Automating high-resolution atomic force microscopy image interpretation

Niko Oinonen
Automating high-resolution atomic force microscopy image interpretation

Niko Oinonen

A doctoral thesis completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall U135a U7 PWC of the school on 8 December 2023 at 12:00.

Aalto University
School of Science
Department of Applied Physics
Surfaces and Interfaces at the Nanoscale (SIN)
Supervising professor
Professor Adam S. Foster, Aalto University, Finland

Thesis advisor
Dr. Fedor Urtev, Aalto University, Finland

Preliminary examiners
Dr. Christian Wagner, Forschungszentrum Jülich, Germany
Professor Limei Xu, Peking University, China

Opponent
Professor Philip Moriarty, University of Nottingham, United Kingdom
Atomic force microscopy (AFM) has become an important tool in nanoscale studies of matter, imaging and characterizing its properties, reaching from micrometer scale down to the sub-nanometer scale. In particular, high-resolution AFM performed in ultra-high vacuum with functionalized tips has achieved a resolution capable of identifying individual atoms in single molecules adsorbed on surfaces. However, applications so far have been mostly limited to simple planar molecules due to the difficult interpretability of the obtained images for more complicated sample molecules.

At the same time, fast and accurate simulations of high-resolution AFM images have become available in the form of the Probe Particle Model (PPM). The PPM simulations make it easy to go from an atomic structure into an AFM image, but the reverse process is not as easy, often requiring a lot of manual labour with testing different candidate molecule geometries. An automated approach for solving this inverse problem would be a major benefit for the wider applicability AFM into imaging atomic scale systems.

To this end, one can note the recent emergence of large-scale machine learning (ML) models, especially deep neural networks, enabling major progress in many fields of science. Utilizing a very large dataset, it is possible to train an ML model to perform a task where no known algorithmic approach works. Given the rapid data-generation capability of the PPM simulations, it should be possible to train an ML model to perform the inverse-imaging task in AFM, going from an AFM image into a molecule geometry or some other property of interest.

The work in this thesis utilizes the PPM to generate simulations for a large database of molecules, and uses those simulations to train neural networks for predicting the molecule geometry as well as the electrostatic field from AFM images. The trained models are subsequently tested also on experimental AFM images, where generally good results are found, but in some cases differences in the details between the simulation and the real experiment make the predictions incorrect or ambiguous. At this stage full general automation of the AFM image interpretation process is not yet possible, but the results here present the first steps in this direction. Additionally, an ML method for automatizing the tip functionalization part of the AFM measurement preparation is presented.

Korkean resoluution AFM-kuvien simulointi on kuitenkin varsinkin helppoa käytännössä probe particle model (PPM) -simulaatiomallia, jonka avulla on mahdollista tuottaa hyvin nopeasti tarkkoja AFM-simulaatiokuvia mistä tahansa molekyyleistä. PPM-simulaation mahdollistama edellyttää aineistomäärää soveltuu hyvin myös koneoppimismallien koulutuksessa. Koneoppimismallit on mahdollista olla ja aineistot ovat kehitetty tilastollisia malleja, jotka pystyvät ratkaisemaan ongelmia missä mikään tunnettu algoritmi on kyllä. Erityisesti neuroverkkoja on sovellettu moniin ongelmia eri tieteellisissä aloissa.

Tässä väärtöskiirjassa esitellään koneoppimisen perustuva automaatiomenetelmä AFM-kärjen funktionalisoimiseen.
As I now reflect back on the period of time that I have spent on my doctoral thesis work, it strikes me that it has been quite the unusual period of time in the world in more than one way, and I have found myself in a quite lucky position in all of this. First is the coronavirus pandemic and the subsequent lockdown leading to a great number of people around the world to be mostly stuck in their homes for a long time. Many of our experimentalist colleagues struggled with this reality, being unable to get to the lab to conduct their experiments. However, for the computational scientist this was not very much a problem at all. The university office just turned into a home office, and in-person group meetings turned into video calls, while the supercomputers were running just as they were before. Being mostly an indoors person, I even somewhat enjoyed this arrangement, although I do think it took some toll on me at times, as it did for everyone; it’s nice to have at least some human contact every once in a while.

Another aspect turning the world upside down in the past couple of years is the rise of machine learning and artificial intelligence, which I have been able to learn a great deal about during my doctoral research. When I first applied for a summer internship position at my now-supervisor Prof. Adam Foster’s group I did not really have a very good appreciation of what is to come. I only applied because I saw that his group was involved in the application of machine learning into physics research, which seemed like something cool and new to me at the time. Now it’s all the hype, especially after the release of ChatGPT, and it seems like every new commercial product and scientific proposal has to have some machine learning angle to be fashionable. Somehow unwittingly I am left in a relatively good position in this state of affairs with my education in machine learning and related methods, both in terms of doing well in the job market and generally having an appreciation for how these technologies work, as well as being able to have a healthy level of skepticism towards the most out-there claims coming from the AI hype train.

As is so often the case, the work in this thesis was not a solo effort, so
I will take some time here to acknowledge and express gratitude for the contributions from everyone involved. First, I must thank my supervisor Prof. Adam Foster for all of his guidance and support, and for granting me the opportunity to work in the Surfaces and Interfaces at the Nanoscale (SIN) group at the Department of Applied Physics of Aalto University. I appreciate Adam’s no-fuss style of supervision, which has given me the freedom to explore new ideas and directions with a great degree of independence. I also got the opportunity to travel to many places in the world that I probably would not have gone to if had been just up to me. Especially the several trips to Japan stay in my memory and have left me with a sense of affinity towards the city of Kanazawa. And the several conference and workshop talks that I gave during the trips have given me at least some confidence to think that maybe I am not a completely awful public presenter after all.

Adam has gathered a bunch of skilled and knowledgeable people in the SIN group, who I have learned much from. Of these people, I especially thank Prokop Hapala, Ondřej Krejčí, and Fedor Urtev, who were there from the start to guide me on my path. Prokop’s deep knowledge of AFM simulations and computational methods have greatly benefitted me, and the probe-particle simulation model that much of the work in this thesis relies on would not exist without him. Ondřej is another master of SPM simulations, who throughout my stay in the SIN group has always been offering the most useful suggestions, questions, and criticisms. Fedor started his journey into learning about AFM and ML roughly at the same time as I did, and it was a pleasure to share that journey with him. His always-positive attitude towards life made many of my days brighter, and our weekly video calls during the pandemic were a highlight. I won’t list every name here, but I am grateful for all the other members of the SIN group as well, past and present, for all of their advice and suggestions, however big or small. Outside the group, I thank Prof. Alexander Ilin for introducing me to the concept of graph neural networks, which was very important for the direction of the latter half of my thesis work, and I also thank the experimentalists in Prof. Peter Liljeroth’s group, especially Benjamin Alldritt and Chen Xu, for providing me with the experimental data that I could use to test my models.

I also greatly benefitted from all the computational infrastructure at Aalto and in Finland more broadly. Aalto has its own compute cluster called Triton, managed by the Aalto Science-IT project, which I made extensive use of. I also used the Puhti, Mahti, and LUMI supercomputers from CSC. While acknowledging the use of these resources is a formal requirement, I think it goes without saying that having access to four different supercomputers at any given time is genuinely a great privilege. Especially when LUMI came into production at the beginning of this year and there were not many users there yet, it meant having immediate
access to almost unlimited computation resources: 16, 32, 64, as many GPUs as you dare to ask for, always immediately available.

Lastly, I want to thank my parents and my brother and sister for all of their kindness and support over the years.

Espoo, November 15, 2023,

Niko Oinonen
# Contents

1. Introduction ................................. 1

2. Atomic force microscopy ................. 5
   2.1 Scanning probe microscopy ............. 5
   2.2 AFM operation modes .................. 6
   2.3 Tip functionalization .................. 9

3. The Probe Particle simulation model .... 11
   3.1 Quantum mechanics simulations ........ 13
   3.2 The probe particle .................... 15
   3.3 Computational procedure ............... 16
   3.4 Force-field models .................... 18
      3.4.1 Pair-wise potential ............... 18
      3.4.2 Hartree electrostatic force ....... 21
      3.4.3 Full-density based model ......... 24

4. Machine learning and neural networks ... 29
   4.1 Overview of machine learning concepts .. 30
   4.2 Neural networks ....................... 32
List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.


Author’s contributions

Publication I: “Automated structure discovery in atomic force microscopy”

F.S., P.L., and A.S.F. conceived the research. P.H., N.O., and F.U. developed the machine learning software. P.H., N.O., F.U., and O.K developed the AFM simulation software and ran the AFM simulations. F.F.C ran the quantum simulations. B.A. performed the experiments. J.K, P.L, and A.S.F. supervised the research. All authors were involved in the results analysis and contributed to the manuscript.

Publication II: “Electrostatic Discovery Atomic Force Microscopy”

P.L. and A.S.F. conceived the research. N.O. and F.U. developed the machine learning software. P.H., N.O., F.U., and O.K developed the AFM simulation software and ran the AFM simulations. F.F.C ran the quantum simulations. B.A., C.X., and S.C. performed the experiments. J.K, P.L, and A.S.F. supervised the research. All authors were involved in the results analysis and contributed to the manuscript.

Publication III: “Molecule graph reconstruction from atomic force microscope images with machine learning”

N.O., A.I., and A.S.F. conceived the research. N.O. and L.K. developed the software and ran the simulations. N.O. wrote the original manuscript. A.I. and A.S.F. supervised the research. All authors were involved in the results analysis and reviewed the manuscript.
Publication IV: “Automated tip functionalization via machine learning in scanning probe microscopy”

B.A., M.A., J.K., P.L., and A.S.F. conceived the research. B.A. and F.U. developed the methodology and software, curated the data, and wrote the original manuscript. B.A. performed the experiments. B.A, F.U., and M.A. validated the method. N.O. aided in the software development. J.K., P.L., and A.S.F. supervised the research. All authors reviewed the manuscript.
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Examples of CO-AFM images</td>
<td>10</td>
</tr>
<tr>
<td>3.1</td>
<td>Graphical user interface for the probe particle model</td>
<td>12</td>
</tr>
<tr>
<td>3.2</td>
<td>Components in the probe particle model and probe particle paths</td>
<td>16</td>
</tr>
<tr>
<td>3.3</td>
<td>Comparison of probe particle model simulations using different force field models</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>Schematic of a multi-layer perceptron neural network</td>
<td>33</td>
</tr>
<tr>
<td>4.2</td>
<td>Schematic of a 2D convolution operation</td>
<td>37</td>
</tr>
<tr>
<td>4.3</td>
<td>Schematic illustration of the Attention U-net model</td>
<td>39</td>
</tr>
<tr>
<td>4.4</td>
<td>Schematic illustration of the Attention Gate</td>
<td>41</td>
</tr>
<tr>
<td>4.5</td>
<td>Schematic of a graph neural network</td>
<td>42</td>
</tr>
<tr>
<td>4.6</td>
<td>Example of overfitting a polynomial</td>
<td>44</td>
</tr>
<tr>
<td>4.7</td>
<td>AFM data augmentation techniques</td>
<td>46</td>
</tr>
<tr>
<td>5.1</td>
<td>Descriptors for an atomic system</td>
<td>53</td>
</tr>
<tr>
<td>5.2</td>
<td>Examples of CNN image descriptor predictions from simulated and experimental data</td>
<td>55</td>
</tr>
<tr>
<td>5.3</td>
<td>Identification of the adsorption configurations of 1S-camphor on Cu(111)</td>
<td>56</td>
</tr>
<tr>
<td>5.4</td>
<td>ES Map predictions on simulated AFM images</td>
<td>58</td>
</tr>
<tr>
<td>5.5</td>
<td>Comparison of simulated and experimental predictions of ES Maps for PTCDA</td>
<td>59</td>
</tr>
<tr>
<td>5.6</td>
<td>Comparison of simulated and experimental predictions of ES Maps for BCB</td>
<td>60</td>
</tr>
<tr>
<td>5.7</td>
<td>Comparison of simulated and experimental predictions of ES Maps for a water cluster on Cu(111)</td>
<td>60</td>
</tr>
<tr>
<td>5.8</td>
<td>ES Map prediction and reference with Hartree electrostatics</td>
<td>61</td>
</tr>
<tr>
<td>5.9</td>
<td>Schematic of the graph construction model</td>
<td>63</td>
</tr>
<tr>
<td>5.10</td>
<td>Example graph predictions for simulated AFM images</td>
<td>65</td>
</tr>
<tr>
<td>5.11</td>
<td>Statistics on missing and extra atoms in graph construction</td>
<td>66</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.12</td>
<td>Confusion matrices for classification in graph construction</td>
<td>68</td>
</tr>
<tr>
<td>5.13</td>
<td>Schematic of the one-shot graph construction model</td>
<td>69</td>
</tr>
<tr>
<td>5.14</td>
<td>Workflow for structure discovery in AFM images of ice nanoclusters</td>
<td>72</td>
</tr>
<tr>
<td>5.15</td>
<td>Geometry predictions for ice nanoclusters</td>
<td>73</td>
</tr>
<tr>
<td>5.16</td>
<td>On-surface geometries of ice nanoclusters optimized with a neural network potential, corresponding simulated AFM images, and comparison to experiment</td>
<td>74</td>
</tr>
<tr>
<td>6.1</td>
<td>Schematic pipeline of the autonomous tip preparation procedure</td>
<td>78</td>
</tr>
<tr>
<td>6.2</td>
<td>Formation of the training database with CO-functionalized tips</td>
<td>79</td>
</tr>
<tr>
<td>6.3</td>
<td>Example predictions of CO tips from the test set</td>
<td>80</td>
</tr>
<tr>
<td>6.4</td>
<td>A summary of statistics of the CO tip classifier</td>
<td>81</td>
</tr>
<tr>
<td>A.1</td>
<td>ES Map predictions using Hartree potential electrostatics</td>
<td>101</td>
</tr>
<tr>
<td>A.2</td>
<td>Full 3D position grid predictions</td>
<td>102</td>
</tr>
<tr>
<td>A.3</td>
<td>Graph construction sequences</td>
<td>103</td>
</tr>
<tr>
<td>A.4</td>
<td>Graph predictions with shifted coordinates</td>
<td>105</td>
</tr>
</tbody>
</table>
List of Tables

5.1 Precision and recall for atom type classification in molecule graph construction ........................................ 67
5.2 Precision and recall for bond connection classification in molecule graph construction ............................. 67
6.1 CO-tip classifier model architecture ............................... 78
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFM</td>
<td>Atomic force microscope/microscopy</td>
</tr>
<tr>
<td>STM</td>
<td>Scanning tunneling microscope/microscopy</td>
</tr>
<tr>
<td>SPM</td>
<td>Scanning probe microscope/microscopy</td>
</tr>
<tr>
<td>FM</td>
<td>Frequency modulation</td>
</tr>
<tr>
<td>AM</td>
<td>Amplitude modulation</td>
</tr>
<tr>
<td>PP</td>
<td>Probe particle</td>
</tr>
<tr>
<td>PPM</td>
<td>Probe particle model</td>
</tr>
<tr>
<td>DFT</td>
<td>Density functional theory</td>
</tr>
<tr>
<td>vDW</td>
<td>van der Waals (force)</td>
</tr>
<tr>
<td>LJ</td>
<td>Lennard-Jones (potential)</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier transform</td>
</tr>
<tr>
<td>FD</td>
<td>Finite difference</td>
</tr>
<tr>
<td>FDBM</td>
<td>Full-density based model</td>
</tr>
<tr>
<td>ML</td>
<td>Machine learning</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-layer perceptron</td>
</tr>
<tr>
<td>NN</td>
<td>Neural network</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional neural network</td>
</tr>
<tr>
<td>GNN</td>
<td>Graph neural network</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent neural network</td>
</tr>
<tr>
<td>MPNN</td>
<td>Message-passing neural network</td>
</tr>
<tr>
<td>NNP</td>
<td>Neural network potential</td>
</tr>
<tr>
<td>PTCDA</td>
<td>Perylenetetracarboxylic dianhydride</td>
</tr>
<tr>
<td>BCB</td>
<td>1-bromo-3,5-dichlorobenzene</td>
</tr>
</tbody>
</table>
Symbols

\[ E \quad \text{Energy, edges in a graph} \]
\[ V \quad \text{Electrostatic potential, nodes in a graph} \]
\[ F \quad \text{Force} \]
\[ \rho \quad \text{Charge density} \]
\[ \varepsilon_0 \quad \text{Vacuum permittivity} = 8.8541878128 \times 10^{-12} \text{F/m} \]
\[ \mathbf{r}_{ab} \quad \text{Vector from } a \text{ to } b \]
\[ \hat{x} \quad \text{Unit vector} \]
\[ \nabla_r \quad \text{Gradient with respect to } \mathbf{r} \]
\[ f \star g \quad \text{Cross correlation of function } f \text{ with function } g \]
\[ \mathcal{F}(f), \hat{f} \quad \text{Fourier transform of function } f \]
\[ \mathcal{O}(f(N)) \quad \text{The "Big O" notation for computational complexity} \]
\[ W \quad \text{Neural network weight tensor} \]
\[ b \quad \text{Neural network bias vector} \]
\[ \mathcal{N}(v) \quad \text{Neighbours of a node } v \text{ in a graph} \]
\[ h_v \quad \text{Embedding vector for a node } v \text{ in a graph} \]
\[ g_e, g_{uv} \quad \text{Embedding vector for an edge } e = (u, v) \text{ in a graph} \]
1. Introduction

Seeing and understanding matter at ever smaller scale and higher level of detail has been one of the driving factors of scientific inquiry since its inception. The first microscopes were invented in the renaissance era when the astronomers of the time thought to turn their telescopes around to look down at the small instead of up at the big. The optical microscopes soon discovered the cell structure of biological organisms and through refinement of the lens systems could see ever smaller objects, but ultimately ran into a fundamental wall of physics, the diffraction limit. Alternative techniques using the diffraction of X-rays and neutrons could resolve the atomic structure of matter, but at the cost that the sample needs to have a periodic crystal structure, therefore giving up the goal of resolving individual atoms or molecules. Among the first measurement techniques that could resolve individual atoms below the diffraction limit was the scanning tunneling microscope (STM), invented in 1981 [1–3] and soon followed by the closely related atomic force microscope (AFM) in 1986 [4]. Both STM and AFM, as well as the wider umbrella of scanning probe microscopy (SPM), stand to this day among the primary techniques for imaging and characterizing as well as manipulating matter at the atomic scale [5]. The focus here is on AFM, which especially stands out as the only method of atomic-scale imaging without any principal material restrictions.

The main component in all SPM devices is an atomically sharp probe tip, which is used to scan a sample at an extremely close distance. In AFM specifically, the measured signal originates from the atomic-scale forces between the tip and the sample. As an additional trick, high-resolution AFM imaging uses an additional atom or small molecule, often a CO molecule, attached to the apex of the probe tip, thus functionalizing it and providing a reliable atomically well-defined tip and a chemically non-reactive interaction with the sample. When performed in ultra-high vacuum at a near-absolute-zero temperature, this measurement technique has offered a reliable way of achieving atomic resolution capable of identifying atoms within individual molecules adsorbed on surfaces [6]. AFM imaging with
Introduction

Functionalized tips can be used for identifying the structures of molecules [7, 8], but is in general useful for the characterization of atomic-scale systems and processes, finding use in for example bond-order discrimination [9] and studying the reactants and products of on-surface chemical reactions [10–12]. In addition to imaging capabilities, the sharp tip of the SPM device can be used for manipulating atoms and molecules, individually picking them up and placing them down to desired positions [5]. This was delightfully demonstrated by researchers at IBM, who made a stop-motion animation titled *A Boy and His Atom* where the characters were drawn by carefully placing CO molecules on a copper surface [13].

Despite the successes, there remain challenges that limit the applicability of AFM. One of the key challenges is the generally difficult interpretability of the obtained images. Applications so far have been mostly focused on planar molecules with a carbon backbone, where the chemical structure of the molecule is seen clearly as sharp lines almost like a chemical diagram for the molecule [6]. However, for molecules with a more three-dimensional (3D) geometry and more diverse chemical structure, the obtained AFM images become much more difficult to interpret [11, 14], and often require much manual labor with expensive calculations to decipher the sample structure [8]. On the other hand, accurate simulations of tip-functionalized AFM have become readily available with the Probe Particle simulation model [15]. The rapid data-generation capability of the simulation model provides a good target for machine learning as a solution to the image interpretation problem.

