Machine Learning at the Nanoscale

Oliver Miles Gordon

ID: 4341846

Submitted to the University of Nottingham

for the Degree of *Doctor of Philosophy*

September 2021

University of Nottingham

UK | CHINA | MALAYSIA

Abstract

Although scanning probe microscopy (SPM) techniques have allowed researchers to interact with the nanoscale for decades now, little improvement has been made to the incredibly manual, time consuming process of setting up, running, and analysing the results of these experiments, often arising due to the constantly varying shape of the probe apex. Unlike traditional computing methods, machine learning methods (with neural networks in particular) are considerably more capable of automating subjective tasks such as these, and we are only just beginning to explore the potential applications of this technology in SPM.

In this thesis we explore a number of areas where machine learning could potentially massively change the way we go about SPM experimentation. We begin by discussing the history, theory, and experimental concepts of scanning tunnelling microscopy (STM), atomic force microscopy (AFM), and normal-incidence-x-ray standing wave (NIXSW). We then explore the makeup of a neural network and demonstrate how they can be applied to a variety of use-cases in SPM, including classification and policy prediction. Moving to the experimental chapters, we first discuss how we can successfully distinguish between STM tip states of the H:Si(100), Au(111) and Cu(111) surfaces. We also show that by adapting this network to work in real time, we improve performance while requiring on the order of 100x less data.

We next discuss our attempts to combine these networks with expert examples to intelligently maintain tip apex sharpness during experimentation, envisioning an end-to-end automatic experiment. Because one of the main difficulties in applying machine learning is the frequent need to manually label data, we then show how we can use Monte Carlo simulations of self-organised AFM nanostructures to automatically label training data for a network, and then combine it with classical statistics and preprocessing to find specific structures in a mixed, messy dataset of real, experimental AFM images. As part of this, we also build a network to denoise experimental images. Finally, we present NIXSW results from an investigation into the temperature dependence of H20@C60, discussing the potential to use unsupervised clustering techniques to distinguish between noisy human-indistinguishable spectra to overcome limitations in data collection.

Acknowledgements

The last few years have been some of the most enjoyable of my life, and I owe this to a great number of hugely important people. First and foremost is of course my supervisor, Phillip. You are, quite simply, just inspiring. I will never forget the sheer amount of faith you put in me and the way you took my stubborn refusal to ever enter a lab and provided me with nothing but kindness and multiple coffee-cups of your trademark enthusiasm in return. When I first started this PhD I said I wouldn't do one for anybody else, and I maintain that to this day.

For Matt, Filipe, Jo, Abi, Ellie, the Alexes, the rest of the nano group, and the many people I have had the pleasure of working and/or sharing an office with, thank you for being so much fun and always such supportive friends. For Jo (Melton), Chris, Roshany, Mum, Dad, Beth, and all other family and friends past and present too great in number to mention - but Matt (double mention!) and Dom *especially* - I truly cannot thank you enough or put into words just how much I appreciate you. You made me the person I am today, so as far as I'm concerned it's all your fault really.

Massive thanks obviously also goes to all those I have worked with in admissions, demonstrating and outreach for being such generally awesome people, the technical staff for their timely help, Walther and the British Council team for a hugely enjoyable week in Pakistan, and everybody who put up with me overexcitedly talking about it non-stop. And finally I must of course also congratulate the Omicron VT for being such a worthy opponent (and COVID-19 for ruining what would have been an amazing year of international travel, cheers for that).

Ultimately however, I wish to dedicate this thesis to my late grandparents, Harold and Fay Cohen. I miss you both, and I hope this would make you smile.

List of Publications

O Gordon, P D'Hondt, L Knijff, SE Freeney, F Junqueira, P Moriarty, and I Swart. Scanning Tunnelling State Recognition with Multi-Class Neural Network Ensembles. *Review of Scientific Instruments*, 90(10):103704, 2019.

O Gordon, F Junqueira, and P Moriarty. Embedding Human Heuristics in Machine-Learning-Enabled Probe Microscopy. *Machine Learning: Science and Technology*, 1(1):015001, 2020.

O Gordon, and P Moriarty. Machine Learning at the (Sub) Atomic Scale: Next Generation Scanning Probe Microscopy. *Machine Learning: Science and Technology*, 1(2):023001, 2020.

O Gordon, J Hodgkinson, S Farley, E Hunsicker, and P Moriarty. Automated Searching and Identification of Self-Organized Nanostructures. *Nano Letters*, 20(10):7688-7693, 2020.

S Farley, J Hodgkinson, **O Gordon**, E Hunsicker, and P Moriarty. Improving the Segmentation of Scanning Probe Microscope Images Using Convolutional Neural Networks. *Machine Learning: Science and Technology*, 2(1):015015, 2020.

S Jarvis, H Sang, F Junqueira, **O Gordon**, J Hodgkinson, A Saywell, P Rahe, S Mamone, S Taylor, A Sweetman, J Leaf, D Duncan, T. L Lee, P Kumar, R Whitby, G Held, L Kantorovich, P Moriarty, and R Jones. Chemical Shielding of H_2O and HF Encapsulated Inside a C_{60} Cage. *Communications Chemistry*, 4(1):135, 2021.

O Gordon. Machine Learning at the Nanoscale. Invited book chapter in Supramolecular Nanotechnology: Advanced Design of Self-Assembled Functional Materials, edited by M Conda and O Azzaroni. *VCH-Wiley (In Production)*, 2021.

Reuse of Materials Statement

For grandma and grandpa. For everything.

Contents

1 Introduction

1.1 Are the Nanobots Nigh?

Nanobots! They break objects down into their component atoms and then recombine those atoms to repair damaged circuits. Nanotechnology! This is a lead pencil; it's made of graphite, which is a particular arrangement of carbon atoms. This is diamond; it too is made of carbon atoms. Nanobots can rearrange atoms, so they could take this lead pencil, move the atoms around a bit, and turn it into diamond!

Thus spoke Kryten 2X4B-523P in the episode 'Nanarchy' of the classic 1980s comedy Red Dwarf. Incredibly for a neurotic cleaning mechanoid with a bachelor's degree in Sanitation Studies and a head shaped like an amusing ice cube, Kryten's nanobots are closer to nanoscience research today than some of the more popular predictions of our nano-future, from the now infamous[1] 'nanolouse' to thoughts of implanting nanobots in our brains to "expand human intelligence by factors of thousands or millions" by 2030[2].

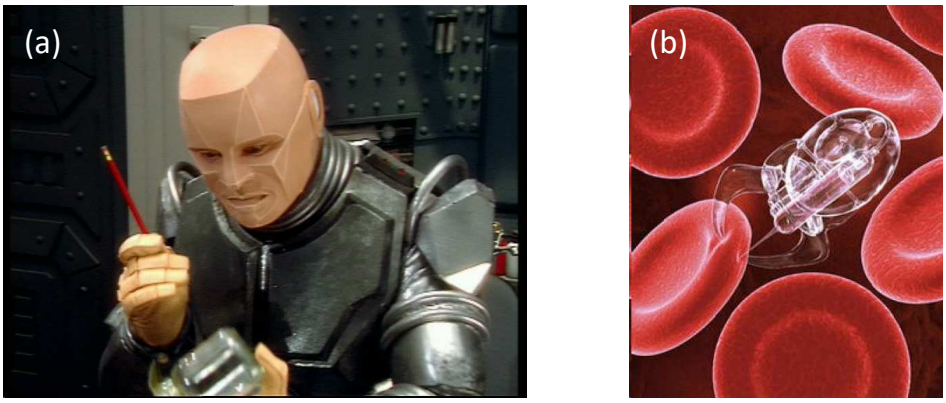

Figure 1.1.1: Two very different interpretations of the 'nano-future'. In **(a)**, Kryten from the BBC comedy 'Red Dwarf' shouts at his nanobots for deserting him, while **(b)** depicts the infamous 'nanolouse', which has accompanied news articles for almost 20 years. Kryten image is © BBC Worldwide (1997). Nanolouse image is © Coneyl Jay (2002).

It is easy to appreciate the excitement, however. Manipulating the very building blocks of the universe is an inherently engaging concept, especially when it seems so intuitive; rearranging the atoms that make up grass into a steak sounds remarkably futuristic, until we realise that the name of this incredible piece of technology is a 'cow', first farmed 10,500 years ago[3]. Yet while many nanoscience concepts in popular media are far-fetched, the major difference between research today and Red Dwarf's nanobots two million years into deep space is, remarkably, largely one of scale. Today's scanning probe microscopy (SPM) instruments, albeit nowhere near small enough to store millions of in a glass jar, already allow us to poke, prod, push and probe at the atomic level, create dominoes of computer code[4], build quantum corrals[5], write our employer's name in Xenon atoms[6], and perhaps even image inter-atomic bonds[7]. All of this, routinely, and with atomic precision.

So, what is the catch? Why can we not yet fabricate atomic scale gears[1]? The answer, somewhat surprisingly, is our inability to obtain and maintain control of the tools and data we already have available to us. Kryten can scare his nanobots into intelligently and collectively rebuilding a spaceship, but sadly real SPM techniques are much less forgiving. Today's state of the art involves spending hours upon tedious hours of busywork trying to gradually (and *semi-randomly*) coerce our probes into a usable form, where 'usable' means we decided our data 'looks good'. In effect, we are rebuilding an increasingly unstable house of cards over and over until we reach our own fallible interpretation of correct. The limiting factor is not one of precision, but of patience, luck, reliability, and *trust*.

However, where human stubbornness and conventional computing techniques have failed[8], machine learning concepts with roots dating back to WW2[9,10] can become a key method of asserting our control over the nanoscale. Key to this are neural networks[10], which, thanks to significant increases in computing power and widespread adoption of free, open source tools such as Python and Tensorflow[11,12], have resulted in fundamental improvements to a huge variety of predictive[10,13], classification[10,14,15] and strategical[16,17] tasks. One of the most famous of these is the 70,000 sample MNIST[18] challenge to classify hand-written digits from 0-9. Simple linear classifiers circa 1998 have a 12% error rate,[18,19], but convolutional neural networks (CNNs) have completely solved this, with the best performing published result having just 0.16% error rate[14]. There is immense information to be inferred from images and multi-dimensional SPM datastreams, and machine learning may prove an important means of unlocking this.

Indeed, one of the key tools in the probe microscopist's box of tricks, the scanning tunnelling microscope (STM), heavily relies on constantly maintaining the sharpness of the probe apex. At its core, this involves noticing a difference in image resolution, then jolting and prodding the tip and hoping we change its shape for the better (but often for the worse). While there have been a number of traditional computing based attempts[8,20,21] to automate this time consuming process, none of these bore much fruit until 2018, when Bob Wolkow's group in Alberta, Canada, took a crucial first step towards this by detecting the presence of 'double-tip' defects on the H:Si(100) surface at fixed scan parameters[22] using a CNN. Since then, the group has expanded this to enable detection, classification, and avoidance of various defect structures on this surface[23].

Further, once we acquire our data, we still have the issue of *subjectively* processing all of it; unlike computers, humans cannot perform billions of calculations per second, and worse still are inherently biased, bottlenecking the research process. Artefacts and noise may also make this even more difficult. Once again, machine learning again provides an exciting means to make better use of the abundance of SPM data, using AI methods often used in reverse image search or detecting anomalous bank transactions. These are just some of the areas of machine learning we will cover in this thesis.

1.2 Thesis Outline

In this thesis we will cover a wide variety of areas in which machine learning techniques have the potential to be transformative to SPM, but with a particular focus on assessing and selecting STM tip states.

We begin with Chapter 2, where we first introduce the history of and key physics behind the major experimental instruments we have applied machine learning methods to; the scanning tunnelling microscope (STM), a device of particular focus, the atomic force microscope (AFM), and normal incidence x-ray standing waves (NIXSW).

Moving onto Chapter 3, we continue our theoretical and literature discussions, but with a focus on key machine learning methods (namely neural networks), their mathematical makeup, and their applications. We will first introduce the most basic neural network use-case of supervised classification. We then expand on this to discuss semi-supervised and fully unsupervised strategies of autoencoders and reinforcement learning, along with many of the common pitfalls and optimisations of neural network design.

From here, we then move into our applications of this theory. In Chapter 4 we discuss our attempt to automatically detect a variety of STM tip states and defects on the H:Si(100), Cu(111) and Au(111) surfaces. After discussing these surfaces, introducing our dataset, and discussing our filtering and pre-processing routines, we compare the classification performance of variety of neural network structures at this task. We then narrow our focus to the H:Si(100) surface and demonstrate and verify how these classification routines can be adapted to allow for real-time classification of tip state; crucial for these methods to be let loose in live experimentation. We also introduce an open-source Python controller for Scienta Omicron's Matrix; nOmicron.

Next, in Chapter 5 we follow up our ability to *assess* STM tip state, and then attempt to *intelligently sharpen* it using reinforcement learning, as aided by the demonstrations of multiple human experts.

In Chapter 6, we then shift gears to a new instrument and a new problem area; overcoming the data we waste and potentially misinterpret due to the need to manually pre-process and trawl through sheer numbers of scan data. We discuss Monte Carlo modelling, and how the Rabani et. al[24] algorithm can be implemented and exploited to produce a diverse range of distinct structures that can be automatically labelled and used as training data for a classifier neural network. We then show how, combined with simple statistics and denoising autoencoders, we can automatically preprocess and classify an example dataset of over 5000 historic AFM scans.

Finally, we conclude the thesis with Chapter 7 with a brief investigation of the temperature dependence of $H_2O@C_{60}$. Particularly, we focus on an example of the surprising ability of unsupervised clustering techniques to theoretically distinguish between otherwise human-indistinguishable spectra. We then finish with a conclusion in Chapter 8.

2 Experimental Techniques

2.1 Scanning Tunnelling Microscopy (STM)

2.1.1 Basic Principles & Tersoff-Hamann Theory

Originally developed by Binnig and Rohrer in 1981[25,26] (for which they were subsequently awarded the Nobel Prize in 1986), scanning tunnelling microscopy (STM) is now one of the primary methods by which the atomic structure of conducting and semiconducting surfaces can be investigated.

In STM, an atomically sharp tip is brought to a distance, z, within the order of sub-nm from the surface to be investigated. This gap forms a 1D potential barrier, $U(z)$. At this distance, we enter the quantum regime in which it is possible for quantum tunnelling to take place between electronic states with energy E at the tip and surface. This is visible in Figure 2.1.1.

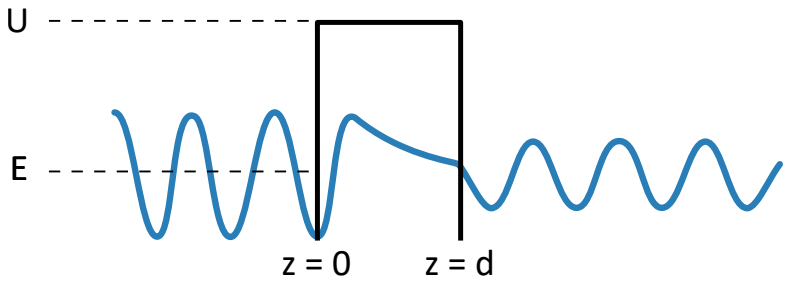

Figure 2.1.1: Quantum tunnelling through a 1D potential barrier. If an electron quantum tunnels between a surface of interest and a scanning tip, the resulting current can be measured and over several positions translated into an STM image.

The resulting wavefunction, ψ, can be described by the equations[27]

$$\psi(z) = \begin{cases} Ae^{ikz} + Be^{-ikz} & z < 0 \\ Ce^{\alpha z} + De^{-\alpha z} & 0 < z < d \\ Fe^{ikz} & d < z, \end{cases} \quad \{2.1\}$$

where $k = \frac{\sqrt{2mE}}{\hbar}$ and $\alpha = \frac{\sqrt{2m(U-E)}}{\hbar}$. m is the mass of an electron, and A, B, C, D and F are constants.

To appreciate how this idealised function results in a tunnel current, therefore ultimately making STM possible, we can exploit the fact that both $\psi(z)$ and $\frac{d\psi}{dz}$ must be continuous at the boundaries of the potential barrier. At $z = 0$, we therefore find that

$$Ae^{ik\cdot 0} + Be^{-ik\cdot 0} = Ce^{\alpha\cdot 0} + De^{-\alpha\cdot 0}, \quad \{2.2\}$$

$$\therefore A + B = C + D, \quad \{2.3\}$$

and

$$\frac{d}{dz}Ae^{ik\cdot 0} + Be^{-ik\cdot 0} = \frac{d}{dz}Ce^{\alpha\cdot 0} + De^{-\alpha\cdot 0}, \quad \{2.4\}$$

$$\therefore ik(A - B) = \alpha(C - D). \quad \{2.5\}$$

Similarly, for $z = d$, it follows that

$$Ce^{\alpha d} + De^{-\alpha d} = Fe^{ikd}, \quad \{2.6\}$$

$$\alpha(Ce^{\alpha d} - De^{-\alpha d}) = ikFe^{ikd}. \quad \{2.7\}$$

We can then solve these equations to allow us to determine the probability of an electron in this system tunnelling. To do so, we begin by multiplying Equation 2.3 by ik, allowing us to equate it to Equation 2.5 and find

$$2ikA = (ik + \alpha)C + (ik - \alpha)D. \quad \{2.8\}$$

Next, we make the important assumption that the barrier is large and wide enough such that $\alpha z \gg 1$. Because this means that $e^{-\alpha d} \approx 0$, by multiplying by $e^{-\alpha d}$ we can see from Equation 2.6 that $C \approx 0$. Equation 2.8 therefore simplifies to

$$2ikA \approx (ik - \alpha)D. \quad \{2.9\}$$

Next, we can combine Equations 2.6 and 2.7 to find

$$2\alpha De^{-ikd} = Fe^{ikd}(\alpha - ik). \quad \{2.10\}$$

Combining Equations 2.9 and 2.10, we eventually determine

$$\frac{4ik\alpha}{ik - \alpha}Ae^{-\alpha d} \approx Fe^{ikd}(\alpha - ik). \quad \{2.11\}$$

Finally, because the probability of finding an electron at a given z is proportional to $|\psi(z)|^2$, it follows that the transmission coefficient for the tunnelling electron, T, is

$$T = \frac{|F|^2}{|A|^2} \approx \frac{16k^2\alpha^2}{(k^2 + \alpha^2)^2}e^{-2\alpha z}. \quad \{2.12\}$$

Given that transmission probability is proportional to transmission current, I, it therefore additionally follows that[27,28]

$$I \propto eVe^{-2\alpha z}. \quad \{2.13\}$$

Because I has an exponential dependence on z, we find that for typical materials, moving the tip by 1Å results in a reduction in current of approximately one order of magnitude. It is for this reason that incredibly high resolution in z, of the order of pm, can be obtained in STM, allowing us to image atomic scale features.

While the 1D tunnelling model captures the essential physics of the problem, it inevitably does not capture the finer details of the true interactions. It is therefore important to recognise

the seminal work of Tersoff and Hamann[29]. As formulated by Bardeen[30] in 1961, the tunnel current for two metals in vacuum can be described as

$$I = \frac{2\pi e}{\hbar} \sum_{t,s} f(E_t)[1 - f(E_s + eV)]|M_{ts}|^2 \delta(E_t - E_s), \quad \{2.14\}$$

where $f(E)$ is the Fermi function. Additionally, the $f(E_t)$ and $(1 - f(E_s + eV))$ terms ensure that tunnelling must be between filled and empty states by giving the probability for an electron to occupy a filled energy level in the tip. M_{ts} is the tunnelling matrix between the tip, t, and the sample, s, which computes the probability of the tip and sample wavefunctions overlapping, an obvious requirement for tunnelling to occur. The δ function is required for conservation of energy for elastic scattering.

By assuming that the tip and sample are only weakly coupled, V is small, and that the Fermi functions can be approximated by their 0K counterparts, Tersoff and Hamann find[29]

$$I = \frac{2\pi e^2 V}{\hbar} \sum_{t,s} |M_{ts}|^2 \delta(E_t - E_F)\delta(E_s - E_F). \quad \{2.15\}$$

Assuming that the tip wave function can be modelled as a single point at position $\mathbf{r}_0$, this then simplifies further[29] to

$$I \propto \sum_{s} |\psi_s(\mathbf{r}_0)|^2 \delta(E_s - E_F). \quad \{2.16\}$$

where $|\psi_s(\mathbf{r}_0)|^2$ is the probability density of the surface wavefunction at the position of the tip, for a perfectly sharp probe. I is therefore directly proportional to the local density of states (LDOS) of the surface[29]. Given that only the electron states between E_F and $E_F + eV$ contribute to tunnelling, we can adapt this to the slightly more realistic equation of

$$I \propto \int_{E_F}^{E_F+eV} |\psi(\mathbf{r}_0)|^2 \delta(E - E_F) T dE. \quad \{2.17\}$$

As we will discuss in Subsection 2.1.3, it is a very poor assumption that tip apex is a perfect point. Without reverse atomic imaging of the tip and density functional theory (DFT) codes it is exceptionally difficult to analytically model the effect of tip shape[21] on surface interaction - a major *raison d'être* for the use of machine learning in this thesis.

Regardless, it can be seen from these equations that there must be a difference between the Fermi levels of tip and surface for a tunnel current to occur. Tunnelling itself still occurs, as thermal fluctuations result in the electrons having a distribution of energies (and so E_F is defined by the average energy at which half of states are filled, and half are unfilled). However, on average the same number of electrons tunnel from the sample to the tip as the tip to the sample, and so there is no net (i.e. measurable) tunnel current. To create this difference in Fermi levels we can apply a bias voltage[27] V, which shifts them relative to each other by eV. This makes the potential barrier trapezoidal, as shown in Figure 2.1.2.

