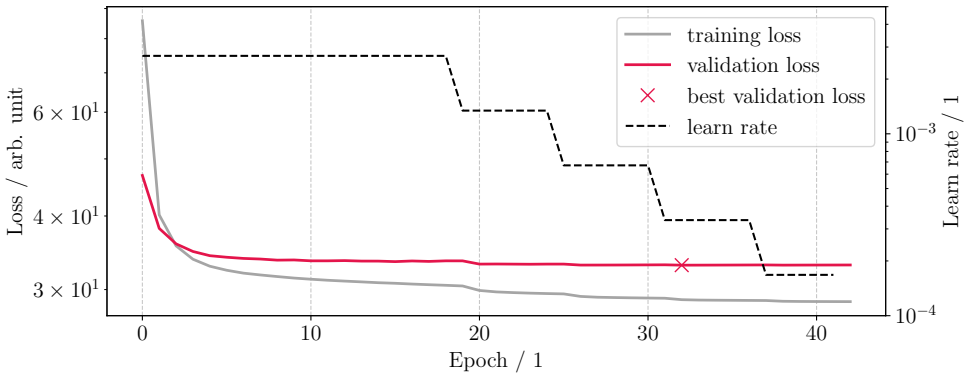

Figure 5.1.: Initial GNN training/ validation loss and corresponding learn rate per epoch on QM9-isomers.

Both losses are reduced further by stepwise decrease of the learn rate. This run produced the best model in epoch 33, with a validation loss of 33.00 and a training loss of 28.83, indicating slight overfitting which is to be expected especially without data augmentation in the training samples. The performance of the model $\mathrm{GNN}_{\mathrm{initial}}$ on the test set is compared to other models and the various guessing schemes in a summarizing Table 5.5 placed at the end of this chapter.

5.1.2. Hyperparameter tuning

Validation loss is used as a benchmark to select the best model from a hyperparameter run. While we will also prefer models with lower loss, we must be very careful not to select models which look good on paper but perform worse due to the lack of sufficient correlation between MSE and iteration count. For this reason, we will base our hyperparameter search on the $\text{GNN}_{\text{initial}}$ model and explore the hyperparameter space in a structured way.

Data augmentation

$\text{GNN}_{\text{initial}}$ already performed quite well in terms of iterations without using any data augmentation. One might argue that there is already some data augmentation intrinsic to the training set due to the different orientations of atoms in various molecules.

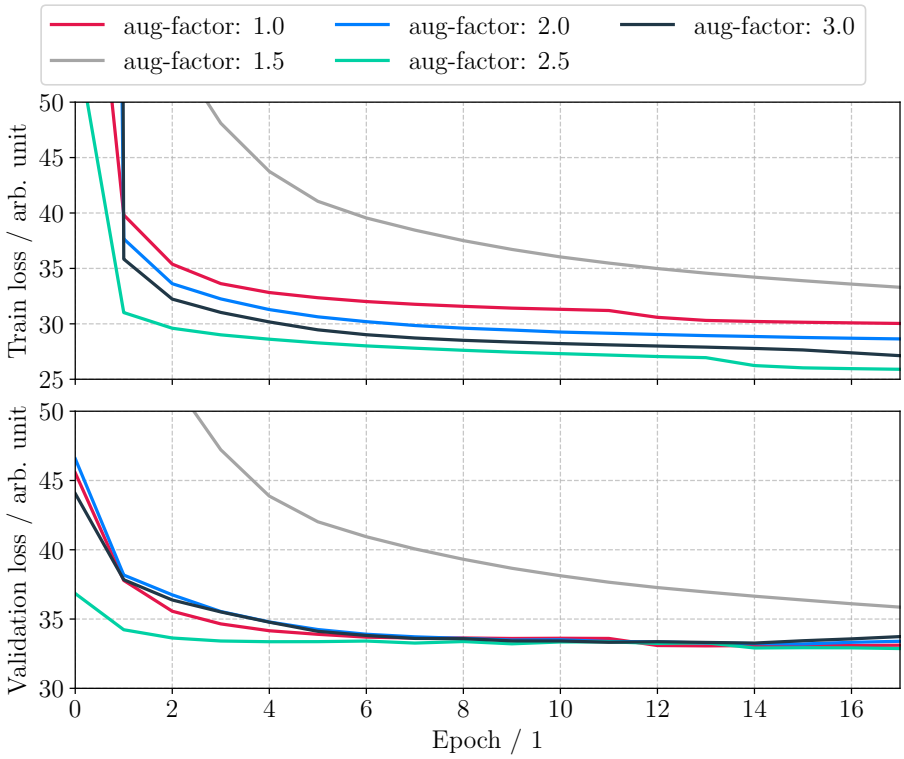

Figure 5.2.: GNN loss for different data augmentation factors on QM9-isomers. All other hyperparameters are chosen as in Table 5.1.

A comparison of the training and validation loss between different augmentation factors in Figure 5.2 shows no clear trend regarding the choice of the augmentation factor. While a factor of 2.5 initially outperforms no augmentation and other factors, they all converge in validation eventually. Investigating different metrics on the test set in Table 5.2 does not yield a conclusive insight. Our initial model has the best performance in terms of iterations; however, other models do not fall far behind in this metric. Furthermore, no statistically significant correlation was found between the number of iterations and other metrics, including the energy errors ($|\Delta E_{\text{DFT}}|$ and relative $|\delta E_{\text{DFT}}|$), the DIIS error, and the RMSE.

Table 5.2.: GNN performance on the QM9 $C_7H_{10}O_2$ isomers test set for different data augmentation factors[2]. For each factor, the table lists the mean and standard deviation of the following metrics: iterations until convergence, $|\Delta E_{\mathrm{DFT}}|$ (absolute energy difference to the reference DFT calculation), $|\delta E_{\mathrm{DFT}}|$ (relative energy difference to the reference DFT calculation), DIIS error (as defined in Equation 2.32b), and root mean square error (RMSE) between the predicted and converged density matrices. Energies are given in Hartree (E_{h}).

Augmentation factor / 1	Iterations / 1	$\lvert\Delta E_{\mathrm{DFT}}\rvert$ / E_{h}	$\lvert\delta E_{\mathrm{DFT}}\rvert$ / 1	DIIS error / E_{h}	RMSE / 1
1.0	**11.2(6)**	**0(3)**	**0.0000(16)**	0.027(4)	**0.0078(6)**
1.5	11.3(6)	1.4(17)	0.0008(10)	0.027(4)	0.0079(6)
2.0	11.5(6)	**0(3)**	0.0001(14)	**0.027(3)**	0.0079(6)
2.5	12.0(7)	1(4)	0.001(2)	0.029(3)	0.0080(6)
3.0	11.4(8)	1(3)	0.0006(18)	0.031(7)	0.0081(6)
3.5	11.3(6)	2(3)	0.0013(18)	0.029(3)	0.0080(6)
4.0	11.6(10)	2(4)	0.001(3)	0.029(3)	0.0080(6)

Investigating Edge Threshold Distance

The edge threshold distance has an immediate impact on message passing between nodes. It essentially acts as a cutoff that determines whether an edge between two nodes is formed. Keeping it rather short, in the vicinity of bond-lengths will only connect directly bound neighbours. On the other hand, a higher threshold distance allows the formation of longer ranging connections in the GNN. Table 5.3 shows the influence on the metrics for varying threshold distances.