Machine learning (ML) has been emerging as a new paradigm for problem-solving in science and engineering. Traditionally, a computer is programmed to perform fixed set of actions, whereas an ML system rather discovers a good algorithm from a dataset of examples. Especially the field of deep learning [16], employing artificial neural networks, has progressed quickly in the last decade, solving many problems under the broad category of artificial intelligence research. The most notable recent example is the development of large language models such as the GPT-3 [17] and GPT-4 [18] models underlying OpenAI’s ChatGPT service, which has quickly become a household name for its capability to understand and generate text, displaying a degree of problem-solving capability [19]. Other applications include computer vision [20–22], speech recognition [23, 24], playing video games and board games [25, 26], planning and decision-making [27], and robotics [28].

My objective in this thesis is to develop ML-based methods for solving the interpretation problem in tip-functionalized AFM image analysis. More specifically, the goal is to extract properties of molecules, such as the atomic coordinates, chemical elements, and the electrostatic field, from high-resolution AFM images. This is achieved by establishing workflows of simulated AFM data generation with the Probe Particle model for ML
model training, and applying the trained models to experimental AFM images. An additional application of ML for automating the AFM tip functionalization is also presented.

Following this introduction, the thesis is divided into six chapters of which the first three provide a background on relevant methods, the next two describe the new applications of these methods, and the last chapter concludes the thesis. The methods used can be divided into two topics: AFM and ML. First, the physics and operating principle behind the AFM device are presented in chapter 2 which describes AFM as an experimental tool, and chapter 3 which describes the Probe Particle model for simulating high-resolution AFM images. Following that, the mathematics and computational methods of machine learning, and deep learning in particular, are presented in chapter 4. The next two chapters present the new contributions of this thesis to the scientific literature. The bulk of these are contained in chapter 5 which describes the application of machine learning into automatically discovering the structure and other properties of molecular systems from AFM images. The following chapter 6 describes another application of machine learning into automatically preparing an AFM device for operation. Finally, chapter 7 summarizes the results and concludes the thesis with remarks on the future directions that the research could take.
2. Atomic force microscopy

Atomic force microscopy (AFM) is the experimental technique that is the focus of this thesis. This chapter discusses the basics of AFM as an experimental technique, starting from the wider umbrella of techniques known as scanning probe microscopy, going into the different operation modes of an AFM device, and finally discussing tip functionalization as the major underpinning of high-resolution AFM studies in recent times. Many parts of this chapter are based on the excellent review by Giessibl [29].

2.1 Scanning probe microscopy

The main idea behind scanning probe microscopy (SPM) is that there is an extremely sharp probe tip that is brought into close proximity with a sample, and some interaction between the tip and the sample is measured. A scan over the sample produces an image of the interaction, which can be interpreted as the topography or some other localized property of the sample. The first SPM technique invented was the scanning tunneling microscope (STM), which was introduced by Binnig, Rohrer, Gerber, and Weibel in 1981 [1–3], winning the Nobel Prize in physics for Binnig and Rohrer in 1986. In an STM device the measured interaction is the tunneling current between the tip and the sample, which has an exponential characteristic,

\[ I_T \propto V \exp(-C\sqrt{\phi d}) \]

where \( V \) is the voltage between the tip and the sample, \( \phi \) is the height of the tunneling barrier, and \( d \) is the distance between the tip and the sample. The constant \( C \) that determines the distance scale is typically such that a 1 Å change in the distance results in an order of magnitude change in \( I_T \) [3]. This incredible sensitivity to distance gives the STM the ability to probe the topography of samples at the atomic scale, and indeed, in 1983 Binnig et al. managed to resolve the 7x7 reconstruction of the Si(111) surface using STM [30], the first real-space determination of this structure.

During the development of the STM, it was noticed that when the tip gets close enough to the sample to produce a measurable current, there
are also significant forces present between the tip and the sample. Binnig et al. [4] put these forces into use when they introduced AFM in 1986 as a complimentary technique to STM. The advantage of AFM is that the sample can be insulating, which does not work for STM which requires a current to flow through the sample. However, achieving atomic resolution with AFM was more challenging than with STM. One of the reasons for the early success of the STM is that the simple exponential characteristic of the signal makes it relatively easy to construct a feedback loop that maintains a stable signal. In contrast, for AFM the force interaction between the tip and the sample is more complicated, arising from many sources such as the short range Pauli repulsion, the longer range electrostatic forces, and in ambient conditions effects such as the meniscus force, caused by surface tension on adhesion layers on the tip and sample. Indeed, it took until 1994 before the same Si(111) 7x7 reconstruction that was easily imaged with STM, could also be imaged with AFM [31, 32].

In the STM instrument, the tunneling tip can be attached to a scanner that can freely position the tip anywhere on the sample. In contrast, the AFM tip must be attached to a cantilever, a flexible beam that bends like a spring under the force interaction. The cantilever should be flexible only in one direction and stiff in other directions so that the motion of the cantilever is one-dimensional. It should be noted that the AFM device can also be simultaneously operated as an STM as long as the cantilever is stiff enough.

Over time SPM has developed into an indispensable tool for surface science studies, allowing the imaging and characterization of materials and their properties, as well as enabling the manipulation of matter on the level of individual atoms [5]. Furthermore, AFM is not only used for atomic scale studies, but can also, for example, capture the dynamics of biologically relevant materials such as cell membranes and proteins at sub-second time resolution [33, 34].

2.2 AFM operation modes

The AFM device, true to its name, is probing a sample by its force interaction. However, a force interaction between atoms is not something that can be directly read off a measuring scale, and rather the measurement is of some other quantity that can be related to the force interaction. There are several operation modes for AFM device that relate the force to the measurement in different ways.

The original and perhaps the simplest AFM operation mode is the static mode, also known as the contact mode, where the tip is brought close to the sample, and the force interaction statically deflects the cantilever up or down depending on the force that is exerted on it. If the mechanical
properties of the cantilever are known, specifically the harmonic spring constant $k_c$, then the magnitude of the deflection $\Delta z$ can be related to the tip-sample force $F_{ts}$ by Hooke’s law

$$F_{ts} = -k_c \Delta z.$$  \hfill (2.1)

The tip is then dragged across the surface, recording the deflection at every point, thereby producing a map of the forces.

The deflection must also be converted into an electrical signal to be measured. A typical method uses a laser beam that is reflected off the back of the cantilever into several photodetectors, such that the deflection of the cantilever can be measured as relative change in signal strength in the different photodetectors. Alternative measurement methods include the use of piezoresistive and piezoelectric cantilevers where the deflection of the cantilever alters its electric resistance or electric potential. The piezoelectric quartz tuning forks are especially used in high-resolution AFM.

While the static operation mode was able to achieve atomic-scale contrast on some samples [35], the technique faced several difficulties in its operation that limited its general applicability. One of these difficulties is the snap-to-contact problem where the tip-sample forces overcome the restoring force of the cantilever and the tip sticks to the surface. This can deform both the tip and the surface and is especially a problem on reactive surfaces such as the Si(111) surface. In part to address these problems, the other major category of operation modes for AFM, the dynamic mode, was introduced. In the dynamic mode the cantilever is driven to oscillate perpendicular to the sample, and the force interaction is measured indirectly by monitoring changes in the oscillation. The dynamic mode measurements can be further divided into the amplitude modulation (AM) and frequency modulation (FM) modes, which will be discussed in the following.

The AM mode, also called the tapping mode, was introduced in 1987 by Martin, Williams, and Wickramasinghe [36] as a method for measuring the force derivative arising from the attractive van der Waals interaction between the tip and the sample. In the AM mode, the cantilever is driven at a constant amplitude at a frequency close to the eigenfrequency $f_0$ of the cantilever, and the modulation of the oscillation amplitude is used as the feedback signal. The amplitude changes on a timescale $\tau_{AM} \approx 2Q/f_0$, where $Q$ is the quality factor of the oscillating cantilever system [29]. In vacuum the Q-factor can reach up to $Q \sim 10^5$, which makes the amplitude change very slowly. For this reason, the AM mode is not well suited for high-resolution imaging in vacuum conditions. However, in ambient conditions where the viscous drag on the cantilever naturally lowers the quality factor ($Q \sim 100$), the AM mode is more commonly used.
The FM mode, also called the non-contact mode was introduced in 1991 by Albrecht et al. [37] as a measurement technique more suitable for systems with a high Q-factor. In FM-AFM the cantilever is driven at its eigenfrequency such that the amplitude remains constant, and the modulation of the frequency is used as the measurement signal. The change in frequency happens on a timescale $\tau_{FM} \approx 1/f_0$ proportional to a single oscillation cycle regardless of the Q-factor, which makes the use of FM-AFM in vacuum a better choice than AM-AFM. This is also the operation mode of choice for all of the AFM data presented in this thesis, and is discussed in more detail in the following.

The oscillating AFM cantilever can be modelled as a harmonic oscillator with spring constant $k_c$, mass $m_c$, and eigenfrequency $f_0 = \frac{1}{2\pi} \sqrt{\frac{k_c}{m_c}}$. (2.2)

The mass here is not the real mass of the cantilever, which is distributed over its whole length, but rather a smaller effective mass that would be attached to the end of a massless cantilever. The oscillation frequency is modified by the interaction between the tip and the sample and becomes $f = f_0 + \Delta f$. The frequency change $\Delta f$ is the target measurement signal in an FM-AFM setup. Using a simple approximation where the negative tip-sample force gradient $k_{ts} = -\partial F_{ts}/\partial z$ is constant over the whole oscillation path and is much smaller than the cantilever spring constant, $k_{ts} \ll k_c$, then the frequency change to first order is given by

$$\Delta f = \frac{k_{ts}}{2k_c} f_0.$$  (2.3)

Roughly speaking, the force gradient is positive in the attractive regime and negative in the repulsive regime, which leads to the conclusion that the frequency shift is positive for repulsive interaction and negative for attractive interaction. In practice the tip-sample interaction does not remain constant, in which case the interaction has to be integrated over the oscillation path as shown in Eq. (3.4) in the next chapter.

In addition to the oscillation frequency, another important aspect is the oscillation amplitude $A$. In order to avoid the snap-to-contact problem, the restoring force $-k_c A$ needs to be higher than the tip-sample force. This means that for soft cantilevers the amplitude needs to be high in order to ensure stable operation. On the other hand, if a stiff cantilever is used, then the amplitude can also be much smaller. A common choice is the qPlus sensor [38], made from a quartz tuning fork of a watch, which has a large spring constant, typical value $k \sim 1800 \text{N/m}$ [29], enabling small-amplitude operation which is important when probing short-range forces.

If the cantilever was allowed to oscillate freely, the tip-sample interaction and other sources of damping would eventually reduce the oscillation...
amplitude to zero. The cantilever therefore has to be driven to maintain the oscillation, and in FM operation specifically, the amplitude needs to remain constant. This is achieved by connecting the cantilever system into a positive feedback loop where the driving signal is adjusted based on the current amplitude in order to keep it constant [39]. The cantilever deflection is first measured and the root-mean-square amplitude is converted into a digital signal by an analog-to-digital converter. Based on the difference between the set-point amplitude and the current amplitude, a PI (proportional-integral) controller adjusts the gain on the driving signal, which is phase-shifted by 90° from the cantilever oscillation. Finally, the signal is fed into a piezoelectric actuator which the cantilever is attached to, thus completing the circuit. Since the cantilever is driven at its resonance frequency, the required driving amplitude is much smaller (by a factor of \(1/Q\)) than the cantilever oscillation amplitude. Additionally, the frequency shift can be measured with a phase-locked loop [39].

In addition to the operating modes discussed here, there exist several more specialized operation modes and variations on the AFM technique. To list a couple, there is the lateral force microscope [40, 41] where the cantilever is mounted vertically and oscillates in a direction parallel to the sample surface allowing measurement of frictional forces at the atomic scale, and the Kelvin probe force microscope [42] where the tip-sample system functions like parallel-plate capacitor and the contact potential difference, related to the work function of the sample, can be measured.

### 2.3 Tip functionalization

While AFM was able to resolve the atomic structure of solid surfaces within a decade of its inception, resolving the chemical structure of single molecules adsorbed on surfaces remained a great challenge for much longer. The break-through development in this direction came in the form of tip functionalization where an additional atom or other small particle is attached to the tip apex. This technique was known to enhance the resolution of STM images [43, 44] and was first applied to AFM in 2009 by Gross et al. [6] who used a CO-terminated tip with FM-AFM to resolve the chemical structure of a pentacene molecule on Cu(111) at atomic resolution.

Imaging with a bare-metal tip can locate the molecule as an individual object on the surface, but atomic resolution is impossible because the tip will influence the position of the sample molecule when getting too close, either laterally displacing it or picking it up on the tip [6]. In contrast, the functionalized tip is chemically inert and flexible, which makes it possible to get much closer to the sample and probe the short range Pauli repulsion [45] without significantly disturbing the sample, yielding a dramatic increase in the obtainable resolution. Figure 2.1A
shows an example of how it can be quite easy to see the overall structure of a molecule from an AFM image obtained with a CO-terminated tip. I note here that in the following chapters whenever I am talking about AFM images, I am specifically referring to this kind of tip-functionalized high-resolution FM-AFM images obtained in ultra-high vacuum.

While the functionalization simplifies the measurement in that it is less dependent on the exact nature of the tip, it introduces other complicating factors to the interpretation of the images. For example, different functionalizations result in different contrast for the same molecule [6], and the flexibility of the tip distorts the observed geometry so that the apparent size of features in the molecule may be significantly different from their real size. These problems become even more apparent when the molecule has a more three-dimensional structure, as is exemplified in the CO-AFM image of 1S-camphor in Fig. 2.1B. This is the problem that the work in this thesis is trying to solve.
3. The Probe Particle simulation model

This chapter discusses the Probe Particle Model (PPM), which is the simulation model and software I used for the AFM simulations in this thesis. PPM was originally invented by Prokop Hapala [15], who was also one of the key figures in getting the research in this thesis started (co-author on Publications I and II). The simulation model plays a very important part in the whole work, since the machine learning models are trained using the simulation data. Additionally, a significant part of the work for this thesis went into making implementations for the simulation methods in order to extend the capabilities of PPM. Therefore, it is worth spending some time discussing the details of how PPM works and all the different force-field models that can be employed.

Every computational model needs an implementation, and the original implementation of PPM was made in Fortran, but was then rewritten in C++ with a Python interface, which is the current main version of PPM and is publicly available as an open-source software [46]. More recently there has been effort to make a more efficient implementation of PPM in OpenCL, which enables the use of a graphics processing unit (GPU) as an accelerator. This reduces the computation time by several orders of magnitude, going from tens of seconds to tens of milliseconds. To make a distinction between the two implementations, in the following I will refer to the C++ implementation as the CPU version and the OpenCL implementation as GPU version (even though OpenCL code can also be made to run on CPUs).

When starting the work on this thesis, many of the features present in the CPU version did not yet exist in the GPU version, and the acceleration of the simulations would be very useful for the machine learning application which benefits from a large amount of training data. Additionally, a faster-running simulation would allow for shorter cycles in experimenting with different parameters of the simulation and expanding the dataset. Therefore, it was worth investing effort into making implementations of the missing force-field models in the GPU version. A significant effort on my part, as well as others, has recently also gone into restructuring the
The Probe Particle simulation model

Figure 3.1. Graphical user interface for the probe particle model. The GPU version of the PPM running under the hood can generate a simulated AFM image nearly instantly after changing any of the parameter values.

PPM code repository to be a proper Python package that is installable with the official *pip*-installer for Python packages. As an additional side-project, I also contributed to making a graphical user interface to PPM (Fig. 3.1), which should help in making the simulation model more accessible to people with less coding experience, as well as just being an easy interface for quickly testing the effect of different parameters for the simulation.¹

The previous chapter described the operating principle behind atomic-resolution AFM measurements, being reliant upon engineering the AFM tip down to the point of having only a single molecule standing at the apex of the tip. However, understanding what is actually going on at the tip apex at the atomic scale is extremely difficult, practically impossible from just the experimental data alone. For this reason, we require a good theoretical understanding of the tip-sample system in order to answer questions about the true origin of the contrast formation in the AFM images. To this end, we will start by considering a first-principles quantum-mechanical approach before arriving at the more practical approach of classical force

¹At the moment of writing, we are working on a manuscript on the recent advancements in PPM under the title *Probe-particle model: Advances in simulations of Scanning Probe Microscopy with flexible tip-apices* with the authors Niko Oinonen, Aliaksandr Yakutovich, Aurelio Galardo, Martin Ondráček, Prokop Hapala, and Ondřej Krejči.
fields used in PPM.

3.1 Quantum mechanics simulations

If one wants to accurately understand the structure and dynamics of systems at the atomic scale, then in principle one always has to solve the Schrödinger equation for the system. In the case of high-resolution AFM, the system is typically cooled to a temperature close to absolute zero, so the system stays (mostly) in its ground state and we can simplify by just considering the time-independent Schrödinger equation

\[ \hat{H} |\Phi\rangle = E |\Phi\rangle, \tag{3.1} \]

where \( \hat{H} \) is the Hamiltonian operator, \( |\Phi\rangle \) is the wave function, and \( E \) is the energy of the system. In AFM the interesting quantity is the force, which can be defined for a particle in the system as the negative change in the total energy with respect to the change in position \( r_i \) of the particle,

\[ F_i = -\nabla_{r_i} E. \tag{3.2} \]

As an additional simplification, quantum simulations very often use the Born-Oppenheimer approximation, which assumes that the nuclei of the atoms are fixed in space so that only the electrons need to be considered. This can be justified in most cases by just considering the fact that the nuclei are much heavier than electrons, so their dynamics are much slower than those of the electrons. Even with these approximations, the Schrödinger equation is very difficult (or impossible) to solve exactly for most systems of interest, and one has to use numerical computational methods instead.

The task of the computation is to find the ground state of the electron wave function. The wave function is a function of the coordinates of all of the electrons in the system, so in three spatial dimensions the wave function has a \( 3N \)-dimensional variable space for an \( N \)-electron system. If this variable space is discretized to \( M \) points on each axis, then the size of the grid grows exponentially as \( M^{3N} \). This makes a naive solution of the equation intractable for all but the smallest systems. Therefore, better methods are required to do calculations on practically sized systems. Perhaps the most commonly used of these numerical methods is density functional theory (DFT) [47]. While DFT is not used directly for the AFM simulations in this thesis, it is still used for generating the prerequisite data for the PPM simulations, so we will look at some of the basics of DFT in the following.

As a starting point, DFT simplifies the problem of determining the electron wave function by not considering all of the electrons as separate
entities, but rather asking what is the density $\rho(\mathbf{r})$ of the electrons at all points $\mathbf{r} \in \mathbb{R}^3$ in the usual three-dimensional space: $\Psi(\mathbf{r}_1, \mathbf{r}_2, \ldots) \rightarrow \rho(\mathbf{r})$. The reason why this approach works lies in the two theorems referred to as the Hohenberg-Kohn theorems [48]. The first theorem states that the ground-state density $\rho_0(\mathbf{r})$ uniquely determines the external potential $v_{\text{ext}}(\mathbf{r})$ experienced by the electrons and the second theorem states that there exists a functional of the density, $E[\rho(\mathbf{r})]$, that determines the energy of the system for any external potential $v_{\text{ext}}$, and the ground-state density $\rho_0(\mathbf{r})$ minimizes this energy. The consequence of these theorems is that DFT is, at least in principle, an exact theory for the ground-state of the system. However, in practice the energy functional contains a part that needs to be approximated, the so-called exchange-correlation functional, where all of the unknown quantities of the system are packed into. There exists a whole zoo of different forms for this functional with varying degrees of computational complexity and accuracy optimized for different kinds of systems [49]. Additionally, other approximations have to be made in the name of numerical implementations of the algorithms, such as the choice of a basis set and an integration grid. This aspect of DFT makes it so that practical application of DFT often requires a lot of care and experience to ensure that the found result can be relied upon.

In the most basic form, DFT has a computational time complexity that scales as $\mathcal{O}(N^3)$ in the number of particles, which is a marked improvement over the exponential scaling of the direct solution. However, the computational cost still becomes prohibitive very quickly with the increasing size of the system, and in practice, the biggest systems computable with DFT are of the order of $\sim 10^3$ atoms. This is still very far from simulating anything approaching macroscopic scale, but it is a very useful tool for investigating many practically interesting systems at the atomic scale.