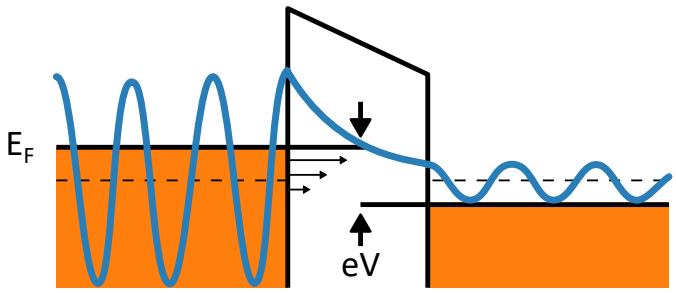

Figure 2.1.2: Quantum tunnelling through a 1D potential well with an applied voltage. When the Fermi levels of the two materials do not align, there is a quantum mechanical tunnelling current, which can be measured over several positions and translated into an STM image.

If the Fermi level of the tip is higher than the sample (i.e. positive sample bias), electrons will tunnel from filled electron states in the tip, to unoccupied electron states of the sample. Conversely, if the Fermi level of the tip is lower than that of the sample (i.e. negative bias), electrons tunnel out of the filled sample states in the sample into empty electron states in the tip. STM does not explicitly map sample topography but actually the (typically highly complex) local density of states (LDOS). This is shown in Figure 2.1.3. One convenient consequence of this is that changing scanning bias allows for tunnelling of different states, and so we can observe different features at different scanning biases.

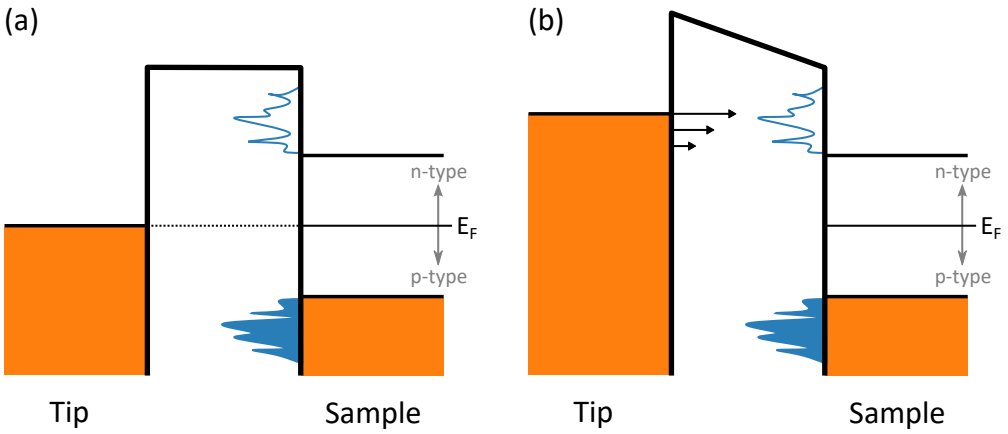

Figure 2.1.3: Tunnelling from a metallic tip to a semiconducting sample. In **(a)**, the semiconductor is undoped, there is no voltage bias between tip and sample, and so the Fermi levels align. In **(b)**, the semiconducting sample has a positive voltage relative to the tip, and so there is a net tunnel current from the tip to the sample.

It can also be seen from Figure 2.1.3 that we can still have a tunnel current when we have a semiconducting sample such as silicon. With a p-type dopant, electron holes are created low in the band gap, lowering the energy required to promote an electron from the valence to the conduction band. E_F is therefore lowered. With an n-type dopant, electron states are created high in the band gap, lowering the energy required to promote to the conduction band, and therefore raising E_F.

2.1.2 Taking a Scan

Now that we appreciate that the current measured by the probe is directly proportional to the electron density of the surface being investigated, we can exploit this to probe the surface.

Firstly, we can capture spectra by holding the tip over a fixed point, adjusting a parameter such as V, and monitoring the tip current or height. This will give us a spectra indicative of the subject of interest. Alternatively, we produce an image by rastering the tip back-and-forth (with nm precision through the use of piezoelectric crystals). The current (or tip height for reasons we will discuss next) can then be mapped to pixel intensity on a display, producing a false colour image of our (semi-) conductive[26] sample. This process is shown in Figure 2.1.4.

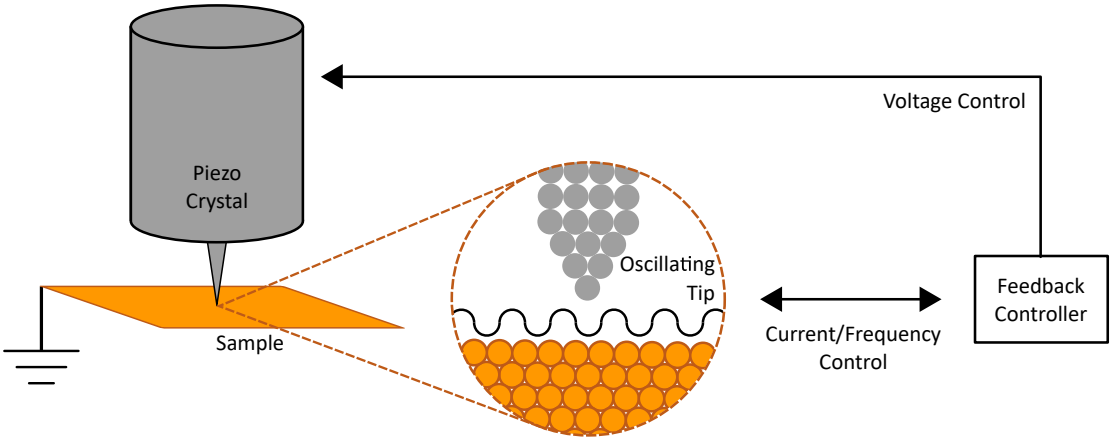

Figure 2.1.4: Figure to demonstrate the basic operation of an STM. A voltage-biased tip (ideally atomically sharp) is brought close to a conductive sample, resulting in a quantum tunnelling current. By rastering the tip back-and-forth across the sample, a false-colour image can be produced by mapping current to pixel-brightness. Feedback control is used in constant-current mode to raise/lower the tip to prevent tip-surface contact. (N.B. the drawn oscillation of the tip is for illustrative purposes only, and should follow the atoms below).

Further, we can position the probe with atomic precision, allowing us to simultaneously manipulate and image at the atomic scale[6], producing images such as that of Figure 2.1.5. Regardless, we rarely use current measurements when producing images, as the tip can constantly 'crash' into the sample, resulting in tip (and sample) degradation, especially when the surface is extremely rough. Instead of operating in the so-called 'constant height' mode, we instead operate in 'constant-current' mode. As depicted in Figure 2.1.6, a Proportional-Integral-Differential (PID) controller can be used to raise and lower the tip height to meet a target of a user-defined current setpoint. This slows the scan considerably. Parameters of the controller must also be set carefully, or non-physical artefacts can be introduced. Examples of this include artificial 'ringing', the potential presence of which has sometimes led to significant debate in the literature[31].

Given the extreme precision of UHV SPM, a high degree of vibration isolation is required. This leads scanners to be placed in low-foot-traffic areas, on heavy tables and air legs to decouple them from the floor and nearby UHV pumps, with careful setup of power supplies to eliminate 50Hz noise. Further, scan heads are also mounted to be freestanding on carefully balanced springs, with large eddy current dampeners used for further vibration isolation.

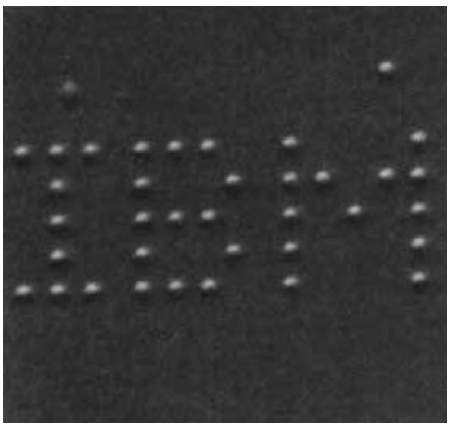

Figure 2.1.5: The now famous 'IBM logo' STM image of Xe on Ni(110), one of the very first STM images published involving atomic manipulation, demonstrating the ability of STM to simultaneously manipulate and image at the atomic scale[6]. T=4K, V=0.01V, I=1nA. Each letter is 5nm tall. Reprinted by permission from Springer Nature: D. M. Eigler et al, Positioning single atoms with a scanning tunnelling microscope. © Springer Nature (1990).

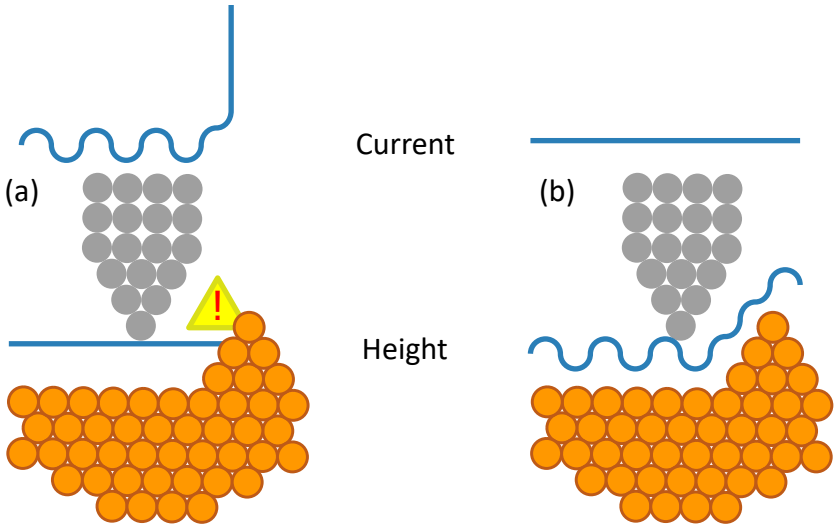

Figure 2.1.6: Major modes of STM operation. Because of the high likelihood of tip-sample collisions in constant height mode **(a)**, we use constant current mode **(b)**, in which a PID controller raises and lowers the tip height, avoiding collisions. (N.B. the drawn path of the tip is for illustrative purposes only, and should follow the atoms below).

2.1.3 Tip States & Tip Sharpening

Crucially, there is a strong, non-linear relationship between the shape of the tip apex and the recorded current. Further, this is certainly not as simple an issue as 'sharp' vs 'blunt'[21]. Not only can multiple visual states of surfaces, such as those of Figure 2.1.7, be observed depending on tip shape, but it is sometimes the case that tips that maximise visual resolution may not necessarily be the best for spectroscopy or atomic manipulation[28,32–37].

Moreover, not only are 'poor' images common, but they can take a number of vague and non-specific forms, as shown in Figure 2.1.8. In particular, double (or even multiple) tip apexes can produce 'ghosting' artefacts. Others (such as step edges, absorbed impurities, or

reconstruction defects) can be due to flaws with the specific surface itself, and be wholly unrelated to the sharpness of the tip apex. Further, there is likely a non-linear relationship between state and tip bias, making state recognition more complex still.

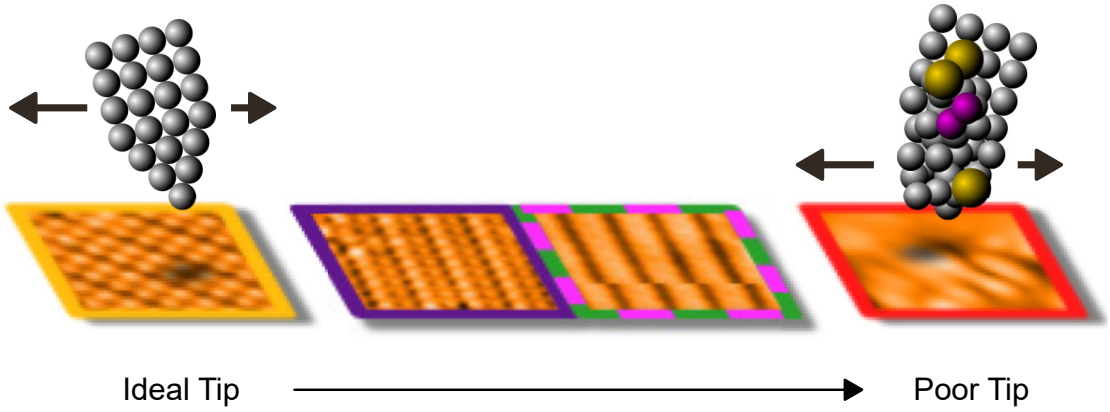

Figure 2.1.7: Demonstration of the profound effect the shape an STM tip has on a scanned image, in this case the common H:Si(100). Note that image quality does not degrade linearly, but instead progresses through several distinct tip 'states'. Further, a tip ideal for imaging is not necessarily the best for other tasks such as manipulation, and there is therefore a significant desire to coerce the tip into one which produces images of a specific state.

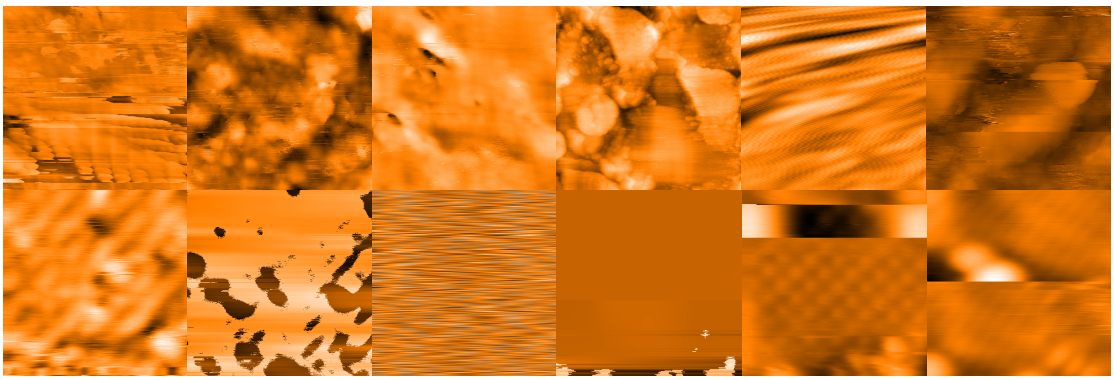

Figure 2.1.8: Various STM images of H:Si(100) taken with imperfect tips at 300K. Note that the tip often changes in-and-out of multiple states during a scan. Parameters right to left, up to down: area = 10x10, 10x10, 10x10, 300x300, 70x70, 9x9, 32x32, 10x10, 20x20, 4x4, 4x4, 5x5nm, V=-1.0, -1.9, -1.9 -0.5, -1.4, -2.0, -2.4, -2.0, 4.0, 1.6, 1.6, 1.6V, I=0.5, 0.15, 2.5, 0.1, 0.15, 0.14, 0.1, 1, 0.01, 0.01, 0.01nA.

Compounding the issue even further, while it is relatively simple to create atomically sharp tips *ex-situ*[38,39], the constant tip-surface interactions means that there is a constant need to sharpen the tip *in-situ*. Unlike ex-situ techniques such as ion beam lithography, in-situ methods can only be generously described as 'somewhat low-tech'. Here, microscopists are not unlike a caveman with a club; we can crash the tip into the surface in the hope of knocking something off (or picking something up), apply a sudden pulse in the hope of encouraging the tip into a more co-operative shape, or more likely waste half a day attempting a vague combination of the two to perform this vital operation. It is these unfortunate truths that are a significant motivator of this thesis, particularly Chapters 4 and 5.

2.1.4 The Omicron VT

The University of Nottingham Nanoscience Group are fortunate to have access to a large, diverse range of STM systems. However, experimentation in this thesis was primarily focused on one system; a Scienta Omicron UHV variable temperature (VT) system, pictured in Figure 2.1.9. The VT is an extremely versatile, reliable SPM platform, with Scienta Omicron reporting over 500 VT SPM systems installed worldwide[40] to date.

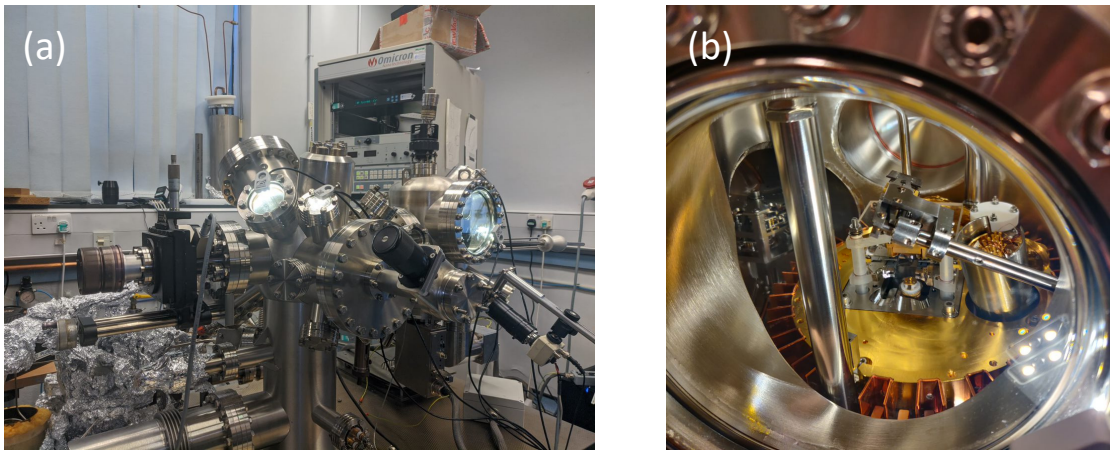

Figure 2.1.9: Photos of the Scienta Omicron VT SPM System. **(a)** overview, **(b)**, close up of scan head, manipulator and eddy current dampeners.

The VT system has many features beyond its low base pressure of 10^{-11} mBar through the regular use of its titanium sublimation pump (TSP), ion and roughing pumps. In our case, it also has a manipulator arm capable of direct and indirect sample heating to $>$1200K, a (non-standard) hydrogen gas cracker and attached cold trap, along with an argon sputter gun for tip preparation.

2.2 Atomic Force Microscopy (AFM)

2.2.1 Basic Principles

Besides STM, atomic force microscopy (AFM) is also a popular method to explore insulating materials with atomic resolution, as developed by Binnig et al. in 1986[41]. While insulating surfaces cannot be probed with STM, quartz based qPlus sensors allow for STM and AFM to be performed on a sample in parallel[26]. Using techniques such as frequency-modulated AFM, it is also possible to achieve sub-atomic resolution beyond that of STM[26].

Using both techniques, there exist a wide variety of exciting research areas, such as creating logic circuits on the atomic scale[4,42] using hydrogen-terminated silicon surfaces. Many modern SPM experiments also involve functionalisation of the tip, wherein an additional molecule is "picked up" from the surface[43]. Notably, CO was shown in 2009 by Gross et al. to dramatically increase AFM resolution for the pentacene molecule[7], and is now being used to image many other surfaces[44,45].

At its core, AFM exploits interatomic, and more generally, tip-sample forces such as the Lennard-Jones potential[46] (depicted in Figure 2.2.1). Tip-sample interactions are dominated by short range Pauli repulsion forces and longer range Van-Der-Waals attraction forces (other forces such as covalent, ionic and hydrogen bonding between tip and sample, as well as electrostatic, magnetic, and lateral forces such as friction and dissipation also contribute to a lesser extent). By bringing a cantilever[41] of spring constant k close to the surface, it therefore experiences a force $F = kz$. This deflects the cantilever, which can be detected e.g. with a laser, as shown in Figure 2.2.2. However, in the more often used qPlus sensors[26], which allow for both AFM and STM in parallel, 'self-senses' using a quartz crystal which both drives the tip and provides an electrical signal proportional to the deflection, allowing for sensing.

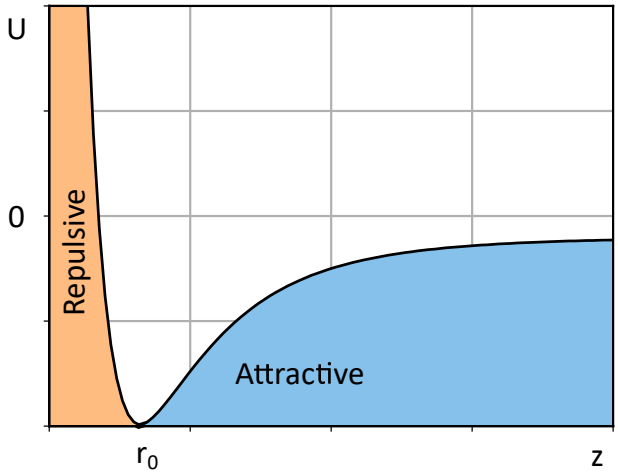

Figure 2.2.1: The Lennard-Jones potential curve.

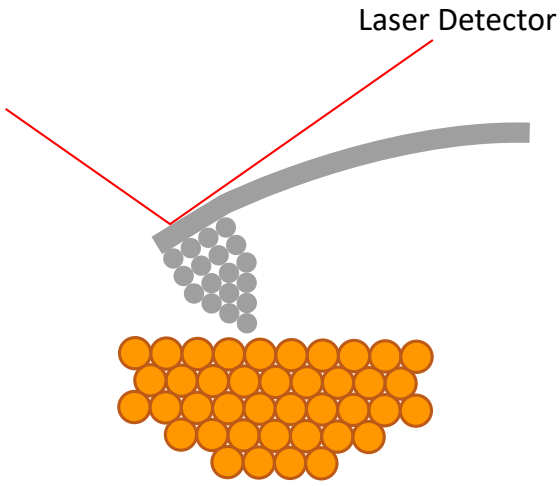

Figure 2.2.2: Basic operating principle of an Atomic Force Microscope (AFM). By bringing a tip mounted on a cantilever incredibly close to a surface of interest, Pauli exclusion/Van Der Waals interactions will cause the cantilever to deflect, which can then be observed with a photodiode.