Table 5.3.: GNN performance on QM9 $C_7H_{10}O_2$ isomers test set for different edge threshold distances (given in Å). Other hyperparameters are set according to Table 5.1.

Threshold distance / Å	Iterations / 1	$\lvert\Delta E_{\mathrm{DFT}}\rvert$ / E_{h}	$\lvert\delta E_{\mathrm{DFT}}\rvert$ / 1	DIIS error / E_{h}	RMSE / 1
1.5	15.1(19)	1(13)	0.001(8)	0.045(5)	0.0104(5)
2.0	12.3(9)	4.1(8)	0.0024(5)	0.034(5)	0.0084(5)
2.5	**11.2(6)**	1(2)	0.0006(12)	0.026(4)	0.0081(6)
3.0	11.8(13)	1(6)	0.001(3)	0.027(3)	0.0079(6)
3.5	12.6(8)	1(2)	0.0010(11)	**0.025(4)**	0.0077(6)
4.0	12.0(9)	2(4)	0.001(2)	**0.025(4)**	0.0078(6)
4.5	12.4(10)	**0(7)**	**0.000(4)**	0.027(4)	0.0080(6)
5.0	11.2(7)	1(2)	0.0005(12)	**0.025(4)**	**0.0077(7)**

For a low distance threshold of 1.5 Å, which lies around the bond length of C-C bonds, worse performance in terms of iterations is attained compared to higher threshold distances.

Further hyperparameter optimization runs

For all training runs up until now, the training loss was evaluated only on the normalized sub matrices included in training. In practice, this means that the network has generally lower loss for low cutoff distances because fewer interaction blocks are included in the loss calculation. In the next step, we investigate whether changing the loss to a full matrix loss will impact GNN prediction performance. Table 5.4 shows ten different configurations evaluated on the test set.

[2]Other hyperparameters are set according to Table 5.1.

Table 5.4.: GNN on QM9 isomers using full matrix loss and a maximum of 30 epochs to train. Metrics on test set and corresponding hyperparameter settings for the five best performing networks (0-4) from search and five sampled ones (5-9) from same hyperparameter search.

Models	Iterations / 1	$\lvert\Delta E_{\mathrm{DFT}}\rvert$ / E_{h}	$\lvert\delta E_{\mathrm{DFT}}\rvert$ / 1	DIIS error / E_{h}	RMSE / 1
$\mathrm{GNN}_{\mathrm{f.\ 0}}$	**11.5(9)**	0.7(14)	0.0004(8)	0.028(4)	**0.0076(6)**
$\mathrm{GNN}_{\mathrm{f.\ 1}}$	11.6(7)	**0.2(17)**	**0.0001(10)**	**0.027(4)**	0.0078(6)
$\mathrm{GNN}_{\mathrm{f.\ 2}}$	11.7(8)	18.0(18)	0.0103(10)	0.029(3)	0.0079(6)
$\mathrm{GNN}_{\mathrm{f.\ 3}}$	13.1(11)	0.3(14)	0.0002(8)	0.035(5)	0.0084(5)
$\mathrm{GNN}_{\mathrm{f.\ 4}}$	13.2(11)	2.3(10)	0.0013(6)	0.040(6)	0.0088(5)
$\mathrm{GNN}_{\mathrm{f.\ 5}}$	13.3(11)	2(3)	0.0014(15)	0.042(4)	0.0090(5)
$\mathrm{GNN}_{\mathrm{f.\ 6}}$	13.6(10)	1(2)	0.0004(13)	0.042(2)	0.0089(5)
$\mathrm{GNN}_{\mathrm{f.\ 7}}$	13.9(11)	1.5(16)	0.0008(9)	0.041(3)	0.0088(6)
$\mathrm{GNN}_{\mathrm{f.\ 8}}$	14.5(14)	0.9(7)	0.0005(4)	0.040(2)	0.0090(5)
$\mathrm{GNN}_{\mathrm{f.\ 9}}$	14.7(12)	7(3)	0.0043(18)	0.042(3)	0.0091(5)

Parameters	$\mathrm{GNN}_{\mathrm{f.\ 0}}$	$\mathrm{GNN}_{\mathrm{f.\ 1}}$	$\mathrm{GNN}_{\mathrm{f.\ 2}}$	$\mathrm{GNN}_{\mathrm{f.\ 3}}$	$\mathrm{GNN}_{\mathrm{f.\ 4}}$	$\mathrm{GNN}_{\mathrm{f.\ 5}}$	$\mathrm{GNN}_{\mathrm{f.\ 6}}$	$\mathrm{GNN}_{\mathrm{f.\ 7}}$	$\mathrm{GNN}_{\mathrm{f.\ 8}}$	$\mathrm{GNN}_{\mathrm{f.\ 9}}$
Hidden dimension	512	256	128	128	128	512	256	128	256	256
Batch size	8	8	8	8	32	32	32	32	32	32
Data aug. factor	1.83	2.46	2.38	1.76	1.50	1.14	1.67	2.45	1.91	2.33
Edge threshold	2.73	3.87	3.21	2.07	2.11	2.11	2.14	2.12	2.11	2.05
Message passing steps	2	4	3	3	4	2	3	2	3	3
Message Net dropout	0.10	0.27	0.21	0.18	0.29	0.14	0.18	0.23	0.22	0.11
Message Net layers	3	5	4	3	5	3	3	3	5	3
Learn rate	6.34e-04	2.31e-04	2.89e-04	4.93e-03	3.97e-03	1.79e-04	3.54e-04	4.36e-03	2.61e-04	1.16e-04
Weight decay	4.56e-04	7.37e-05	4.37e-05	1.69e-05	9.97e-04	8.35e-05	5.21e-05	4.43e-05	2.97e-04	2.35e-05

The best performing networks using full matrix loss perform slightly worse with regard to iteration count. Notably, $\mathrm{GNN}_{\mathrm{f.\ 0}}$ reaches the lowest RMSE of all benchmarked models so far. Correlation between our surrogate metrics and iterations is rather high for DIIS and RMSE, with a Pearson correlation coefficient of 0.91 and 0.94 respectively. Contrary, there is no correlation between the energy errors $\lvert\Delta E_{\mathrm{DFT}}\rvert$ and $\lvert\delta E_{\mathrm{DFT}}\rvert$ and the number of iterations. When interpreting these correlations, caution is necessary because they only hold under the specific conditions (GNN trained using full matrix loss) in which they were measured. Below, we will see that these relationships exhibit limited generalizability across different guessing schemes.

5.1.3. Evaluation & Conclusion

The top performing models as defined above are now compared to PySCF guessing schemes. A summary of this comparison is provided in Table 5.5.

Table 5.5.: Comparison of different models with PySCF and $\overline{P}$ guessing schemes for QM9 - $C_7H_{10}O_2$ Isomers.