Returning to the topic of AFM simulations, we could in principle use DFT for simulating all the dynamics of the AFM tip-sample system, at least for small samples. While some authors have taken this approach [45, 50], due to many of the issues mentioned above, setting up the DFT calculation correctly can be difficult, and the computational cost is very high. This makes it impractical to use DFT as a routine tool for theoretical analysis of AFM images, and certainly makes it impossible to use DFT for high-throughput AFM simulations at the scale of the data required for machine learning applications. Moreover, there is actually little need to do first-principles calculations to perform accurate simulations of AFM images due to how the functionalized-tip AFM experiment works. It turns out that, in order to accurately reproduce the kind of contrast that is observed in experimental AFM images, it is adequate just to simulate the dynamics of the particle on the functionalized tip using classical force-field models. This is the approach taken by PPM, which will be discussed in more detail in the following.
3.2 The probe particle

The idea behind atomic simulations using classical force fields is to forget about the internal electronic structure of the atoms, and just consider the atoms as point particles following the dynamics dictated by Newton’s laws of motion. A force field is formally a vector field that maps points in space to force vectors, but in molecular dynamics simulations a force field also refers specifically to the functional form of the potential energy of the system that defines the forces that the atoms experience. This functional form is empirically constructed to mimic the dynamics of real systems as closely as possible. In PPM, this approach is used for simulating the motion of the particle functionalizing the AFM tip as the cantilever executes its oscillating motion. The reason why this works lies in the fact that the functionalized tip is chemically inert, so the quantum mechanical nature of its interactions with the sample do not play a significant role (aside from Pauli repulsion). This is computationally dramatically more efficient: simulating even a single approach of the oscillating AFM tip with DFT can take from hours to days on 100+ CPU cores, whereas a complete PPM simulation on a single GPU takes $0.01 \text{s} \sim 0.1 \text{s}$.

The central element of the PPM simulation is its namesake probe particle (PP), which is a model of the atom or molecule that functionalizes the tip apex in the AFM device, most commonly a CO molecule. It should be noted that the PP here is just a single point-particle even when the real particle is a composite system. In the case of the CO molecule, the oxygen atom is the one pointing towards the sample [43], so in this case the PP would be the oxygen atom with interaction parameters chosen so that the approximate behaviour of the complete CO molecule is replicated.

The other two elements of the simulation are the tip and the sample that the PP interacts with. The force on the PP is a sum of its interaction with the tip (T) and the sample (S):

$$F_{PP}(r) = F_T(r) + F_S(r). \tag{3.3}$$

We consider here the force that the PP experiences, but ultimately the measured signal of the AFM device comes from the force that the tip experiences. We assume that the tip and the sample are non-interacting, so that the only force on the tip is the one exerted on it by the PP, which is equal and opposite to the force that the tip exerts on the PP, $-F_T$. Since the PP is very small and light compared to the macroscopic tip, it relaxes into an equilibrium position on a much shorter timescale than the movement of the tip. Therefore, at any given point $r$ on the trajectory of the tip’s oscillation path, we can say that $F_{PP} = 0$, and consequently the force on the tip is $-F_T(r) = F_S(r + \Delta r)$. The additional offset $\Delta r$ is to account for the fact that the PP relaxes to some position that is offset from the position of the
tip. The task of the simulation is to find this offset \( \Delta r \) for every position of the tip and the corresponding force \( F_S \) that the sample exerts on the PP at that position, in order to construct a map of the force that the tip experiences. This idea is illustrated in Fig. 3.2.

An additional detail to note here is that the sample can sometimes be soft and could be significantly modified by its interaction with the PP. However, trying to accurately simulate the movement of the atoms in the sample would require a much more expensive calculation, possibly going back to the quantum-mechanics approach. Therefore, we will assume that the sample is static as well, so that we can focus solely on modelling the behaviour of the PP. For small molecules that stick firmly onto the surface, this is not a concern, but for larger molecules that may have more flexible parts this approximation is less valid and can become one of the biggest sources of error in the simulation.

### 3.3 Computational procedure

The procedure that PPM takes to simulate an AFM image can be divided into three parts:

1. Define a grid of positions for the PP, and at each position compute the force that the sample exerts on the PP based on the chosen force field model (Sec. 3.4). The result is a discrete vector field of forces.
2. Do a 3D raster scan over the sample, relaxing the PP at each position...
The Probe Particle simulation model

using the pre-computed grid of forces for the sample-PP interaction, and a harmonic force calculated on the fly for the tip-PP interaction.

3. At each position, integrate the computed equilibrium force on the PP over the oscillation amplitude in the z-dimension, converting the force into the frequency shift signal that would be measured in the experiment.

The separation of the first two steps is for computational efficiency. We could skip step 1. and calculate the sample force on every step of the relaxation process, but since the PP is the only mobile particle in the system, the force can be pre-calculated on a grid, and then during the relaxation process the force can be fetched with much less computational cost by interpolating the value on the pre-calculated grid. One just has to make sure that the grid density is high enough that the interpolation does not result in a significant error, and the physical extent of the grid should be large enough that the PP cannot fall outside of it. If the grid is periodic, the last point is not an issue. The force interaction between the PP and the tip is not pre-computed since it is a simple pair interaction that is as fast to recompute as it would be to interpolate. The interaction model used here is usually a harmonic potential, which is discussed in more detail below in the context of pair-wise potentials.

The goal of the relaxation step (2.) is to find the equilibrium point \( \mathbf{r}_0 \) where the force on the PP vanishes, \( \mathbf{F}_{PP}(\mathbf{r}_0) = 0 \), and to record what force the sample exerts on the PP at that position, which is also the force exerted on the tip, as was discussed above. The process starts by placing the PP in a pre-defined point below the tip and then following the force curve to the local minimum in the energy landscape according to Newton’s laws of motion. The general procedure for doing this is to define some finite time step and compute the new acceleration, velocity, and position of the PP on each time step, although the specifics depend on the choice of the optimization algorithm. An additional frictional force proportional to the velocity of the PP is added so that the motion stops when the minimum is reached. In principle this computation would require defining a mass for the PP, but in practice the optimization algorithm has some parameters that control inertia and friction so that the mass can be arbitrary. There are several different algorithms for doing this, and the current PPM implementation uses one called the Fast Inertial Relaxation Engine (FIRE) [51, 52].

The first two steps already construct a map of the force on the tip, which can be viewed as an image, but the actual signal recorded in an AFM experiment is not the force directly, but the shift in the oscillation frequency of the tip. Fortunately, the force map can be converted into a frequency shift quite straightforwardly, which is the third and final step of the simulation process. This can be done using the formula derived by Giessibl via perturbation theory [53, 54], which leads to the expression for the frequency
The Probe Particle simulation model

shift

$$
\Delta f(r) = -\frac{f_0}{2k_c} \frac{2}{\pi A^2} \int_{-A}^{A} q F_{S,z}(r-q\hat{z}) \sqrt{A^2-q^2} \, dq,
$$

(3.4)

where $f_0$, $k_c$, and $A$ are the eigenfrequency, spring constant, and oscillation amplitude of the cantilever, respectively, $F_{S,z}$ is the $z$-component of sample-PP force, and $\hat{z}$ is a unit vector along the $z$ direction, which is defined to be parallel to the tip oscillation direction.

3.4 Force-field models

In order to calculate the force that the PP experiences, we need to decide on some model of its interaction with the sample. In classical molecular dynamics simulations, the total energy of the system is typically divided into multiple parts that can be calculated independently. In PPM the energy of the PP-sample system, $E_S$, is divided into three parts as

$$
E_S(r) = E_P(r) + E_{vdW}(r) + E_{ES}(r).
$$

(3.5)

The three components are the Pauli energy $E_P$, the van der Waals (vdW) energy $E_{vdW}$, and the electrostatic energy $E_{ES}$. This division of the energy is useful because the vdW and electrostatic terms have analytical expressions that can be motivated from classical physics. Only the Pauli term is fundamentally quantum mechanical and has to be approximated by some other method. Note here that, since the tip and the sample are static, the energy is only a function of the position $r$ of the PP. Once the energy is determined, the force that the sample exerts on the PP can be calculated as the negative gradient of the energy as in Eq. (3.2).

There are several different ways of defining the energy components, and in the following sections three different models will be discussed: a strictly pair-wise potential, a potential where the pair-wise electrostatic term is replaced with a charge distribution model, and a model that also uses a charge distribution for the Pauli interaction. A comparison of the obtained images for the different force field models is shown in Fig. 3.3.

3.4.1 Pair-wise potential

Perhaps the simplest form for a potential for a collection of interacting particles is the pair-wise potential, where each pair of particles interacts with each other independent of the other particles. The total interaction of any given particle with its surroundings is simply a sum of its interactions with each of the other particles, according to the superposition principle.

PPM uses one of the most basic pair-wise potentials, the Lennard-Jones (LJ) potential, which has two different interaction terms, one repulsive
Figure 3.3. Example of $\Delta f$ images from PPM simulations using different force field models with a comparison to experimental images. (A) The system geometry: 1-bromo-3,5-dichlorobenzene on the Cu(111) surface. (B) PPM simulations and experimental AFM images. (C) Approach curves at the positions of the top Cl atom (left) and the bottom H atom (right). The values for the experiment curve are scaled and shifted to be at the same scale with the simulation, so only the overall trend should be compared. The full-density based model (FDBM) simulation (Sec. 3.4.3) here uses the parameters $V_0 = 15[eVÅ^3]$ and $\alpha = 1.1$. 
and one attractive. The LJ potential energy between a pair of particles $a$ and $b$ separated by a distance $r_{ab} = |r_{ab}| = |r_b - r_a|$ is

$$E_{\text{LJ}}(r_{ab}) = \varepsilon_{ab} \left[ \left( \frac{\sigma_{ab}}{r_{ab}} \right)^{12} - 2 \left( \frac{\sigma_{ab}}{r_{ab}} \right)^6 \right],$$

(3.6)

where $\varepsilon_{ab}$ and $\sigma_{ab}$ are fitting parameters. The two terms of the LJ-potential model two different force interactions. The first $r^{-12}$ term models the Pauli repulsion and the second $r^{-6}$ term models the attractive vdW force (or more accurately, the London dispersion force). The form of the vdW term can be justified in (semi-)classical physics by considering the interaction of two induced dipole moments, or more rigorously in quantum perturbation theory where the $r^{-6}$ is the leading term in the energy expression [55]. In contrast, the Pauli term in reality has no simple analytical form, and the $r^{-12}$ form is chosen just for the computational efficiency of computing the 12th power as the square of the 6th power required by the vdW term. The corresponding LJ-force on particle $b$ is

$$F_{\text{LJ}}(r_{ab}) = -\nabla_{r_b} E_{\text{LJ}}(r_{ab}) = \frac{12}{2} \varepsilon_{ab} \sigma_{ab}^2 \left[ \left( \frac{\sigma_{ab}}{r_{ab}} \right)^{14} - \left( \frac{\sigma_{ab}}{r_{ab}} \right)^8 \right] r_{ab}.$$

(3.7)

The parameters $\varepsilon_{ab}$ and $\sigma_{ab}$ depend on what the pair of atoms is. A typical way of fitting these parameters is to define single values $\varepsilon_a$ and $\sigma_a$ for every element on the periodic table, and then derive the pair-parameters by taking an average of the values of the single atom values in the pair as

$$\sigma_{ab} = \frac{\sigma_a + \sigma_b}{2},$$

(3.8)

$$\varepsilon_{ab} = \sqrt{\varepsilon_a \varepsilon_b}.$$ 

(3.9)

This is also the scheme used in PPM, and the parameter values are taken from the OPLS force field [56, 57].

The LJ-potential takes care of the Pauli and vdW components of the total energy in Eq. (3.5), but we still need the electrostatic component. In the strictly pair-wise interaction paradigm the electrostatic interaction can be modelled with the Coulomb interaction between point-charges. The Coulomb force between two particles with charges $q_a$ and $q_b$ is given by

$$F_{\text{ES}}(r_{ab}) = \frac{1}{4\pi \varepsilon_0} \frac{q_a q_b}{r_{ab}^3} r_{ab},$$

(3.10)

where $\varepsilon_0$ is the vacuum permittivity. If we assign a point-charge to the PP as well as to all the atoms in the sample, then we can use this formula to calculate the electrostatic force that each atom exerts on the PP. The problem of how to assign the charges is non-trivial, and a more physically correct way of calculating the electrostatic interaction is presented in the
next section, so this issue won’t be further discussed here.

In principle, all the atoms in the PP-sample system are interacting with each other, but since we assume the sample to be static, we don’t actually have to care about any of the terms in the total energy expression that are related to interactions between the atoms within the sample. The only thing we care about is the force that the sample atoms exert on the PP. Therefore, in the above formulas we can set the particle \( b \) as the PP, and calculate the total force as a sum over all the atoms of the sample, which is

\[
F_S(r_{PP}) = \sum_{a=1}^{N_{at}} F_{\text{LJ}}(r_{aPP}) + F_{\text{ES}}(r_{aPP}),
\]

(3.11)

where \( r_{PP} \) is the position of the PP, and \( N_{at} \) is the number of atoms in the sample.

The tip-PP interaction in principle could be modelled with an LJ potential as well, but this can lead to problems with stability of the simulation, such as the PP escaping the tip and sticking to the sample. Instead, PPM uses a harmonic force with radial and lateral components,

\[
F_T(\Delta r) = -k_r(\Delta r - r_{eq})\hat{\Delta r} - k_x\Delta r_x\hat{x} - k_y\Delta r_y\hat{y},
\]

(3.12)

where \( k_{(r,x,y)} \) are the spring constants in the radial, \( x \), and \( y \) directions, respectively, \( \Delta r \) is a vector from the tip to the PP, \( \Delta r = |\Delta r| \), \( r_{eq} \) is the equilibrium tip-PP distance, and \( \hat{\Delta r} \), \( \hat{x} \), and \( \hat{y} \) are unit vectors in the direction of \( \Delta r \), the x-axis, and the y-axis, respectively. Typically, the lateral force is symmetric in the xy-plane, so that \( k_x = k_y \), and the radial component is much stiffer than the lateral component, \( k_r \gg k_{(x,y)} \). For example, for a CO tip typical values for the spring constants are \( k_x = k_y = 0.24 \text{N/m} \) [58] and \( k_r = 30 \text{N/m} \).

Example simulations using the force field with LJ-potential and point charge electrostatics is shown on the left column of Fig. 3.3B. The charge on the PP in this case is actually composed of a combination of three charges arranged in a \(-q, 2q, -q\) pattern on the z-axis in order to create a quadrupole charge distribution, which better approximates the charge distribution on the oxygen of a CO molecule attached to the AFM tip. A comparison to the experimental images shows that the simulation captures the overall contrast of the real images quite well, although at closer range the central ring of the molecule appears larger in the simulation.

### 3.4.2 Hartree electrostatic force

A partial charge for an atom is not a proper quantum mechanical observable, and therefore there does not exist any single correct way of assigning charges to atoms in a molecule. To calculate the electrostatic force interaction between the sample and the PP properly, we need to consider the
charge of both the sample and the PP as a distribution over space. The equivalent of Coulomb’s law for a charge distribution \( \rho \) is Gauss’s law

\[
\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0},
\]

(3.13)

where \( \mathbf{E} \) is the electric field due to \( \rho \). Additionally, if we write \( \mathbf{E} = -\nabla V \) for an electrostatic potential \( V \), then we arrive at Poisson’s equation

\[
\nabla^2 V = -\frac{\rho}{\varepsilon_0}.
\]

(3.14)

The electrostatic potential, also called the Hartree potential, is solved as a part of the DFT self-consistency cycle, so the potential can be obtained quite easily for systems calculable with DFT. Once obtained, the Hartree potential can be used for calculating the electrostatic contribution to the force on the PP.

If the Hartree potential of the sample is \( V_S \) and the charge distribution of the PP is \( \rho_{PP} \), then the electrostatic interaction energy of the system with the PP located at point \( \mathbf{r} \) is a cross-correlation between the potential and charge distribution:

\[
E_{ES}(\mathbf{r}) = (\rho_{PP} \star V_S)(\mathbf{r}) = \int_{\mathbf{r}' \in \mathbb{R}^3} \rho_{PP}(\mathbf{r}' + \mathbf{r})V_S(\mathbf{r}')d\mathbf{r}'.
\]

(3.15)

The formula presented here is for the continuous domain, but to compute the integral in practice, we would discretize the problem and do it as a sum on a finite grid. If we discretize both \( \mathbf{r} \) and \( \mathbf{r}' \) into grids of \( N \) points, then naively calculating the sum for every one of the grid points takes \( \mathcal{O}(N^2) \) operations. However, there is a better way of calculating the cross-correlation integral using a Fourier transform.

The Fourier transform of a function \( f \) can be defined as

\[
\hat{f}(\omega) \equiv \mathcal{F}\{f\}(\omega) = \int_{-\infty}^{\infty} dx f(x)e^{-i\omega x}
\]

(3.16)

with a corresponding inverse transform

\[
f(x) = \mathcal{F}^{-1}\{\hat{f}\}(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} d\omega \hat{f}(\omega)e^{i\omega x}.
\]

(3.17)

Performing a Fourier transform on a function has the effect of breaking it down into its frequency components, which is useful in many signal processing applications. Here, the important property of the Fourier transform that we can make use of is the fact that the cross-correlation integral in the frequency domain turns into a simple point-wise multiplication of the integrant functions,

\[
\mathcal{F}\{f \ast g\}(\omega) = \hat{f}(\omega)\hat{g}(\omega),
\]

(3.18)
where \( \bar{f} \) is the complex conjugate of \( f \). We can apply this to calculate the energy in Eq. (3.15) as

\[
E_{ES}(\mathbf{r}) = \mathcal{F}^{-1} \left( \mathcal{F} \{ \rho_{PP} \} \mathcal{F} \{ V_S \} \right) (\mathbf{r}).
\]  

(3.19)

The advantage of this formula compared to the direct integration comes from the fact that the discrete version of the Fourier transform can be calculated in \( \Theta(N \log N) \) time using a fast Fourier transform (FFT) algorithm such as the Cooley-Tukey algorithm [59]. For typical grid sizes of \( N \sim 10^7 \) this is the difference between the computation taking 0.1 s or a full day. It should be noted that the transform here is for a univariate function, but the transform (and the computing algorithm) can be extended for multivariate functions by repeated application of the univariate transform for all the variables.

Now with the electrostatic energy in hand, we can again obtain the force on the PP as the negative gradient of the energy,

\[
\mathbf{F}_{ES}(\mathbf{r}) = -\nabla \mathbf{r} E_{ES}(\mathbf{r}).
\]

(3.20)

There are several methods for numerical gradient calculation. Since FFT was used for the cross-correlation integral, one might note that the derivative operation also becomes simpler in frequency domain, namely \( \mathcal{F} \{ f' \} (\omega) = i \omega \mathcal{F} \{ f \} (\omega) \), which could be easily done at the same time as the frequency-domain cross-correlation product. However, because the gradient in three dimensions requires three different derivatives, this in practice would require three inverse transforms, which increases the computation time significantly. In contrast, computing the derivative by a finite difference (FD) formula is an \( \Theta(N) \) operation:

\[
\mathbf{F}_{ES,x}(\mathbf{r}) \approx - \frac{\mathbf{F}_{ES}(\mathbf{r} + h) - \mathbf{F}_{ES}(\mathbf{r} - h)}{2h},
\]

(3.21)

and similarly for the other components of \( \mathbf{r} = (x, y, z) \). This formula specifically is the centered difference formula, which has error that scales as \( \Theta(h^2) \). The step \( h \) in practice is determined by the grid step of DFT computation for the potential \( V_S \). In the current implementation of PPM, both of these methods of gradient computation are used: the CPU version uses the FFT method, and the GPU version uses the FD method.

The charge density for the tip can also be calculated by DFT. However, the sharp variation in the density near the nuclei can cause problems with accurately computing the force. A common approximation is to use the difference of the molecule electron density to the vacuum density of the atoms constituting the molecule. The difference is mostly in the valence electrons, and for CO specifically this density has been noted to have a quadrupolar characteristic around the oxygen, so often an analytical form based on spherical harmonics is used as the tip density in the simula-
The Probe Particle simulation model

The quadrupole moment can also be parametrized for different tip environments (Cu, Ag, Au, etc.).