2.2.2 Contact & Non-Contact Modes

Regardless, we have so far limited our discussion to a 'static' cantilever. However, we more often perform 'non-contact' mode AFM. (While 'contact mode' AFM is more destructive by comparison, it is still possible to probe relatively delicate samples in UHV with this method). This can also be performed with qPlus sensors, which allow us to perform AFM and STM simultaneously[26]. These sensors now see routine use, and have recently been used to explore logic circuits created on the atomic scale[4,42] using hydrogen-terminated silicon surfaces. Using a piezoelectric quartz tuning crystal, the consistent vibrational frequency of quartz often used in watches to keep time can instead be used to self-sense the frequency shift caused by interactions with the surface. To do so, we oscillate the cantilever of quality factor Q at its resonant frequency, f_{res}, with amplitude $|A_{drive}|$. Tip-sample interactions result in a change in the vibrational frequency of the cantilever, Δf. For a small oscillation amplitude, Δf is then proportional to the force gradient, $\frac{dF}{dz}$, with the equation[28]

$$\frac{\Delta f}{f_{res}} = -\frac{1}{2k}\frac{dF}{dz}. \qquad \{2.18\}$$

By using PID controlled feedback to keep dF/dz constant by modulating either amplitude[41] or frequency[47], we can then build up an image. This is known as 'tapping' mode. Amplitude is the simplest means to this end; the changing tip-sample interactions will adjust the amplitude of oscillation, $|A|$, as

$$|A| = \frac{A_{drive}}{\sqrt{\left(1 - \frac{f^2}{f_{res}^2}\right)^2 + \frac{1}{Q^2}\frac{f^2}{f_{res}^2}}}. \qquad \{2.19\}$$

As such, the cantilever height z can be approached or retracted and recorded to match a setpoint value of $|A|$. Instead, we can use a phase-locked loop (PLL) to track the frequency shift. A key advantage of frequency modulation is that the update time in frequency[48] is $\approx 1/f_{res}$. Given that the update time for amplitude moderation[48] is $\approx 2Q/f_{res}$ and typical UHV cantilevers can have Q of the order[48] of 10^4, amplitude moderated AFM can be unusably slow, and so is rarely employed in UHV.

2.3 Normal-Incidence X-Ray Standing Waves (NIXSW)

2.3.1 Standing Wave Production

Normal incidence x-ray standing wave (NIXSW, often simplified as XSW[49]), along with LEED, provides yet another means to determine the structure of surfaces and molecules. Instead of directly probing the material of interest, we exploit interference of the crystal with incident photons, as depicted in Figure 2.3.1. This allows us to determine the position of an adsorbate with respect to the crystal plane it lies in, and by repeating at multiple planes, ultimately triangulate its position.

In XSW, we begin with a crystal aligned in a particular crystallographic direction, with periodic layer spacing d_H. We then direct an incident soft x-ray at it of intensity E_0, (and therefore wavelength, λ, as $E = hc/\lambda$), and order n. For reasons touched upon later, we set up the crystal such that the incoming x-ray beam is at normal incidence to the planes of

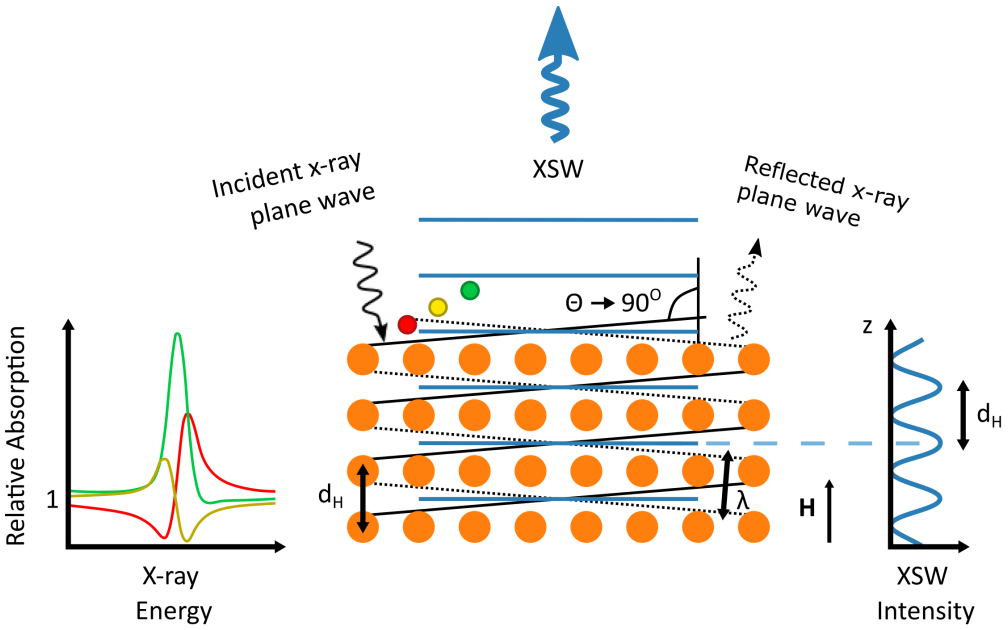

Figure 2.3.1: Production of an X-ray Standing Wave (XSW). After directing an x-ray into the bulk of a surface, the incident and reflected waves, with a constant phase difference, produce a stationary wave with positional intensity that is characteristic of the surface.

interest, $\theta_B = 90°$. By performing the experiment at a synchrotron, we can then tune E_0 so that it matches the Bragg condition for scattering from those planes, defined as[50]

$$n\lambda = 2d_H \sin(\theta_B). \qquad \{2.20\}$$

We then have an incident X-ray beam and a back-scattered X-ray beam, each of which are normal to the planes of interest (hence "normal incidence" XSW). As the incident and reflected waves have a consistent phase difference, ϕ, they therefore interfere together to produce a standing wave of intensity I. At a distance z (in the same plane as d_H) from the crystal, I is given by the equation[50–52]

$$I = \left|1 + \frac{E_H}{E_0} \exp\left(\frac{-2\pi i z}{d_H}\right)\right|^2. \qquad \{2.21\}$$

Because of the photoelectric effect (and to a lesser extent Auger electrons, x-ray fluorescence photons, and other inelastic scattering events[50,53,54]), the standing wave results in x-ray absorption[50], followed by photoelectron emission. Because the emitted photoelectrons have properties dependent on the electron levels of the emitting element, it is possible to distinguish between emissions from the bulk crystal and an elementally different molecule of interest fixed atop the crystal. This is because the emission also depends on the location of emitting atoms to the XSW. Atoms centred at the nodes of the XSW will radiate strongly, while atoms at the anti-nodes will not radiate at all[54], and be "invisible". However, because the incident x-ray is scattered multiple times, the reflectivity, R, (much like ϕ) of the crystal is a function of the energy of the incident x-ray. This dependence produces a flat-topped reflectivity (otherwise known as a Darwin or "rocking") curve[50,52,54].

While it is possible to vary θ_B instead of E, the width of the reflectivity curve, $\Delta\theta$ would be incredibly low[50]. This would not only limit experimental sensitivity, but also place incredibly strict requirements on the crystal quality, and angular spread of the beam source. However, as

$$\Delta\theta = \frac{\Gamma F_0}{\sin 2\theta_B}, \qquad \{2.22\}$$

we can instead position the incident beam such that $\theta_B = 90^\circ$ (i.e. a 'normal' incidence). Here, F_0 is the structure factor of scattering in the (000) plane, and Γ a constant dependent on λ and the unit cell. As a result, $\sin 2\theta_B \to 0$ and thus $\Delta\theta \to \infty$. This produces a much wider reflectivity curve, and so modern XSW experimentation is performed by varying E at fixed $\theta_B = 90^\circ$.

2.3.2 Characterising Materials

Next, we must consider how we can detect this emission. Typically, we can use a fluorescent screen placed normal to the beam to ensure good normal alignment. This is achieved by finding the peak reflection intensity as we sweep through a large range of E past the Bragg energy. In doing so we can ensure maximum signal-to-noise. While a fluorescent screen could be used to detect absorption/emission events, in a modern beamline, photoelectrons are amplified by a microchannel plate (with a 'low pass' minimum energy used to reduce noise) and detected using a CCD not dissimilar to one used in a digital camera. By monitoring photons hitting the CCD (of which one axis corresponds to photon intensity and the other position on the same) over a period of time, we can begin to build spectra.

Importantly, the phase of the XSW, ϕ, shifts by π radians between both ends of the reflectivity curve, meaning that the positions of nodes and anti-nodes of the XSW switch as E of the incident ray is altered. Away from E_B, where the incident beam does not reflect, absorption and emission are roughly equal (with an arbitrary emission intensity of $I = 1$). Because I is proportional to amplitude squared, emission intensity becomes $I = 4$ at maximum interference. As such, the emission intensity with respect to E is characteristic of the emitting element/orbital.

At this point, we can characterise our structure by first substituting the relationship

$$\frac{E_H}{E_0} = \sqrt{R}e^{i\phi}, \qquad \{2.23\}$$

into Equation 2.21, to find

$$I = 1 + R + 2\sqrt{R}\cos\left(\phi - \frac{2\pi z}{d_H}\right). \qquad \{2.24\}$$

However, due to thermal fluctuations and disorder in the surface, z is not single-valued. Instead, we consider emitting atoms[50,52,54] to have a small distribution, dz, over the periodic range d_H. We therefore modify Equation 2.24 into the form

$$I = 1 + R + 2f_{co}\sqrt{R}\cos\left(\phi - \frac{2\pi D}{d_H}\right), \qquad \{2.25\}$$

with the new parameters of coherent position, D, and coherent fraction, f_{co}. ϕ can be calculated from E_0, and R from the rocking curve. We can now determine the position of an adsorbate with respect to the crystallographic planes of the substrate by plotting I against E_0. An example XSW spectrum of Ag(111) is shown in Figure 2.3.2.

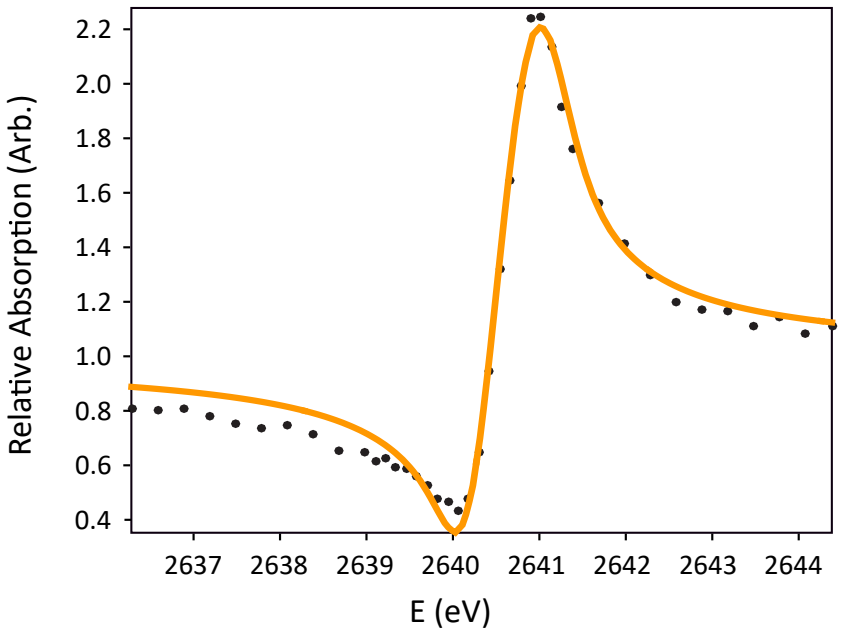

Figure 2.3.2: Example graph of XSW intensity vs Bragg energy (orange) for the Ag(111) substrate. This fit is largely dependent on the coherent position (0.28 ± 0.01), coherent fraction (0.46 ± 0.02), and fit to reflectivity (blue).

Finally, a fitting algorithm[55] can then be used to determine D and f_{co}. Mathematically, these are the phase and amplitude, respectively, of the first Fourier component of the real-space structure[56]. D is proportional to the average height of the absorber atoms above the Bragg planes, and if the system is perfectly ordered (i.e. all the absorbing sites are in the exact same position relative to the crystal surface), $f_{co} = 1$.

2.3.3 Diamond Light Source & Beamline i09

The requirement for soft/hard x-rays means that XSW is not possible outside of a synchotron. For experimentation in this thesis we acquired data from Diamond Light Source in Oxford, originally opened in 2007. Of the 32 experimental beamlines available, beamline i09 (shown in Figure 2.3.3) is specialised towards high resolution studies of atomic surfaces, can provide energies ranging from 100eV - 20keV, and allows for a range of scanning techniques including XSW, x-ray photoelectron spectroscopy (XPS), hard x-ray photoelectron spectroscopy (HAXPES) and near edge x-ray absorption fine structure (NEXAFS).

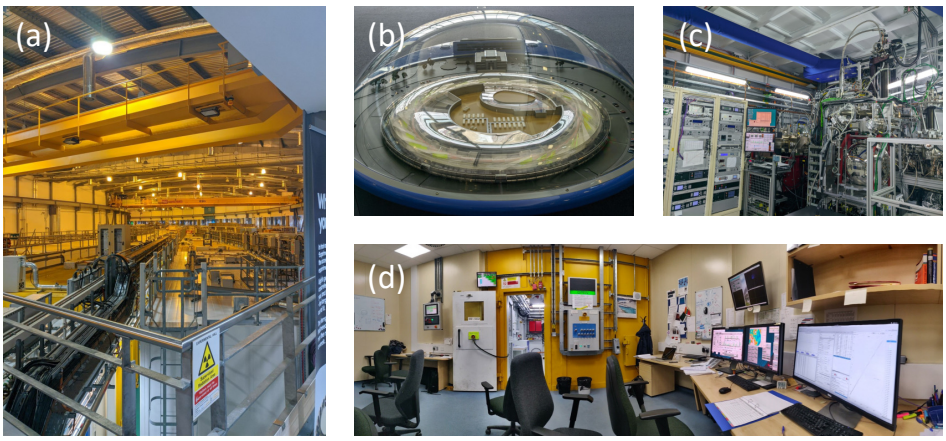

Figure 2.3.3: A collection of images showing **(a)** the main experimental hall and **(b)** a miniature model of Diamond Light Source in Oxford, UK. Also shown in **(c)** and **(d)** is the experimental 'hutch' and control room of beamline i09.

3 Machine Learning Methods

3.1 Introduction

As touched upon in the opening of this thesis, machine learning techniques (particularly neural networks), while sometimes employed as an over-hyped buzzword, have in recent years exploded in popularity and now appear in all aspects of daily life, from improving phone camera processing[57] and adjusting screen brightness[58] to detecting fraud, enhancing web search and predicting lameness in cows[59]. Although the very core of the concept dates back nearly a century[9,10], only over the last few years has the computing power and ease-of-access (via open-source packages such as Tensorflow[12], Keras[11], and OpenAI Gym[17]) become mainstream enough to allow for its widespread use. In particular, convolutional neural networks (CNNs) act as feature extractors with 2D images (or any other multidimensional information). Given a huge number of examples, relationships not even necessarily visible to humans can be understood, and this understanding applied to a subjective task too nuanced for conventional computing analysis.

One of the base-standard tasks in machine learning is to use CNNs to perform the now quintessential supervised classification task of distinguishing 70,000 handwritten digits[10,60,61]. Unlike conventional computing methods which inherently cannot cope with subtle and subjective tasks, neural networks are now trusted enough to automatically read cheques in many bank ATMs[62]. They can even out-perform human doctors in breast cancer biopsies[64] (ethical debate notwithstanding[63]).

The most common use of machine learning; supervised learning (discussed in Section 3.2) can be thought of much like a student and a teacher. We, the teacher, create a textbook on a particular topic. The student (i.e. the neural network), reads the questions *and answers* in the textbook over and over again, learning how to relate the questions to the answers. We then test the students' understanding, by periodically giving them an exam, in which the student *does not know the answers*. If the student passes the exam to our satisfaction, we conclude that they have learnt about the topic (and not just the answers by rote), and can then go apply their knowledge to the real world.

Indeed, many machine learning applications have already been made to SPM. Notably, just before this project was undertaken, in 2018 Rashidi and Wolkow published an influential work[22] in which they claimed to use supervised machine learning to detect the presence of 'double-tip' artefacts by looking at dangling bond defects on images of the H:Si(100) surface taken at fixed scan parameters. Crucially, the task being automated is incredibly routine in STM, and despite the high desire to automate this the subjectivity in STM image recognition[21] means that despite multiple attempts, conventional computing means[8] have never been particular adept at coming anywhere near to automation of otherwise exceptionally simple and tedious routine experimental tasks. The Alberta based group has since extended this work[23] to allow for detection, classification, and avoidance of various defects on this same surface.

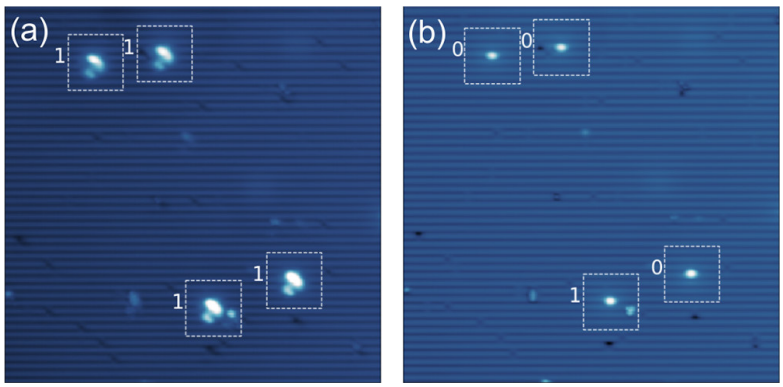

Figure 3.1.1: Use of machine learning to distinguish sharp STM tips (labelled 0) from 'double tip' defects (labelled 1) on the H:Si(100) surface using the appearance of dangling bonds as a reference. 40x40nm images were recorded at V=-1.8V, I=50pA, T=4.5K. Reprinted with permission from M. Rashidi and B. Wolkow, *Autonomous Scanning Probe Microscopy in Situ Tip Conditioning through Machine Learning*, ACS Nano 2018, 12, 6, 5185-5189. © (2018) American Chemical Society[22]

Elsewhere in SPM, other significant machine-learning enabled advances include Aldritt et al[65], who built on previous machine learning protocols developed by Sergei Kalinin and co-workers at Oak Ridge National Laboratories (among others)[66–69], to create what they describe as automated structure discovery AFM (ASD-AFM). This deep learning framework uses a large set of images simulated via DFT optimisation of molecular structures and the probe-particle model of tip-sample interactions developed by Hapala et al[70,71] as a basis to match experimental data to molecular geometry. For similar purposes, Wahl et. al[72] used a slightly modified k-means clustering algorithm to correctly distinguish *and map* the 40:60% ratio of AFM spectra of the $FeSe_{0.4}Te_{0.6}$ surface. Further, they did this in an *unsupervised* fashion (semi-supervised for Aldritt). In other words, there is no need to write the textbook; the network learns by itself to find boundaries that best split the data. This is beneficial, as the requirement to have huge amounts of data is a widely accepted bottleneck in the machine learning process, resulting in the extreme value of data collection *and manual labelling* to companies such as Google and Facebook.

3.2 Supervised Classification

3.2.1 The Basic Classifier

In a supervised network, we begin with a vector input[10], $\mathbf{x}$, containing n real-numbered data points. In the context of SPM, this could be the n pixels of a scanned image, or data points from a spectroscopy sweep. Our ultimate goal is then to map this data to another vector, $\mathbf{y}$, of length k. In this discussion we mainly consider a "classifier" network, which will categorise $\mathbf{x}$ as belonging to one of k distinct states. In other areas such as predictive "regression", $\mathbf{y}$ could instead be a prediction of house price[73] over k timesteps.

We do this by "learning" to approximate a function, $f(\mathbf{x})$, such that

$$\mathbf{y} = f(\mathbf{x}), \qquad \{3.1\}$$

where[10] $f : \mathbb{R} \to \{1, \ldots, k\}$.

For our main particular use case (i.e. assessing the state of an STM tip), each of the k values of **y** represents the probability that the tip was in one of k tip states when **x** was recorded. An outline of a tip-state classifier is shown in Figure 3.2.1.