Guessing schemes	Iterations / 1	$\lvert\Delta E_{\mathrm{DFT}}\rvert$ / E_{h}	$\lvert\delta E_{\mathrm{DFT}}\rvert$ / 1	DIIS error / E_{h}	RMSE / 1
$\mathrm{GNN}_{\mathrm{initial}}$	11.2(6)	**0(3)**	**0.0000(16)**	0.027(4)	0.0078(6)
$\mathrm{GNN}_{\mathrm{f.\ 0}}$	11.5(9)	0.7(14)	0.0004(8)	0.028(4)	**0.0076(6)**
$\overline{P}$	17.1(15)	211(6)	0.1208(19)	0.110(3)	0.0137(4)
`1-e`	18.8(18)	220(30)	0.128(19)	0.1220(10)	0.14(4)
`vsap`	14.2(9)	5.4(4)	0.0030(2)	**0.026(2)**	0.0109(7)
`atom`	16.6(19)	14(4)	0.007(3)	0.044(7)	0.016(2)
`minao`	**10.8(6)**	2.8(2)	0.001 62(12)	0.077(3)	0.0155(4)

A direct comparison of PySCF and $\overline{P}$ guessing schemes with our GNN approach already offers several insights. Examining the relationship between energy errors and the number of iterations shows that guesses with smaller energy errors ($|\Delta E_{\mathrm{DFT}}|$) tend to convergence faster. Furthermore, a low DIIS alone will not necessarily translate into fast convergence of the guessed density. The GNN approach surpasses most conventional guessing schemes and performs nearly on par with `minao` in terms of iterations. Varying the data augmentation factor had negligible effect on the convergence speed, while the distance threshold contributes for distances around the bond length. However, it remains to be seen how well these models generalize to other data.

5.2. QM9 - $C_7H_{10}O_2$ Molecular Dynamics (MD)

So far, all predictions have considered only ground state geometries. While predicted and calculated traits of the ground state give invaluable insight into the chemical properties, the behaviour of the molecules in more attainable environments is of interest. The MD trajectories of $C_7H_{10}O_2$ dataset [50] constitute a randomly sampled set of 113 isomers from the QM9 $C_7H_{10}O_2$ data. The trajectory of every isomer is calculated every 1 fs for 5000 steps at a temperature of 500 K using the PBE exchange-correlation potential (see 2.1.8). From these geometries, 500 are sampled to calculate a reference data set using the 6-31G(2df,p) basis at the B3LYP level of theory. This combination is employed in all following computer experiments.

5.2.1. Zero-shot predictions

Zero-shot predictions are made on inputs outside the training set's scope. In our case, a model previously trained on the QM9 $C_7H_{10}O_2$ isomer set may be used to predict the density for a given MD sample. Prediction metrics using the pre-trained models from Section 5.3 on the MD test set are given in Table 5.6.

Table 5.6.: GNN zero-shot predictions on the QM9 $C_7H_{10}O_2$ isomer MD test set. $GNN_{initial}$ and $GNN_{f.\ 0}$ were trained using the QM9 $C_7H_{10}O_2$ isomer set.

Models	Iterations / 1	$\lvert\Delta E_{DFT}\rvert$ / E_h	$\lvert\delta E_{DFT}\rvert$ / 1	DIIS error / E_h	RMSE / 1
$GNN_{initial}$	11.4(9)	1(2)	0.0007(11)	0.026(3)	0.0079(4)
$GNN_{f.\ 0}$	11.3(8)	1.3(11)	0.0007(6)	0.029(3)	0.0077(4)

Both models perform surprisingly well on the unseen data. $GNN_{f.\ 0}$ even performs slightly better, indicating a generalization to various perturbations of the geometry. Furthermore, $GNN_{initial}$ performs slightly worse on the MD dataset.

5.2.2. Hyperparameter tuning

Metrics for re-trained versions of $GNN_{initial}$ and $GNN_{f.\ 0}$ from Section 5.3 are shown in Table 5.7. While training $GNN_{f.\ 0}$ on the MD dataset leads to improvements in all metrics, the re-trained version of $GNN_{initial}$ performs worse than the original model on the MD test set.

Table 5.7.: GNN predictions on the QM9 $C_7H_{10}O_2$ isomer MD test set. With MD-re-trained[3] versions of $GNN_{initial}$ and $GNN_{f.\ 0}$.

Models	Iterations / 1	$\lvert\Delta E_{DFT}\rvert$ / E_h	$\lvert\delta E_{DFT}\rvert$ / 1	DIIS error / E_h	RMSE / 1
$GNN^{MD*}_{initial}$	12.3(10)	1(4)	0.001(2)	0.035(4)	0.0086(6)
$GNN^{MD*}_{f.\ 0}$	11.1(5)	0.0(10)	0.0000(6)	0.028(3)	0.0073(5)

Analogous to the hyperparameter optimization in Table 5.4, we also use full matrix loss for hyperparameter tuning of the specialized MD models in Table 5.8. Comparing Table 5.4 and Table 5.8, heavier regularization is generally observed in the MD models. Higher data augmentation and a slightly stronger weight decay help with fitting the varied density given by the perturbed geometries. Furthermore, four out of the top five networks have a batch size of eight, introducing more noise and thus regularization into the update steps.

Table 5.8.: GNN on QM9 isomers MD using full matrix loss and a maximum of 30 epochs to train. Metrics on test set and corresponding hyperparameter settings for the ten best performing networks in terms of iterations.

Models	Iterations / 1	$\lvert\Delta E_{\mathrm{DFT}}\rvert$ / E_{h}	$\lvert\delta E_{\mathrm{DFT}}\rvert$ / 1	DIIS error / E_{h}	RMSE / 1
$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 0}}$	**10.9(5)**	1.5(17)	0.001(1)	**0.027(3)**	**0.0075(4)**
$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 1}}$	11.0(4)	1.8(13)	0.001(1)	0.030(3)	0.0076(4)
$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 2}}$	11.2(6)	3.1(22)	0.002(1)	0.031(3)	0.0079(5)
$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 3}}$	11.2(4)	1.9(21)	0.001(1)	0.033(3)	0.0084(3)
$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 4}}$	11.4(6)	3.5(12)	0.002(1)	0.031(3)	0.0081(3)
$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 5}}$	11.6(10)	0.6(23)	**0.000(1)**	0.035(3)	0.0087(5)
$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 6}}$	11.6(9)	**0.0(18)**	**0.000(1)**	0.035(9)	0.0087(12)
$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 7}}$	11.8(9)	1.0(33)	0.001(2)	0.032(4)	0.0084(4)
$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 8}}$	11.8(7)	4.1(20)	0.002(1)	0.037(3)	0.0085(3)
$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 9}}$	11.8(7)	1.0(18)	0.001(1)	0.035(6)	0.0083(9)

Parameters	$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 0}}$	$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 1}}$	$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 2}}$	$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 3}}$	$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 4}}$	$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 5}}$	$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 6}}$	$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 7}}$	$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 8}}$	$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 9}}$
Hidden dimension	512	256	512	128	128	512	512	512	256	128
Batch size	8	8	8	16	8	32	32	16	32	8
Data aug. factor	3.00	4.00	2.00	4.00	3.00	3.00	3.00	3.00	2.00	1.00
Edge threshold	2.65	3.25	3.86	3.11	2.22	3.31	3.86	3.95	2.29	3.82
Message passing steps	3	4	3	3	4	2	2	2	5	4
Message Net dropout	0.14	0.24	0.02	0.05	0.16	0.20	0.10	0.27	0.06	0.02
Message Net layers	3	3	4	4	4	2	3	2	3	3
Learn rate	1.08e-04	3.99e-04	1.95e-04	3.27e-04	4.82e-04	8.66e-04	5.65e-04	2.80e-04	1.21e-03	3.32e-03
Weight decay	7.13e-04	1.24e-04	2.35e-04	1.50e-05	8.57e-04	1.35e-04	4.36e-05	2.06e-04	9.34e-04	7.39e-04

5.2.3. Evaluation & Conclusion

It has become evident that not every model and hyperparameter configuration translates to good results for the geometry-perturbed MD dataset. The best performing model for the isomers, $\mathrm{GNN}^{\mathrm{MD}^*}_{\mathrm{initial}}$, takes an iteration longer to converge than the $\mathrm{GNN}^{\mathrm{MD}^*}_{\mathrm{f.\ 0}}$ model (both are re-trained versions!). Therefore, the latter model generalizes better to the MD geometries. This may be attributed, at least to some extend, to the full matrix loss used during training for this model. Table 5.9 compares the best performing GNNs with established guesses. Besides the well established `minao` scheme, the GNNs outperform all other guessing schemes in terms of iterations.