It should be noted that, since the PP can bend, the charge distribution of the tip should in principle rotate with the bending PP during the simulation. However, as was discussed in Sec. 3.3, the force-field is calculated just once in the beginning, and the alternative would be to recompute the energy and the force for each step of the PP relaxation process, which would be extremely costly. Therefore, the simplifying assumption is made that the bending of the PP does not significantly affect the force on the PP so that the force field can be computed just once in the beginning. This assumption can be justified by noting that in practice the PP starts bending significantly only at the closest range where the interaction is dominated by the Pauli repulsion term.

An example simulation using the Hartree potential electrostatics with a spherical harmonic quadrupole charge distribution on the PP is shown in the middle left column of Fig. 3.3B. Compared to the point-charge case, the central ring appears more clearly defined, and the molecule overall appears slightly smaller.

3.4.3 Full-density based model

It was mentioned briefly in the discussion of the LJ-potential (Eq. (3.6)) that the Pauli repulsion is not something that is computable by any closed-form expression. Therefore, the accuracy of the interaction energy in the LJ-potential at the closest range which is dominated by the Pauli repulsion should be questioned. In order to get a more accurate approximation of the Pauli repulsion term, Ellner et al. [60] introduced the full-density based model (FDBM), where the Pauli repulsion energy $E_P$ is computed by an overlap integral of the PP and sample electron densities,

$$E_P(r) = V_0 \int_{r' \in \mathbb{R}^3} \left[ \rho_{PP}^e(r' + r) \rho_{S}^e(r') \right]^\alpha d'r',$$

(3.22)

where $\rho_{PP}^e$ is the PP electron density, $\rho_{S}^e$ is the sample electron density, and $V_0$ and $\alpha$ are fitting parameters. This formulation of the Pauli repulsion can be motivated by considering that the quantum-mechanical origin of the Pauli repulsion lies in the electrons obeying the Pauli exclusion principle when the electron clouds of the two atoms start overlapping, leading to an increase in the kinetic energy of the electrons. To be clear, this formula is also not an accurate representation of the full quantum version of Pauli repulsion, but it allows for a better approximation compared to the LJ-potential if the fitting parameters are chosen appropriately.

The parameters $V_0$ and $\alpha$ can be fitted per sample by comparing the computed energies with those obtained from DFT calculations. In the original paper, a table of these parameter values are presented for a
selection of small molecules, where the typical values are in range 12...22 [eVÅ³] for \( V_0 \) and 1.07...1.12 for \( \alpha \) [60]. However, in a more recent study, a higher range of values, 45...120 [eVÅ³] for \( V_0 \) and 1.08...1.20 for \( \alpha \), is found to match another set of molecules better [61]. Here, the units [eVÅ³] denote the appropriate units that match the chosen value of \( \alpha \) such that the final energy is in units of eV. Overall, the values for these parameters have not been explored for very many different systems and are a subject of future research.

For practical computation of Eq. (3.22) we can note that by distributing the exponent to the integrants, the integral becomes of the same cross-correlation form as the electrostatic energy in Eq. (3.15): \( E_P = V_0((\rho_{pp})^\alpha \ast (\rho_S)^\alpha) \). Therefore, we can compute the integral using the FFT method discussed above, and the derivative for the force can also be computed in the same way using either the FFT or the FD method. For added efficiency, we can also note that the two energies can be summed first, \( E_{ES} + E_P \), so that the derivative only has to be computed once.

The FDBM replaces the Pauli term in the LJ-potential, but we still need the vdW interaction as well. One could keep using the LJ vdW energy, but Ellner et al. choose to instead use the DFT-D3 [62] vdW correction which is used often with DFT functionals that do not account for the vdW energy well. An implementation for the DFT-D3 energy and force calculation has been recently added to PPM as well. The form of the DFT-D3 energy is similar to the LJ-potential in that it has an \( r^{-6} \) term, but it also has an additional \( r^{-8} \) term. There are a couple of different ways of defining the DFT-D3 energy, depending on the used damping function. The form used here is based on the Becke-Johnson damping [63, 64] and is given by

\[
E_{vdW} = - \sum_{a=1}^{N_a} \sum_{b=a+1}^{N_a} \left( C_{6}^{ab} \frac{s_6}{r_{ab}^6} + f(R_{ab}^0)^6 + C_{8}^{ab} \frac{s_8}{r_{ab}^8} + f(R_{ab}^0)^8 \right) \tag{3.23}
\]

where the sum is over all pairs of atoms \( a, b \) in the system. The additional term \( f(R_{ab}^0) \) in the denominator is a damping function that ensures that the energy cannot diverge and is given by

\[
f(R_{ab}^0) = a_1 R_{ab}^0 + a_2, \tag{3.24}
\]

where \( R_{ab}^0 \) is a cutoff radius which is defined as

\[
R_{ab}^0 = \sqrt{\frac{C_{8}^{ab}}{C_{6}^{ab}}}. \tag{3.25}
\]

The parameters \( s_6, s_8, a_1, \) and \( a_2 \) are additional scaling parameters for different DFT functionals, and there exist tables of fitted values for most of the commonly used functionals [65]. The \( C_{6}^{ab} \) coefficients are originally calculated based on the polarizabilities of the constituent atoms, but in
practice one would use pretabulated values [65]. The $C_{ab}^{8}$ parameters are derived from the $C_{ab}^{6}$ values as

$$C_{ab}^{8} = 3C_{ab}^{6} \sqrt{Q_a Q_b},$$

(3.26)

where $Q_a = 0.5\sqrt{Z_a \langle r^4 \rangle_a / \langle r^2 \rangle_a}$ is calculated based on the nuclear charge $Z_a$ and multipole expectation values $\langle r^4 \rangle_a$ and $\langle r^2 \rangle_a$. Again, pretabulated values exist for these [65].

Another factor that makes the DFT-D3 vdW different from LJ is that the $C_{ab}^{6}$ parameters are adjusted based on the environments of the two atoms in order to account for the changing polarizabilities of the atoms caused by their interactions with their neighbours. To this end, one first calculates a coordination number ($CN$) for every atom in the system, which is roughly a count of the number of neighbours each atom has, and is given by

$$CN^a = \sum_{b \neq a} \frac{1}{1 + \exp(-k_1[k_2(R_{a,cov} + R_{b,cov})/r_{ab} - 1])},$$

(3.27)

where $k_1 = 16$, $k_2 = 4/3$, and $R_{a,cov}$ is the covalent radius of atom $a$ [66]. The $C_{ab}^{6}$ coefficient is then calculated as a weighted average of pretabulated reference values $C_{ab}^{6,ref}$ for different reference coordination numbers $CN_{ref}$, and is given by

$$C_{ab}^{6} = \frac{\sum_{i,j} C_{ab}^{6,ref}(CN_{ref,i}^a, CN_{ref,j}^b)L_{ij}^{ab}}{\sum_{i,j} L_{ij}^{ab}},$$

(3.28)

$$L_{ij}^{ab} = \exp \left( -k_3 \left[ (CN^a - CN_{ref,i}^a)^2 + (CN^b - CN_{ref,j}^b)^2 \right] \right),$$

where the indices $(i,j)$ run over the reference points for the respective atom types $(a, b)$, and $k_3 = 4$. In PPM, one of the atoms in the system is the PP, which we assume to be chemically inert. This means that the PP should not count towards the coordination number of all the other atoms, which are the atoms of the sample. This simplifies the calculation, because the $C_{ab}^{6}$ coefficients for the sample atoms become independent of the position of the PP, so that they can be computed just once in the beginning of the calculation, and the coordination number of the PP is always $CN_{PP} = 0$.

Similar to the LJ case, the only terms in the energy sum (Eq. (3.23)) that we actually care about are the ones related to the PP. Therefore, we can set $b = PP$ and rewrite the energy as a function of the position $r_{PP}$ of the PP,

$$E_{vdW}(r_{PP}) = -\sum_{a=1}^{N_a} \left[ s_6 \frac{C_{aPP}^6}{r_{aPP}^6 + f(R_{aPP})^6} + s_8 \frac{C_{aPP}^8}{r_{aPP}^8 + f(R_{aPP})^8} \right],$$

(3.29)

where there now is only a single sum over the atoms in the sample. Once more taking the negative gradient of the energy, the corresponding vdW
The Probe Particle simulation model

force on the PP is given by

\[
F_{\text{vdW}}(r_{PP}) = - \sum_{a=1}^{N_{at}} \left( 6C_6^a \frac{r_{aPP}^4}{\left[ r_{aPP}^6 + f(R_{aPP}^0)^6 \right]^2} \right) ^2 + \left( 8C_8^a \frac{r_{aPP}^6}{\left[ r_{aPP}^8 + f(R_{aPP}^0)^8 \right]^2} \right) r_{aPP}.
\] (3.30)

An example of a simulation using the FDBM is shown in the middle right column of Fig. 3.3B. Compared to the previous two simulations, here both the relative brightness and apparent size of the central ring remain more constant over the whole distance range, which is more in line with the experimental images. This is also seen in the approach curves in Fig. 3.3C, where the curve for the FDBM is more flat compared to the other two.
This chapter discusses the second major topic in this thesis, machine learning (ML). The impact ML has had in recent times in science, engineering, and society in general, can hardly be overstated. The most visible application is in artificial intelligence (AI) research where large language models in particular have emerged as a leading candidate for creating artificial general intelligence [19], but applications in other research fields, including physics, have become more common as well.

It is interesting to note the difference in the ML approaches taken in AI and physics research. The ML models in AI research have become more and more general in their structure, relying more heavily on learning the structure of the problem from the statistics of the data. It is unlikely that the AI models will ever return to simple rule-based algorithms due to the inherent irreducible complexities of the human world. For example, natural languages, while having a structure in their grammar, also have a lot of ambiguities and arbitrary idiomatic expression which simply have to be learned by heart. In contrast, the ML in physics research has been more conservative and has added more constraints to the model structures. This is especially seen in the equivariant neural network potentials [67–69] which by construction respect the known symmetries of the system. It will be interesting to see whether the ML applications in physics will continue to bake more symmetries into the models, or will also lean more heavily into very large datasets. Perhaps both.

The discussion on ML here starts in Sec. 4.1 with overview of some general concepts, including the model parameters, the loss function, and the fitting algorithm, as well as the categorization for different types of ML models. The specific type of ML model utilized in this thesis is the neural network, which is discussed in Sec. 4.2 in its different forms. The important role of the dataset in ML is discussed in Sec. 4.3, and finally, Sec. 4.4 provides a short review of the literature of ML applications in SPM.
4.1 Overview of machine learning concepts

There is no single definition for what ML is, but on a general level we can think of an ML system as a program that can learn to perform a task based on examples rather than working on a pre-defined set of rules. On a more practical level, doing ML means constructing a statistical model with tunable parameters that modify the behaviour of the model, and fitting those parameters into a dataset of examples such that number of errors that the model makes on those examples is minimized. This capability to learn from examples is often associated with intelligence, and thus ML systems are often thought of as a form of AI, although the simplest ML models perform very simple tasks that one would not intuitively think of requiring any intelligence.

An ML model is a function \( f_\theta : \mathcal{X} \rightarrow \mathcal{Y} \), that has a set of internal parameters \( \theta \). The domain \( \mathcal{X} \) and codomain \( \mathcal{Y} \) can be practically anything. A classic example would be the problem of classifying photos of cats and dogs, where \( \mathcal{X} = \) “Set of all photos containing a cat or a dog” and \( \mathcal{Y} = \{\text{cat, dog}\} \). A perhaps more practically interesting example would be if \( \mathcal{X} \) contains medical records of patients, and \( \mathcal{Y} = [0,1] \) representing the probability that a patient has a given disease, a typical task for a physician. Or on the topic of this thesis, \( \mathcal{X} \) could contain sets of AFM images, and \( \mathcal{Y} \) could represent some property of molecules that we are interested in.

In order to fit the parameters \( \theta \), we require a dataset of examples \( D \subset \mathcal{X} \times \mathcal{Y} \). In a given example \((x, \hat{y}) \in D\), \( x \) is input to the model, and \( \hat{y} \) is the correct output corresponding to \( x \), often called the label or the ground truth. Once the model makes a prediction \( y = f_\theta(x) \), the quality of the prediction is measured by a loss function \( \mathcal{L}(\hat{y}, y) : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R} \), that compares the prediction with the label and outputs a number that is smaller the closer the prediction is to the label. The task of fitting the parameters is to minimize the loss for the whole dataset,

\[
\theta_{\text{optim}} = \arg\min_\theta \sum_{(x, \hat{y}) \in D} \mathcal{L}(f_\theta(x), \hat{y}). \tag{4.1}
\]

The choice of the loss function depends on the task. Typical loss functions include the mean squared error (MSE) loss,

\[
\text{MSE}(y, \hat{y}) = \frac{1}{N_y} \sum_{i=1}^{N_y} (y_i - \hat{y}_i)^2, \tag{4.2}
\]

often used in regression tasks (continuous codomain), and the cross-entropy loss

\[
\text{CrossEntropy}(y, \hat{y}) = \sum_{i=1}^{N_y} \hat{y}_i \log y_i, \tag{4.3}
\]
Machine learning and neural networks

often used in classification tasks. Here $N_y$ is the size of the label vector ($Y \subset \mathbb{R}^{N_y}$).

The method for fitting the model parameters, also referred to as the training of the model, depends on the form of the model and the choice of the loss function. For a simple model such as a linear model with MSE loss, the problem reduces to the well-known linear least-squares problem, which admits an exact solution for the optimal parameters. However, for many models of interest, such as the neural networks discussed below, there is no known method for finding the global minimum, and approximative methods searching for local minima are used instead. Of these, perhaps the most often used is gradient descent, where the parameter values start at some initial guess, and then they are iteratively updated by going to the direction opposite to the gradient of the loss function for some particular example(s), so that the loss gradually falls. Expressed mathematically, the parameters initially have the values $\theta_0$, and on time step $t$ they are given by the recursive formula

$$\theta_t = \theta_{t-1} - \alpha \nabla_{\theta} \mathcal{L}(f_{\theta}(x), y^\ast),$$

(4.4)

where $\alpha$ is the learning rate that controls the size of the step. Generally, a higher learning rate leads to faster training, but may also result in overstepping the minimum, or in some cases even a diverging loss. In principle, a small enough learning rate should always allow converging to a local minimum. The method here using a single example $(x, y^\ast)$ for the training step is known more specifically as stochastic gradient descent, but the loss and gradient can also be averaged over multiple examples, either taking the whole dataset (batch gradient descent), or using the more common practice of dividing the dataset to smaller subsets (mini-batch gradient descent).

There also exist variations of the gradient descent algorithm that do not simply look at the current gradient value, but also remember the past values, maintaining a kind of momentum for each parameter that helps the parameters not to get stuck in low-gradient regions [70]. In particular, the Adaptive Moment Estimation (Adam) [71] algorithm is a common choice, and is also used for the training of all the models in this thesis. Although often used, an obvious limitation of gradient descent is that it requires both the model and the loss function to be differentiable. An alternative that does not require differentiability is the evolutionary algorithm, which creates a population of different parameter values and mixes them iteratively based on a fitness function, mimicking the idea of natural selection in Darwinian evolution [72].

The discussion here so far has focused on what is called supervised learning, where during training each input to the model has an associated correct label that the model is trying to reproduce. However, not every
dataset is labelled, in which case we may try doing unsupervised learning, where we try to find patterns in data without explicit labels. Common types of unsupervised learning include principal component analysis, and auto-encoders, which can be used to compress the data into a space with smaller dimensionality where patterns differentiating different categories of data may emerge. Yet another type of ML is reinforcement learning, where the model is an agent acting in some environment and is trying to learn an optimal policy to perform some task by maximizing a reward function. Reinforcement learning finds use in, for example, robotics [28] and playing games [25, 26], and in the context of SPM has been used for automating atom manipulations on surfaces [73].

The focus here will be on supervised learning, specifically neural networks, but a variety of other types of supervised ML models exist as well. To name a few, there is linear and non-linear regression, kernel ridge regression, Gaussian process regression, support-vector machines, and decision trees. These models come with various different degree of computational complexity and are best suited for different tasks. In general, it is better to start out by trying simpler models before moving on to more complex ones, as more complex models with more parameters tend to be more prone to overfitting (Sec. 4.3), are often less interpretable, and come with higher computational cost.

### 4.2 Neural networks

The (artificial) neural network (NN) is a very general type of ML model that is used in a wide variety of tasks, especially in AI research. ML using NNs is also known as deep learning. The main selling point of NNs is their capability of learning useful representations of the data without the need for domain-specific knowledge. This is in contrast to the idea of feature engineering used in many other forms of ML, where useful features are first extracted from the data using hand-crafted rules before the application of the ML model. While this makes NNs very generally useful, it is also a part of what gives them the reputation of being difficult-to-interpret black boxes. However, NNs can also be constructed to respect certain symmetries, which can help in limiting bad behaviour if designed correctly for the particular problem. In the following I will focus on the three types NNs used in the ML models in this thesis: the multi-layer perceptron, convolutional neural networks, and graph neural networks.

#### 4.2.1 The multi-layer perceptron

The basic idea of an NN model is to form a network structure like those found in the brain, where arrays of neurons are connected to one another
through synapses, and the neurons send electric signals to each other, performing what can be seen as a kind of computation. Although, the comparison is more of an analogy given that an NN model of substantial size is required to simulate even a single neuron in the brain [74]. One of the earliest ideas in this direction were the artificial neurons first studied theoretically by McCulloch and Pitts in 1943 [75], and later in 1958 built by Rosenblatt into a machine that he called the perceptron [76, 77]. The perceptron is a linear binary classifier

\[ f(x) = \begin{cases} 1 & w^T x + b > 0, \\ 0 & \text{otherwise.} \end{cases} \] (4.5)

Here, the weight vector \( w \) and the bias constant \( b \) are the parameters of the model. In its original implementation, the weights and biases of the perceptron were actually potentiometers whose resistances could be tuned to modify the output signal of the circuit. While useful for some simple tasks, the perceptron is too limited in capability to be a general-purpose NN model.

The more modern software counter-part of the perceptron is the multi-layer perceptron (MLP) that uses multiple layers of perceptrons to construct a more complicated network. A single layer in an MLP can be written as

\[ f_l(x) = g_l(W_l x + b_l), \] (4.6)

where \( W_l \) and \( b_l \) are the weight matrix and bias vector for the layer \( l \), analogous to the weight vector and bias constant in the perceptron, and \( g_l \) is an activation function, discussed in more detail below. An MLP is constructed by composing a bigger network of these layers,

\[ f(x) = f_n \circ \cdots \circ f_2 \circ f_1(x). \] (4.7)
This can be schematically drawn as a graph like in Fig. 4.1. The input vector and the output vectors from the layers are drawn as sets of circles, *neurons*, such that each element in the vectors is represented by a single neuron, and the weights and biases are drawn as lines connecting the neurons in the adjacent layers. The layers between the input and output layers are called *hidden layers*. The number of neurons in each layer and the number of hidden layers are *hyperparameters* of the model, meaning that they are not fitted to the data, but rather chosen when constructing the model.

The importance of the activation function is that it makes the network as a whole non-linear. Without the activation functions, all the linear transformations could be reduced to a single matrix multiplication. In the perceptron, the activation function is a step function that performs the binary classification task. However, a step function is not generally a good choice in an MLP, because it is not suited for gradient-based training. One commonly used activation function is the sigmoid function

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$  \hfill (4.8)

This function is a kind of smooth step function that approaches 1 when $z \to \infty$ and approaches 0 when $z \to -\infty$. It matches the intuition of a neuron either firing or being inactive, while also being a differentiable function. A physicist may also note that this is the Fermi-Dirac distribution when $z = (\mu - E)/(k_B T)$. The sigmoid function has the problem that its gradient quickly approaches zero when $|z| \gg 0$, which becomes a problem in deeper networks where the earlier layers train more slowly. Another activation function that mitigates this problem is the Rectified Linear Unit (ReLU) function

$$\text{ReLU}(z) = \max(0, z).$$  \hfill (4.9)

Despite its simplicity, the ReLU function is one of the most commonly used activation functions in deep learning. However, problems can still arise with the gradients, when the parts of the network get stuck at negative values where the gradient is zero and the parameters never get updated, causing some of the neurons in the network to "die". In cases where this is a problem, one may use the LeakyReLU function

$$\text{LeakyReLU}_a(z) = \begin{cases} z & z \geq 0, \\ az & z < 0, \end{cases}$$  \hfill (4.10)

where $a \ll 1$. Here the gradient is never zero, which prevents the dying of the neurons. Yet another activation function is the SoftMax function,

$$\text{SoftMax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}},$$  \hfill (4.11)
which is used often in the final layer of classification networks, due to how the output of the function is normalized to sum to unity, representing a probability distribution.