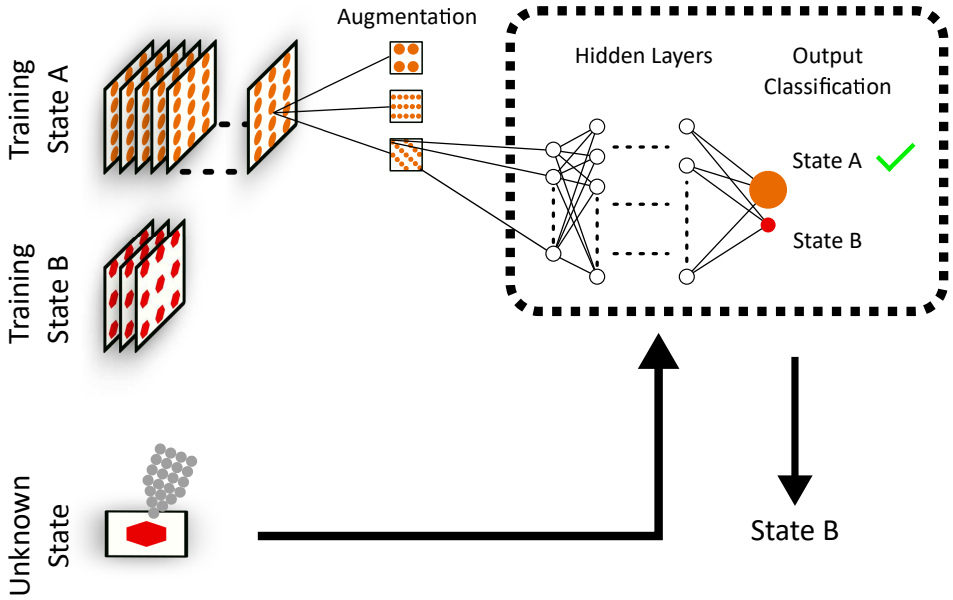

Figure 3.2.1: A supervised neural network used to classify STM tips. By manually labelling (and later augmenting) training data, parameters of nodes and neurons can be "trained" to best map the pixel-wise inputs of the scan images to their manually labelled classifications. This network can then be used to correctly classify new scans[74].

Regardless, in supervised learning[10], we determine $f(\mathbf{x})$ such that $\mathbf{y} = \hat{\mathbf{y}}$, where $\hat{\mathbf{y}}$ is the "correct" output of a given **x**. This typically requires manually collecting M samples of **x** and $\hat{\mathbf{y}}$.

3.2.2 Nodes & Neurons

Looking at $f(\mathbf{x})$ more closely, **x** is linked to **y** through a series of nodes and neurons, as inset in Figure 3.2.1. L horizontal layers, l, of interconnected nodes, link input to output. Each layer passes a weighted average of the values in its vector of inputs, $\mathbf{a}^{(l)}$, to its output, $\mathbf{a}^{(l+1)}$ (where $\mathbf{a}^{(0)} \equiv \mathbf{x}$, and $\mathbf{a}^{(L)} \equiv \mathbf{y}$), via the equations[10]

$$\mathbf{z}^{(l)} = \left(\mathbf{w}^{(l)\mathsf{T}} \cdot \mathbf{a}^{(l-1)}\right) + \mathbf{b}^{(l)} \qquad \{3.2\}$$

$$\mathbf{a}^{(l)} = g^{(l)}\left(\mathbf{z}^{(l)}\right). \qquad \{3.3\}$$

where **b** and **W** represent the learnable hyperparameters of biases and weights of the nodes in each layer. This is visualised in Figure 3.2.2.

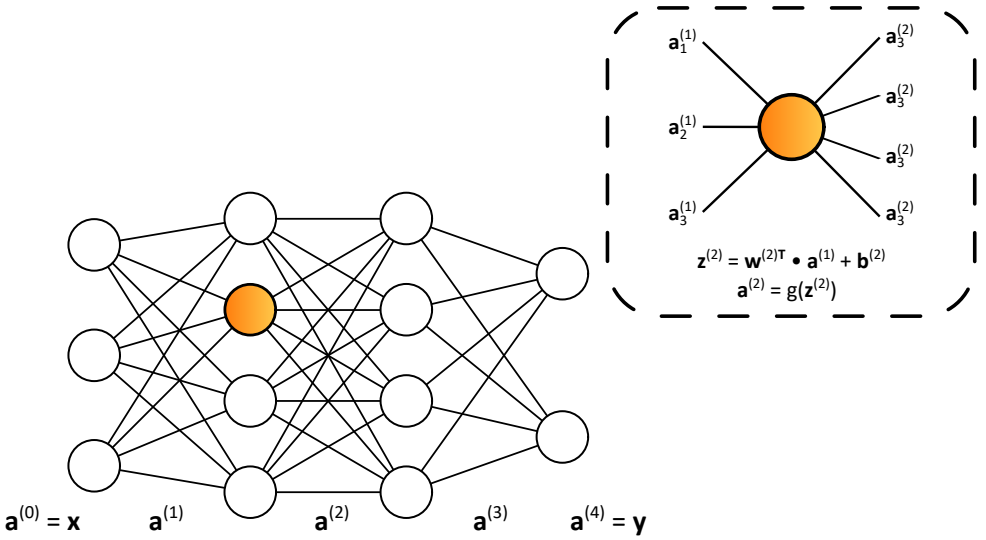

Figure 3.2.2: Pictorial representation of how neurons transfer information from input, **x**, to output, **y**, via multiple layers, l, of nodes, $\mathbf{a}^{(l)}$, in a neural network.

Our ultimate goal is to determine values of **W** and **b** that will best map **x** to **y**, a process described in detail in Subsection 3.2.5. For now, we will discuss how the output of each node (sometimes called the transfer function), **z**, is input into a non-linear activation function, $g(\mathbf{z})$, which will link our node to further levels of nodes, and eventually to **y** at the last layer to give us an output.

3.2.3 Activation Functions

Because the intermediary output **z** is linear and can take any value $-\infty \leq \mathbf{z} \leq \infty$, it is difficult to make either very large or very small changes to the flow of information, and so a network is likely to rapidly tend to $\pm\infty$. As such, we apply a non-linear activation function, $g(\mathbf{z})$, which bounds the output from -1 or 0 to 1, after each layer as in Equation 3.3. Besides $\tanh$, common intermediary activation functions include Rectified Linear Units (ReLU) and $\mathrm{Leaky\ ReLU}$, and are described by the equations[10]

$$\mathrm{ReLU}(\mathbf{z}) = \max(0, \mathbf{z}) \qquad \{3.4\}$$

$$\mathrm{Leaky\ ReLU}(\mathbf{z}) = \max(0.1 * \mathbf{z}, \mathbf{z}). \qquad \{3.5\}$$

Crucially, these functions are differentiable, and have first derivatives in the range $0 \leq g'(\mathbf{z}) \leq 1$, which assists with the back-propagation routine discussed later in Subsection 3.2.5. Additionally, the activation function of the final layer is typically chosen such that $0 \leq g(\mathbf{z}^{(L)}) \leq 1$. By doing so, the network will output a confidence for each of the k classifications. A confidence of 1 corresponds to high confidence that **x** is a given category, while 0 denotes high confidence that **x** does not belong to a given category of classification.

Typically, sigmoid or softmax functions are used for $g(\mathbf{z})$, which typically defined as[10]

$$\mathrm{sigmoid}(\mathbf{z}^{(L)}) = \frac{1}{1 + e^{-\mathbf{z}^{(L)}}} \qquad \{3.6\}$$

$$\text{softmax}(\mathbf{z}^{(L)}) = \frac{e^{\mathbf{z}^{(L)}}}{\sum_k e^{z_k^{(L)}}}, \qquad \{3.7\}$$

respectively. Sigmoid functions are typically used in multiclass classifiers (in which only one classification is possible for each image), and softmax in multilabel (in which multiple classifications can be made for each image). This is as the sigmoid function must sum to 1 across the k outputs, but the softmax limits each value of z_k be 0 to 1 independent of each other. Equations 3.4 to 3.6 are graphed in Figure 3.2.3 (Equation 3.7 varies with input size, so cannot be comparably graphed).

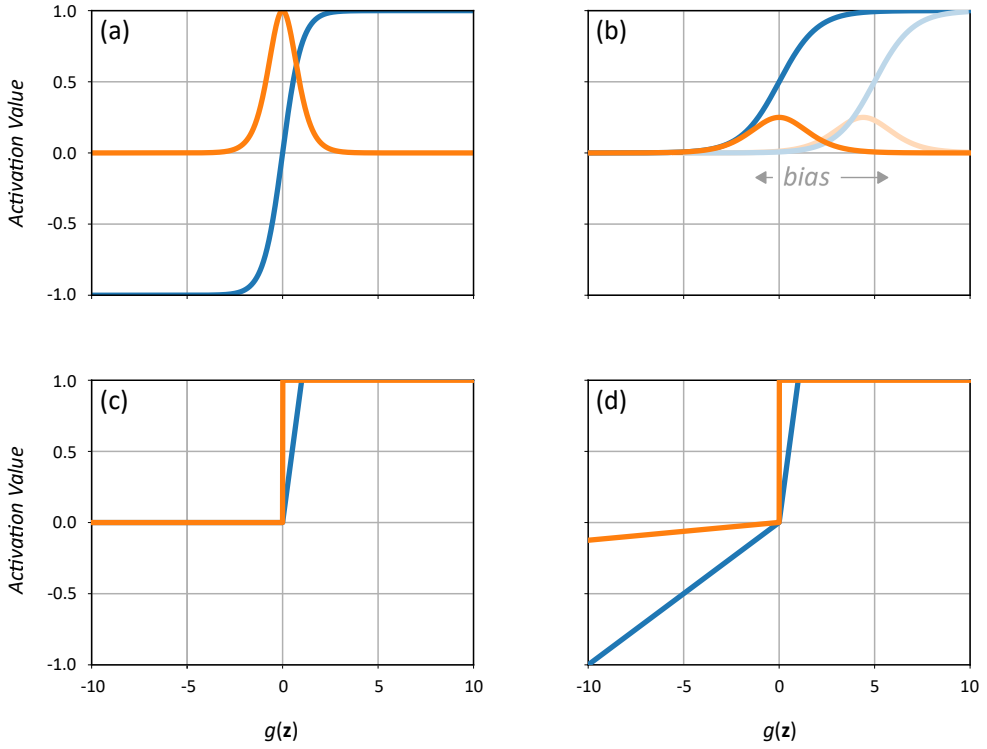

Figure 3.2.3: Figure demonstrating the values (blue) and first derivatives (orange) of the **(a)** tanh, **(b)** sigmoid, **(c)**, ReLU, and **(d)** Leaky ReLU activation functions. Biases can also be added to these functions to vary the output depending on $g(\mathbf{z})$, as depicted in (**b**).

3.2.4 Convolutional Layers

One of the major performance enhancements that can be made when **x** is image-based is to apply convolutions between the layers with a convolutional neural network (CNN). By applying a series of convolutions to Equation 3.2 at the same time[10] as **W**, **x** can be compressed into its raw visual features. At the same time, it reduces the number of parameters in $f(\mathbf{x})$ to optimise, therefore simplifying the learning task and in turn improving performance.

To apply convolutions, three additional parameters are required: a convolutional kernel, its size, and its stride. Different kernels extract different features, such as the direction of edge detection[10,75]. Not only does the kernel size have the same effect, but larger kernels allow

for greater feature compression. However, smaller kernels maintain more detail, meaning that the kernel size must be carefully selected to reasonably represent its input. The same also applies for strides, in which the kernel is shifted by more than one pixel between convolutions to allow for greater feature compression. Examples are shown in Figure 3.2.4.

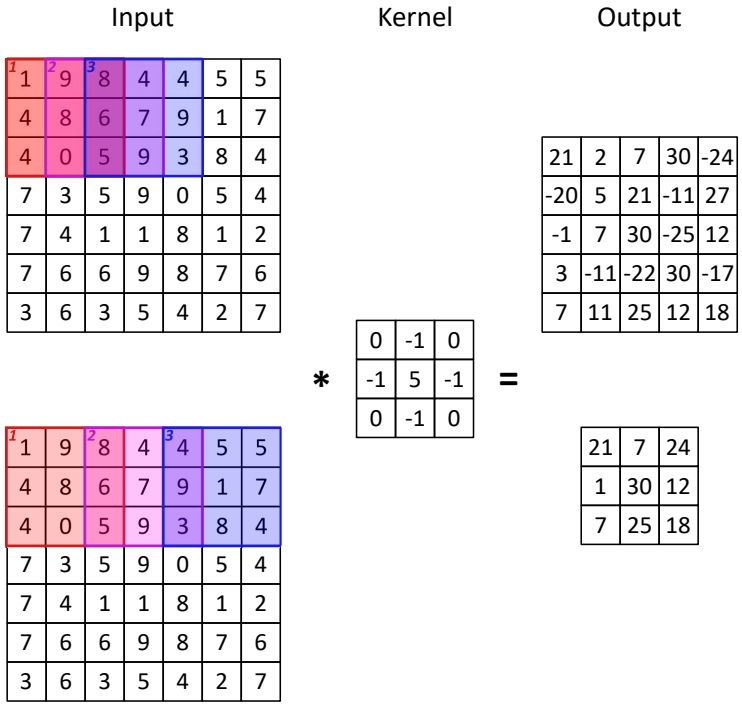

Figure 3.2.4: Figure demonstrating the effect stride length (top row = 1, bottom row = 2), kernel values, and kernel size has on the convolutions of a 2D object in machine learning.

3.2.5 Loss, Back-Propagation and Learning

Returning to Equation 3.2, recall that **b** and **W** represent the biases and weights of the nodes in each layer that control the transfer of information from input **x** to classification **y**. The learning phase of a neural network is therefore simply discovering the values of **b** and **W** that best map **x** to **y**. However, given that even the basic VGG16[76] network has 1.38×10^8 parameters of $0 \leq \mathbf{b} \leq 1$ and $0 \leq \mathbf{W} \leq 1$ to tweak, finding an optimal combination cannot be done through brute-force alone.

Instead, we can optimise this process by utilising the non-discrete nature of predictions. Recall from Subsection 3.2.2 that **y** is not a discrete value of 1 = 'yes' or 0 = 'no' for a k category classifier, but instead a continuous "confidence" of $0 \leq \mathbf{y}_k \leq 1$ (which we typically round to 0 or 1 for a final classification). It stands to reason that a network that is both confident and incorrect needs its parameters adjusting more than a network that is both confident and correct.

For this reason we first introduce a loss function, $J(\mathbf{y}, \hat{\mathbf{y}})$, which compares the network prediction **y** to the manually created 'correct' $\hat{y}$. The exact form of J depends on the task being solved; regression tasks will often[10] use the mean square error (MSE) of

$$J_{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\sum_{i=1}^{m} (\mathbf{y} - \hat{\mathbf{y}})^2}{m}. \quad \{3.8\}$$

For the classification tasks here, J is typically binary cross-entropy (a simplification of categorical cross-entropy, used for multi-label), as given by the equations[10]

$$J_{categorical}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{m} \sum_{i=1}^{m} \Big(\sum_{k} \mathbf{y}_k \log(\hat{\mathbf{y}}_k) \Big). \quad \{3.9\}$$

$$J_{binary}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{m} \sum_{i=1}^{m} \Big(\mathbf{y} \log(\hat{\mathbf{y}}) + (1 - \mathbf{y}) \log(1 - \hat{\mathbf{y}}) \Big), \quad \{3.10\}$$

After '*forward*-propagating' from **x** to **y** and calculating J, we can then use '*back*-propagation'[10], an algorithm derived in the 1970's[77,78] to periodically update **W** and **b** by a small amount $d\mathbf{W}$ and $d\mathbf{b}$, based on how much J has changed with the last update and if it has gotten greater or smaller. We then iterate over several repeats of the dataset ("epochs") until we are satisfied that J is sufficiently small or has reached a minimum. This routine forms the basis of the supervised learning process.

Using back-propagation allows us to determine new values of **W** and **b** with the equations

$$\mathbf{W}^{(l)} = \mathbf{W}^{(l)} - \alpha \, \mathrm{d}\mathbf{W}^{(l)} \quad \{3.11\}$$

$$\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha \, \mathrm{d}\mathbf{b}^{(l)}, \quad \{3.12\}$$

where α is the learning rate hyperparameter. This is not adjusted by back-propagation, but slowly reduced over time by an optimiser such as the Adam function[79] to allow the system to gradually converge on a set of parameters, helping to prevent the minima being local, rather than global. Regardless, using the chain rule, we can differentiate J to find $\mathrm{d}\mathbf{W}$ and $\mathrm{d}\mathbf{b}$ with the equations

$$\mathrm{d}\mathbf{A}^{(l-1)} = \frac{\partial J}{\partial \mathbf{A}^{(l-1)}} = \mathbf{W}^{(l)\mathsf{T}} \mathrm{d}\mathbf{Z}^{(l)} \quad \{3.13\}$$

$$\mathrm{d}\mathbf{Z}^{(l)} = \mathrm{d}\mathbf{A}^{(l)} \, g'(\mathbf{Z}^{(l)}), \quad \{3.14\}$$

and therefore

$$\mathrm{d}\mathbf{W}^{(l)} = \frac{\partial J}{\partial \mathbf{W}^{(l)}} = \frac{1}{m} \mathrm{d}\mathbf{Z}^{(l)} \mathbf{A}^{(l-1)\mathsf{T}} \quad \{3.15\}$$

$$\mathrm{d}\mathbf{b}^{(l)} = \frac{\partial J}{\partial \mathbf{b}^{(l)}} = \frac{1}{m} \sum_{i=1}^{m} \mathrm{d}\mathbf{Z}^{(l)(i)}. \quad \{3.16\}$$

It is for this reason that the transfer functions of Subsection 3.2.3 must be differentiable. Regardless, the forward-back-propagate-update algorithm is visualised in Figure 3.2.5.

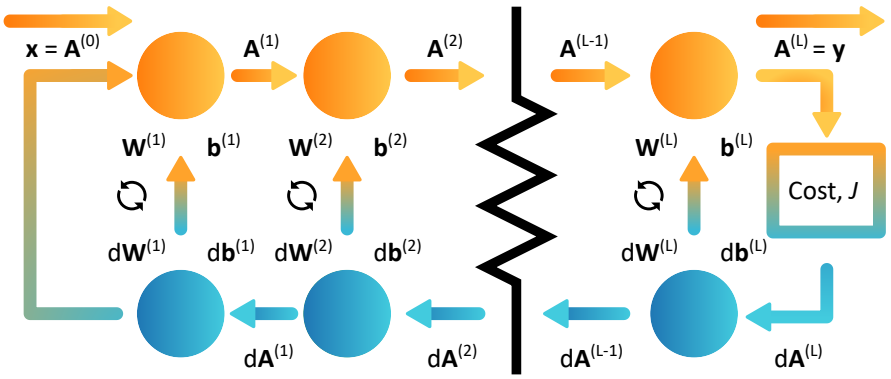

Figure 3.2.5: Demonstration of the forward-back-propagation algorithm used to train a neural network. To forward propagate, we start with input $\mathbf{x}$, and transfer it through the network via nodes and neurons with weights and biases $\mathbf{W}$ and $\mathbf{b}$ to an output $\mathbf{y}$. We then compare $\mathbf{y}$ to $\hat{\mathbf{y}}$ to compute a loss, J. The change in J over the last iteration is then back-propagated through the nodes and neurons, updating $\mathbf{W}$ and $\mathbf{b}$ by a small amount $d\mathbf{W}$ and $d\mathbf{b}$.

Further optimisations can be made to the learning process with features such as regularisation, in which a penalty is added to J based on $\mathbf{W}$, or dropout, in which a percentage of $\mathbf{W}$ and $\mathbf{b}$ are not updated, further helping the system converge and avoid local minima in its complex multi-dimensional energy landscape.

Once this computationally expensive process is complete, the weights and biases can be saved, and the network given new data to rapidly classify by performing a cheap matrix multiplication; this is why training is slow, but inference is fast. This could be further examples of $\mathbf{x}$ and $\hat{\mathbf{y}}$ in a validation dataset to quantify the true performance of the classifier on unseen data, or deployed into a real-world system where $\mathbf{x}$ is known but $\hat{\mathbf{y}}$ does not exist.

3.3 Semi-Supervised Autoencoders

3.3.1 Anomaly Detection

While supervised classifiers are extremely usable when we are able to explicitly detect features we *can* explicitly label, we are not able to detect what we *cannot* explicitly label. This is known as 'anomaly detection' (sometimes called one-class classification)[80]. This semi-supervised technique only requires the use of "positive" features (i.e. actively desired). By determining underlying patterns within the positive samples, otherwise unseen samples not conforming to these patterns are then classified as negative. This technique is often used in areas such as fraud detection[80,81] observing abnormal medical scans[80,82], and also in content-based-image-retrieval in internet searching[83]. While not all anomaly detection algorithms are machine-learning based[80], such as the k-means clustering algorithm[84], here we focus on the machine learning 'autoencoder'.

To create an autoencoder, we begin with a neural network not dissimilar to Subsection 3.2.1. As pictured in Figure 3.3.1, the first half encodes the input, $\mathbf{x}$, into a compressed representation of itself. For the second half, the encoder is then inversed, decoding to recreate the original image, $\mathbf{x}'$. Because $\hat{\mathbf{y}} = \mathbf{x}$, training data does not need to be manually made. The network therefore learns to optimise Equation 3.1, but with $\mathbf{y} \rightarrow \mathbf{x}'$ (i.e. output its input). For an image, the loss function, J, is a modified form of Equation 3.8,

$$J_{mse}(\mathbf{x}', \mathbf{x}) = \frac{(\mathbf{x}' - \mathbf{x})^2}{m}. \quad \{3.17\}$$

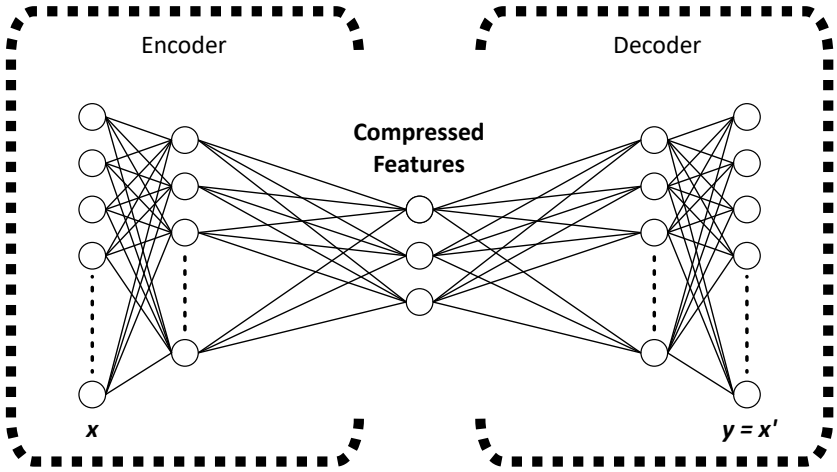

Figure 3.3.1: Generic structure of an autoencoder used to perform anomaly detection.