Table 5.9.: Comparison of different models with PySCF and $\overline{P}$ guessing schemes for QM9 - $C_7H_{10}O_2$ MD.

Guessing schemes	Iterations / 1	$\lvert\Delta E_{\mathrm{DFT}}\rvert$ / E_{h}	$\lvert\delta E_{\mathrm{DFT}}\rvert$ / 1	DIIS error / E_{h}	RMSE / 1
$\mathrm{GNN}^{\mathrm{MD}}_{\mathrm{f.\ 0}}$	10.9(5)	1.5(17)	0.001(1)	**0.027(3)**	0.0075(4)
$\mathrm{GNN}^{\mathrm{MD}^*}_{\mathrm{initial}}$	12.3(10)	1(4)	0.001(2)	0.035(4)	0.0086(6)
$\mathrm{GNN}^{\mathrm{MD}^*}_{\mathrm{f.\ 0}}$	11.1(5)	**0.0(10)**	**0.0000(6)**	0.028(3)	**0.0073(5)**
$\overline{P}$	16.9(10)	211(6)	0.1208(19)	0.110(3)	0.0137(4)
`1-e`	18.2(17)	227(19)	0.129(10)	0.126(8)	0.14(3)
`vsap`	14.5(7)	5.8(5)	0.0033(2)	0.028(2)	0.0114(6)
`atom`	16.6(17)	14(3)	0.0080(19)	0.047(6)	0.0167(16)
`minao`	**10.7(5)**	2.7(4)	0.001 54(19)	0.077(3)	0.015 38(18)

5.3. QM9 - Full Dataset

So far, all experiments and tests focused on structural isomers of a single molecule, initially $C_5H_4N_2O_2$ (see Section 3.2) and $C_7H_{10}O_2$ later. The GNN architecture performed very well against established guessing schemes, but still has to prove its capabilities in a more general setting. For this task, the 134k molecules of QM9 [31] provide a broad chemical space with up to nine heavy atoms and the five elements, H, C, N, O and F.

5.3.1. Sampling the QM9 dataset

For a fair evaluation of the various guessing types, some effort has to be made with respect to the QM9 dataset sampling. Contrary to the previous isomer datasets, we have to ensure a balanced distribution of traits. This is achieved as follows. We stratify the train, validation and test sets by the number of atoms in the molecules. This stratification helps to prevent the models from being unduly biased toward any particular molecular size due to an unbalanced train/validation/test split.

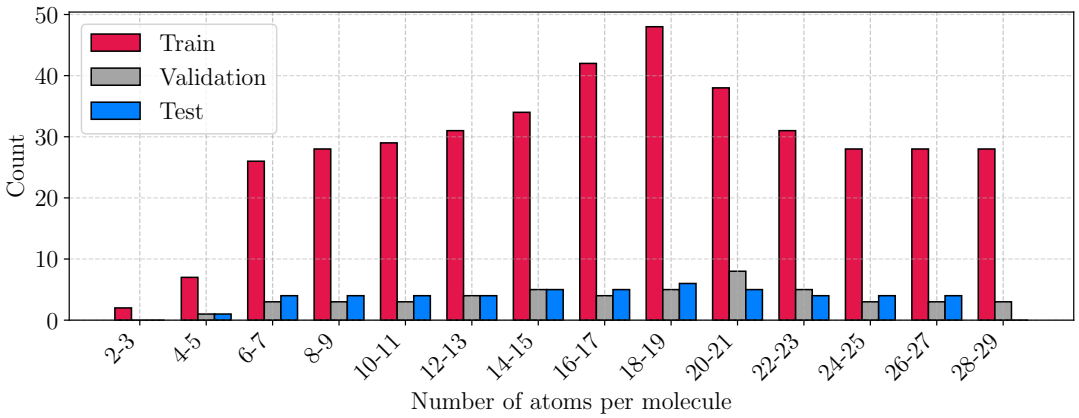

Figure 5.3.: Stratified sample of QM9 dataset by number of atoms per molecule for each set.

We sample 500 molecules from the QM9 set in a stratified manner and additionally ensure ample representation for various molecule sizes. Therefore, compared to the actual distribution seen in Figure 2.5, the representation of molecules with 6 to 13 and 24 to 29 atoms per molecule is increased. This results in the distribution for the train, validation and test sets shown in Figure 5.3. Once again, the train/validation/test split is 80/10/10 per cent.

Zero-shot predictions like those devised in the previous section are not possible on the full dataset due to missing encoders/decoders for N, F and their respective interaction blocks. Therefore, we will proceed directly to hyperparameter tuning.

5.3.2. Hyperparameter tuning

Network hyperparameter tuning was carried out in the same manner as above, except that the maximum number of training epochs was increased to 50 to accommodate the more diverse dataset. Table 5.10 shows the results for the ten best performing models.

Table 5.10.: GNN on full QM9 dataset sample using full matrix loss and a maximum of 50 epochs to train. Metrics on test set and corresponding hyperparameter settings for the ten best performing networks in terms of iterations.

Models	Iterations / 1	$\lvert\Delta E_{\mathrm{DFT}}\rvert$ / E_{h}	$\lvert\delta E_{\mathrm{DFT}}\rvert$ / 1	DIIS error / E_{h}	RMSE / 1
$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 0}}$	12(2)	1.2(55)	0.001(3)	0.034(11)	0.009(3)
$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 1}}$	**12.0(17)**	0.9(17)	0.0010(11)	0.034(13)	**0.009(2)**
$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 2}}$	12.0(18)	1.3(42)	0.001(2)	0.035(11)	0.009(3)
$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 3}}$	12.0(19)	1.6(27)	0.0010(17)	0.034(13)	0.009(3)
$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 4}}$	12.0(17)	0.2(21)	0.0002(17)	0.044(48)	0.009(3)
$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 5}}$	12.0(18)	0.3(27)	0.000(2)	**0.033(13)**	0.009(3)
$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 6}}$	12(2)	0.3(19)	0.0002(13)	**0.033(13)**	0.009(3)
$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 7}}$	12.1(16)	0.7(25)	0.0007(19)	0.038(18)	0.009(3)
$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 8}}$	12(2)	**0.1(19)**	**0.0000(13)**	0.035(12)	0.009(3)
$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 9}}$	12.1(18)	1.3(23)	0.0008(13)	0.036(14)	0.009(2)