The differentiability of the activation functions allows NNs to be trained by gradient descent, which is the most common method of NN training. Since an NN is a function composition, the gradient is calculated by the chain rule of derivation. There is a particular efficient algorithm of calculating the gradient, called backpropagation or back-prop, due to how the gradient has to be computed working backwards from the end of the network to the beginning. The first back-prop type algorithms were proposed in 1970s for accumulating rounding errors in floating-point arithmetic [78, 79], but the first applications to NN training came in the 1980s [80]. Today, there exist several software frameworks that implement the backprop algorithm for all the common functions used in NNs. Currently, the most popular ones are Tensorflow [81] and Pytorch [82], both open-source software libraries for Python that offer GPU acceleration for both the NN layers and the gradient calculation. These libraries allow for quick implementations of both the model and its training process without the big upfront cost of implementing everything from scratch, which is likely one of the biggest contributing factors for the rapid development of ML-related technologies in recent times.

4.2.2 Convolutional neural networks

A common task in ML is image analysis, where the model gets an image as an input and outputs some interesting property of the image, commonly a class label, as in the example of images of cats and dogs. With the MLP as a hammer, image analysis may seem like a good nail to hit. After all, an image is simply an array of pixels represented by numerical values, which could well be input into an MLP model. However, this is not a very good idea for a couple of reasons. One is that images tend to be very large in size, consisting possibly of millions of pixels, which would make the weight matrix in the first layer of the network extremely large, leading to a very high computational cost and likely overfitting. Another issue is that different images come in different sizes, but the MLP input vector needs to be of a fixed size. Finally, an MLP in a fundamental way does not respect the inherent geometrical structure of the image array, where the pixels are arranged in a definite order. To stress this last point, consider what would happen if we have an image of some object, and we shift the image a few pixels to the right by taking a different crop of a larger image. A human would hardly see the difference as long as the main content of the image stays the same. However, for an MLP, the image changes completely, since a shift in the pixels corresponds to a different permutation of the input vector elements, so that completely different weights in the network
will be assigned to the same pixels. This means that the network needs to indirectly learn that the pixel shifts do not matter for the overall content of the image, which in practice would require a very large network and a very large dataset. However, there is an alternative way of constructing the network that respects this symmetry in the data. This is achieved in the convolutional neural network (CNN). Since AFM data is also of the image type, the CNN is the main type of ML model used in this thesis.

The main component of a CNN is the convolution layer that transforms an input $X$ as

$$\text{Conv1D}(X)_i = \sum_{p=0}^{K-1} X_{i+p} W_p + b \quad \forall i \in \{0,\ldots,N-1\}, \quad (4.12)$$

where $W \in \mathbb{R}^K$ are the weights of the layer, also known as the convolution kernel or the filter, and $b$ is a bias constant similar to an MLP. This is actually the discretized version of the cross-correlation operation in Eq. (3.15), but the operation is called convolution in the context of deep learning by convention. The input $X \in \mathbb{R}^{N+K}$ is typically padded at the start and the end by half the kernel size $K$, so that the output size has the same size as the original input. The operation in Eq. (4.12) is the 1D convolution, but the concept generalizes straightforwardly to higher dimensions. For 2D and 3D:

$$\text{Conv2D}(X)_{ij} = \sum_{p=0}^{K_x-1} \sum_{q=0}^{K_y-1} X_{i+p, j+q} W_{pq} + b \quad \forall (i,j) \in \{0,\ldots,N_x-1\} \times \{0,\ldots,N_y-1\}, \quad (4.13)$$

$$\text{Conv3D}(X)_{ijk} = \sum_{p=0}^{K_x-1} \sum_{q=0}^{K_y-1} \sum_{r=0}^{K_z-1} X_{i+p, j+q, k+r} W_{pqr} + b \quad \forall (i,j,k) \in \{0,\ldots,N_x-1\} \times \{0,\ldots,N_y-1\} \times \{0,\ldots,N_z-1\}. \quad (4.14)$$

The convolutions typically also have multiple channels, just as digital images have red, green, and blue colour channels, although the exact meaning of the channels becomes abstract after the first layer. Having multiple channels is a way of expanding the information flow through the convolution layers by attaching not just a single value but a vector of values to each pixel. Every channel is assigned its own weights in the convolution kernel. The output can also have multiple channels, which also get their own weights each. Therefore, in total for a 3D convolution, the convolution kernel would be a 5D array. Such multidimensional arrays, be it the weights or the input data, are usually called tensors in deep learning, although the concept of a tensor finds more general use in mathematics beyond being simply a multidimensional array. The input and output
Arrays for convolution layers specifically are also called feature maps.

The convolution operation can be visualized as a process where the convolution kernel is slid across the input feature map such that at each position the overlapping elements are first multiplied point-wise and then summed together to form a single element in the output feature map. This is shown schematically in Fig. 4.2 for the 2D case. Here it can be seen that the convolution layer does not suffer from the same problem as the MLP for shifts in the image. If we were to shift the pixels into another position in the input, the output pixels would just shift by the same amount into the same direction. Respecting this symmetry, the convolution layer is translationally equivariant. The sharing of the weights at all positions in the feature map also means that the convolutional layer has significantly fewer parameters than an MLP layer. Additionally, the convolution operation has the property of locality where a pixel in the output feature map only depends on the pixels in the immediate vicinity in the corresponding position in the input feature map. This can be either an advantage or a disadvantage depending on how local the relevant features in the feature maps are. Because of this sparse connectivity in
convolutional layers, the MLP layers are in contrast often called fully-connected layers.

Another important type of layer in CNNs is the pooling layer. Unlike the other layer types, pooling layers do not have any trainable parameters. Rather, it is a fixed operation, whose purpose is to down-sample the feature maps in the network. A pooling layer is a function $f : \mathbb{R}^K \rightarrow \mathbb{R}$ that condenses several elements into one, and is usually applied for non-overlapping sequences of elements in the input feature map, so that the number of output elements is less than in the input. A common type of pooling layer in CNNs is the average pooling layer, which in 1D can be expressed as

$$\text{AvgPool}(X)_i = \frac{1}{K} \sum_{j=1}^{K} X_{si+j},$$

where $K$ is the pooling kernel size, and $s$ is the stride. Typically, $s = K$ so that there is no overlap and all elements in the input are covered. Another common pooling layer type is max pooling, where the maximum element instead of average is taken. The pooling operation generalizes to higher dimension by simply making the pooling kernel multidimensional, similar to the convolution layer.

Putting all these pieces together, a basic CNN is formed by stacking multiple convolution and pooling layers in sequence, similar to the layers of the MLP. Naturally, the convolution layers here also need the activation functions in order to make the network as a whole non-linear. A typical CNN architecture has a number of convolution layers arranged into blocks that are interleaved with pooling layers to gradually reduce the size of the feature maps with an increasing number of channels deeper into the network. A common use case is image classification models, where the CNN ends in fully connected layers that get as input the final low-resolution feature maps from the last convolution layer. An early example of this kind of network is LeNet, proposed by LeCun et al. in 1989 [20], which achieved highly accurate classification of handwritten digits. A more recent example is AlexNet [21], which became famous for winning the 2012 ImageNet image classification competition by a large margin, marking a significant milestone in the development of image classification as well as NNs in general.

In the image classification networks, the final output is a vector representing the class probabilities. However, in some tasks the output of the network is also an image, which is the case for example in image segmentation, where the task is to label every pixel in the input image. This allows the use of fully convolutional networks where all the layers are of the convolution type instead of switching to fully connected layers at the end. An example of this kind of network is the U-Net [22], which was originally developed for segmentation of medical images. The main
The specific variant of the U-net that is used in this thesis is the Attention U-net architecture. The encoder is similar to the convolution part of the image classification models with convolution blocks and gradual down-sampling of the feature maps. The decoder reverses the process by using up-sampling operations between the convolution blocks, so that in the end the feature map size returns back to the size of the original input. Another important aspect of the U-net are the skip-connections, which add a second path of information propagation to the network. Each of the corresponding stages in the encoder and decoder have their own skip connection where the feature maps from the encoder are directly copied to the decoder. These two paths of information propagation serve two different purposes. The path going long way through the encoder-decoder network allows learning more global features in the data since in the low-resolution space the convolution kernels can cover a bigger part of the feature maps at once, while the skip connections allow some of the finer details that might have been lost in the encoder-decoder branch to propagate to the final layers. The skip connections also allow a better back-propagation of the gradients to the earlier layers, allowing for more efficient training. A variant of the U-net architecture is shown schematically in Fig. 4.3.
U-net [83], which was also first applied to medical image segmentation. The difference of this model to the standard U-net is the use of the so-called **Attention Gate** in the skip-connections. The concept of **attention** is very important in the recent developments in NNs, as it forms the basis for one of the currently most powerful NN models, the Transformer, which is used in large language models like ChatGPT. The basic idea behind attention in NNs is to construct a layer in such a way that in the process of producing the output it also has to produce weights for every element in the input, in a way showing which parts it is paying most attention to. In its original formulation in the Transformer network, the attention layer is given by [84]

\[
\text{Attention}(Q, K, V) = \text{SoftMax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V, \quad (4.16)
\]

where \(Q\), \(K\), and \(V\) are the so-called query, key, and value, and \(d_k\) is the size of the key vector. The values represent the information in the input vector, and the key and query perform a kind of information retrieval on the values. The SoftMax activation ensures that the attention weights for the values sum to unity. The origin of the key, query, and value depend on the specific application. In the case of **self-attention** they are all derived from a common input \(X\) to the layer: \(Q = f_q(X)\), \(K = f_k(X)\), and \(V = f_v(X)\), where \(f_q\), \(f_k\), and \(f_v\) are some differentiable functions, typically simple linear transformations. Analogously, if \(X\) is a feature map in a CNN, we can form an Attention Gate layer

\[
\text{AG}(X, Q) = \text{SoftMax}(f_a[f_x(X) + f_q(Q)]) \odot X, \quad (4.17)
\]

where \(f_x\), \(f_q\), and \(f_a\) are convolution layers, and \(\odot\) denotes an element-wise multiplication. In this case, the feature map produced by the SoftMax activation is an **attention map** that highlights the important regions in the input feature maps. This is illustrated schematically in Fig. 4.4. The Attention Gate is applied to all the skip connections in the U-net, and the query \(Q\) comes from a feature map at the end of the encoder, as shown in Fig. 4.3.

### 4.2.3 Graph neural networks

The CNNs introduced in the previous section are well-suited for dealing with AFM data. However, there is another type of data that we are interested in dealing with: the coordinates and elements of atoms in molecules. There exists a mathematical object, the **graph**, which is particularly well-suited for representing molecules. There also exists a type of NN that is specifically designed for processing graph-structured data, the **graph neural network** (GNN), which is the topic of this section.

Starting with a definition, a graph is a pair \((V,E)\). The first component
of the pair is a set of nodes or vertices \( V = \{v_i\}_{i=1}^{N_V} \) that represents some objects of interest. It should be noted that even though we label the nodes here by numbers \( \{1, \ldots, N_V\} \) the set is actually unordered, so the labelling is arbitrary. The second component is a set of edges \( E \subset V \times V \) that connect the nodes in the graph as pairs. The edge pairs can be either ordered, in which case the graph is directed, or unordered, in which case the graph is undirected. We can also define a neighbourhood of a node \( v \in V \) as \( \mathcal{N}(v) = \{u \mid u \in V \land (u, v) \in E\} \), which is the set of nodes that have an edge connecting to \( v \). A graph is often visualized as a network where the nodes are drawn as circles and the edges as lines or arrows connecting the circles. In fact, the schematic of an MLP in Fig. 4.1 is a graph. The example of a graph that we are most interested in here is the molecule graph, where the nodes are atoms and the edges are covalent bonds.

A GNN is an NN that operates on a graph in such way that the transformations are done on a per-node or per-edge basis. To this end, every node \( v \in V \) and edge \( e \in E \) in the graph has an associated embedding vector \( h_v \) and \( g_e \). In the beginning, the embeddings only contain information of the node or edge itself, but the GNN gradually propagates and mixes the information with the neighbours, so that in the end every embedding contains a description of its neighbourhood. Using the definition of a graph network from Battaglia et al. [85], a GNN in its most general form can be described by the set of recursive relations

\[
\begin{align*}
    g^t_e & = \phi_e(g^{t-1}_e, h^{t-1}_u, h^{t-1}_v, \gamma^{t-1}) \quad \forall e = (u, v) \in E \\
    h^t_v & = \phi_v(h^{t-1}_v, \rho_v(G^t_v), \gamma^{t-1}) \quad \forall v \in V
\end{align*}
\]  

(4.18)
Figure 4.5. Schematic of a GNN. All nodes and edges in the graph have their own embedding vectors, $h_v$ and $g_{uv}$, which get updated iteratively. The update steps alternate between updating the edge and node embeddings, and the iteration continues for a fixed number of $n_t$ steps. There may also be a global information vector $\gamma$, which is not shown here.

\[ \gamma^t = \phi_\gamma(\rho_{G}(G^t), \rho_{H}(H^t), \gamma^{t-1}) \]

where $G^t_v = \{ g_e^t | e = (u, v) \in E \}$ is the set of embeddings for edges connected to a node $v$ at time $t$, $G^t = \{ g_e^t | e \in E \}$ is the set of all edge embeddings at time $t$, $H^t = \{ h_v^t | v \in V \}$ is the set of all node embeddings at time $t$, and $\gamma$ is an optional global information vector. This process is illustrated in Fig. 4.5. The $\phi$ functions are NNs that update the node and edge embeddings as well as $\gamma$ on each time step $t \in \{1, \ldots, n_t\}$. The $\rho$ functions are aggregation functions that combine the information from multiple embedding vectors into a single vector. The aggregation functions should be chosen in such a way that they respect the inherent permutation invariance of the nodes in the graph and can accept a variable number of input vectors, usually utilizing operations such as taking the sum, mean, or maximum of the input vectors. In early GNNs the number of time steps was not fixed, but rather the iteration continued until the embeddings converged to a fixed point [86]. The convergence can be guaranteed by choosing the update functions to be contraction maps. However, in more recent GNNs, the iteration count is fixed, and it is a hyperparameter of the model [85]. It is important to make the number of iterations high enough that every node has the opportunity to get information from far enough away. For example, if one expects that a node should receive influence from 5 steps away on the graph, then the number of iterations should be at least $n_t = 5$.

An example of a GNN relevant to the topic of this thesis is the Message-passing neural network (MPNN) introduced by Gilmer et al. [87]. The authors applied the MPNN for predicting various physical and chemical properties of molecules such as the HOMO and LUMO energies, atomiza-
tion energies, and polarizabilities, replacing expensive DFT calculations for the same purpose. In the MPNN, the update rule can be written as

\[ m^t_v = \sum_{u \in N(v)} M(h^{t-1}_v, h^{t-1}_u, g_{uv}) \quad \forall v \in V \tag{4.19} \]

\[ h^t_v = U(h^{t-1}_v, m^t_v) \quad \forall v \in V \tag{4.20} \]

Here, the function \( M \), that plays a similar role to \( \phi_e \) above, computes a message from all the neighbours of \( v \), which get aggregated by a sum to a single vector \( m^t_v \). These vectors are then used for updating the embeddings for each node with the function \( U \). Note here that, although the edges have their own embeddings \( g_{uv} \), the embedding is not updated. While there is no global vector here, the final step in the MPNN uses a readout step that is analogous to the global update step,

\[ y = R(H^n). \tag{4.21} \]

This readout value would be the model prediction for the molecule property of interest.

Another application for GNNs that has been gaining attention is neural network potentials (NNP), where the idea is to use an ML model as a replacement for expensive energy and force calculations, for example, from DFT. One of the important aspects of the problem are the inherent symmetries that the system has with respect to the properties of interest. Specifically, the energy of the system is invariant to translations, rotations and reflections of the system, and the forces are equivariant with respect to rotations and reflections, meaning if the system rotates by some amount, the force vectors should rotate by the same amount. Traditionally, the way to encode these symmetries into the ML system has been to first engineer some transformations that extract features from the molecule system in a way that respects the relevant symmetries [88]. However, GNNs offer an alternative where no feature engineering is required, and instead the model itself is designed in such a way that the prediction respects the desired symmetries with respect to the input. An example of this kind of model is NequIP [67–69], which is an E(3)-equivariant GNN model, meaning that the update rule in the GNN iteration loop is constructed in such a way that it respects the 3D translation, rotation, and reflection symmetries. The details of how this model works are beyond the scope of this thesis, but we make use of it for building a database of simulation systems in Sec. 5.5.
4.3 Data, preprocessing, and augmentation

For successful application of ML, the dataset is often as important as the model itself. It is used for training the model as well as evaluating the model performance. In recent times, the focus has been on big data, where very large datasets are used for training very large models. Especially in deep learning these developments have been driven by the availability of GPUs and other more specialized hardware that can leverage the large datasets by offering efficient realizations for the model training algorithms.

As one increases the number of parameters in an ML model, one of the factors that limit the ultimate performance is overfitting. An overfitted model performs well on the training dataset, but generalizes poorly to new examples outside the training set. A clear example of this is fitting a high degree polynomial to a number of points less than the polynomial degree, which allows for a perfect fit at the fitting points, but comes with the cost that the polynomial is likely highly oscillating between the points, as illustrated in the example in Fig. 4.6. Traditionally, the rule of thumb is that the number of training data points should be 10 times as many as the number of parameters in order to avoid overfitting. On the other hand, if a very large dataset is available, it is possible to train a model with very large number of parameters without overfitting.

In order to test whether the trained model is overfitted, we require a separate set of examples that were not used in the training. For this reason the dataset is usually divided into a training set and a test set, with a typical
ratio of 80/20, where most of the data is in the training set. An additional factor comes from the hyperparameters, such as the number of layers or channels in a CNN. These are not directly fitted to the training data, but rather the optimum is found by testing which hyperparameters yield the lowest test loss. However, after a number of trial-and-error test cycles, the hyperparameters could be fitted to the test set, and the test set ceases to be a good measure of the generalization performance. For this reason, another intermediate split is made into a validation set, which is used for monitoring overfitting and choosing the best hyperparameters, and the test set is reserved only as the final test of performance. Sometimes, especially for small datasets, the validation set is added back into the training set before training the model for a final time.

In supervised learning the dataset needs to be completely labelled. In some applications the data is naturally labelled, which would be the case for example in text generation, where the label is simply the next word in the sentence. In other applications, such as image classification, the data comes unlabelled, and the labels need to be added after the fact by manual labour, which often presents a challenge for gathering large datasets. However, the size requirement for the dataset can be reduced by regularization, where some constraints are placed on the model or its parameters, and augmentation, where the examples in the dataset are multiplied by applying transformations to the inputs that preserve the same output. In NNs one very generic form of regularization are Dropout [89] layers, where random activations along with their connected parameters in the preceding layer are set to zero according to a predefined probability. A higher dropout rate leads to higher regularization, but usually also lower performance on the training set.

In the specific example of AFM image analysis considered in this thesis, the problem of data gathering is even more severe: performing an experiment and labelling even a single example requires a very large amount of work. This makes it practically impossible to gather a dataset large enough to train a model of sufficient complexity for the task, even with augmentation. For this reason the dataset is instead generated with PPM simulations of AFM images. While this approach allows for generating large datasets, it produces another challenge, which is that the simulations do not perfectly represent the reality of the experiment. There are differences arising from the limits of the physical model in the simulation as well as any imperfections in the experimental setup leading to noise and artefacts in the experimental images. Therefore, an ML model naively fitted to the simulated images is unlikely to generalize well to the experimental images, being in a way overfitted to the simulation. For this reason as well as the general tendency of large ML models to overfit, we apply several augmentations and preprocessing steps to the simulated AFM images before and during the training of the model. These are described
Machine learning and neural networks

Randomized simulation parameters

The simulation has several parameters that affect the final image, including the tip charge, lateral spring constant, amplitude, and distance from the sample. The optimal parameter values depend on the specifics of the tip as well as the sample and other conditions in the experiment. When generating a large dataset of simulations, it is impossible to carefully choose the optimal parameters for each example. Instead, the parameters are randomized, so that a distribution of different physical conditions are covered in the training set.