Because the network will only be capable of recreating data with commonly appearing features, non-anomalous data will have a low J, while outliers will have a high J. By thresholding, it is possible to detect anomalous data without knowing exactly what is anomalous.

3.3.2 Denoising

One alternative use of autoencoders is denoising. Instead of inputting data such that $\mathbf{x} = \hat{\mathbf{y}}$, we can add significant amounts of noise to $\mathbf{x}$, leaving $\hat{\mathbf{y}}$ unchanged. Our autoencoder then learns to remove noise from data, during compression/reconstruction. As a prelude to Chapter 6, Figure 3.3.2 shows how a denoising autoencoder can be trained on simulated AFM images, then used to remove noise and artefacts from automatically preprocessed experimental data.

3.4 Assessing & Improving Performance

3.4.1 Testing & Validation Datasets

Quantifying the performance of a CNN can be a surprisingly misleading endeavour. While we may naively look at loss with the data used for training, our network can deceive us incredibly easily. Indeed, how do we know if it has actually learnt to solve our task as we understand it, or if it has learnt the right answers 'by rote' from random noise in the training dataset? To return to the student analogy from the beginning of this chapter, has our student simply learnt to quote from the answers in their textbook, fooling their teacher but leaving them completely unable to apply the lesson to new situations? Our solution in machine learning is exactly as we would do for our student; we constantly give them additional 'testing' questions in the form of an exam where they do not yet know the answers. To be doubly sure, we can additionally create a 'validation' dataset; the final exam taken after all learning has finished. We therefore do not monitor learning with the training loss, but with the testing loss instead.

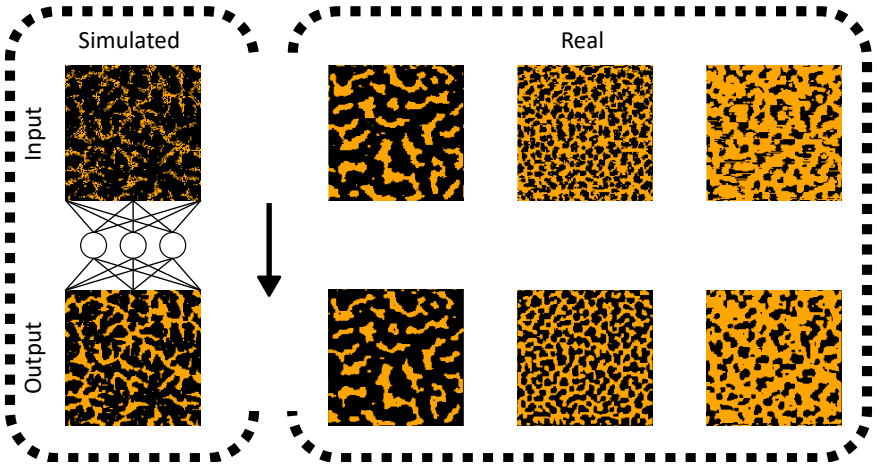

Figure 3.3.2: Figure of a denoising autoencoder used to remove noise and artefacts from AFM dewetting images. Starting with simulated scans and adding noise, the autoencoder learns to remove noise to reconstruct the original simulated scan. This network can then be applied to real, preprocessed and binarised scans, where it was able to remove noise and preprocessing artefacts. (See Chapter 6 and Gordon et al., Nano Letters 20(10), 2020.)

3.4.2 Under & Overfitting

The key advantage of having testing and validation datasets is that we can discover the presence of so-called 'overfitting', in which a CNN uses random trends to correctly classify training data, but at the expense of misclassifying unseen data[10]. By continuously examining our classifier during training, we can detect overfitting by seeing that training loss improves while testing loss degrades, often rapidly, as seen in Figure 3.4.1(b). During training, we therefore look at the loss of the testing dataset, rather than the training dataset, when determining when to stop training.

Once we have discovered overfitting, we must account for it as otherwise our network will not function at its actual task when presented with new information. Besides shuffling to remove meaningless order between disparate data points and adjusting hyperparameters such as learning rate, the simplest way to reduce overfitting is to reduce the complexity of the network, either by reducing the number and/or size of the hidden layers. This reduces the number of trainable parameters, reducing the capacity of the network to learn worthless noise.

However, we can sometimes go too far and cause underfitting. As seen in Figure 3.4.1(b),this is when the network performs poorly on the training set, so does not generalise to perform well on the testing set. Unfortunately, there is currently insufficient understanding[10] to know what shape and structure of network is most suitable for a given task. Indeed, there are in fact an abundance of attempts to use machine learning methods to optimise the architecture of machine learning methods for a variety of tasks[85]. Instead, individuals typically use one of a number of "known-good" architectures, such as Oxford VGG[76] or Google's AlexNet[86].

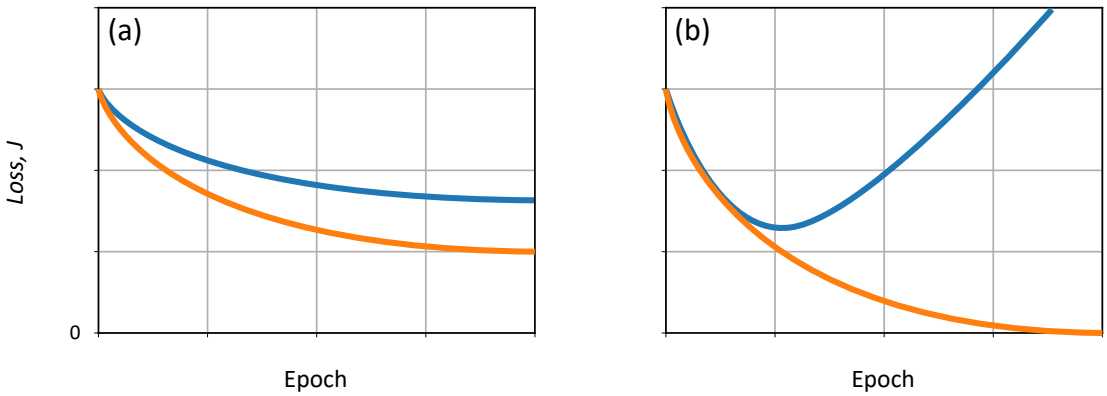

Figure 3.4.1: In under fitting, **(a)**, testing loss (blue) lags behind training loss (orange), which converges to an overly high value, indicating that the network has not fully learnt to solve its task. In over-fitting, **(b)**, the network learns from random noise in the training set, causing training loss to tend to a very low value, but making the system fail to generalise to new data, causing testing loss to rise.

3.4.3 Data Augmentation & Feature Engineering

Alternatively, we could try to improve learning performance by increasing the relative signal to noise of our dataset. This is typically an area worth the time investment. While one of the most basic (yet effective) methods to reduce overfitting is to simply use more data in the hope of drowning out noise, this is rarely viable. While Facebook has access to 3.5 *billion* Instagram[87] images on which to train their networks, SPM researchers do not and will never possess labelled data at this scale, regardless of efforts to encourage open data-sharing[88]. Instead, we can "invent" new data from existing data, which has been shown to be an exceptionally effective way of improving performance[10]. From Figure 3.2.2, it can be seen that pixels in a flipped image will take a different route through the network from $\mathbf{x}$ to $\mathbf{y}$. It therefore provides the same learning potential as a completely different image. For example, a horizontally flipped photo of a cat still clearly contains a cat regardless of whether the cat's head is on the right side or the left side of the image (we would not consider vertical flips on cat photos as cats do not naturally exist upside down with the sky on the floor). Other common augmentations include rotation, pan, crop, zoom and Gaussian noise, as demonstrated in Figure 3.4.2. We can also augment depending on context, such as adding "artificial" tip artefacts to an AFM image[89].

Finally, for strong neural network performance it is imperative to preprocess data to match the constraints of a network. Of particular importance is the scaling of data. By the definition of nanoscience, SPM data is typically of the order of 10^{-9}, barely above floating-point accuracy. Thinking of the transfer functions shown in Figure 3.2.3, calibrated data of STM tip height (i.e. constant current) scans are of the order of 10^{-9}, so would almost always produce the exact same activation value, making it extremely difficult for our network to learn to distinguish between different values of $\mathbf{x}$. Instead, we can scale our data to the range $-1 \leq \mathbf{x} \leq 1$ or $1 \leq \mathbf{x} \leq 1$. While this scaling is physically somewhat arbitrary, we should remember that visual features are retained. Another consideration is if we should scale each item of data relative to itself, or relative to each other. This can cause issues in SPM especially, where we need to balance between maximising fine detail in worthwhile data, and distinguishing between differing data ranges of distinct categories.

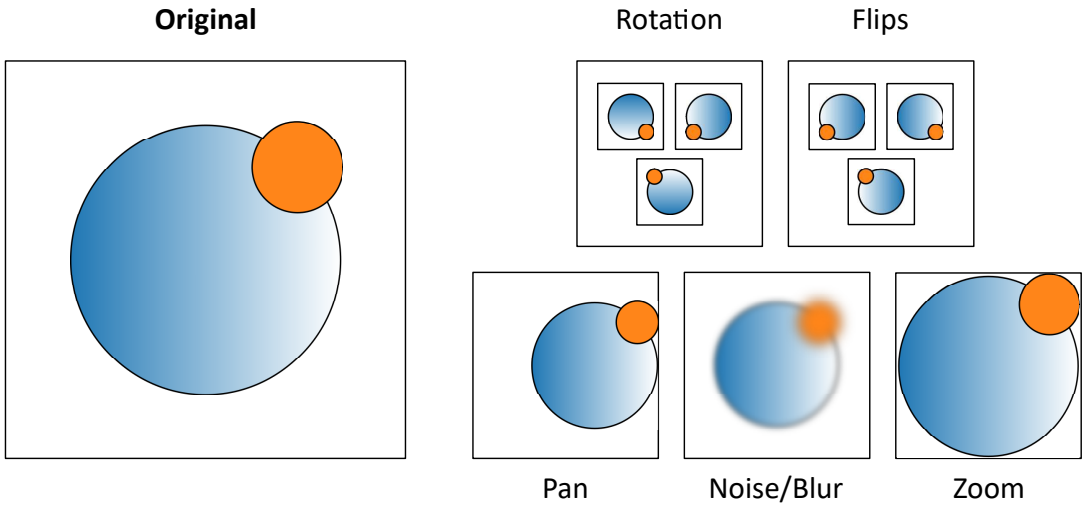

Figure 3.4.2: Some of the most common ways to artificially increase the number of data samples in a neural network, potentially improving performance significantly.

3.4.4 Moving Beyond Pure Accuracy

Further, we should also consider alternative metrics for considering performance. Again, we may wish to continue our student-exam analogy, by naively giving them a % score of 'correct' answers. However, we will fall into yet another trap. Imagine we wish to detect fraud in a financial dataset where 99% of transactions are not fraudulent. If we guessed everything as not fraud, or flipped a 99:1 biased coin toss, our network would be near perfectly accurate while in reality being utterly useless at its actual job. This is of particular importance to us - the raison d'etre for this thesis is of course that it is rare that any piece of SPM equipment works as desired. Some methods to negate this specific issue include weighting loss by the reciprocal of the percentage of each class present, and using a weighted accuracy metric (where the reciprocal of the number of classes is defined as guessing)[90]. It is for this reason that other authors warn against using solely accuracy to judge the performance of weighted datasets[22,91].

However, the best practice for determining performance is to consider alternative metrics. Recall that $0 \leq \mathbf{y} \leq 1$; to make a decision we must first round y to 0 (i.e. no) or 1 (i.e. yes) for each of the k states. There is no reason why we must do this rounding at $\mathbf{y} = 0.5$. As such, we do not just have correct (True) and incorrect (False), but True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). In the context of STM, a True Positive would be our classifier saying our tip is good when it is good, a True Negative saying it is bad when it is bad, a False Positive saying it is good when it is bad, and a False Negative saying it is bad when it is good. As in the medical diagnosis field[80,82], there are different consequences to positive and negatives, so we do not always wish to round from 0.5, and wish to explore the effect of this rounding on True and False Positives and Negatives. For this reason we can make our system more practical by tuning the confidence threshold required to make a positive classification. For example, we could reduce false positives at the cost of increased false negatives and therefore decreased accuracy. One way to quantify this is to first calculate the sensitivity (otherwise known as True Positive Rate (TPR)), precision (otherwise known as Positive Predictive Value (PPV)), recall (otherwise known as False

Positive Rate (FPR)), with the equations

$$TPR = \frac{TP}{TP + FN}, \tag{3.18}$$

$$PPV = \frac{TP}{TP + FP}, \tag{3.19}$$

$$FPR = \frac{TN}{TN + FP}. \tag{3.20}$$

By comparing sensitivity and recall, and precision and recall, at different rounding thresholds, we form the Receiver-Operator-Characteristic (ROC) and Precision-Recall (PR) graphs, as depicted in Figure 3.4.3. Not only do we have a visual means with which to tune our rounding threshold to suit our given task, but we can quantify all-around performance by the area under ROC (AUROC), in which a perfect classifier has an AUROC of 1 and guessing 0.5. Crucially, these metrics are also not affected by class imbalance[91] and are therefore superior to accuracy.

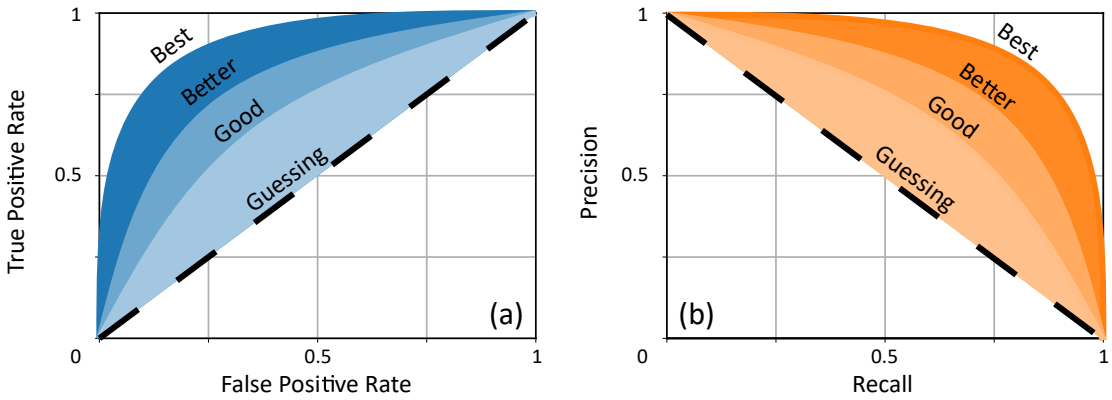

Figure 3.4.3: The **(a)** Receiver Operator Characteristic (ROC) and **(b)** Precision Recall (PR) curves. Rather than simple true/false accuracy, these allow us to compare the rates of true and false positives and negatives in a neural network as the confidence threshold to make a positive prediction is varied. This not only allows us to tune a network for a given task where there are different implications of positive and negative predictions, but by integrating allows us to account for class imbalance and view the overall performance of the classifier as a whole. A perfect network will have an area-under-curve of 1, while a guessing network will have area = 0.5.

3.4.5 Ensemble Networks

One final method capable of improving model performance is to use multiple trained models for a task. Here, we can train multiple distinct models with different architectures to perform the same task (e.g. classification), using majority voting to allow a democratic vote based on shared and separate experiences[10]. We can then extend this even further to have multiple models specialising in slightly different tasks, then bring them together to perform a more complex task that is difficult to train with a single network. For example, in self-driving cars, one model will detect road signs, while another may control emergency braking.

3.5 Semi-Supervised & Unsupervised Reinforcement Learning

While supervised methods can be used to make static assessments/predictions of a system, they are not capable of optimising routines[92] (also known as 'policies'). This is particularly true when the outcome of actions is time-dependent. This makes them poorly suited to tasks ranging from making a bowl of cereal to controlling self-driving cars[93], playing video games[17,94], or choosing when to make financial investments[95]. In our case, while we and others[22,96–98] have shown (and will show in Chapter 5) that supervised classifiers are highly capable of *assessing* an STM tip, they are poor at actually *improving* it based on that situation. Reinforcement Learning (RL) provides a means to solve these situations.

3.5.1 Markov Decision Making

To accomplish a task in RL, we begin with an *environment*[92]. Inside it, we place an *agent*, responsible for the decision-making process. At a discrete timestep, $t = \{1, 2, 3, \ldots, \infty\}$, the agent observes the environment's *state*, S_t. Accordingly, it then performs an *action*, A_t. As the agent begins a new step at time $t + 1$, it then receives a numerical *reward*, R_{t+1}, based on the new of the system, S_{t+1}. R can be positive (i.e. a reward), or negative (i.e. a punishment). The agent only knows about the value of R, not the function calculating it. The agent can then make, take, and receive further observations, actions, and rewards[92]. For a system to be Markovian, the probability of a given event depends solely on the state of the system during the previous event. This cycle is pictured in Figure 3.5.1.

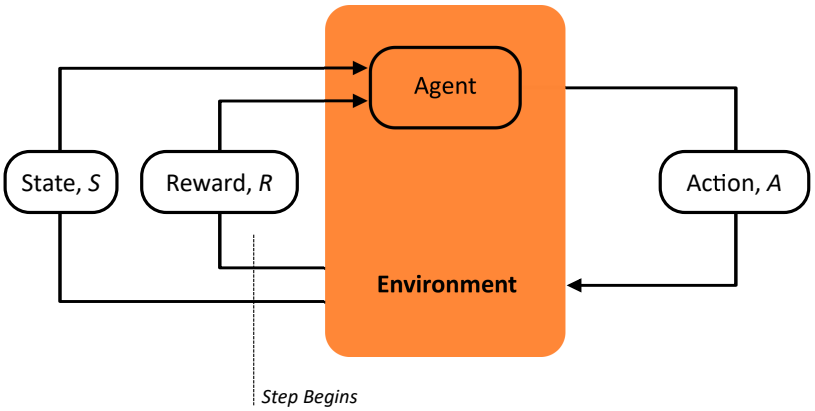

Figure 3.5.1: A reinforcement learning agent interacts with its Markovian environment.

The goal of RL is to find a policy to maximise[92] return, G, over a given time, T, where

$$G_T = \sum_{t=0}^{T} \gamma^t R_t. \qquad \{3.21\}$$

It is for this reason that an RL agent to work towards long term goals, accepting that it may have to perform an action with negative R at a given t, but expecting a delayed reward of greater R. Further, a discount rate, γ, where $0 \leq \gamma \leq 1$, can be applied. By increasing γ by the power of the number of timesteps taken, the patience of the network to receive a return on its investment can be constrained.

Regardless, devising a policy to maximise G would seemingly require knowledge of the system in the future. As this is obviously impossible, we instead exploit our assumption that the system is Markovian. Because of this assumption, the probability of the system being in state S_t (and therefore receiving reward R_t) depends on S_{t-1} and A_{t-1} preceding that step. By exposing the agent to sufficient S, A, and R, we can find a policy that optimises G, thus completing the task. Commonly, this is achieved with the Q learning algorithm [16]. We can therefore take an untrained agent performing random A, and allow it to learn how to optimally complete its task by exploring the environment it lives in.

3.5.2 Reward Function

One of the most crucial aspects in reinforcement learning is how the agent is rewarded or punished. It is imperative to remember that the 'goal' of an agent during training is to receive maximum reward over time; it does not know nor 'care' if it performs the task in the way we intuitively expect, or not. While this in theory opens up the possibility of obtaining greater-than-human performance at a task, this can be problematic. For example, an agent may devise a mathematically greater-than-human policy to quickly make a pot of tea, except the strategy may be to set the kitchen on fire to faster boil a greater volume of water.

In general, there are two means of providing a reward to an agent: 'sparse' and 'dense'. For a sparse reward, the agent in our above analogy could score 0 after every step until the very end of the episode, when it scores $+1$ if it makes tea and -1 if it does not. While there is a reduced chance of accidentally encouraging a bad strategy, the agent has no instant response to its decisions. This is fine if we have a game like noughts-and-crosses where it is reasonably likely to 'chance' into a positive outcome in a small number of timesteps (i.e. a small state space), this is not viable in more complex scenarios. Instead, in these cases and games like go [99] and chess we may employ a 'dense' reward. Here, we manually create and 'shape' a function to apply a reward/penalty after every timestep in an attempt to highlight good moves and poor moves. However, reward shaping can be a very vague, subjective task, made only more frustrating by the sensitivity of an agent to the specific function, which can rapidly become overly-complicated.

Applying this to STM tips, we must therefore appreciate that while the reward function is inherently arbitrary, its 'shape' and means of delivery is absolutely vital to ensuring effective learning. Generally, this means making the reward linearly and simply correlated with the state observations. We can also use reward shaping to encourage or discourage certain behaviours; applying a penalty after every action could encourage the agent to progress further into scans, while applying a reward to observations corresponding to flat areas could encourage the agent to prioritise obtaining them.