Parameters	$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 0}}$	$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 1}}$	$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 2}}$	$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 3}}$	$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 4}}$	$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 5}}$	$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 6}}$	$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 7}}$	$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 8}}$	$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 9}}$
Hidden dimension	128	256	256	256	512	128	512	512	512	128
Batch size	8	8	8	8	8	8	8	16	8	8
Data aug. factor	4.00	1.00	1.00	1.00	1.00	2.00	4.00	3.00	1.00	2.00
Edge threshold	3.40	2.58	3.98	3.21	2.85	3.29	3.27	2.72	3.44	2.30
Message passing steps	4	5	3	2	4	2	3	3	2	4
Message Net dropout	0.09	0.16	0.25	0.21	0.04	0.11	0.25	0.04	0.18	0.19
Message Net layers	3	3	3	3	3	4	3	2	3	2
Learn rate	6.90e-04	9.31e-04	1.61e-03	2.75e-04	8.03e-04	1.51e-04	1.26e-04	5.85e-04	1.09e-03	6.50e-04
Weight decay	7.15e-05	5.46e-05	3.58e-05	4.69e-05	3.99e-04	6.06e-04	2.56e-04	1.45e-04	1.84e-05	1.74e-05

Both the mean iteration count and its variability increased by about one cycle compared to the best-performing isomer models. Further analysis of iteration performance by molecule size revealed no clear correlation between atom count and the number of iterations to converge.

5.3.3. Evaluation & Conclusion

Using our stratified QM9 sample, we show that our GNN implementation trained on just 400 molecules achieves lower SCF iteration counts than every other guessing scheme except `minao`. The greater chemical diversity in this subset leads to a uniformly higher standard deviation in cycle counts across all methods. Table 5.11 compares our top-performing GNN against the established guessing schemes.

Another notable finding emerges when contrasting the full dataset results for $\overline{P}$ with those from the isomer only runs. While the average number of iterations remained essentially unchanged, with only their spread increasing for the more diverse dataset, the DIIS error in the full dataset

differs substantially from that in the isomer-only case. Furthermore, `minao` has the second highest DIIS error while simultaneously offering the fastest converging guess. Crucially, it once again underscores that DIIS error and iteration count are largely uncorrelated.

Table 5.11.: Comparison of GNN model with calculations employing PySCF and $\overline{P}$ guessing schemes on the full QM9 dataset. Here, $\overline{P}$ is computed blockwise across all molecules by averaging over each C-C, O-H, etc. block.

Guessing schemes	Iterations / 1	$\lvert\Delta E_{\mathrm{DFT}}\rvert$ / E_{h}	$\lvert\delta E_{\mathrm{DFT}}\rvert$ / 1	DIIS error / E_{h}	RMSE / 1
$\mathrm{GNN}^{\mathrm{Full}}_{\mathrm{f.\ 1}}$	12.0(17)	**0.9(17)**	**0.001(11)**	0.034(13)	**0.009(2)**
$\overline{P}$	17(3)	40(19)	0.024(10)	0.042(6)	0.014(3)
`1-e`	19(3)	200(80)	0.13(4)	0.122(15)	0.16(12)
`vsap`	14(2)	4.3(19)	0.0027(9)	**0.024(3)**	0.0106(14)
`atom`	16(2)	10(5)	0.006(3)	0.039(7)	0.015(2)
`minao`	**11.1(11)**	**2.4(8)**	**0.0016(5)**	0.081(12)	0.017(3)

6. Conclusion & Outlook

This thesis presents a novel approach to density matrix prediction of molecular systems using machine learning techniques. Initially, an indirect way of predicting the density matrix from the overlap was explored, which involved predicting the Fock matrix and subsequently deriving the electron density in a given atom-centred basis set. Ridge- and Kernel Ridge Regression methods performed well against established schemes when employing a minimal basis set. However, moving to larger basis sets, small deviations in the Fock matrix led to enormous errors in the density matrix. Linear models, as well as a simple MLP network, failed to deliver adequate Fock predictions to derive accurate density matrices.
To mitigate these issues, a message passing Graph Neural Network (GNN) was designed in Chapter 4, to directly predict the sparse density matrix from the overlap matrix. A flexible per-atom-type encoder/decoder architecture was implemented, which allows for a handling of molecular systems of varying size. Furthermore, invariance to molecular rotation was ensured by augmenting the training data with rotated versions of the molecules.
In Chapter 5, the GNN was tested on three distinct datasets and achieved iteration-to-convergence performance on par with the best classical guessing schemes.

The primary limitation for all learning models is their reliance on a surrogate loss (mean squared error of the density matrix prediction) to minimize iterations. Training directly on the iteration count would likely boost network performance, but at the expense of immense computational effort. Moreover, the GNN restricts message passing to atoms, so matrix elements representing interatomic interaction (interaction blocks) are passively mapped via encoder/decoder and do not engage in message passing. While initial experimentation using updaters during message passing for interaction blocks showed promising results, it was not pursued further due to the larger complexity of the model.

Various avenues for future work are suggested, including further development of the GNN architecture, extension to larger and more varied systems, and the incorporation of a novel loss function to better align the model with the SCF iteration count.
Improvements of the architecture could include the addition of the aforementioned interaction block updaters, as well as additional embeddings to better capture the chemical environment. Implementing equivariant capabilities intrinsically in the GNN would eliminate the need for data augmentation, which did not improve network performance on our isomer dataset. Consequently, such an architectural change is advantageous only for systems that do not already offer a high diversity in molecule orientations.
While excellent prediction capabilities were observed for organic compounds including H, C, O, N and F, the model's performance on transition metals and larger systems remains to be explored. Promising results for the MD-trajectory dataset suggest that the application to transition state geometries might be feasible. Furthermore, application on hard to convergence transition metal complexes and heavy-elements with strong relativistic effects could be a worthwhile endeavour.

Finally, a hybrid loss function that combines RMSE with energy- and/or DIIS error might be promising. To avoid the costly derivation of the Fock matrix, crucial for these metrics, one might make use of twin networks, where one network predicts the density matrix and the other the Fock matrix. Combining the RMSE from one network with the DIIS error given by their predictions[1], one could simultaneously minimize DIIS and RMSE without directly training on iteration count.

[1] DIIS may be calculated using the density guess of one network and the Fock guess of the other (see Equation 2.32b).

A. Appendix

A.1. Initial guessing methods in PySCF [34]

The PySCF package offers a variety of initial guessing methods for the density matrix for SCF calculations, as described in the excerpt of the documentation [51] below:

- `minao` (default): A superposition of atomic densities [52, 53] technique, in which the guess is obtained by projecting the minimal basis of the first contracted functions in the cc-pVTZ or cc-pVTZ-PP basis set onto the orbital basis set, and then forming the density matrix. The guess orbitals are obtained by diagonalizing the Fock matrix that arises from the spin-restricted guess density.

- `1e`: The one-electron guess, also known as the core guess, obtains the guess orbitals from the diagonalization of the core Hamiltonian, thereby ignoring all interelectronic interactions and the screening of nuclear charge. The 1e guess should only be used as a last resort, because it is so bad for molecular systems; see [8].

- `atom`: Superposition of atomic densities [52, 53]. Employs spin-restricted atomic HF calculations that use spherically averaged fractional occupations with ground states determined by fully numerical calculations at the complete basis set limit in [54].

- `huckel`: This is the parameter-free Hückel guess described in [8], which is based on on-the-fly atomic HF calculations performed analogously to `atom`. The spherically averaged atomic spin-restricted Hartree-Fock calculations yield a minimal basis of atomic orbitals and orbital energies, which are used to build a Hückel-type matrix that is diagonalized to obtain guess orbitals.