Normalization

Normalizing the inputs to the model is a very general technique in ML for gaining better performance. This is often due to different elements in the input having their values in different orders of magnitude, making it difficult to adjust the parameters to balance the contributions. The scale of the pixel values in AFM images do not have a very large variance, but
normalization can still help with dealing with unknowns in the experiment. Specifically, there is a background attractive interaction between the tip and the sample arising from the vdW interaction. The exact magnitude of this force contribution depends on the specifics of the tip geometry, which is unknown, but the force is constant when the tip remains at a constant distance from the sample. In order to remove this unknown factor from the equation, we normalize the input AFM images per constant-height slice by subtracting the mean and dividing by the standard deviation of the pixel values.

**Constant resolution**

The AFM simulations are all performed at a predefined pixel resolution, which in this thesis, somewhat arbitrarily, is chosen to be 0.125 Å/pixel in the lateral direction with a 0.1 Å step between the slices in the vertical direction. In experimental AFM images the physical dimensions of the images are known, and when there is a mismatch, the image is interpolated to the predefined resolution before it is input to the model. The constant resolution is used for the model training with the intuition that it would be difficult for the model to infer the physical scale of the features in the image. Although, the AFM tip may undergo some amount of drift during the scan, causing a distortion in the image that is not accounted for. Random rescaling of the simulations could be added to the training and would be a good target for investigation in future studies.

**Random noise**

Experimental AFM images usually have at least some amount of random noise in the pixel values, which is something that is not present in the simulations. Therefore, we apply random noise to the simulations using a uniform random distribution. We have found that randomizing the amplitude of the noise is also beneficial for better generalization (more details in the supplementary information (SI) of Pub. II).

**Rotations and reflections**

While the construction of the CNN accounts for the translational symmetry of the image data, the predictions could still be sensitive to other important symmetric transformations, namely, rotation and reflection. In order to enforce these symmetries, in each training batch, we add to the batch random rotations and reflections of the input AFM images, while transforming the outputs in an equivalent way.
Cutouts

Cutouts erase small rectangular regions in the input images [90]. This works well as a general regularization technique for CNNs, and in the case of AFM images, the cutout regions also bear some similarity to the artefacts seen in some experimental AFM images. A random number of cutouts with random size are added to each of the input AFM images.

Pixel shifts

We train our models with several input AFM images at different distances from the sample. Due to the drift of the tip in the experiment, the different height images are not always perfectly aligned. In order to account for this, the simulated AFM images are randomly shifted by one or two pixels laterally between the different height images. The shift can be done either by erasing and padding pixels at the edges or periodically rolling the pixels from one edge to the other.

Background gradient

The imaging plane in experimental AFM images is not always perfectly parallel with the surface plane of the substrate. This causes a gradient to appear in the background regions of the image. In order to account for this possibility, we augment the simulated AFM images by additive random gradient planes. Both the direction and magnitude of the gradient are randomized.

4.4 Machine learning in scanning probe microscopy

While the application of ML into SPM is still in its early days, there is a growing body of literature in this direction [91–98], much of which came into existence during the work that went into this thesis. This section briefly lists some this literature.

The ML applications in SPM can be roughly divided into two categories: data analysis and device operation. Focusing first on data analysis, we can note that the idea of using ML in analysing SPM data is not new. As early as 1995, Wang and Whitehouse applied an NN into deconvolving tip effects from AFM images [99]. Much of other pioneering work can be attributed to the group of Sergei V. Kalinin, who have, among other things, analysed spectroscopic SPM data by principal component analysis [100], explored polarization switching in ferroelectric capacitors by analysing hysteresis loops in SPM measurements with NNs [101, 102], classified SPM images of ferroelectric domains using NNs and cross-correlation of
principal components [103], and explored electronic transport at oxide interfaces via clustering and Bayesian inference on SPM data [104, 105]. Most of the early work used feature engineering or unsupervised feature extraction in combination with basic ML methods such as clustering and small fully-connected NNs. With the advent of deep CNNs, there has been a growing interest in the last few years for inference directly from the image data. Among these efforts, the most closely related to the work in this thesis is the work by Carracedo-Cosme et al. who have generated a large dataset of AFM images for use in ML training, called the QUAM-AFM dataset [106], which they subsequently used for training a generative adversarial CNN for predicting a ball-and-sticks representation of the molecule structure [107], as well as a recurrent neural network (RNN) for predicting the IUPAC names of molecules from AFM images [108]. In a similar vein, Tang et al. [109] used a CNN for predicting the structure of interfacial ionic hydrate systems from AFM images. Some other applications include the identification of nanostructures, nanoparticles, and defects [110–115], the analysis of surface lattices with distortions and defects [116], image reconstruction and super-resolution [117–119], and the classification of molecules [120–122]. For a more extensive list of publications, see the review article by Nguyen and Liu [95], who have tabulated publications by the used ML method. In some adjacent studies where ML was not used for analysing SPM images themselves, Järvi et al. [123] used Bayesian optimization for accelerating the DFT search for a molecule structure matching AFM images (using the same AFM data of 1S-camphor as in Pub. I in this thesis), and Ranawat et al. [124, 125] trained a U-net CNN to predict the structure of hydration layers over surfaces in order to elucidate the contrast formation of AFM in liquids.

In addition to analysis of SPM data and images, there is a more general interest in automating the whole process cycle of SPM operation for use as a tool in designing, building, and characterizing materials and nanostructures. This includes the automation of device preparation and control, image acquisition, and sample manipulation. One of the biggest hurdles in preparing an SPM device for operation is the preparation of a suitable tip, which ideally should only have a single atom at its apex interacting with the sample. This time-consuming trial-and-error process has recently been automated by Rashidi et al. [126] who used a CNN to detect the tip state from the obtained STM images and used an automated loop that conditions the tip until a suitable tip state is reached. This method was expanded on by Gordon et al. [127], who also found that accurate tip-state classification can be made even from partial scans [128], enabling fast real-time assessment of the tip state. Several authors have tackled the problem of automated device operation more generally. Ma [129] explored the use of both an NN and a Markov decision process for controlling the feedback loop for tip-sample interaction. Others have developed systems for vibration
cancellation in the $z$-feedback loop using linear models [130] as well as RNNs [131]. A combination of automated device control and image analysis can facilitate fully autonomous imaging operations. In this direction, Krull et al. [132] developed a system for long-running autonomous STM imaging with tip state correction using magnesium phthalocyanine on Ag(100) as an example system, and similarly, Sotres et al. [133] reported high-resolution AFM imaging of DNA on mica without human intervention. Finally, in addition to autonomous imaging, ML can enable autonomous sample manipulation and nanofabrication using the SPM tip. Examples of this include Leinen et al. [134] using deep reinforcement learning to autonomously find an optimal tip trajectory for lifting a PTCDA molecule off an Ag(111) surface, and Chen et al. [73] demonstrating autonomous construction of an artificial kagome lattice out of Ag adatoms on Ag(111), also utilizing deep reinforcement learning.
5. Automated AFM image interpretation

In the previous chapters I discussed the theoretical background for the methods used in this thesis, and in this chapter I will present the application of these methods to AFM image analysis. The applications can be roughly divided into two categories: sample geometry prediction (Pubs. I and III), and electrostatic field prediction (Pub. II). The research into geometry prediction represents the bulk of the work in this thesis, and it went through a couple of stages, first using pure CNN models for image descriptor predictions (Pub. I), and then evolving into a more complicated model combining a CNN and a GNN for molecule graph prediction (Pub. III). The later publications in this direction in some sense supersede the earlier work, but all of the results are presented here in a roughly chronological order.

5.1 Descriptors of atomic systems

The overall goal of the research is to, given a set of AFM images of a sample, extract as much useful information about the sample as possible, and to do it in an automated way to the maximum degree possible. The kinds of information we are interested in are the physical and chemical properties of the sample, such as the positions and types of atoms in the sample or the charge distribution in the sample. This sounds simple on paper, but in order for us to apply the computational machinery of ML into the problem, we need to find some way of representing the interesting information in a format that can be processed by an ML model. In other words, we need to decide on a descriptor of the data.

The starting point is always a set of AFM images at varying distances from a sample. The images are simply 2D arrays of single-valued data, and the images at different distances can be stacked into a single 3D tensor of data, which is very straightforward to use as an input to an ML model. The challenge comes with the output of the model that represents the information about the sample that we are interested in. The main case
that we discuss here is the problem of representing atomic positions as a target output. Perhaps the most intuitive descriptor would be just a simple list of $xyz$ coordinates for every atom, which would be represented as an $N_{\text{at}} \times 3$ matrix. However, this descriptor comes with a couple of problems. The first is that the number of atoms $N_{\text{at}}$ varies with the sample, which makes the model output have a variable size, requiring the use of more complicated model architectures. The second problem is that the list of positions is permutationally invariant, meaning that the structure of the underlying system does not depend on which order the atoms are listed in. This presents a difficulty for the ML model training, because the model has to somehow learn that all the factorial number of different permutations that look like different objects are in fact the same one.

In order to side-step these problems, we need to find a descriptor that is independent of the number and ordering of the atoms in the system. There is a lot of existing literature on descriptors of atomic systems designed for ML applications [88]. Some of the commonly used ones include Atom-Centered Symmetry Functions [135], the Many-Body Tensor Representation [136], and the Smooth Overlap of Atomic Positions [137]. While these descriptors satisfy the requirements that they are fixed-size and permutation invariant, they are mostly designed to be used as an input to rather than an output of an ML model. For this reason they are not readily convertible back to an atomic geometry, which is what we really want to get at. From the point of view of the existing types of deep learning models described in the previous chapter, the one most suitable for AFM image analysis would be the CNN, which is commonly used for image recognition tasks. Therefore, it would be convenient if the output of the model would be in image format as well so that we could simply use a fully-convolutional NN as the ML model.

For this purpose, we designed a number of image descriptors for atomic systems, whose purpose is to show the geometry or other properties of the system in an intuitive way. Examples of these are shown in Fig. 5.1. The first three of these are the Atomic Disks, the vdW-Spheres, and the Height Map. The one which we primarily use is the vdW-Spheres descriptor, which shows the atoms as spheres whose radius is proportional to the vdW radius of the element and the brightness corresponds to the relative depth of the atoms. The Atomic Disks descriptor has a similar idea except that, instead of spheres, the atoms are presented as conical disks and the sizes of the disks are proportional to the covalent radius of the elements. The Height Map represents an isosurface of the total force experienced by the scanning probe. The application of these image descriptors is presented in Sec. 5.2. The final image descriptor is the Electrostatic (ES) Map, which shows the $z$-component of the electrostatic field of the sample at a set distance of 4 Å above the highest atom, within a lateral region close to the sample. The observed contrast in the ES Map indicates the distribution of charges
Atomic Disks  vDW-Spheres  Height Map

ES Map  Mol. Graph  Mol. Geometry

**Figure 5.1.** Descriptors for an atomic system.

in the sample. The application of the ES Map descriptor is presented in Sec. 5.3.

In Sec. 4.2.3, another type of ML model was discussed, the GNN, which can operate on graph-structured data. Molecules are quite naturally represented as graphs with the atoms as the nodes and the bonds as the edges of the graph. The molecule graph (see Fig. 5.1) is the final type of descriptor that we consider. The graph representation overcomes the limitations of the list-of-coordinates representation by considering the atoms as an unordered set and also adds the additional information of bond connections. However, this also requires a more complicated model structure, since we need to start with image data and then convert it into a graph. The application of the graph descriptor is presented first in Sec. 5.4 for purely simulated data and then in Sec. 5.5 for a more specific application of reconstructing the geometries of small ice clusters from experimental AFM data.

### 5.2 Atomic structure discovery

This section presents a summary of the results in Pub. I, which represents our first attempt at atomic geometry reconstruction from AFM images by machine learning. Here, we employ a CNN model that takes a set of
10 AFM images as an input and translates them into one of the three structural image descriptors: the Atomic Disks, the vdW-Spheres, or the Height Map.

The model has a relatively simple encoder-decoder structure that is similar to the U-net, but without the skip connections. The encoder uses 3D convolutions with $3 \times 3 \times 3$ kernels gradually reducing the feature map size by interleaved average pooling layers with $2 \times 2 \times 2$ kernels. The 3D feature maps are reduced to 2D in the middle of the network by pooling, and the decoder, mirroring the encoder, uses 2D convolution interleaved with nearest-neighbour upscaling layers to return the feature map to the original lateral image size. All of the layers use LeakyReLU$_{0.1}$ activations, apart from the final layer which uses a ReLU activation. The model is implemented in Keras [138] using the Tensorflow backend.

One of the main challenges in applying deep learning into AFM image analysis is the lack of large labelled datasets of AFM images. The reason for this is clear: AFM experiments are extremely time-consuming and accurate labelling of the images is extremely difficult even for experts in the field. Therefore, we choose to train the model using data simulated with PPM for a large database of 134,000 small organic molecules with DFT-optimized geometries, covering the elements H, C, N, O, and F [139]. The simulation model employed here is the pair-wise potential with LJ force field and point-charge electrostatics (Sec. 3.4.1). The point charges are obtained from Mulliken charge populations calculated by the coupled-cluster method using the Psi4 chemistry software [140]. The simulated images are preprocessed by normalization, random noise, and cutouts (as described in Sec. 4.3).

We first test the model on simulated data taken from a test set separate from the training and validation sets. On the simulation benchmarks, we can directly compare the prediction made by the model to the reference that we know to be the correct answer for the particular example. Figure 5.2(A-R) shows vdW-Spheres predictions for simulations of three example systems, an isomer of $C_7H_{10}O_2$, dibenzo[a,h]thianthrene, and fullerene $C_{60}$. The first molecule exemplifies the typical kind of small 3D molecule found in the dataset. The prediction matches well with the reference, only missing the hydrogen on the hydroxyl group, which would be also impossible for a human expert to identify. It should be noted that the reference does not contain all the atoms in the geometry, because some of the low-lying atoms are not represented in the AFM image in any way, and we cannot ask the model to predict atoms that it could not know about. In the second example with the dibenzo[a,h]thianthrene molecule, the model is able to correctly identify the protruding sulfur atoms and the overall structure of the carbon rings is found, but we can also observe some blurriness in the prediction which makes the separation of the carbon atoms less clear. This kind of blurriness, especially of deeper atoms, is a
somewhat common feature in the predictions. The last example is the C<sub>60</sub> molecule with a pentagon pointing upward at a slight angle. Here, the top ring is reconstructed accurately and three of the top-most connecting atoms are found as well, while a couple of the lower atoms are missing.

The performance of the model on the simulations is overall very good, but naturally the real test is on the experimental images. To this end, we first compare the simulation result on the C<sub>60</sub> molecule to a prediction on experimental AFM images for the same molecule in Fig. 5.2(S-X). The overall structure in prediction for the experimental image is similar to the simulation finding all the top atoms in the molecule, but with slightly less definition to the individual atoms. We also observe some noise in the prediction on the dark attractive regions around the molecule that are not well represent by the simulation. The figure also shows the Height Map and Atomic Disks predictions for the experiment, which show overall similar structure to the vdW-Spheres prediction. The Atomic Disks descriptor in particular is useful for showing the atom positions more distinctly.
Automated AFM image interpretation

Figure 5.3. Identification of the adsorption configurations of 1S-camphor on Cu(111). Rows (1-5) refer to distinct molecular configurations with experiments in columns (A-D) and simulations in columns (E-I). Selected experimental AFM images (out of the 10 slices used for input): at (A) – far, (B) – middle, (C) – close tip-sample distances and NN prediction (D) for the vdw-Spheres descriptor. The vdw-Spheres representation shown in (E) corresponds to the full molecular configuration (F) resulting from the best match to experiment. The corresponding simulated AFM images are given in panels (G-I) (far–middle–close). Reproduced from Pub. 1 (CC BY-NC 4.0).

For a second example, we consider several experiments of the 1S-camphor molecule in different orientations on a Cu(111) surface, shown in Fig. 5.3(A-C). We task ourselves with finding the correct orientations of the molecule based on the experimental AFM images. To this end, we first perform the ML prediction for the vdw-Spheres descriptor, and then compare the predictions to descriptors corresponding to a set of different orientations of the molecule chosen from points uniformly distribution on a sphere. The comparison is done by cross-correlation and the orientations are ranked based on the cross-correlation score. We pick the highest ranked orientations for each of the experiments (Fig. 5.3F), and perform the PPM simulation for each of these (Fig. 5.3(G-I)), giving us a direct comparison between the experiment and the simulation. The simulations qualitatively reproduce all the major features of the experimental images in every case, but the simulations tend to be more sharp at the close range compared the experiment.
5.3 Electrostatic field discovery

Stepping away from the task of geometry discovery, this section presents the results of Pub. II that considers the problem of identifying the sample electrostatic field from AFM images. The electrostatic properties of molecules are important for various processes in nature and in technology, and have been previously studied especially with Kelvin probe force microscopy (KPFM) [141]. However, KPFM has faced difficulties in widespread adoption due to the challenging experimental setup as well as the lack of rigorous theoretical understanding of the signal at atomic scale preventing quantitative measurements. Here, we present an ML-based method for obtaining accurate electrostatics based on solely standard AFM measurements.

The idea is similar as in the previous section, a CNN model gets a set of AFM images as input and translates it into an image descriptor, in this case the ES Map descriptor. The major difference here is that the input consists of two sets of AFM images, obtained with two different tip functionalizations. We consider here the case of a CO and a Xe tip, but other combinations should work as well (Sec. "Other tip combinations" in the SI of Pub. II). The concept of using two different tips for determining the sample electrostatics is inspired by the observation that the different charges on the differently functionalized tips result in significant modifications to the observed contrast in the obtained images [142], even completely inverting the contrast in some cases [143, 144]. In particular, Hapala et al. [145] mapped the differently distorted positions of features in images obtained with CO and Xe tips in order to reconstruct the electrostatic potential of sample molecules.

The model architecture is a CNN similar to the one used for the atomic structure prediction above, with the difference that we include here the skip connections and attention layers of the Attention U-net model (Fig. 4.3). The input of the model is also changed from a single stack of 10 AFM images to two stacks of 6 AFM images. In order to accommodate the two input channels, we split the first convolution block into two, one for each AFM tip, and combine their outputs in the channel dimension before applying the rest of the CNN model as before (Fig. 4.3). This model and all the models in the following sections are implemented in Pytorch.

One of the limitations in the first study was the limited chemical space covered by the dataset of molecules used for the simulations. Here, we build our own dataset of molecules that includes 81k molecules containing the elements H, C, N, O, F, Si, P, S, Cl, and Br. The distribution of the elements in the molecules is very uneven, so in order to balance the dataset, we choose particular orientations of the molecules that include rarer elements more often on the upward-facing side of the molecule (details in the SI of Pub. II). In the end this results in a dataset of 235k AFM simulations.
that we divide into training/validation/test sets as 180k/20k/35k. The PPM simulations use the same force field model as above with LJ-potential and point-charge electrostatics, and the molecules are in vacuum. During training, the dataset is augmented with random noise, cutouts, pixel shifts, rotations and reflections, and background gradients (Sec. 4.3).

In order to benchmark the model, we first consider three test cases with simulated data, shown in Fig. 5.4. The molecules are chosen because of their non-planar structure and the presence of interesting functional groups yielding non-homogenous patterns in the electrostatic field. In all three cases the agreement between the prediction and the reference is excellent. For a more quantitative comparison, we consider a relative error metric

$$\text{RelError}(y, \hat{y}) = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{\max \hat{y} - \min \hat{y}}$$

(5.1)

where $y$ is the predicted ES Map, $\hat{y}$ is the reference ES Map, and the sum is over the $N$ pixels of the image descriptor. For the three examples in

Figure 5.4. ES Map predictions on simulated AFM images. Predictions are shown for three test systems, (A) N2-(2-Chloroethyl)-N-(2,6-dimethylphenyl)-N2-methylglycinamide, (B) 2-[(1E)-2-thienylmethylene]hydrazide, and (C) tetrathiafulvalene thiadiazole. In each case are shown, from left to right, the 3D structure of the molecule, three out of six input AFM images at different tip-sample distances for both tip functionalizations, and the predicted and reference ES Map descriptors. The colorbar scale for the prediction and the reference is the same on each row. Reproduced from Pub. II (CC BY 4.0).
Figure 5.5. Comparison of simulated and experimental predictions of ES Maps for perylenetetracarboxylic dianhydride (PTCDA). On the left are shown three out of six input AFM images at different tip-sample distances for both tip functionalizations, and on the right are the model predictions for both simulation and experiment and the reference descriptor. Both predictions and the reference are on the same colorbar scale. The molecule geometry used in the simulation is shown on the bottom right. Reproduced from Pub. II (CC BY 4.0).