3.5.3 Reinforcement Learning with Expert Examples

One of the major reasons for the success of RL in tasks such as playing video games, besides the simple, logical reward function, is that multiple mutations of the same agent can be created at once, then directly compared with one another by implementing with multiple copies of the exact same starting environment. This allows for rapid, mass exploration of S and A. This is not possible with SPM, as the environment cannot be simulated, the physics

relating tip shape to tip state is not fully known, and the environment will subtly change over time so cannot be perfectly 'reset'. Further, while a round of 'pong' could be won through a small number of fully random inputs, the huge state space of an STM tip makes random success near-impossible.

However, these pitfalls are not unique to STM. For example, consider self-driving vehicles; how does one learn to "drive well" and do so with imperfect simulation?[93,100]. One could also consider the issue of AI-devised healthcare treatment programs[101]; how can an agent be ethically allowed to learn when it has no initial knowledge and can only interact with real patients who may have different outcomes despite identical health data?

One alternative is that of Inverse Reinforcement Learning (IRL)[94,100,102], which is a general term for more specific methods such as imitation learning, behaviour cloning, experience replay and many other "expert-example" algorithms and extensions (although, confusingly, these terms are often used interchangeably). While RL has an initially untrained agent learn a policy to maximise its reward function by exploring its environment, IRL does the opposite. Instead, an agent uses expert examples from a human to attempt to inversely learn the state Markovian, by observing the humans' policy to maximise G_T by performing A at given S. The trained agent may also be further improved by allowing it to explore its environment through normal RL, but with the advantage of having a policy to begin with. (Indeed, this 'heatup' phase often forms parts of conventional RL, where initial off-policy exploration occurs through purely random action selection) Overall, this method is analogous to how an intern may "shadow" a trained scientist before they are given autonomy and the chance to devise their own ways of working.

There are a number of algorithms that can be used for inverse learning, such as behaviour cloning[103], or general supervised classifiers, which both explicitly require expert examples. However, and while supervised classification is sometimes used as a pre-training step, more success is seen with algorithms such as DQN[104] or DDPG[105]. While a full theoretical discussion of these algorithms is outside the scope of this thesis, these algorithms operate by building up a memory (known as a 'replay buffer') of experiences, and searching for a state in memory similar to what is happening in the present moment. In RL, this replay buffer initially consists of the outcome of random actions (hence why the task must be solvable with random actions), but in IRL it consists of expert actions.

3.5.4 Partially Observable Markovians

However, it is at this point we encounter a major issue. Recall the Markovian assumption from Subsection 3.5.1, which states that the probability of a given event depends solely on the state of the system during the previous event.

A good analogy of this issue is to consider a Markovian person in a maze. At some time t they come to a clearing. The only information available to them is that they see a path to the left and a path to the right. They turn left. At some time $t + \tau$ they come to a clearing. The only information available to them is that they see a path to the left and a path to the right. They turn left. At some time $t + 2\tau$ they come to a clearing. The

only information available to them is that they see a path to the left and a path to the right. They turn left. At some time $t + 3\tau$ they come to a clearing. The only information available to them is that they see a path to the left and a path to the right. They turn left.

While it is obvious to the astute reader what is going on, our Markovian person cannot infer that they have gone in a circle and will never escape until they adjust their actions. Specifically, they would need a top-down view of the maze to know their progress over time as a function of the action they choose. However, this information is not available at eye level, and the 'correct' decision does not solely depend on the state of the system during the previous event; the agent needs to know about its 'trajectory'. The state is only 'partially observable', and is therefore not fully Markovian.

In reality, this surprisingly tricky issue[106] applies to pretty much any worthwhile problem in RL. Even the most suitable, classic RL problems of beating Atari games such as Breakout[16,17] has some amount of time dependence (i.e. the direction and speed of the ball) which not encoded into a single frame of pixels. In these cases it has been shown that by 'stacking' three frames' worth of pixels as the observations (i.e. enough to classically calculate position, velocity,W and acceleration), learning performance improves substantially[16].

Compounding the issue, any SPM experiment has some stochasticity to it. In an Atari game, if we press the right arrow, our agent inherently understands that this causes the character to move to the right. With an STM tip, if we perform a tip pulse (as described in Subsection 2.1.3, we may sharpen the tip, blunt the tip, or do nothing at all. Even if our agent chooses to passively watch the progress of the scan, our tip may still spontaneously change for better or for worse. The agent mistakenly believes that doing nothing *directly causes* the tip to change. Further still, the probability of each outcome is at least semi-random itself; a problem for an algorithm based on probability distributions. This is an extremely important area of research in which no single ideal solution has been identified, and may not be for at least several more years.

4 Automated Identification of STM Tip State

4.1 Introduction (The Trouble with Tips)

Scanning probe microscopy (SPM) is a powerful set of techniques that have allowed researchers to make observations at the atomic level for decades[25,107,108]. Many modern SPM experiments combine STM and AFM using qPlus sensors, and can also involve functionalisation of the tip, wherein an additional molecule is "picked up" from the surface[43]. Notably, CO was shown in 2009 by Gross et al. to dramatically increase AFM resolution for the pentacene molecule[7], and is now being used to image many other surfaces[44,45].

However, despite these advances, success in these methods is highly reliant on the production of atomically sharp STM tips. Although these are readily created *ex situ*[38,39], imperfections in the tip apex including the presence of "double" or multiple tips mean that image artefacts often appear spontaneously during experimental sessions. Examples can be seen in Figure 2.1.8. Maintaining tip shape is therefore a vital part of routine SPM operation. To maintain resolution, human operators (notably often PhDs) spend significant time correcting apex flaws *in situ* through a repeated combination of controlled voltage pulsing and/or tip crashing. This is a high effort, time consuming and manual process with a low skill requirement, making it ideal to automate.

Despite the impetus to automatically detect and correct for apex flaws in STM, one of the major reasons for the lack of SPM automation is the large number of visually distinct features present in any given STM image. For the H:Si(100) surface alone, these include[21]: 'atoms' (for the sharpest tips), 'dimers', 'asymmetries', and 'rows'. Example STM images for these different states are shown in Fig 4.1.1.

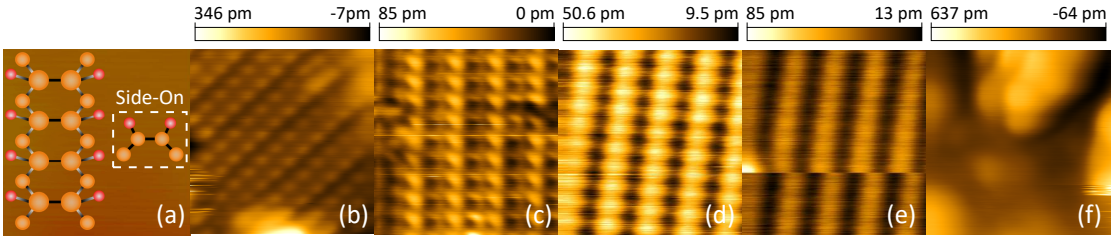

Figure 4.1.1: Selection of images demonstrating the classified tip states for STM imaging of H:Si(100) (See Section 2.1.3). **(a)** ball-and-stick model (not to scale), **(b)** atoms, **(c)** asymmetries, **(d)** dimers, **(e)** rows (and tip change), and **(f)** generic defect. Typical scans in this thesis were approx. V=-2V, I=-1nA.

Not only do microscopists have to assess the impact of tip state on data collected, but this often leads to uncomfortable questions regarding the objectivity of microscopists in tip formation and control; which of the images shown in Figure 4.1.1 is the 'right' image? These images clearly illustrate the dramatic effect that the convolution of the tip and surface structure gives rise to a wide variety of different image types; it is the tip, not the surface, which is playing the key role in determining the 'sample' structure seen in the images. Forcing experiment to match theory is always a dangerous endeavour, and while H:Si(100) is understood, how can we know what is a sharp tip apex image for an unknown surface? Any scientist with even a modicum of experience with scanning probe techniques knows that

this is a risky game to play, and the literature contains many examples of misinterpreted SPM images in which the influence of the probe was not taken into consideration. Moreover, is the highest spatial resolution image necessarily the most appropriate when it comes to spectroscopic analysis or single atom/molecule manipulation, and to what extent are we assuming that our data is physically worthwhile and not due to coincidental interactions with the tip[33]?

Further, there are occasions when we do not even necessarily desire the highest resolving tip apex, which may not be best suited to spectroscopy and/or atomic/molecular manipulation experiments. (Indeed, some probe microscopists have deliberately blunted the tip apex so as to trade off spatial resolution for higher spectroscopic energy resolution[28]). On this point, Hofer[32] makes some interesting and important observations regarding the role of the uncertainty principle in probe microscopy.) A previous analysis of over two thousand STM-induced hydrogen desorption events by the Nottingham Nanoscience group[33] for the H:Si(100)-(2x1) surface provided strong evidence that the probability of hydrogen extraction was highest for a tip apex that did not yield optimal spatial resolution. Similarly, previous experiments with tip-induced pushing and pulling of C60 molecules[34–37] also often involved drawing a compromise between highest imaging capability and the efficacy of single molecule manipulation.

Regardless, the concept of automated tip restoration is not in any way new or novel. Discussions of the desire to negate manual tip improvement can be seen as far back as the early-mid 1990s[109]. However, since then there have been relatively few attempts since then to achieve the "holy grail" of automatically obtaining atomic resolution[8,20–22,110], let alone to extend this to an 'optimise tip' button capable of intelligently coercing the tip into a specific state. These attempts suffer from similar pitfalls, such as high computational cost, real-world impracticality, failing when multiple tip flaws are present, and failing when the tip spontaneously changes the visible resolution mid-image. Convolutional neural networks (CNNs) are a highly promising alternative, routinely achieving high accuracy in complex vision tasks such as medical, satellite and digit recognition[111–113]. In 2018, Rashidi and Wolkow[22] were the first to consider CNNs for tip-assessment, and were able to detect double-tip defects on the H:Si(100) surface with fixed scan parameters. In a 2019 follow up[23], the Alberta based group has extended their approach to enable detection, classification, and avoidance of various defect structures on the H:Si(100) surface, which, they argue, is an important step towards autonomous atomic-scale manufacturing.

As such, we can break down the concept of automatically improving an STM tip into two distinct stages: assessing the state of the tip, and then acting on that assessment. While we save the latter for Section 4.5, in this chapter we discuss our attempts to significantly extend the work of Rashidi and Wolkow to detect multiple tip states of multiple surfaces under more generic scanning conditions and with greater performance. We then discuss how this was extended to a real-time implementation, potentially speeding up the assessment process by two orders of magnitude.

4.2 Surfaces Considered

4.2.1 Si(100)

The second most abundant element on earth (behind oxygen), silicon makes up 28% of the mass of the Earth's crust[114]. Silicates are used to form glass, cement, and building materials, while pure silicon is not only incredibly cheap, but semiconducts. Indeed, for this reason it underpins pretty much all of modern computing hardware, and has resulted in the geographic hotspot of the major technology conglomerates being named 'Silicon Valley'. Looking closer to home in SPM where it was first imaged using STM in 1985[115], silicon has long been studied[21,116,117], and is often used as a substrate in AFM studies such as that of Chapter 6. As such, it provides an ideal platform to develop machine learning protocols with.

While its face-centred cubic structure is theoretically simple in the (100) plane, the presence of dangling bonds makes the Si(100) crystal very reactive, meaning that outside of UHV, the bare untreated silicon surface is extremely reactive and will oxidise immediately in air.

Before continuing, we briefly review the common Wood's notation. Wood's notation provides a simple way to quantify the periodicity of a lattice based on its substrate, $(\mathbf{a}_1, \mathbf{a}_2)$, and adsorbate, $(\mathbf{b}_1, \mathbf{b}_2)$, basis vectors with the equation

$$\left(\frac{|\boldsymbol{b_1}|}{|\boldsymbol{a_1}|} \times \frac{|\boldsymbol{b_2}|}{|\boldsymbol{a_2}|} \right). \qquad \{4.1\}$$

As an example, we can see from Figure 4.2.1(a) that for this silicon-like fcc(100) surface, $|\boldsymbol{a_1}| = |\boldsymbol{a_2}|$, $|\boldsymbol{b_1}| = |\boldsymbol{b_2}|$, and $2|\boldsymbol{a_1}| = |\boldsymbol{b_1}|$. Further, this is a 'primitive' structure as both unit cells contain only one lattice point in total. Figure 4.2.1(a) is therefore a p(2x2) cell. Figure 4.2.1(b), meanwhile, can be described by overlapping two repeat units for the adsorbate cell, and so it is a 'centred' c(2x2) unit cell.

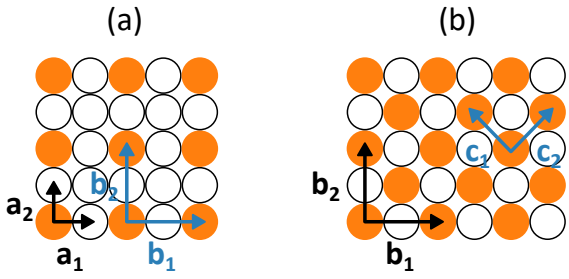

Figure 4.2.1: Explanation of the common Wood's notation, showing the lattice (orange) and adsorbate (white) unit cells for the **(a)** p(2x2), and **(b)** c(2x2) periodicity of the fcc(100) lattice of silicon.

Regardless, for the Si(100) reconstruction, Si atoms at the surface level of the crystal 'pair-up' to produce rows of dimers in a p(2x1) periodicity, as shown in Figure 4.2.2(a). In doing so, the surface reaches its energetically favourable ground state by reducing the number of dangling bonds per atom from two to one (or from four to two dangling bonds per dimer). As a result, there is a σ bond between both silicon atoms in the dimer pair, as well as a weaker π bond between the two remaining bonds. This π bond is much like those formed in planar carbon structures (e.g. benzene).

However, in reality this dimer pair is not symmetric, but instead has an asymmetrical 'buckling'[115,118] (as depicted in Figures 4.2.2(b) and (c)). Because it is more energetically favourable for electron states to be fully filled or fully empty, by transferring charge from one dangling bond to another (and therefore buckling the dimer pair), the reconstruction can be in a more energetically favourable state. As such, the reconstruction changes from being in a p(2x1) periodicity to either a p(2x2) or a c(4x2) structure, depending on the relative orientation of the buckled dimers.

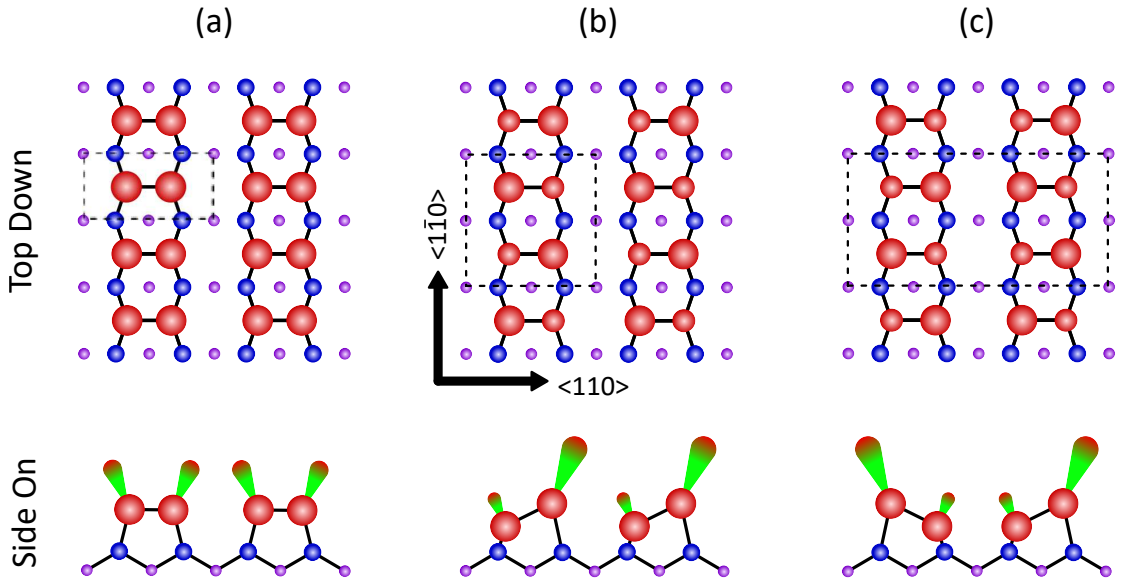

Figure 4.2.2: Ball-and-stick diagram of the **(a)** symmetric p(2x1), **(b)** asymmetric p(2x2) in-phase buckling, **(c)** asymmetric c(4x2) out-of-phase buckling, for the Si(100) reconstructions.

Further, (and particularly at non-cryogenic temperatures[119]), the two atoms in each dimer pair constantly alternate between being higher or lower than the other. This flickering happens proportionally to $\exp(-\Delta E/k_B T)$. At room temperature and an energy barrier between the two configurations of $\Delta E \approxeq 100$meV, we find a flipping frequency on the order of THz, much higher than the sampling frequency of STM scanners[120] which are at best on the order of kHz (assuming that the tip is static above a fixed point, and that the bandwidth is limited purely by the pre-amp). As such, STM actually images the 'average' position of the atoms.

Additionally, a number of surface defects for the Si(100) surface have long been debated and described in the literature[121–123]. Besides step edges, the most common of these are 'missing-dimer' defects. These can be seen in Figure 4.2.3. It is also extremely common for impurities (particularly water[124]) to bond to the clean Si(100) surface. Finally, nickel[125] contamination of the Si(100) can prevent the above reconstructions from perfectly forming. As will be discussed later, all of these defects can be problematic for the machine learning systems discussed in later chapters, as it is difficult to determine if a poor image is due to defects of the surface, or blunting of the tip.

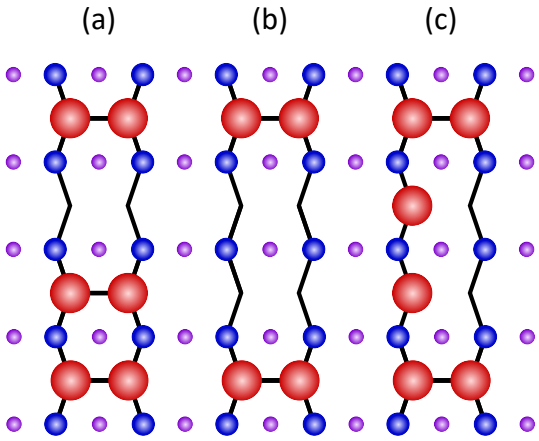

Figure 4.2.3: Common missing dimer vacancy (DV) defects of the Si(100) surface.

To provide the lowest possible base pressure and therefore reduce the number of impurities that could adsorb at the prepared surface, the VT was baked for periods of several days, or in some cases, weeks, at 140°C. This lengthy bakeout period was particularly helpful when hydrogen passivating samples (as discussed in Subsection 4.2.2).

To ensure low pressures during annealing of the sample, a sample plate with a piece of sacrificial silicon was first roasted on the manipulator arm at ∼1120°C for 15 minutes via resistive heating. After cleaning and sonicating a Si(100) sample ex-situ and placing it on the sample plate, it was then placed inside the system load lock (vented to nitrogen), pumped back down to UHV and heated overnight to ∼140°C. The sample was then degassed at ∼600°C for a number of hours using direct heating to remove a large amount of the impurities (e.g. water and latent solvent) from the surface. Periodic firing of the titanium sublimation pump (TSP) also helps with this. Once the pressure of the system recovered sufficiently, the oxide layer on the Si(100) surface was then removed by using direct heating to flash the sample to ∼1200°C, revealing the silicon beneath. By reducing the heating current, the sample was then cooled to ∼900°C, then extremely slowly to ∼600°C. These steps allow for remaining defects to diffuse across the surface, and gradually reconstruct with minimum surface roughening. At this point, the sample was then be left to cool completely, producing the Si(100)-(2x1) reconstruction. An STM image is shown in Figure 4.2.4.

4.2.2 H:Si(100)

One common modification to the Si(100) is to hydrogen passivate it, as depicted in Figure 4.2.2. This produces the H:Si(100)-(2x1) surface[126], which has been studied extensively in the context of atomic scale lithography[4,127–130], single atom electronics[131,132], and fundamental quantum mechanics (including quantum information[133,134]). Additionally, desorption of single H atoms is possible via injection of electrons from the STM tip, leading to vibrational heating, and ultimately dissociation, of the H-Si bond. Key for our purposes is that it is significantly less reactive than Si(100), making for more stable samples easier to use to train machine learning networks, and is therefore key to the themes of this thesis.

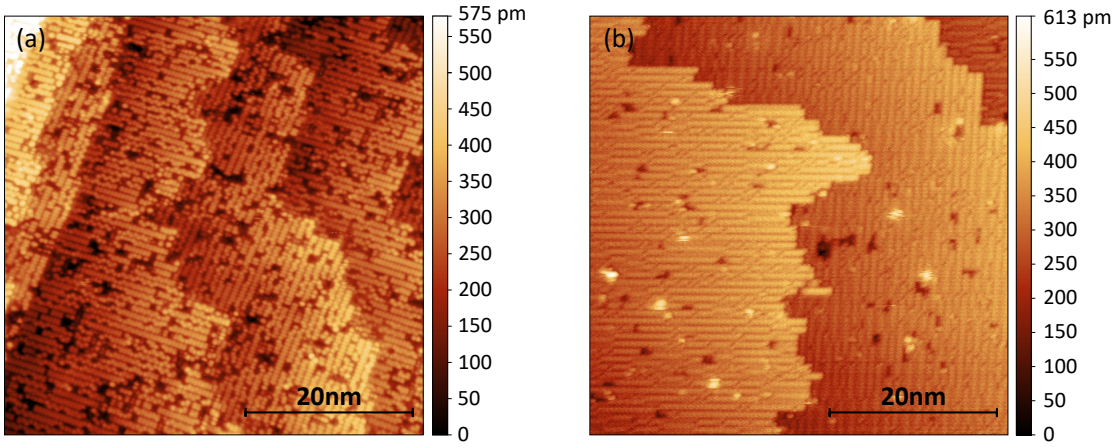

Figure 4.2.4: STM image of **(a)** the Si(100)-(2x1), and **(b)** the passivated H:Si(100) reconstructions, showing dimers, dimer defects, and multiple step edges. Scans were taken at 300K, with parameters from right to left of $I = 100$, 250pA, $V = 1.5$, 2.25V.