- `vsap`: Superposition of atomic potentials as described in [8]. A sum of pre-tabulated, fully numerical atomic potentials determined with the approach of [54] is used to build a guess potential on a DFT quadrature grid; this potential is then used to obtain the orbitals. Note: this option is only available for DFT calculations in PySCF.

A.2. Comments on second-order Newton solvers & iterations

A second-order Newton solver essentially uses the Newton algorithm to converge an electron density by constructing density gradients g and approximating the Hessian H of the energy. The implementation in PySCF uses an outer macro loop which takes the Newton steps and an inner loop which approximates the solution to the linearized Newton equation:

$$H\Delta x = -g$$

to obtain the update Δx which is to be applied to the orbitals.

Faster convergence in terms of iterations was claimed by Schütt et al. for their neural network. [55] This prompted investigations on the $C_7H_{10}O_2$ isomer set. For our isomer test set, a comparable reduction of iterations from 1st order in Figure 4.3 to 2nd order[1] in Figure A.1 can be seen.

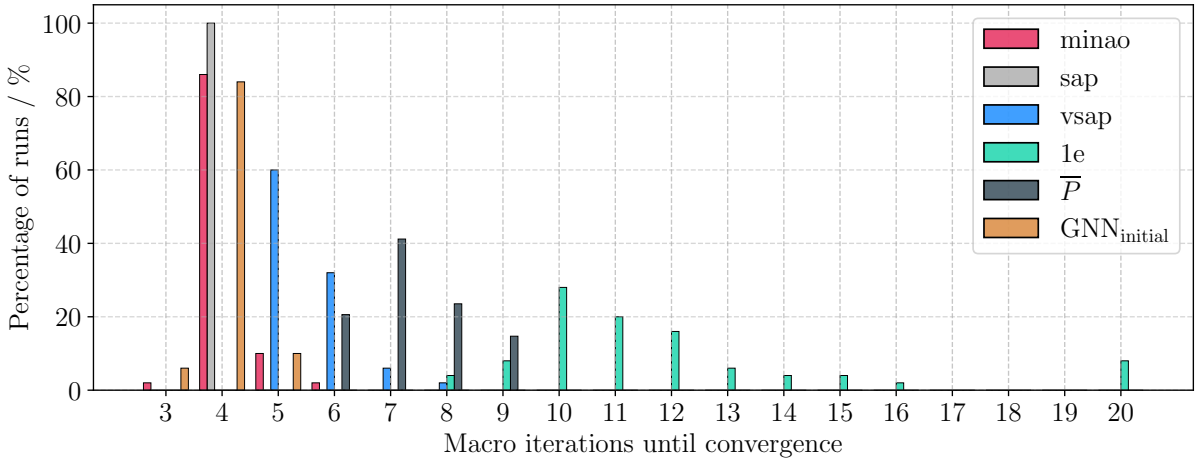

Figure A.1.: Macro iterations until convergence distribution for QM9-isomers for PySCF, $\overline{P}$ and $GNN_{initial}$ guesses. All entries with ≥ 20 macro iterations are grouped at 20.

Yet, this direct comparison is not entirely fair. Contrary to DIIS, which builds the Fock matrix once per iteration, the Newton-solver repeatedly rebuilds the Fock matrix in its inner loop. Summing all Fock matrix builds, one obtains different behaviour depicted in Figure A.2.

[1] using PySCF's `newton_ah` solver

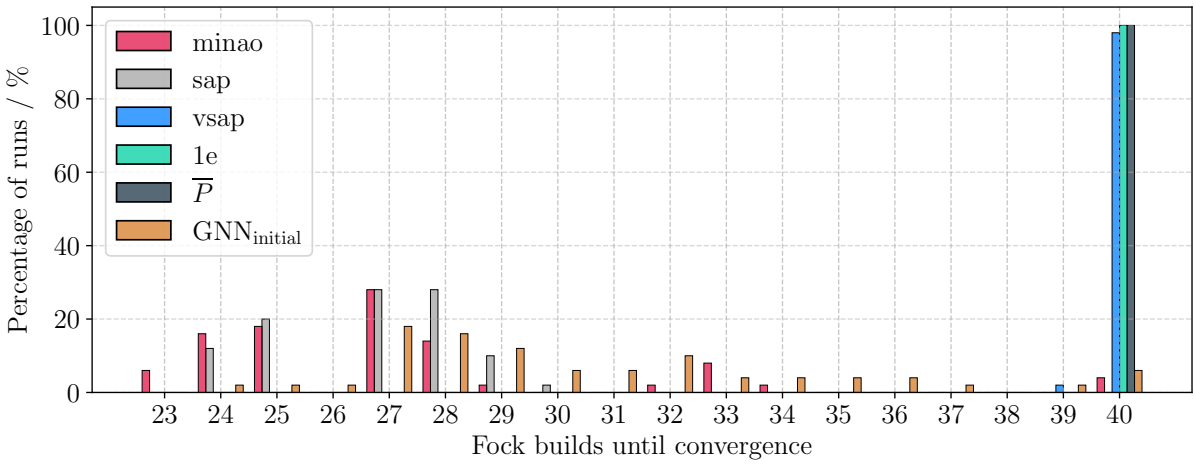

Figure A.2.: Fock matrix build count until convergence for QM9-isomers for PySCF, $\overline{P}$ and $\mathrm{GNN}_{\mathrm{initial}}$ guesses. All entries with ≥ 40 Fock builds are grouped at 40.

Given this fact, one needs to be cautious in making predictions regarding wall clock time of simulations from 'iteration' counts, especially when comparing different algorithms.

A.3. Source Code & Packages used

The source code developed in the course of this thesis was entirely written in Python 3.11.11 [56] and can be accessed online. [57]

Models in Chapter 3 were trained using SCIKIT-LEARN [36] and TENSORFLOW [38] with the KERAS frontend [39] and KERASTUNER [40] for optimization. The GNN in Chapter 4 was implemented using the PYTORCH GEOMETRIC library. [44] Hyperparameter optimization was performed using the RAY library. [48] Dataset handling was performed using a custom written SCF_GUESS_DATASETS package, which is available online. [46] Data augmentation partially used APIs of the custom developed SCF_GUESS_TOOLS package (release v1.0), also to be found online. [58]

List of Figures

2.1 Perceptron . . . 16
2.2 Commonly used activation functions. . . . 17
2.3 Multilayer Perceptron . . . 17
2.4 Schematic Message Passing Neural Net . . . 18
2.5 QM9 dataset overview . . . 20

3.1 Schematic overview data flow . . . 21
3.2 Density of dsgdb9nsd_022700 in STO-3G basis & theory level B3LYP . . . 23
3.3 Normalized difference of density guesses . . . 24
3.4 MLP vs. reference Fock . . . 27
3.5 `minao` vs. reference Fock . . . 27
3.6 `minao` vs. MLP density . . . 28

4.1 Matrix block regions of H_2O . . . 32
4.2 GNN pipeline design . . . 35
4.3 Iterations until convergence distribution for QM9-isomers . . . 37

5.1 Initial GNN loss on QM9-isomers . . . 40
5.2 GNN loss for different augmentation factors on QM9-isomers . . . 41
5.3 Stratified sample of QM9 dataset . . . 47