Fig. 5.4, the errors, in order, are 1.04%, 1.79%, and 2.29%. The performance of the model on simulated data is clearly excellent.

We validate the model performance on three example system with experimental AFM data. The first one is perylenetetracarboxylic dianhydride (PTCDA), shown in Fig. 5.5. The reference shows negative field over the oxygens at the ends of the molecule and positive field in the region between, which is well reproduced in the simulation prediction. The experimental prediction is also qualitatively correct with similar pattern as the reference, but the magnitude of the field is weaker over the whole molecule, roughly half of the reference value.

The second example we consider is the 1-bromo-3,5-dichlorobenzene (BCB), shown in Fig. 5.6. The reference ES Map here shows negative field over the chlorines and the middle of the molecule, close to neutral over the bromine, and positive over the hydrogens between the halides. Again, the simulation prediction is in excellent agreement with the reference. The experimental prediction has overall good agreement with the reference apart from the missing positive region on the hydrogen opposing the bromine, and is also quantitatively very close to the reference in many parts of the molecule.

The last example is of a cluster of seven water molecules on Cu(111) arranged in a pentagon with two legs, shown in Fig. 5.7. Unlike the above two examples, in this case we include the surface in the simulation, since the cluster is only stable on a surface. The stability of the structure is confirmed with DFT calculations and is considered more carefully below in Sec. 5.5. The ES Map here is found to be mostly positive over the whole
Figure 5.6. Comparison of simulated and experimental predictions of ES Maps for 1-bromo-3,5-dichlorobenzene (BCB), similar to Fig. 5.5. Reproduced from Pub. II (CC BY 4.0).

Figure 5.7. Comparison of simulated and experimental predictions of ES Maps for a water cluster on Cu(111), similar to Fig. 5.5. Reproduced from Pub. II (CC BY 4.0).
cluster with neutral regions on the left and right sides. The simulation prediction matches the reference quite well, except for the more negative regions on the sides that should be neutral. The experimental prediction is reasonable but deviates much more from the reference than the other two examples. The prediction captures the positive regions on the top water molecule and the left leg but also has a significantly more negative region in the middle of the cluster.

In the above examples we considered both the simulations and reference ES Maps using the point-charge model of electrostatics that the CNN model was trained on and found overall good performance. However, the point-charge model does not always accurately represent the real charge distribution. In order to test the validity of the point-charge approximation, we perform the PPM simulations and construct the ES Map descriptors for the above test systems using the Hartree potential from the full electron density obtained by DFT calculations (Sec. 3.4.2). The DFT calculations are performed using FHI-aims [146] with the PBE functional [147], Tkatchenko-Scheffler dispersion correction [148], and the "tight" basis set provided by FHI-aims.

As a first test, we redo the predictions on the three simulation benchmark molecules using the Hartree electrostatics, as shown in Fig. A.1. The predictions are not as good as with the point-charge model, with the relative errors going up to 6.12%, 4.78%, and 7.34%. By visual inspection, predictions are qualitatively still mostly correct but fail to capture the magnitude of the brightest areas in the reference.
Next, we test the Hartree electrostatics on an on-surface geometry of the BCB molecule, the result of which is shown in Fig. 5.8A. On the surface the bromine-side of the BCB molecule tilts down slightly, yielding an uneven contrast that matches better with the experimental images. The reference ES Map is altered as well, showing a strong positive field over the middle of the molecule as opposed to the negative field of the point-charge model. Performing the prediction for this simulation, we find that it correctly captures the positive region in the middle but misses the negative regions on the chlorines and is overall too small in magnitude. The experimental prediction actually better matches with the point-charge reference here even though the Hartree potential should in principle provide a better physical model of the electrostatic field. However, since the model is trained on in-vacuum point-charge data, we would not expect it to reproduce the Hartree-potential field for on-surface structures in those cases where they significantly deviate from each other. This is one of the limitations of the current model that needs to be addressed.

We also perform the simulation with the Hartree potential for the water cluster, shown in Fig. 5.8B. In this case, the overall pattern in the reference ES Map is largely the same as for the point-charge reference. The qualitative agreement in the simulation prediction is still very good, but the magnitude of the field is too weak, similar to the BCB case. The weaker field in the predictions seems to be a general pattern for the cases using the Hartree potential.

5.4 Molecule graph reconstruction

The image descriptor approach has so far worked quite well for better identifying the geometries of the molecules that we have studied. However, the pixel-space of the image descriptors does not directly indicate the information that we are interested in, namely the coordinates and elements of the atoms. The image descriptor still leaves a layer of interpretation to the result, for example, requiring a human to identify the sphere-shapes in the predicted vdW-Spheres descriptor. In the camphor example we overcame this limitation by matching the geometries by cross correlation in the descriptor space, but this approach only worked because we knew what the molecule was. In the more general case with a wider candidate space of molecules this approach is unlikely to work and is inefficient. In this section, we explore the use of an alternative descriptor for molecule geometries, the molecule graph, as reported in Pub. III. Compared to the image descriptors, the molecule graph is a more precise description of the molecule geometry, and allows for the application of symbolic reasoning based on physical rules and intuitions in an automated way.

The fundamental difference between the image descriptor and the mole-
Automated AFM image interpretation

1. Find atom positions

   - AFM Image
   - Convolutional neural network
   - Position grid
   - Point Cloud

2. Construct graph

   - Point Cloud
   - Label node + Add edges
   - Updated graph
   - Fully labelled graph

**Figure 5.9.** Schematic of the graph construction model. The first part of the model uses a CNN and a peak-finding algorithm to find the positions of the atoms in the AFM image. The second part takes the found positions and constructs a graph out of them using an iterative process which at each step labels one node and adds edge connections to the node. Reproduced from Pub. III (CC BY 4.0).

cule graph is that the image describes a continuous domain whereas the graph consists of a finite set of discrete objects. This creates a challenge for the design of the ML model, since the AFM input is also of the image format, which means that the model has to transition between these two domains somewhere in the middle. A function that discretizes a continuous function is not differentiable, which means that such a function cannot be used in an NN model that is trained by gradient-based methods.

One way to overcome this problem is to use an RNN that constructs the molecule one atom at a time, similar to how RNNs were previously used for constructing sentences in natural language processing and other sequence construction tasks [149, 150]. This way the model can predict the coordinates and the type of just one atom at a time, which can be done with a CNN-based RNN. However, we have found that just naively applying this approach does not work very well. The problem becomes that the iteration process is not very stable: a single mistake on an earlier step makes the whole rest of the sequence fail, not knowing when to stop, the prediction often spirals into some clearly physically impossible structure, especially for larger molecules. The problem is found to originate especially from the coordinate prediction task, where the model can easily get stuck in placing the atoms in some small region very close to each other. Inspired by this finding, we instead choose to construct a model that separates coordinate prediction task from the graph construction task.

The overall idea of the method is illustrated in Fig. 5.9. The first part of the model uses an Attention U-net to translate a set of AFM images into a 3D grid where the atom positions are represented by Gaussian peaks. The positions of the peaks are detected using a peak-finding algorithm in order
to arrive at a simple list of coordinates for the atoms. The approach here is very similar to the one with the image descriptors, and in principle one of the image descriptors could be used in place of the grid of Gaussian peaks. However, in practice we found the peak finding to work more reliably for the Gaussian peaks than any of the existing image descriptors. It should also be emphasized that the position grid is three-dimensional, even though it is not shown in the figures here, so that the atom positions can also be identified in the depth direction. The magic step of going from continuous to discrete domain happens in the peak-finding step which is based on a template matching algorithm. This algorithm is not based on ML, so it does not need gradient propagation. In fact, it is a post-processing step that can be swapped out for another algorithm if desired even after the model has been trained.

The second part of the model takes the found list of atom positions and constructs a molecule graph out of them. It does so by an iterative process that considers each of the atom positions at a time, predicting for it a label and edge connections to all the existing atoms in the graph. The labels we use here are based on the elements of the atoms, one class for each column in the periodic table. The model inside the loop is a combination of a GNN that processes the existing graph, and a CNN that accounts for information from the AFM image. I skip here the details of how this model works, which can be found in Pub. III, in favor of focusing on the improved version of the model in Sec. 5.5.

The dataset of molecules is the same one as the one used above for the electrostatics prediction task with the same augmentations. However, the PPM simulation is upgraded from the point-charge electrostatics to the Hartree potential electrostatics also for the training set, in part to address the issues encountered above with the ES Map predictions. The Hartree potentials are calculated by DFT (FHI-aims, PBE, Tkatchenko-Scheffler, tight basis) for all the molecules in the dataset. The calculations are done in vacuum, saving the Hartree potential on a large grid, which allows for interpolating the potential for a simulation scan from any direction.

We benchmark the method here with simulated data. We first look at predictions for some example systems familiar from the previous sections, the first of which is the BCB (Fig. 5.10A). Here, we find that the position grid prediction perfectly captures the positions of all the atoms, and in the predicted graph all the atoms are correctly labelled, and the graph is complete with all the bonds between the atoms.

The second case is the same cluster of seven water molecules as in the previous section (Fig. 5.10B). Compared to the planar BCB molecule, this example has a more 3D geometry, making the interpretation of the atomic structure from the AFM images more difficult. The prediction correctly identifies the top oxygen atoms and the two top hydrogens present in the reference. Note that the reference here does not contain all the atoms
Figure 5.10. Example predictions for (A) BCB, (B) a cluster of water molecules, and (C) PTCDA. In each case, from left to right, is presented the 3D structure of the system, three out of ten of the simulated input AFM images, the predicted and reference position grid, and the final predicted and reference molecule graphs. The structures of the graphs are presented as projections to the xy-plane (top) and to the xz-plane (bottom). The position grids here have been reduced down to 2D by averaging over the z-dimension. See Figs. A.2 and A.3 in Appendix A for the full 3D position grids and graph construction sequences, respectively. Reproduced from Pub. III (CC BY 4.0).
in the complete sample geometry. We choose to cut all atoms 0.8 Å below the top atom, because there is typically no direct signal of these atoms in the AFM images, and in practice we find that the performance of the model starts to drop if deeper atoms are included in the reference during training. However, in this particular case the prediction actually contains hydrogens from three of the deeper water molecules not present in the reference, indicating that these molecules could also be predictable. This example is considered again with experimental data in Sec. 5.5.

The final example is the PTCDA, which represents a fairly large graph size in comparison to our training dataset that does not contain many large planar molecules. In this case we find that the position grid prediction is very good, correctly identifying the positions of all the atoms in the reference, although with one extra atom on the left side of the molecule. However, the graph prediction in this case has mixed accuracy. While the left side of the graph is constructed flawlessly, apart from the discrepancy of the extra atom, the right side has many mistakes despite the symmetry of the molecule. The reason for this asymmetry is the relatively large width of this molecule causing the required AFM size to be larger than in the training set. The model is exhibiting a dependence on the chosen coordinate system, which is illustrated in Fig. A.4 where a shift in the absolute coordinates is shown to make the PTCDA prediction more symmetric and the predictions for the other two examples are also altered into worse direction. This is one of the primary limitations of the current model that needs to be addressed.

Next, we turn to more quantitative metrics with statistical analysis on the predictions made on the test set. The ability to provide meaningful
### Table 5.1. Precision and recall for atom type classification for the graph construction models trained with three different graph construction orders: random, decreasing y-coordinate, and decreasing z-coordinate.

<table>
<thead>
<tr>
<th>Class</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Random</td>
<td>y</td>
</tr>
<tr>
<td>1 (H)</td>
<td>0.992</td>
<td>0.987</td>
</tr>
<tr>
<td>2 (C, Si)</td>
<td>0.937</td>
<td>0.917</td>
</tr>
<tr>
<td>3 (N, P)</td>
<td>0.684</td>
<td>0.733</td>
</tr>
<tr>
<td>4 (O, S)</td>
<td>0.842</td>
<td>0.825</td>
</tr>
<tr>
<td>5 (F, Cl, Br)</td>
<td>0.905</td>
<td>0.907</td>
</tr>
</tbody>
</table>

### Table 5.2. Precision and recall for bond connection classification for the graph construction models trained with three different graph construction orders: random, decreasing y-coordinate, and decreasing z-coordinate.

<table>
<thead>
<tr>
<th>Class</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Random</td>
<td>y</td>
</tr>
<tr>
<td>Bond</td>
<td>0.997</td>
<td>0.994</td>
</tr>
<tr>
<td>No bond</td>
<td>0.999</td>
<td>0.999</td>
</tr>
</tbody>
</table>

statistical metrics on the model performance is one of the advantages of the graph descriptor compared to the image descriptors. As an additional factor, we note that the iterative nature of the graph construction process leaves a choice in the order in which the graph is constructed. We test here three different orderings: random order, ordered by decreasing y-coordinate, and ordered by decreasing z-coordinate (increasing depth).

We first consider the quality of the atom position predictions by counting the average number of missing and extra atoms in the predictions compared to the references, shown in Fig. 5.11. These counts are made by finding a one-to-one mapping between the predicted and reference atom positions by matching atoms within a threshold distance 0.35 Å, which corresponds to half of the smallest possible distance between two covalently bonded atoms (H-H bond). Atoms that do not match any other atoms within this threshold are counted as missing or extra atoms. We find that both the number of missing and extra atoms is roughly 20% of the graph size for small graphs and the number saturates to around two missing/extra atoms for larger graph sizes.

Next, we consider the accuracy of the atom type classification in the graph construction process. For each atom there is an outcome with the predicted class and the reference class, which can be gathered into a confusion matrix $M$, whose entries are defined as

$$M_{ij} = \text{number of nodes with reference class } i \text{ and predicted class } j.$$  \hspace{1cm} (5.2)

Using the confusion matrix, we can further define the precision and recall...
Automated AFM image interpretation

**Figure 5.12.** Confusion matrices of (A) atom classification and (B) bond connections for random-order graph construction. The values have been normalized by the total number of true cases of the corresponding class, and the values in parentheses show the total number of cases with the corresponding true and predicted class.

for each class $k$:

\[
\text{Precision}(k) = \frac{M_{kk}}{\sum_{i=1}^{n_c} M_{ik}} \tag{5.3}
\]

\[
\text{Recall}(k) = \frac{M_{kk}}{\sum_{j=1}^{n_c} M_{kj}} \tag{5.4}
\]

The precision and recall of the atom classification for each of the graph construction orders is shown in Table 5.1. The H-group has the highest accuracy with both high precision and recall for all construction orders. Of the other classes, only the N-group has significantly lower accuracy with especially low recall rate compared to the others. The confusion matrix in Fig. 5.12 shows that most misclassifications are made with N-group atoms being falsely labelled as C-group atoms. Comparing the different construction orders, the biggest differences are in the lower precision for the halogen group and the lower recall for the O-group for z-order, and the lower recall for N-group for both y- and z-order, but otherwise there are no significant differences. Lastly, we also consider the precision and recall for the bond connection classification, shown in Table 5.2. Here, the only notable detail is that the recall for the z-order is slightly lower for the positive bond class, but otherwise both the precision and the recall are extremely high for all construction orders.
Automated AFM image interpretation

Figure 5.13. Schematic of the one-shot graph construction model. The process starts on the left with overlaying the found atom positions (gray dots) onto the AFM image and selecting rectangular patches around these positions. A CNN then turns the image patches into the initial embedding vectors (coloured squares) for every node of the graph, and edges (dashed lines) are added to graph based on the proximity of the nodes. Finally, a GNN processes the information in the node vectors and does a classification of the nodes into atom types and a binary classification on each edge whether it corresponds to a chemical bond.

5.5 Ice nanocluster geometry reconstruction

The previous section introduced the graph construction model and tested it on simulated data. In this section the model is applied to experimental data for the specific case study of small ice clusters on Au(111) and Cu(111) surfaces.

The prediction model is modified in a couple of ways to address the issues encountered above. The overall structure is still the same: the first part of the model predicts the atom coordinates, and the second part constructs the graph. The position prediction model is still an Attention U-net, but the attention layers are modified to be able to accept variable z-size inputs, so that we can make predictions on AFM stacks of arbitrary size. The second part of the model is more significantly changed: the graph is constructed in one shot instead of iteratively.

In order to deal with variable z-sizes for the AFM image stacks, we construct here a version of the Attention Gate layer that can accept variable z-size inputs and produces fixed z-size outputs. Given a feature map $X$ with z-size $K$ and a desired output z-size $K'$, the output of the attention layer is

$$X'_{k'} = \sum_{k=1}^{K} \sigma(f_{k'}(X))_k \odot X_k \quad \forall k' \in \{1...K'\},$$  \hspace{1cm} (5.5)

where the sum is over the z dimension of the feature map, $\sigma$ is the sigmoid activation function, $f_{k'}$ is a CNN block with unique weights for each of the output z-layers $k'$, and $\odot$ denotes an element-wise product. Here, we choose to use three layers with ReLU activations in the CNN block. Adding these layers to all the skip connections and the end of the encoder branch in the U-net allows the decoder to work with a fixed-size input regardless of the AFM image stack size.

The structure of the graph construction model is simplified by going from
Automated AFM image interpretation

an iterative scheme to a one-shot graph construction, which is schematically illustrated in Fig. 5.13. The model starts by overlaying all the atom positions found in the first phase onto the AFM image and selecting patches around the atom positions within a cutoff distance. These patches are then processed by a CNN into fixed-size vectors, also using here the attention layers described above in order to deal with the variable AFM stack size. The output vectors from the CNN are used as the initial embedding vectors $h^0_v$ in a GNN that mixes the information in the node embeddings through a number of iterations. Finally, an MLP does a per-node classification to produce the predicted labels for each atom. The edges in the GNN are based on the positions of the atoms, such that atoms within a cutoff distance from one another have an edge between them. The GNN also maintains an embedding vector for each of the edges, and we perform a binary classification in the end to determine whether each edge corresponds to a chemical bond or not.

Looking at the GNN in more detail, we use here a message-passing scheme given by the set of equations

$$g^0_{uv} = \frac{(h^0_u + h^0_v)}{2} \quad \forall (u,v) \in E, \quad (5.6)$$

$$m^t_v = \sum_{u \in \mathcal{N}(v)} f_m(h^{t-1}_u, h^{t-1}_v, r_v - r_u) \quad \forall v \in V, \quad (5.7)$$

$$h^t_v = f_h(h^{t-1}_v, m^t_v) \quad \forall v \in V, \quad (5.8)$$

$$g^t_{uv} = f_g(g^{t-1}_{uv}, (m^t_{uv} + m^t_{vu})/2) \quad \forall (u,v) \in E, \quad (5.9)$$

where $h_u$ and $r_u$ are the embedding vector and the position of a node $u$ in the graph, $f_m$ is an MLP, $f_h$ and $f_g$ are Gated Recurrent Units [150], and $t \in \{1 \ldots n_t\}$ is the time step. Importantly, here the message function in Eq. (5.7) uses the relative coordinates of the atoms, which makes the predictions translationally invariant. This addresses the issue with coordinate-system dependence that we encountered for the PTCDA prediction in Fig. 5.10C. After $n_t$ iterations, the class of each node $c^v_v$ and edge $c^e_e$ is read out as

$$c^v_v = f_c(h^{n_t}_v) \quad \forall v \in V, \quad (5.10)$$

$$c^e_{uv} = f_e(g^{n_t}_{uv}) \quad \forall (u,v) \in E, \quad (5.11)$$

where $f_c$ and $f_e$ are both MLPs. The loss for both of the predictions is the cross-entropy loss (Eq. (4.3)).