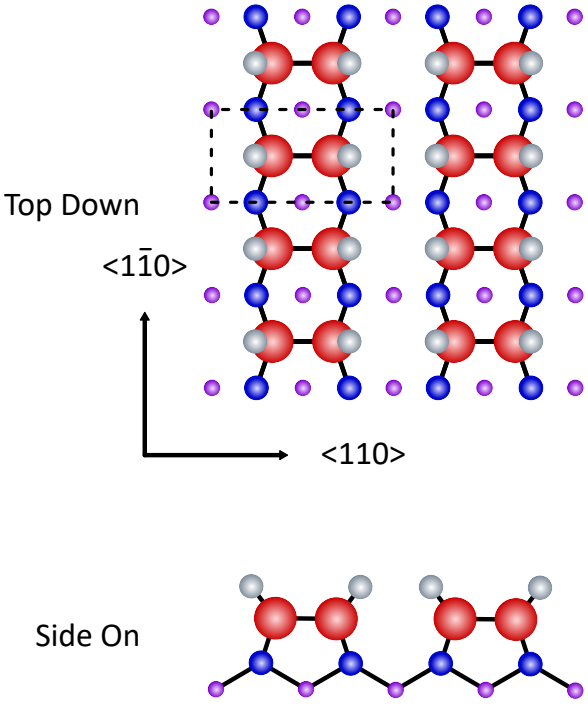

Figure 4.2.5: Ball-and-stick diagram of the H:Si(100)-(2x1) reconstruction.

Unsurprisingly, there are also a number of common surface defects seen in the H:Si(100) surface. Besides vacancy defects analogous to Si(100), we observe 'dangling-bond' defects when passivation is incomplete, leaving chemically reactive areas on an otherwise unreactive surface. While not useful for our purposes, these sites are highly interesting, with one notable study demonstrating the ability to 'switch' dangling bond features, allowing for the 'healing' of individual defect sites with atomic precision [135].

To create the H:Si(100) surface, we first prepare for hydrogen passivation by allowing the system base pressure to recover. By first degassing the manipulator arm, then leaving the pressure to recover overnight, then degassing the cracker for several hours, a clean, minimal pressure environment was created in which to perform the passivation.

During the last stage of degassing, the hydrogen line (connected to the gas cracker) was also baked overnight while opened to the turbo and left to cool, and the line flushed with 2 bar of pure hydrogen. Flushing the line of gaseous contaminants is an essential step to producing a clean H:Si(100) surface. During cooldown the following morning, a section of the line was immersed in liquid nitrogen for one hour, which has also been shown to reduce surface defects significantly. This section of the line is known as a 'cold-trap', using a series of coils with high surface area in contact the liquid nitrogen to trap non hydrogen gas.

The hydrogen line was then refilled, the sample placed in the manipulator arm, heated to $\sim$375°C, exposed to the cracker at $\sim$ 2000°C for $\sim$1 minute, and then allowed to cool completely. This allows the hydrogen to react with the dangling bonds, forming the H:Si(100)-(2x1) reconstruction shown in Figure 4.2.6.

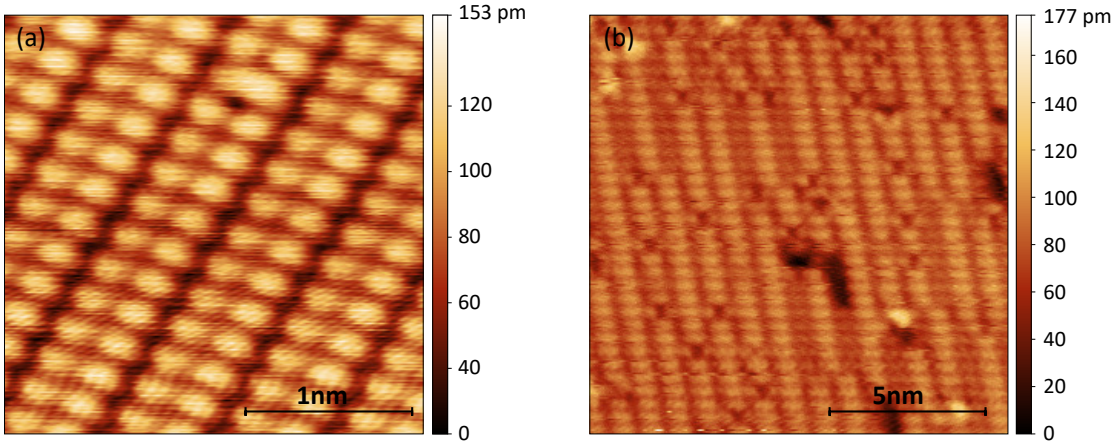

Figure 4.2.6: STM images of the H:Si(100)-(2x1) reconstruction, showing rows of dimer pairs of Si atoms taken at 300K. Scan parameters (right to left) are I = 10, 500pA, V = 1, -2.25V.

4.2.3 Cu(111) & Au(111)

While H:Si(100) was the primary surface of interest and the only one directly imaged in this thesis, our long-term goal is to produce a system that can detect tip states for a variety of crystallographic surfaces. We therefore employed a dataset of additional images of Au(111) and Cu(111) collected and classified by Freeney and Swart et al. in Utrecht (provided without scan parameters). Example images of each surface are shown in Figures 4.2.7 and 4.2.8. Like H:Si(100), these substrates are very commonly used in SPM research. Indeed, H_2O @ C_60 was deposited onto Cu(111) for the experimental work performed in Chapter 7.

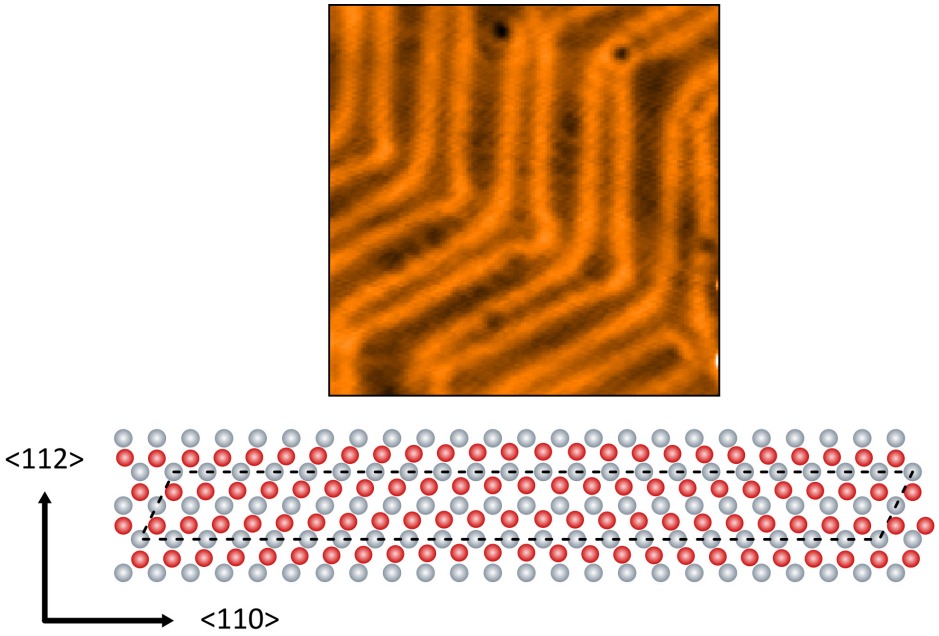

Figure 4.2.7: The Au(111) $22 \times \sqrt{3}$ herringbone reconstruction. As seen in the theoretical image (bottom), 23 surface atoms (grey) must occupy just 22 lattice sites (red), which results in a raising in the 112 plane to provide strain relief. Over a large area, this produces Au(111)'s iconic herringbones, as pictured under STM (top). This image was used as an example from which to train a neural network to detect sharp STM tips[96]. Image courtesy Swart et al.

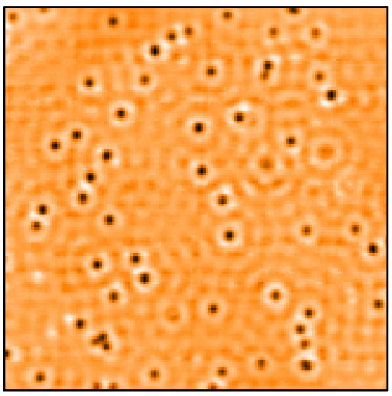

Figure 4.2.8: The Cu(111) surface with a relatively high concentration of adventitious Co adsorbates. STM data courtesy Swart et al (provided without scan parameters).

4.3 Historic Experimental Dataset

4.3.1 Acquisition & Scan Parameters

Before even beginning to think about training a CNN, we must carefully consider the surfaces we will investigate. As discussed above, H:Si(100) is an ideal test-bed for developing CNN automation techniques. Furthermore, because it has been previously studied in the context of machine-learning-enabled SPM[22,110], a good comparison can be formed with existing machine learning approaches based on full scans. However, these classifications are surface-specific. To study non-surface-specific defects, and to demonstrate the general applicability of our CNN protocol, we also consider two other commonly studied surfaces[136,137] of Cu(111) and Au(111), with data provided by Ingmar Swart's group in Utrecht.

To make use of historical data and ensure that we can demonstrate our CNN can work under a variety of generic scanning conditions, H:Si(100) images were obtained from historical scans from the Nottingham Nanoscience Group, acquired at room temperature between March 2014 and November 2015 on the Scienta Omicron VT STM described in Subsection 2.1.4. Because these were historical scans, scanning parameters were not fixed, but had various rotations, length scales between 3 x 3 nm^2 and 80 x 80 nm^2, and resolutions up to 512 x 512 pixels. The Au(111) and Cu(111) images were acquired similarly on an Omicron LT, but at a fixed scan size (30 x 30 nm), resolution (150 x 150 pixels), and temperature (4.5 K). This gave a total of 13,789 images.

4.3.2 Manual Classification

As of the requirement to have 'correct' labels, $\hat{\mathbf{y}}$, (discussed in Subsection 3.2.1), all 13,789 images had to be manually classified by hand. As shown in Figure 4.3.1 H:Si(100) was classified into the four tip states of "atoms", "dimers", "asymmetries", and "rows", along with two other categories of "tip changes" and "generic defects". Similarly, Au(111) and Cu(111) images were classified by the Utrecht group into five categories of undesirable defects ("Double tips," "tip changes," "step edges," "impurities," and image corruption "defects"), along with the one desirable state of sharp resolution[21].

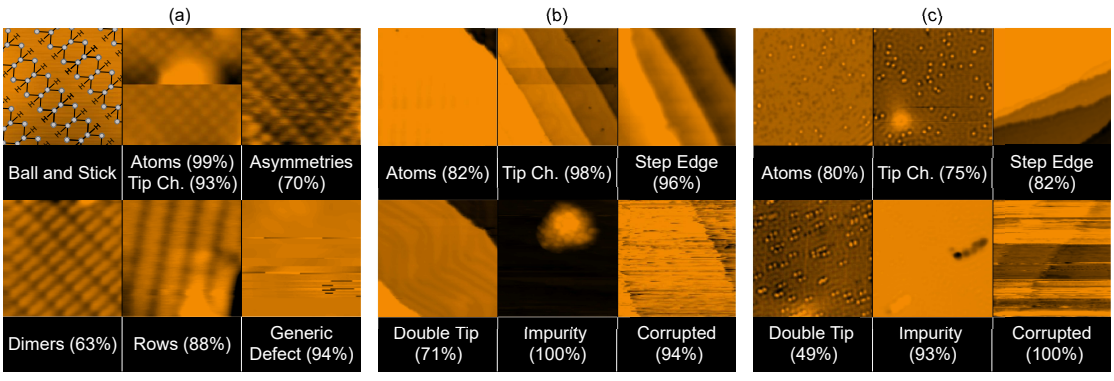

Figure 4.3.1: Selection of images demonstrating key tip states for STM imaging of **(a)** H:Si(100), **(b)** Au(111), **(c)** Cu(111), and the confidence thresholds of convolutional neural networks used to classify them. We note that in many examples, features can appear to strongly blend between images, such as with asymmetries and dimer-like modulation in rows in **(a)**. Because creating unambiguous training data was impractical, we therefore combined these classes. Typical scans in this thesis were approx. V=-2V, I=-1nA.

4.3.3 Filtering, Preprocessing & Augmenting

As discussed in Subsection 3.4.3, careful feature engineering is key to maximising CNN performance. We must also take into account the fact that any adjustments we make must be physically compatible with real-world measurements.

Firstly, although in practice STM images are multilabel (in which images can belong to multiple categories), we classified and discarded images such that we only trained with multiclass (in which images can belong to only one category). This was beneficial as CNNs learn from the relationship between categories and so did not have to learn to ignore impossible relations (for example, an image can be both 'atomic' and 'tip change', but not 'atomic' and 'generic defect').

It is also known that although a CNN can learn with ambiguous or misleading training labels, performance is reduced[10]. However, because undesirable tip changes can occur even when observing a desirable tip state, these were not excluded. Instead, tip changes were trained in a separate binary yes/no CNN for H:Si(100), and the remaining images trained in a four-class CNN. Splitting datasets by tip changes was not applied to the Cu (111) and Au (111) datasets as the aim here was to explore the relations between undesirable defects.

Another complication comes in the form of variety of removing ambiguous images. While we did not train on images that the human classifiers did not agree on, the small number of human classifiers and large degree of ambiguity in classification meant that many ambiguous images remained. In the absence of a perfect classification system and greater number of classifiers, these imperfect human classifications formed the training data that the network had to learn from. As such, no CNN could achieve 100% accuracy without overfitting. For example, given that 78% of the silicon dataset was agreed upon, it could be tentatively argued that a human-like CNN would score similarly. (A poll carried out in our group which involved the manual classification of a small subset of the Si dataset by nine scanning probe microscopists, similarly found only 73% agreement.)

To further improve training performance, the training and testing data were repeated and augmented. Expanding on the simple vertical and horizontal flips used by Rashidi, we also applied rotations from 0° to 360°, and cropped, panned, and added random amounts of Gaussian noise. For the tip change categories, only horizontal flips and Gaussian noise were applied as in our case tip changes were horizontal shears, and zooming in might crop off the discontinuity. Additional random trends created by the physical scan environment were negated with minimal processing[138] by flattening data on a line-by-line basis along the x axis. Each individual image was also scaled to the order of -1 to 1.

After randomly selecting a subset of images to use as validation data, train and testing data were then created by randomly splitting the remaining images with an 80:20 ratio. Ultimately, 3386 H:Si(100), 3600 Cu(111), and 2470 Au(111) images were used for training/testing and 431, 1120, and 432 images for verification, respectively.

4.4 Offline Classification

4.4.1 Training Methods & Models

In addition to the network described by Rashidi and Wolkow[22] (RW), we tested models similar to the popular visual geometry group (VGG) network with and without batch normalisation. We also tested a model highly similar to Google's Squeezenet[139]. This network had ten back-to-back convolutional layers, filters increasing in number from 32 to 1024, strides alternating between 1 and 2, and 3x3 convolutions. Between layers, batch normalisation and the elu[140] activation function were applied. A more traditional Random Forest Classifier (RFC) was also implemented for comparison.

We note that although multi-class networks are typically trained with a sigmoid activation function and categorical cross-entropy loss function, we did not use these. Because future data would be multi-label, we instead opted for the multi-label standard of softmax and binary cross-entropy[141]. This made the confidence prediction of each category 0-1 independent of each other, instead of mathematically linking all the predictions for each category to sum to 1*.

Another adjustment had to be made to prevent overfitting by learning about the highly different number of images in each class. For example, 5.6% of the images in the filtered H:Si(100) dataset were atoms, compared to 41.9% generic defects. Large variety was also observed in the Cu(111) and Au(111) sets. We therefore weighted the loss by the reciprocal of the percentage of each class present, used a weighted accuracy metric[142] (where the reciprocal of number of classes is defined as guessing), and randomly shuffled data. Without weighting, our networks would not assess each image purely on its own merits, but partly on the luck of previous users in obtaining good scans.

To determine an optimal ensemble CNN, a variety of model structures, optimisers[143–146] and learning rates, were trained and analysed for each model structure. In all cases, training was done at a batch size of 128, and downscaled image sizes of 128x128 pixels. This downscaling was required as the fixed number of nodes and neurons in a CNN makes it impossible to pass variable sized inputs (an issue we return to in Subsection 4.5. At higher resolutions training time massively increased, but with little to no improvement in performance.

The top performing models were then combined to create a majority voting ensemble, which have been shown to further improve performance[147]. For subjective data such as in STM images, this was also more analogous to a majority human vote with different models having different preferences.

Training and analysis were performed with Python 3.6.3, Keras[11] 2.2.2, Tensorflow 1.11.0, and an Nvidia Titan Xp.

*We also note that although RW used sigmoid and categorical cross-entropy functions[22], they were not the standard choices for their binary classification scheme because both the confidence of the positive and negative classification could be high, degrading performance.

4.4.2 Performance Comparison

First, the individual models were compared. Table 4.4.1 displays the best results obtained for all the desirable/undesirable multi-class models. Although all networks performed significantly better than RFC and weighted random guessing, the RW CNN performed poorly and similar to the more traditional RFC. Furthermore, at the 32x32 image size described by Rashidi and Wolkow[22], RW performed comparable to random guessing, indicating the high difficulty of this task.

We also found a wide variety in performance between different surfaces, indicating that *CNN architectures respond differently to different surfaces*. For example, whilst Squeezenet was the best performer for H:Si(100), only VGG like networks were suitable for Au and Cu. Furthermore, batch normalisation improved performance on H:Si(100), whilst negatively impacting Au(111) and Cu(111). This variance is understandable, given the current lack of consensus on how network structure relates to performance on a given data set[10].

The best performing networks were then taken and turned into an ensemble. Three were chosen as this gave a good balance between performance and memory usage. As expected, small performance improvements were seen when moving to ensembles. For H:Si(100), the top performer was an ensemble of two SqueezeNets and one batch-normalised VGG, with adam, sgd and rmsprop optimisers, and learning rates of 0.001, 0.0001 and 0.0001 respectively. However, our ensemble structure did not train well with Cu(111) and Au(111) (65% balanced accuracy, 0.64 precision, 0.89 AUROC on Cu(111)). This is likely because of the low performance of the component networks on these surfaces. As such, ensembles for Au(111) and Cu(111) were therefore created from multiple repeats of the VGG like network.

Although Table 4.4.1 indicated strong overall performance, this was likely underestimated. Considering Figure 4.3.1, there was a high degree of feature overlap (such as Si dimer images with bright, asymmetric edges) which made the classification task subjective. While these categories were eventually combined as they were routinely misclassified together[†], the filtered multi-class images still contained acceptable multi-label answers. Because we only allowed one correct classification for any image, the network was often punished despite producing a sensible distribution. This would have been avoided were significantly more classifiers employed.

Despite this, Figure 4.4.1 shows that the AUROC for all categories and surfaces was very high. This indicated that the classifier had a low false positive rate, but at the cost of a high false negative rate. Although decreasing accuracy, this characteristic is not detrimental to areas such as ours when only positive predictions are to be acted upon. Furthermore, ambiguous classifications often had confidences <0.5, increasing false negative count and reducing accuracy further still. Unambiguous cases, such as corruptions, impurities and individual atoms of Au(111) and Cu(111) and generic defects of H:Si(100) were otherwise classified extremely well with near perfect AUROC.

[†]We note that although it seems like asymmetries/atoms should be grouped because they are visually similar, the end goal was to observe atoms.

	Ensemble			SqueezeNet			VGG (Batchnorm)			VGG			RW			RFC			Random		
	Si	Au	Cu	Si	Au	Cu	Si	Au	Cu	Si	Au	Cu	Si	Au	Cu	Si	Au	Cu	Si	Au	Cu
AUROC	0.95	0.98	0.94	0.94	0.95	0.88	0.92	0.93	0.85	0.91	0.98	0.93	0.87	0.82	0.77	0.79	0.88	0.83	0.50	0.50	0.50
Bal. Acc.	0.78	0.86	0.80	0.77	0.71	0.67	0.74	0.74	0.59	0.72	0.86	0.76	0.62	0.55	0.50	0.46	0.53	0.52	0.25	0.16	0.16
Precision	0.89	0.95	0.75	0.88	0.82	0.67	0.82	0.77	0.57	0.82	0.92	0.72	0.71	0.54	0.47	0.57	0.62	0.51	0.25	0.18	0.17

Table 4.4.1: Table to compare the performance of a variety of machine learning methods for classifying desirable and undesirable tip states for six classes of Au(111) and Cu(111), and four classes of H:Si(100). The SqueezeNet, VGG, Rashidi-Wolkow (RW) and ensemble networks are examples of convolutional neural networks. These all performed significantly better than the more traditional Random Forest Classifier (RFC) with 5000 trees, and random guessing, which performed as expected.