A.1 Macro iterations until convergence distribution for QM9-isomers . . . 54
A.2 Fock matrix build count until convergence QM9-isomers . . . 55

List of Tables

3.1 $C_7H_{10}O_2$ subset - iterations to convergence ridge regression . . . 24
3.2 $C_7H_{10}O_2$ subset - iterations to convergence Kernel Ridge Regression . . . 25
3.3 Hyperparameter Kernel Ridge Regression . . . 25
3.4 Hyperparameter & Parameters of the MLP model . . . 26
3.5 $C_7H_{10}O_2$ subset - iterations to convergence MLP . . . 28

5.1 Hyperparameters - initial MGNN training (manually selected) . . . 40
5.2 GNN on QM9 isomers with different data augmentation factors . . . 42
5.3 GNN on QM9 isomers with different edge threshold distances . . . 42
5.4 GNN on QM9 isomers training with full matrix loss . . . 43
5.5 Models compared to PySCF and $\overline{P}$ schemes - $C_7H_{10}O_2$ Isomers . . . 44
5.6 GNN zero-shot predictions on QM9 $C_7H_{10}O_2$ isomer MD . . . 45
5.7 GNN predictions on QM9 $C_7H_{10}O_2$ isomer MD . . . 45
5.8 GNN on QM9 isomers MD training with full matrix loss . . . 46
5.9 Models compared to PySCF and $\overline{P}$ schemes - $C_7H_{10}O_2$ MD . . . 46
5.10 GNN on full QM9 dataset sample with full matrix loss . . . 48
5.11 Models compared to PySCF and $\overline{P}$ schemes - full QM9 dataset . . . 49

Bibliography

[1] E. Schrödinger. "An Undulatory Theory of the Mechanics of Atoms and Molecules". In: *Physical Review* 28.6 (1926). Archived (PDF) on 17 December 2008, pp. 1049–1070. DOI: 10.1103/PhysRev.28.1049.

[2] D. R. Hartree. "The Wave Mechanics of an Atom with a Non-Coulomb Central Field. Part II. Some Results and Discussion". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 24.1 (1928), pp. 111–132. DOI: 10.1017/S0305004100011920.

[3] J. C. Slater. "Note on Hartree's Method". In: *Physical Review* 35.2 (1930), pp. 210–211. DOI: 10.1103/PhysRev.35.210.

[4] V. A. Fock. "Näherungsmethode zur Lösung des quantenmechanischen Mehrkörperproblems". In: *Zeitschrift für Physik* 61 (1930), pp. 126–148. DOI: 10.1007/BF01340294.

[5] A. Szabo and N. S. Ostlund. Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory. Dover ed. Dover Publications, 1996. ISBN: 9780486691862.

[6] J. H. Van Lenthe et al. "Starting SCF calculations by superposition of atomic densities". In: *Journal of Computational Chemistry* 27.8 (2006), pp. 926–932. DOI: https://doi.org/10.1002/jcc.20393. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.20393. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/jcc.20393.

[7] M. Wolfsberg and L. Helmholz. "The Spectra and Electronic Structure of the Tetrahedral Ions MnO_4^-, CrO_4^-, and ClO_4^-". In: *The Journal of Chemical Physics* 20.5 (1952), pp. 837–843.

[8] S. Lehtola. "Assessment of Initial Guesses for Self-Consistent Field Calculations. Superposition of Atomic Potentials: Simple yet Efficient". In: *Journal of Chemical Theory and Computation* 15.3 (Jan. 17, 2019), pp. 1593–1604. DOI: 10.1021/acs.jctc.8b01089.

[9] T. Koopmans. "Über die Zuordnung von Wellenfunktionen und Eigenwerten zu den einzelnen Elektronen eines Atoms". German. In: *Physica* 1 (1934), pp. 104–113.

[10] R. S. Mulliken. "Electronic population analysis on LCAO–MO molecular wave functions. I". In: *The Journal of Chemical Physics* 23.10 (1955), pp. 1833–1840. DOI: 10.1063/1.1740588.

[11] W. J. Hehre, R. F. Stewart, and J. A. Pople. "Self-Consistent Molecular-Orbital Methods. I. Use of Gaussian Expansions of Slater-Type Atomic Orbitals". In: *The Journal of Chemical Physics* 51.6 (Sept. 1969), pp. 2657–2664. ISSN: 0021-9606. DOI: 10.1063/1.1672392. eprint: https://pubs.aip.org/aip/jcp/article-pdf/51/6/2657/18864378/2657_1_online.pdf. URL: https://doi.org/10.1063/1.1672392.

[12] J. Dunning Thom H. "Gaussian basis sets for use in correlated molecular calculations. I. The atoms boron through neon and hydrogen". In: *The Journal of Chemical Physics* 90.2 (Jan. 1989), pp. 1007–1023. ISSN: 0021-9606. DOI: 10.1063/1.456153. eprint: https://pubs.aip.org/aip/jcp/article-pdf/90/2/1007/18974738/1007_1_online.pdf. URL: https://doi.org/10.1063/1.456153.

[13] F. Jensen. “Unifying General and Segmented Contracted Basis Sets. Segmented Polarization Consistent Basis Sets”. In: *Journal of Chemical Theory and Computation* 10.3 (2014), pp. 1074–1085. ISSN: 1549-9618. DOI: 10.1021/ct401026a. URL: https://doi.org/10.1021/ct401026a.

[14] F. Jensen. “Estimating the Hartree—Fock limit from finite basis set calculations”. In: *Theoretical Chemistry Accounts* 113.5 (2005), pp. 267–273. ISSN: 1432-2234. DOI: 10.1007/s00214-005-0635-2. URL: https://doi.org/10.1007/s00214-005-0635-2.

[15] P. Hohenberg and W. Kohn. “Inhomogeneous Electron Gas”. In: *Phys. Rev.* 136 (3B Nov. 1964), B864–B871. DOI: 10.1103/PhysRev.136.B864. URL: https://link.aps.org/doi/10.1103/PhysRev.136.B864.

[16] W. Kohn and L. J. Sham. “Self-Consistent Equations Including Exchange and Correlation Effects”. In: *Physical Review* 140.4A (Nov. 15, 1965), A1133–A1138. DOI: 10.1103/PhysRev.140.A1133.

[17] J. P. Perdew and K. Schmidt. “Jacob’s ladder of density functional approximations for the exchange-correlation energy”. In: *AIP Conference Proceedings* 577.1 (July 2001), pp. 1–20. ISSN: 0094-243X. DOI: 10.1063/1.1390175. eprint: https://pubs.aip.org/aip/acp/article-pdf/577/1/1/12108089/1_1_online.pdf. URL: https://doi.org/10.1063/1.1390175.

[18] J. C. Slater. “A Simplification of the Hartree-Fock Method”. In: *Phys. Rev.* 81 (1951), pp. 385–390. DOI: 10.1103/PhysRev.81.385.

[19] S. H. Vosko, L. Wilk, and M. Nusair. “Accurate spin-dependent electron liquid correlation energies for local spin density calculations: a critical analysis”. In: *Canadian Journal of Physics* 58.8 (1980), pp. 1200–1211. DOI: 10.1139/p80-159.

[20] J. P. Perdew, K. Burke, and M. Ernzerhof. “Generalized Gradient Approximation Made Simple”. In: *Physical Review Letters* 77.18 (1996), pp. 3865–3868. DOI: 10.1103/PhysRevLett.77.3865.