In addition to the coordinate system dependence, another important issue that we ran into in the context of the ES Map predictions was that the training data did not adequately represent the distribution of the experimental AFM data, so that the predictions failed to accurately reproduce the reference descriptor when the simulation was changed to use a more physically correct model. We already switched from using point-charge
Electrostatics into using the Hartree potential for the training dataset. However, a second limitation still exists in the simulations, namely, the absence of the surface, which in some cases can make significant difference to the geometry or the electronic structure of the sample. Including the surface in the DFT calculation is significantly more expensive, especially since the sample geometry needs to be relaxed on the surface. In order to address this issue of finding the optimal geometry for on-surface samples, we train here an NNP for water molecules interacting on metallic surfaces, use it to generate thousands of realistic on-surface ice nanocluster geometries with a small computational cost, and train the geometry prediction model on simulations of these generated geometries. Furthermore, the trained NNP can also be used for validating the predicted geometries of the geometry prediction model.

The NNP that we train here is NequIP, an E(3)-equivariant GNN model that has been shown to be effective for yielding state-of-the-art level performance for atomistic energy and force predictions at very high data efficiency [67]. In order to train the NNP, we construct a dataset of ice cluster on both Au(111) and Cu(111), calculating the energies and forces for each atom with DFT. Here, the DFT calculations are carried out using the Vienna Ab-initio Simulation Package (VASP) [151] with the optB86b-vdW functional [152] that has been shown to accurately describe water adsorbing on metallic surfaces [153, 154]. For the NNP training set, we generate a total of 700 structures for both surfaces that cover typical configurations seen during the relaxation process. The per-atom mean absolute errors for the energies and forces for the trained NNP are 0.414 meV and 0.00558 meV/Å for the Au(111) model, and 0.859 meV and 0.00950 meV/Å for the Cu(111) model.

Once the NNP has been trained, it can be used for force predictions allowing a rapid generation of on-surface relaxed geometries. We still need to run a single DFT calculation for the final relaxed structure in order to obtain the Hartree potential required for the PPM simulation, but the total computational cost is reduced by one to two orders of magnitude. We use this method to generate ~2000 on-surface ice cluster geometries with between 7 and 43 water molecules, mostly arranged in single layers, but sometimes containing water molecules sitting on top of the bottom layer of water molecules. For each of the geometries, we perform the PPM simulation 10 times, with randomly chosen simulation parameters for augmentation. The randomized parameters are the lateral spring constant, the oscillation amplitude, the tip-sample distance, and the lateral equilibrium position of the PP.

We apply the model trained on the simulated ice clusters to experimental images. The overall workflow that we use for determining the ice cluster geometries from the experimental AFM images is shown in Fig. 5.14. We start by predicting a geometry with the CNN/GNN model from the
Figure 5.14. Workflow for structure discovery in AFM images of ice nanoclusters. Initially, an atomic structure prediction is inferred from an experimental AFM image stack. Incomplete molecules are then fixed, and possibly unrealistic ones are eliminated. Next, multiple hydrogen bond networks and surface positionings are rapidly evaluated by carrying out Neural Network Potential (NNP) relaxations. The most stable resulting structure is further optimized with DFT, from which simulated AFM images can be obtained and compared to the initial experiment.

We test the workflow on several experimental AFM image stacks of ice nanoclusters on metallic surfaces, shown in Fig. 5.15. Experiments A-G are performed on the Au(111) surface, and experiment H is on the Cu(111) surface. The initial geometry predictions and corresponding PPM simulations with these results is in works under the title Structure discovery in Atomic Force Microscopy imaging of ice with the authors Fabio Priante, Niko Oinonen, Ye Tian, Dong Guan, Chen Xu, Shuning Cai, Peter Liljeroth, Ying Jiang, and Adam S. Foster.
### Figure 5.15.
Geometry predictions for ice clusters. Each row corresponds to one experiment. On the left are the farthest and closest distance experimental AFM images, in the middle the geometries predicted from the experimental AFM images, and on the right the simulations based on the predicted geometries. The tip-sample distances in the simulations are manually chosen in each case to visually match the experimental images. The sizes of the atoms in the geometry indicate the relative depths of the atoms.
Automated AFM image interpretation

Figure 5.16. On-surface geometries of ice nanoclusters optimized with an NNP, corresponding simulated AFM images, and comparison to experiment. The rows A-H correspond to the experiments with same labels in Fig. 5.15. On each row, on the left is the final optimized geometry, in the middle is the simulated AFM image, and on the right the experimental image.

Simulations are also shown in Fig. 5.15. In all cases, the overall contrast seen in the simulations is remarkably similar to the one observed in the experimental images. The single big flaw is the missing bright top atom in experiment F. We can also observe many spurious lone atoms in the geometries, many of them lying significantly deeper than the rest so that they are hardly seen in the simulated images. These atoms clearly need to be filtered out or filled into complete water molecules.

The final geometries produced by the NNP optimization and DFT relaxation are shown in Fig. 5.16 along with the PPM simulations. In these final geometries, all the spurious atoms are eliminated, and the geometry is stable on the surface. Compared to the initial predictions, the overall structures of the clusters are not significantly changed, but the optimized geometries have a somewhat more 3D structure. This is especially seen in experiment C, where the bottom five-ring in the final simulation (Fig. 5.16C) only shows the brightest top atom, whereas in the initial simulation (Fig. 5.15C) has most of the ring visible. This apparent large difference in atom depths may be due to the limit in the PPM simulation that the sample geometry is fixed during the simulation. The relaxation of the sample under the influence of the probe can smooth the differences
in contrast due to height differences, making the sample seem more flat. Another factor not studied here is the possibility of water molecules forming a bilayer structure instead of a single layer, which is indicated by the relative height measurement in the STM. The geometry prediction here is not capable of detecting the second layer because the prediction has a depth cutoff, and the training dataset also contains mostly single-layered geometries.
6. Automated AFM tip functionalization

This chapter discusses a different application of ML into AFM. Whereas in the previous chapter the topic was on the interpretation of already obtained AFM images, here the focus is on preparing the AFM device for the experiment. The following presents an automated procedure for the CO functionalization of the AFM tip, as reported in Pub. IV.

6.1 Automated loop for CO functionalization

As was discussed in chapter 2, the crucial step for obtaining atomic resolution in AFM imaging of individual molecules is the functionalization of the tip. However, this is much easier said than done, and in practice this process is very time-consuming. Even when one manages to pick up the CO molecule on the tip, it is not guaranteed to produce sharp images, most commonly due to asymmetric adsorption of the CO molecule causing asymmetric distortions in the obtained images. If the tip is bad, then the experimentalist simply needs to try again until a good tip is found. The goal here is to automatize this trial-and-error process in order to save time in the preparation of AFM experiments.

The overall idea of the workflow is presented in Fig. 6.1. The process is an automated loop where the system repeatedly picks up a CO molecule, obtains an STM image of another CO molecule, assesses the quality of the CO-tip based on the STM image, and then makes the decision to either return to the beginning or terminate the loop. The crucial part here is the tip quality classifier, which looks at the STM image to determine whether the image contains significant asymmetry or other artifacts indicating a poor quality tip. The details of this model are presented in the following section.
Automated AFM tip functionalization

**Figure 6.1.** Schematic pipeline of the autonomous tip preparation procedure. The forward direction is from left to right: a bare metal tip picks up an individual CO molecule from the Cu(111) substrate; a tip terminated with a CO molecule scans over the surface to get the STM image of CO molecules on the substrate; STM images are used as input to a CNN which classifies the CO-tip as good or bad based on its centeredness; based on the quality of the tip, a decision is made whether to clean it and repeat the functionalization procedure, or to continue to the next step of the experiment preparation. Reproduced from Pub. IV (CC BY 4.0).

### 6.2 CO-tip quality assessment

The CO-tip quality classifier is a binary-classifier CNN that takes a single STM image as an input and outputs the probability of whether the tip is good or not. The layers of the network are listed in Table 6.1 and visualized on the right side of Fig. 6.1. The network has a basic CNN image classifier structure with two convolution blocks, an AvgPool layer, and two fully connected layers in the end. Additionally, following the convolution blocks there are Dropout layers with 0.25 dropout rate in order to reduce overfitting. All layers use the ReLU activation except the last layer that uses the sigmoid activation. The loss function is the cross-entropy loss, and the model is trained using the Adam optimizer.

With only \( \sim 2k \) parameters, the model here is considerably smaller than the models in the previous chapter that had hundreds of thousands to millions of parameters. This is due to the considerably smaller training

<table>
<thead>
<tr>
<th>Layer type</th>
<th>Output dimension</th>
<th>Kernel size</th>
<th>Stride</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>( 16 \times 16 \times 1 )</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Conv2D</td>
<td>( 14 \times 14 \times 4 )</td>
<td>( 3 \times 3 )</td>
<td>(1, 1)</td>
<td>40</td>
</tr>
<tr>
<td>Conv2D</td>
<td>( 12 \times 12 \times 4 )</td>
<td>( 3 \times 3 )</td>
<td>(1, 1)</td>
<td>148</td>
</tr>
<tr>
<td>AvgPool</td>
<td>( 6 \times 6 \times 4 )</td>
<td>( 2 \times 2 )</td>
<td>(2, 2)</td>
<td>-</td>
</tr>
<tr>
<td>Dropout</td>
<td>( 6 \times 6 \times 4 )</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Conv2D</td>
<td>( 4 \times 4 \times 8 )</td>
<td>( 3 \times 3 )</td>
<td>(1, 1)</td>
<td>296</td>
</tr>
<tr>
<td>Conv2D</td>
<td>( 2 \times 2 \times 8 )</td>
<td>( 3 \times 3 )</td>
<td>(1, 1)</td>
<td>584</td>
</tr>
<tr>
<td>Dropout</td>
<td>( 2 \times 2 \times 8 )</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Flatten</td>
<td>32</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Fully-connected</td>
<td>32</td>
<td>-</td>
<td>-</td>
<td>1056</td>
</tr>
<tr>
<td>Fully-connected</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>33</td>
</tr>
</tbody>
</table>

Table 6.1. CO-tip classifier CNN architecture. Amount of trainable parameters: 2157
dataset, which, unlike in the previous chapter, is obtained here from experimental STM images. The dataset consists of 21 images obtained with bad tips and 45 images obtained with good tips, each image classified by a human expert. Each image contains multiple CO molecules, which are divided into their own smaller images using the SURF algorithm [155], as shown in Fig. 6.2, and are then rescaled to a fixed size of $16 \times 16$ pixels like in the images shown in Fig. 6.3. This results in a total of 346 sample images which we divide into training and test sets in an 80/20 split, making sure that samples originating from the same experiment do not end up on different sides of the split. To further reduce overfitting, the dataset is augmented by rotations and reflections of the images during training.

The classifier is validated on the predictions on the test set. A few predictions from the worst and best predicted examples are shown Fig. 6.3, where we see that the classification is overall very good, especially for good tips, but we also see a few cases of wrong predictions for bad tips.

For a more quantitative analysis, we count the predictions in four different categories: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). In this case, the positive predictions are of good tips and negative predictions are of bad tips. The overall accuracy is the ratio of true predictions to all predictions,

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$  \hfill (6.1)

We also consider the other standard metrics of Precision, Recall (or True Positive Rate (TPR)), F-score, and False Positive Rate (FPR):

$$\text{Precision} = \frac{TP}{TP + FP},$$  \hfill (6.2)
Automated AFM tip functionalization

Figure 6.3. An illustration of a trained CNN classifier performance on STM images of individual CO molecules from the test set. Samples are sorted by the accuracy of the predicted class: with the best predictions (on the left) and the worst predictions (on the right) for the both classes. Bad CO tip examples are on the top row and the good CO tips are on the bottom. Reproduced from Pub. IV (CC BY 4.0).

Recall = \frac{TP}{TP + FN}, \quad (6.3)

F-score = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}, \quad (6.4)

FPR = \frac{FP}{FP + TN}. \quad (6.5)

The Precision and Recall here are same as the definitions in Eqs. (5.3) and (5.4), but reduced down to the binary classification case.

When making a prediction, the model outputs a value between 0 and 1 that indicates the probability of whether the tip is good or not. The actual classification is done by choosing a threshold above which the prediction is considered in the good category, and the values of the above metrics depend on the chosen detection threshold. The usual initial choice is in the middle, 0.5, but the threshold can be tuned to optimize the desired metrics. In this case we want to set the threshold fairly high in order to maximize the precision, meaning that when the model predicts a good tip, it is likely to actually be a good tip, and minimize the FPR, so that bad tips get through less often. At the same time, the higher threshold leads to a lower recall rate, meaning that some good tips may be classified as bad ones, but this is not catastrophic, because in this case the automated loop simply tries again with another tip.

The relationship between the TPR and FPR can be visualized in the receiver operating characteristic (ROC) curve, where the TPR is plotted against the FPR for different values of the threshold. The ROC-curve for our classifier is shown in Fig. 6.4, where we see that we can achieve a very good TPR at a low FPR, and the high area under the curve (AUC = 0.993) indicates that the classifier is generally of good quality. In our tests with the current test set, we find a reasonable choice for the threshold to be 0.9. The confusion matrix and the values of the accuracy metrics for this
Figure 6.4. A summary of prediction results for the CO-tip classifier: a confusion matrix at 0.9 confidence level, numerical metrics of success and a ROC curve. In the confusion matrix, the number of correct and incorrect predictions are summarized with count values and broken down by each class: true negative prediction (TN), false positive (FP), false negative (FN) and true positive (TP). Reproduced from Pub. IV (CC BY 4.0).

threshold, shown in Fig. 6.4, demonstrate that the current classifier can be used for reliable detection for the quality of CO tips.
7. Conclusion

In summary, I have presented in this thesis multiple applications of ML for the automated interpretation of AFM images. The main objective was the extraction of the atomic geometry and the electrostatic field from AFM images. This was first explored with the use of a CNN for predicting image descriptors of the atomic geometry with a specific application to finding the adsorption geometries for a camphor molecule on a Cu(111) surface. The same methodology was then applied to predicting another image descriptor of the electrostatic field of the molecule, where we found a good result for the point-charge electrostatics model but a mixed result when switching to use the more physically accurate Hartree potential model. Stepping away from pure CNN models, the following section presented another method for the geometry prediction task using a GNN in combination with the CNN model in order to construct a molecule graph starting from an AFM image. The performance of this model was first demonstrated on simulated data and was then applied to experimental images in the specific task of finding the geometries of small ice clusters. The predicted geometries produced simulated images very closely resembling the experimental images, and after a following step of refinement by an NNP, fully relaxed geometries could be produced, validating the original predictions. The final application in the following chapter stepped to a slightly different direction of applying ML to the AFM device preparation by automating the tip functionalization process, and thus helping the AFM experimentalist save time.

One of the important aspects that was highlighted in the results for the ES Map predictions in Sec. 5.3 is the importance of the training data in ML applications. We saw that despite the near-perfect performance on the simulated test set, the interpretation of the experimental predictions was left ambiguous due to the differing reference ES Maps for point-charge and Hartree electrostatics. This difference in the training distribution of the simulated images and the final test distribution of the experimental images is perhaps the main difficulty at this point, and it speaks to the importance of accurate simulation data. To this point, we also saw quite accurate predictions for the geometries of ice clusters from experimental
AFM images in Sec. 5.5, where we were able to produce simulations for on-surface geometries of very similar kind as seen in the experiments via the use of an NNP. However, we should not expect this level of performance to naively generalize to wider chemical spaces for a couple of reasons. First, water molecules tend to form very specific kinds of networks, so the ML model does not need to learn a very big space of possible configurations, and second, the training of the NNP becomes more difficult the more different atomic species are present in the geometries. Despite these difficulties, the general workflow of using an NNP for generating data for the training of another ML model seems a promising direction for future studies.

In addition to the accurate geometries, another important factor for the accuracy of the PPM simulations is the employed force-field model. In the applications in this thesis, all the PPM simulations used the Lennard-Jones potential supplemented either with point-charge or Hartree potential electrostatics. However, another force-field model was also presented in the theory part in Sec. 3.4.3, namely, the FDBM that provides a more accurate approximation for the Pauli repulsion force. This force-field model has already been employed for ML applications in the QUAM-AFM dataset [106], which has been used by other authors for predicting both image descriptors [107] and IUPAC names [108] of molecules from AFM images. We have been experimenting with this dataset in our group as well, finding a degree of success in improving the accuracy of the trained model, but also finding that the trained model generalizes poorly to more 3D molecules, since the QUAM-AFM dataset was consciously chosen to include only quasi-planar molecules. The FDBM generally remains a relatively unexplored simulation model, specifically in the choice of the two parameters $V_0$ and $\alpha$, which have been fitted to DFT calculations for various systems, but there is no general rule for a good choice of the parameter values. Given the wide range of fitted values in the existing literature [60, 61], the choice to use fixed values for $V_0$ and $\alpha$ for every molecule in the QUAM-AFM dataset becomes questionable. Perhaps a better choice would be to find some distribution of reasonable parameter ranges for both $V_0$ and $\alpha$, and randomize the values similarly to how some of the other simulation parameters are randomized. Given that the FDBM is now implemented also in the GPU version of PPM, its application should become easier and faster for future datasets.

Another important aspect is of course the ML model itself. The models used in the work for this thesis saw many changes along the way, starting from the simple encoder-decoder CNN model, ending at the combined CNN/GNN model, and having many iterations in between which never made it to any publication. The current graph construction model is already highly accurate on the simulation test sets, for example achieving near perfect accuracy on the ice cluster simulations. Therefore, the much more pressing concern is the problem of generalizing from simulations to
experiment that was highlighted above. However, there are improvements to be made to the ML model as well. To this end, one can look into the equivariant networks used in the NNPs such as the NequIP model, which has a remarkable data efficiency of requiring as few as some hundreds of data points to train an accurate potential. A big part of this efficiency can be attributed to the equivariance properties which get rid of the need to augment the dataset with symmetric transformations, since the model actually sees those transformed data points as being equivalent. Our current graph model already respects the translational symmetry of the problem, since both the CNN and the update rule in the GNN have this symmetry property. However, the rotation and reflection symmetries are not respected. This means that, for example, rotating the atom coordinates by 90 degrees before the graph prediction could change the predicted atom labels, and this symmetry has to be rather learned by augmenting the dataset with rotations. Some of the ideas from the NNPs could be used here, but not directly, since our problem also includes the AFM image that is processed by a CNN, so the symmetry would somehow need to be retained throughout the transformation from the image domain to the GNN node embeddings. This is going to be a challenge, but the benefit is clear in requiring less training data and ensuring correct behaviour, which may also help with the problem of generalization to experimental data.

In the end looking at the bigger picture, it is still relatively early days for ML applications in SPM. At the moment the struggle is to bridge the world of simulations to the real world of the experimental apparatus, because there is not enough data for the model to learn in the real world, but perhaps it does not have to be so. Reminiscent of Feynman’s famous idea of small robots building down to atomic scale, one could imagine bootstrapping a process for SPM, where real world data is gradually added to the ML model training, and as the model gets better, more sample systems of increasing complexity can be analyzed with gradually increasing level of automation. We are only in the beginning stages, but given the results in this thesis for automated AFM image interpretation and tip functionalization as well as the recent demonstration for automated atom manipulation with SPM [73], at least the initial building blocks are there.
References


References


[121] Jiali Li et al. “Machine Vision Automated Chiral Molecule Detection and Classification in Molecular Imaging”. In: *Journal of the American Chemical Society* 143.27 (2021), pp. 10177–10188. DOI: 10.1021/jacs.1c03091.


[138] François Chollet et al. Keras. https://keras.io. 2015.


References


Figure A.1. Predictions for the electrostatics benchmark examples using the DFT Hartree potential for electrostatics in the simulations. Compare to Fig. 5.4. Reproduced from the SI of Pub. II (CC BY 4.0).
Additional figures

**Figure A.2.** Full 3D position grids for the predictions in Fig. 5.10. Reproduced from the SI of Pub. III (CC BY 4.0).
Figure A.3. Graph construction sequences for the three example systems in Fig. 5.10. Light-grey circles without outline correspond to positions of atoms that have not yet been added to the graph. Colors of the circles correspond to the atom classes as in Fig. 5.10. Reproduced from the SI of Pub. III (CC BY 4.0).
Figure A.3. (Continued) Graph construction sequences for the three example systems in Fig. 5.10. Light-grey circles without outline correspond to positions of atoms that have not yet been added to the graph. Colors of the circles correspond to the atom classes as in Fig. 5.10. Reproduced from the SI of Pub. III (CC BY 4.0).
Figure A.4. Graph predictions on the three test systems with coordinates shifted by −2Å in the x direction. Reproduced from the SI of Pub. III (CC BY 4.0).