[21] J. Sun, A. Ruzsinszky, and J. P. Perdew. “Strongly Constrained and Appropriately Normed Semilocal Density Functional”. In: *Physical Review Letters* 115.3 (2015), p. 036402. DOI: 10.1103/PhysRevLett.115.036402.

[22] C. S. Lee, W. Yang, and R. G. Parr. “Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density”. In: *Physical Review B* 37.2 (Jan. 15, 1988), pp. 785–789. DOI: 10.1103/PhysRevB.37.785.

[23] A. D. Becke. “Density-functional thermochemistry. III. The role of exact exchange”. In: *The Journal of Chemical Physics* 98.7 (Apr. 15, 1993), pp. 5648–5652. DOI: 10.1063/1.464913.

[24] S. Grimme. “Semiempirical hybrid density functional with perturbative second-order correlation”. In: *The Journal of Chemical Physics* 124.3 (2006), p. 034108. DOI: 10.1063/1.2148954.

[25] D. E. Knuth et al. “Comments on “Digital Typography””. In: *TUGboat* 11.4 (1990), pp. 512–528. URL: https://www.tug.org/TUGboat/tb11-4/tb30knut-samuel.pdf.

[26] C. M. Bishop. Pattern Recognition and Machine Learning. Springer, 2006. ISBN: 978-0-387-31073-2.

[27] I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT Press, 2016. ISBN: 978-0-262-03561-3.

[28] F. Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain”. In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519.

[29] J. Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873 (2021), pp. 583–589. DOI: 10.1038/s41586-021-03819-2.

[30] J. Gilmer et al. “Neural Message Passing for Quantum Chemistry”. In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1263–1272.

[31] R. Ramakrishnan et al. Quantum chemistry structures and properties of 134 kilo molecules. figshare. 2014. DOI: 10.6084/m9.figshare.c.978904.v5.

[32] L. Ruddigkeit et al. “Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17”. In: *Journal of Chemical Information and Modeling* 52.11 (Nov. 1, 2012), pp. 2864–2875. DOI: 10.1021/ci300415d.

[33] R. Ramakrishnan et al. “Quantum chemistry structures and properties of 134 kilo molecules”. In: *Scientific Data* 1 (2014).

[34] Q. Sun et al. “PySCF: the Python-based simulations of chemistry framework”. In: *WIREs Computational Molecular Science* 8.1 (Sept. 2017). ISSN: 1759-0884. DOI: 10.1002/wcms.1340.

[35] P. Peduzzi et al. “A simulation study of the number of events per variable in logistic regression analysis”. In: *Journal of Clinical Epidemiology* 49.12 (Dec. 1996), pp. 1373–1379. DOI: 10.1016/s0895-4356(96)00236-3.

[36] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python ”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[37] R. S. Mulliken. “Electronic Population Analysis on LCAO-MO Molecular Wave Functions. I”. In: *The Journal of Chemical Physics* 23.10 (Oct. 1955), pp. 1833–1840. ISSN: 0021-9606. DOI: 10.1063/1.1740588. eprint: https://pubs.aip.org/aip/jcp/article-pdf/23/10/1833/18807925/1833_1_online.pdf. URL: https://doi.org/10.1063/1.1740588.

[38] M. Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Version 2.12.0. Software library. Oct. 5, 2023. URL: https://www.tensorflow.org/.

[39] F. Chollet et al. Keras: The Python Deep Learning API. Version 2.12.0. Software library. Nov. 1, 2023. URL: https://keras.io/.

[40] T. O’Malley et al. KerasTuner. https://github.com/keras-team/keras-tuner. 2019.

[41] S. Hazra, U. Patil, and S. Sanvito. “Predicting the One-Particle Density Matrix with Machine Learning”. In: *Journal of Chemical Theory and Computation* 20 (2024), pp. 4569–4578. DOI: 10.1021/acs.jctc.4c00042.

[42] X. Shao et al. “Machine learning electronic structure methods based on the one-electron reduced density matrix”. In: *Nature Communications* 14 (2023), p. 6281. DOI: 10.1038/s41467-023-41953-9.

[43] K. Schütt et al. “SchNet - A deep learning architecture for molecules and materials”. In: *The Journal of Chemical Physics* 148 (June 2018), p. 241722. DOI: 10.1063/1.5019779.

[44] PyTorch Geometric Development Team. PyTorch Geometric: Deep Learning on Graphs and Irregular Structures. Version 2.3.0. 2024. URL: https://pytorch-geometric.readthedocs.io/.

[45] M. Fey and J. E. Lenssen. “Fast Graph Representation Learning with PyTorch Geometric”. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019. URL: https://arxiv.org/abs/1903.02428.

[46] D. Milacher. scf_guess_datasets. Accessed: 2025-06-27. 2025. URL: https://github.com/hauser-group/scf_guess_datasets.

[47] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. 2019. arXiv: 1711.05101 [cs.LG]. URL: https://arxiv.org/abs/1711.05101.

[48] R. Liaw et al. “Tune: A Research Platform for Distributed Model Selection and Training”. In: *arXiv preprint arXiv:1807.05118* (2018).

[49] L. Li et al. Massively Parallel Hyperparameter Tuning. 2018. URL: https://openreview.net/forum?id=S1Y7OOlRZ.

[50] Quantum Machine. MD Trajectories of $C_7O_2H_{10}$. Contains MD trajectories of 113 QM9 $C_7O_2H_{10}$ isomers, 500 K, 15 fs resolution; energies in ‘id.energy.dat‘. 2025. URL: http://quantum-machine.org/datasets/ (visited on 07/15/2025).

[51] P. Developers. Self-consistent field (SCF) methods. PySCF User Guide. Accessed: 2025-04-09. 2025. URL: https://pyscf.org/user/scf.html.

[52] J. Van Lenthe et al. “Starting SCF calculations by superposition of atomic densities”. In: *Journal of Computational Chemistry* 27.8 (Mar. 23, 2006), pp. 926–932. DOI: 10.1002/jcc.20393.

[53] J. Almlöf, K. Faegri, and K. Korsell. “Principles for a direct SCF approach to LICAO-MO ab-initio calculations”. In: *Journal of Computational Chemistry* 3.3 (Sept. 1982), pp. 385–399. DOI: 10.1002/jcc.540030314.

[54] S. Lehtola. “Fully numerical calculations on atoms with fractional occupations and range-separated exchange functionals”. In: *Physical Review A* 101.1 (Jan. 28, 2020). DOI: 10.1103/physreva.101.012516.

[55] K. T. Schütt et al. “Unifying machine learning and quantum chemistry with a deep neural network for molecular wavefunctions”. In: *Nature Communications* 10.1 (Nov. 15, 2019), p. 5024. ISSN: 2041-1723. DOI: 10.1038/s41467-019-12875-2. URL: https://doi.org/10.1038/s41467-019-12875-2.

[56] Python Software Foundation. Python Language Reference, version 3.11.11. 2024. URL: https://www.python.org.

[57] E. Wachmann. MolGraphNetwork. Accessed: 2025-09-02. 2025. URL: https://github.com/hauser-group/MolGraphNetwork.

[58] D. Milacher and E. Wachmann. scf_guess_tools: Tools for SCF initial guesses. Accessed: 2025-07-30. 2025. URL: https://github.com/hauser-group/scf_guess_tools.