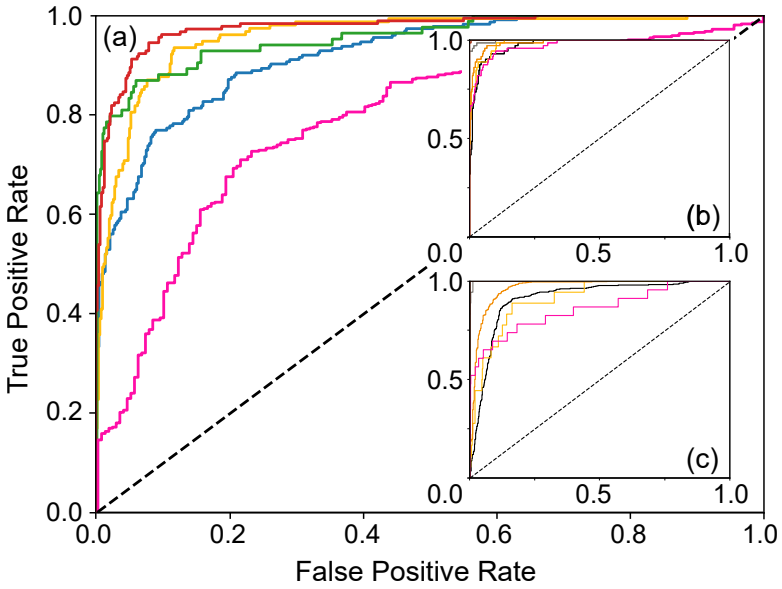

Figure 4.4.1: Receiver Operator Characteristic graphs demonstrating the overall performance and area under curve as the confidence threshold required to make a positive prediction is varied for CNN ensembles. Classification performance is compared for scanning tunnelling images of **(a)** H:Si(100), **(b)** Au(111) and **(c)** Cu(111). A perfect classifier has an area under curve of 1, with guessing 0.50 (black dash, theoretical). For **(a)** we find asymmetry/dimer = 0.92 (blue), individual atoms = 0.96 (yellow), rows = 0.95 (green), tip change = 0.79 (pink), and generic defect = 0.98 (red). For **(b)** and **(c)** respectively we find impurities = 1.00, 1.00 (grey), double tip = 0.98, 0.91 (black), corruption = 1.00, 1.00 (brown), individual atoms = 0.98, 0.91 (yellow), step edges = 0.99, 0.97 (orange), and tip change = 0.97, 0.86 (pink).

Furthermore, misclassifications were often between sets of desirable/undesirable states, rather than with desirable states being misclassified as undesirable and vice versa. To demonstrate this, the four class H:Si(100) and Au(111) ensembles were simplified into "good/bad" classifiers. They then achieved improved balanced accuracies of 93% and 91%, mean precisions of 0.96 and 0.97, and AUROCs of 0.98 and 0.98 respectively. Cu(111) did not improve owing to poor PR of individual atoms and tip changes, as visible in Figure 4.4.2(c).

However, although tip changes were classified respectably with Au(111) and Cu(111), this was not the case with H:Si(100). When including the separate binary network to cover all classes for H:Si(100), performance was significantly poorer, with a balanced accuracy of 77%, mean precision of 0.88, and average AUROC of 0.92. This is particularly visible in Figure 4.4.1, with the tip change category having an ROC line below the other categories and AU of 0.80. This is likely because when augmentations were limited to simple flips and noise, the network rapidly overfit. Regardless, few false positives were made for tip changes when increasing confidence thresholds. This is because precision was only seen to decrease at high values of recall, as visible in Figure 4.4.2. As such, tip states can still be distinguished with a low false positive rate by raising the confidence threshold.

4.5 Online Classification in Real-Time

4.5.1 Justification

Whilst we have now shown that CNNs are capable of assessing SPM tips[22,96,110], we must consider its practicality. Currently, they require (as described in Section 3.2) to make an

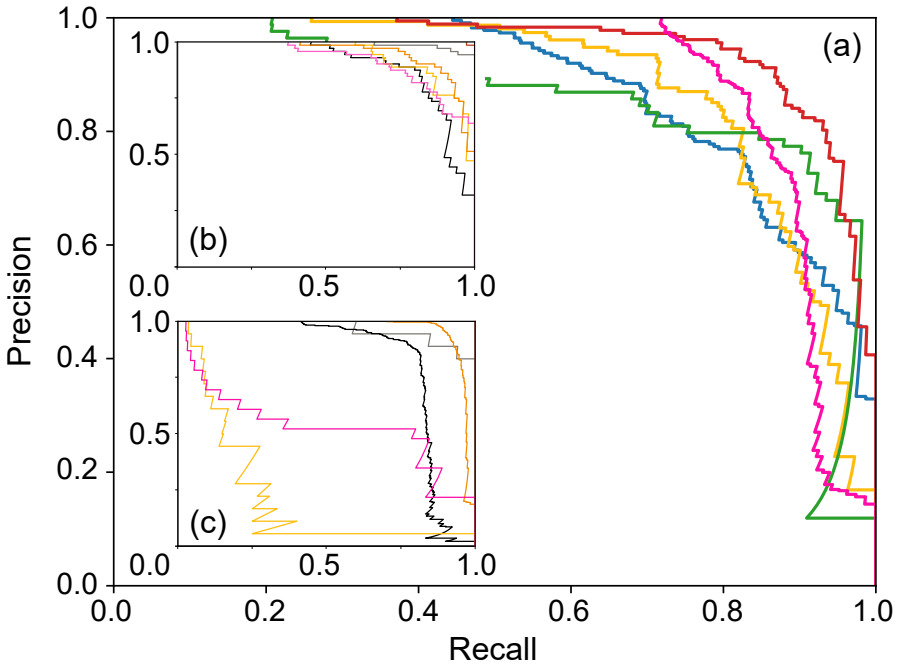

Figure 4.4.2: Precision-Recall graphs to demonstrate the overall performance of ensemble CNNs when classifying the known tip states for images of **(a)** H:Si(100) **(b)** Au(111) **(c)** Cu(111) as the confidence threshold required to make a positive classification was varied. Precision is the percentage of true positives compared to total positive classification, while recall is the percentage of positive classifications that have been correctly identified as positive. Some tip states are desirable and surface specific, such as asymmetry/dimer (blue), individual atoms (yellow), and rows (green). However, tip changes (pink), impurities (grey), double tips (black), corrupted (brown), step edges (orange), and generic defect (red), are undesirable. Performance is strong, except for individual atoms and tip change in (c).

assessment, and so we cannot use partial scans. This method of CNN assessment *after complete* scans compares extremely poorly to human-based assessment, in which SPM operators routinely perform accurate assessment *during in-process* scans by observing individual line profiles as the image is acquired. Indeed, as little as 1-2% of a full scan may be required to correctly assess a particularly poor tip. Furthermore, because the majority of time spent maintaining the tip is spent acquiring the data to assess, manual maintenance by a human is beyond an order-of-magnitude faster than any current CNN protocol. Given that manual maintenance can take several hours as-is, automated tip assessment with full-scan CNN protocols may be unable to keep up with the demands of SPM experimentalists unless an alternative strategy is introduced.

Furthermore, there is a wealth of data embedded in SPM scans, which can be exploited with neural network structures. For example, Burzama et al. have very recently shown[148] that single layer neural networks can be used to extract meaning in Ising model images, which are otherwise difficult for humans to interpret. A key difference in our case is that we focus on accurately observing atomic resolution, which is lost (or at best aliased) for larger scan areas. This is not the case for Burzawa et al's work, where close to a critical point the correlation length becomes effectively "system spanning". These patterns at criticality therefore are described by power law behaviour. Indeed, our group has previously examined this type of power law behaviour and the associated structural correlations for nanoparticle assemblies[149,150].

4.5.2 Training Method

For easy comparison, we use the H:Si(100) dataset from before, along with a VGG-like[76] networks found above to have strong all-round performance. This network[76] begins with two 2D convolutional layers of 32 output filters, 3x3 convolutional filters, and 3x3 strides. This is followed by a third max pooling layer with 2x2 convolutional filters and 2x2 strides. This three layer block is then repeated, but with output filters of 64 and then 128 layers respectively. The very first convolutional layer in the model was then altered to have 7x7 convolutional filters and 2x2 strides.

This structure was then trained three separate times to create a majority voting ensemble. Not only does this allow for the performance benefits seen when taking a majority vote of a subjective task, but also reduces variance in CNN performance which was found to vary by about 1% between repeats.

4.5.3 Padding & Masking

In many neural network applications, data are often of varying length. For example, in natural language processing[151], some words and sentences are inevitably longer than others. In these cases, shorter pieces of data are lengthened by "padding" them with a marker value[151] until they are as long as the longest piece of data. The marker value is chosen such that it cannot naturally appear in the real data. Training and testing then continues as normal, as the network learns to ignore the marker value. In the context of SPM, we can exploit the fact that images are sequentially generated one linescan at a time, and that completed images contain the same number of linescans, regardless of scan parameters. During an incomplete scan, the missing linescans can therefore be replaced with a marker value to allow the network to produce an output. Figure 4.5.1 demonstrates how data could be padded during scanning to form a full sized image.

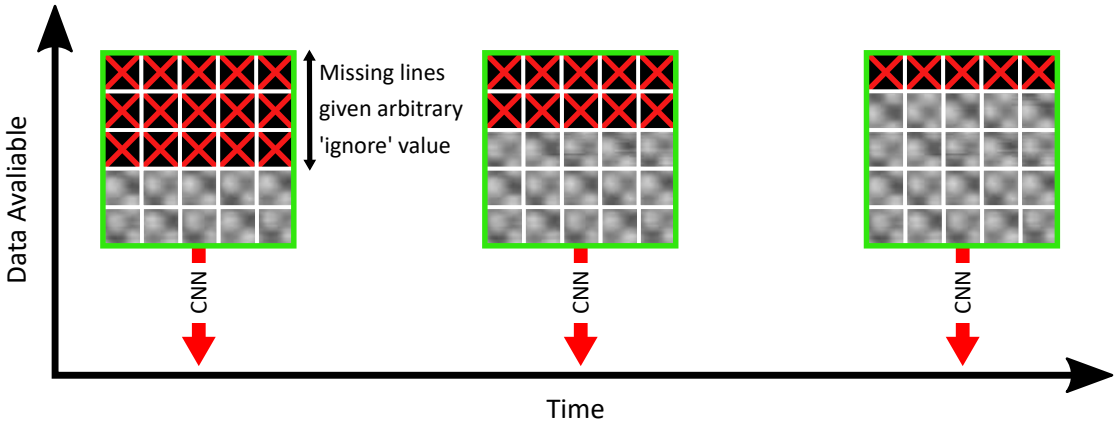

Figure 4.5.1: Figure to demonstrate a potential method to allow neural networks to predict the state of an SPM tip using incomplete scans. Because CNNs can only make predictions if given the same number of data points used during training, it is not possible to make predictions using incomplete scans. It is also computationally wasteful to create multiple CNNs for each stage of scan completeness. Instead, partial scans can be "padded" with an arbitrary marker value until there are enough data points to equal a full sized scan. After each successive linescan, less padding is required. This allows the CNN (green border) to train/predict using incomplete scans.

As such, it is possible simulate and test partial scans with the original dataset of complete scans. To do this, a random number of linescans from the end of the scan were "masked" during training by replacing the real data with the marker value. To do this, we let

$$\begin{bmatrix} \mathbf{A}^i_j \\ \mathbf{A}^i_{j+1} \\ \mathbf{A}^i_{j+2} \\ \vdots \\ \mathbf{A}^i_N \end{bmatrix} = M, \qquad \{4.2\}$$

where N is the total number of lines in a full image, and M the marker value. This produces an array, $\mathbf{A}^i_j$, for the i^{th} image of a dataset, in which only j linescans appear to have been produced. Data is then further augmented by repeating $\mathbf{A}^i$ with random j.

Although this method is simple and can easily be applied to existing protocols, the use of a marker value is of course highly problematic. In the context of SPM, data can theoretically contain any positive or negative value within the operating range of the acquisition hardware. As such, no marker value exists that could not show up in the actual dataset, without being so large as to make the actual data miniscule by comparison. As such, the network will likely become insensitive to some of the actual data. Given that each line was pre-processed to have mean of 0 and standard deviation of 1, we therefore consider arbitrary marker values of $M = 0$ and $M = 10$. As an alternative, we also consider "tiling" by repeating $\mathbf{A}^i_j$ to full scan-size, negating the need for an arbitrary marker value.

To test this method, the performance of the CNN ensemble was calculated as one additional line was unmasked at a time. We do this by masking from the j^{th} line of $\mathbf{A}^i_j$ using Equation 4.2 for all 648 images in the verification dataset. The CNN ensemble was then used to predict the tip state, $\mathbf{P}(\mathbf{A}^i_j)$, from $j = 2$ to $j = N$. By assuming the human prediction to be perfectly correct, performance was calculated by comparing CNN predictions to the corresponding human predictions. Performance is shown as a function of j in Figure 4.5.2.

As such, it can clearly be seen that for all types of padding, the CNNs successfully learnt to make correct observations with limited data. Furthermore, when comparing the performance difference of small amounts of data with $j = 2$ to full scans with $j = N$, the performance of all padding types only decreased by an average of $4 \pm 1\%$, $8 \pm 6\%$, and $7 \pm 2\%$ for mean AUROC, mean precision, and balanced accuracy respectively. Given that these are significantly better than the 0.50, 0.25 and 0.25 of guessing respectively, it is clearly entirely possible to assess SPM tip state with only a small number of linescans.

However, at $j = N$ the padding-enabled-CNNs performed significantly worse than an identical ensemble trained without padding. Here, padding reduced full size performance by up to 12%, 23% and 22% for the mean AUROC, mean precision, and balanced accuracy respectively, when compared to the worst performing padding methods. Giving CNNs the ability to classify partial scans therefore significantly harms performance, reducing the real-world effectiveness of such systems. We also note that this architecture also performed better than the ensembles

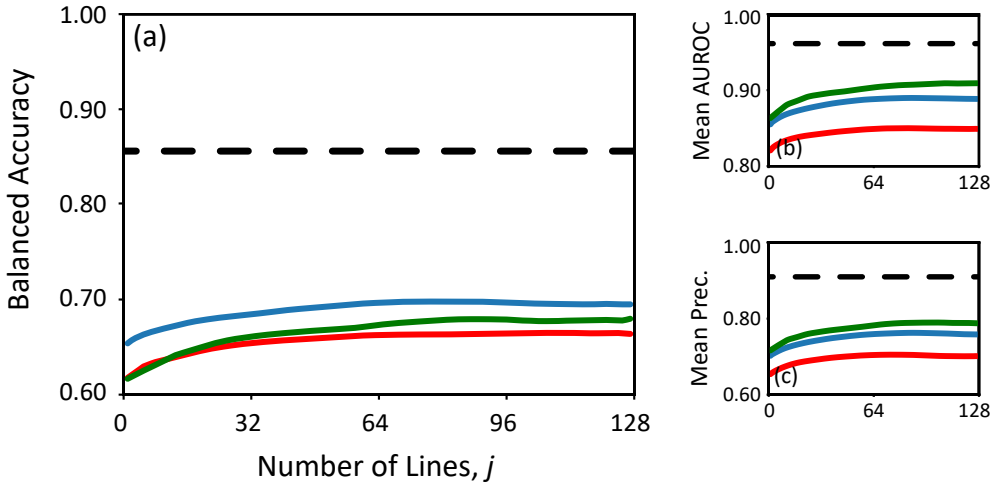

Figure 4.5.2: Figure to demonstrate the balanced accuracy **(a)**, mean area-under the receiver-operator-characteristic curve **(b)**, and mean precision **(c)** of a neural network trained to classify partial STM images of the H:Si(100) surface. Given that SPM data is generated one line at a time, incomplete scans can be padded to full-size with a marker value that the otherwise identical network then learns to ignore. In this way, the data requirements for neural network automated state detection can be reduced significantly. Here, we consider marker values of 10 (red), 0 (blue), and also tile the data to size (green). However, performance is far below an identical network trained exclusively on full size data (black)

presented in Section 4.4. The large initial convolutional window may have caused this. Besides the reduced maximum performance, there was also a large computational inefficiency due to training the CNNs to perform (and subsequently ignore) a large number of expensive calculations on meaningless data.

One advantage of partial-scan methods is that tip changes can be instantly detected by looking for changes and impulses in CNN output, as visible in Figure 4.5.3. This is a significant improvement on the previous full-scan methods which requires a secondary "tip change" network[96]. We note that without manual labelling of all tip change locations on all images, a quantitative analysis of tip-change detection is not possible. However, the imperfect ignoring of the marker values meant that some of the horizontal shears due to tip changes caused little-to-no-change in network output. The change in prediction to reflect a new tip state was also often small, and tended to "drift" rather than instantly "snap" to the new value. This was particularly problematic for tip changes later on in a scan. One explanation is that the network learnt to heavily rely on earlier scan-lines because training images often had early scanlines present, but later scanlines did so increasingly rarely. It was also impossible to detect tip changes using the "tile" method of data padding, which created a horizontal shear (visually identical to a tip change shear) between every tile. As such, padding should only be employed early on in scans and when the tip state is likely stable.

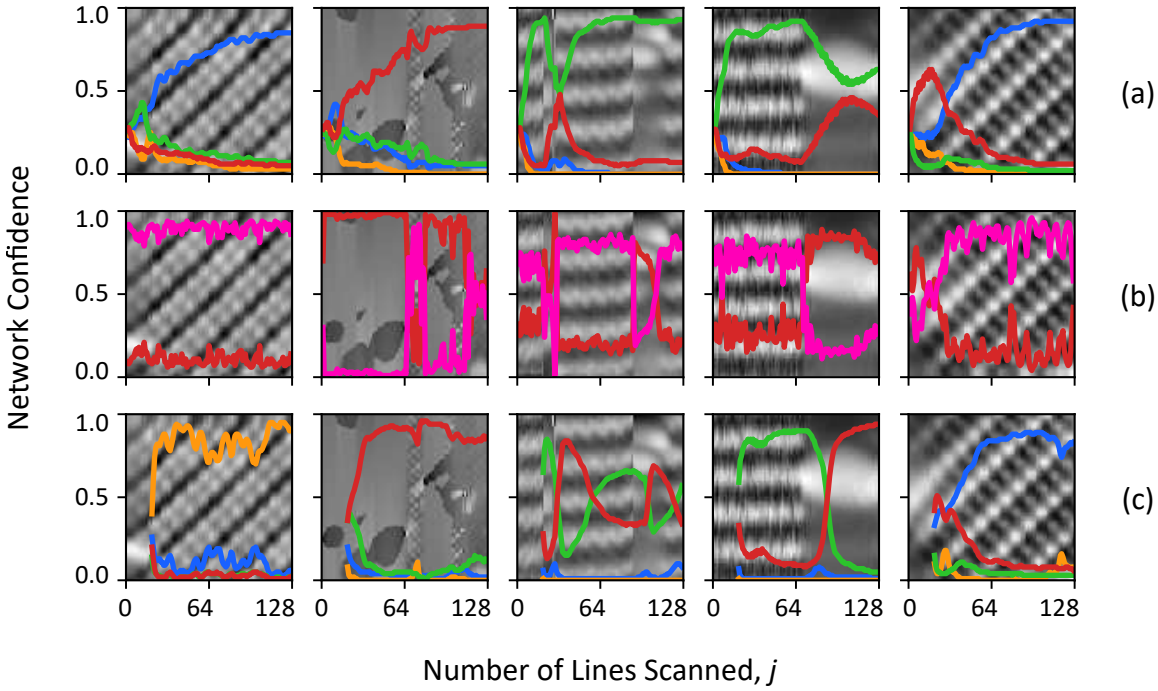

Figure 4.5.3: Figure to demonstrate a variety of methods by which the H:Si(100) tip states of individual atoms (yellow), asymmetries/dimers (blue), rows (green), and generic defects (red) can be recognised from incomplete SPM scans. Instead of detecting SPM tip states using complete scans, neural networks were taught to recognise partial scans by zero padding **(a)**, or by classifying single linescans **(b)**. In this case, non-defect categories had to be combined together (pink). However, optimal results were found by forming a "window" with a small group of 20 consecutive linescans, and giving additional predictive power by using a second LSTM network as the window is "rolled" over time **(c)**. This network was found to perform the strongest at single class classifications, and showed good responsiveness with varying tip state. Typical scans in this thesis were approx. V=-2V, I=-1nA.

4.5.4 Individual Linescan Windows & Cumulative Averages

One alternative to padding incomplete scans is to train to classify the individual linescans that form an image, rather an image in its entirety. As new lines are scanned, they could immediately be predicted. This negates much of the insensitivity and computational wastefulness caused as a result of padding, and is demonstrated in Figure 4.5.4.

However, one consequence of basing predictions on individual linescans is that each linescan is stripped of its context to the rest of the scan. Acquiring more linescans should therefore not improve network performance. As such, a small amount of context can be applied to the other scanlines in the image by applying an additional layer to cumulatively average the network predictions using the equation

$$\mathbf{P}(\mathbf{A}_j^i) = \frac{\sum_{k=1}^{j} \mathbf{P}(\mathbf{A}_k^i)}{j}, \qquad \{4.3\}$$

where $\mathbf{P}(\mathbf{A}_j^i)$ is the cumulatively averaged vector describing the predictions of the j^{th} linescan of the i^{th} image in the dataset. A prediction for the entire image is therefore found when the condition $j = N$ is met.

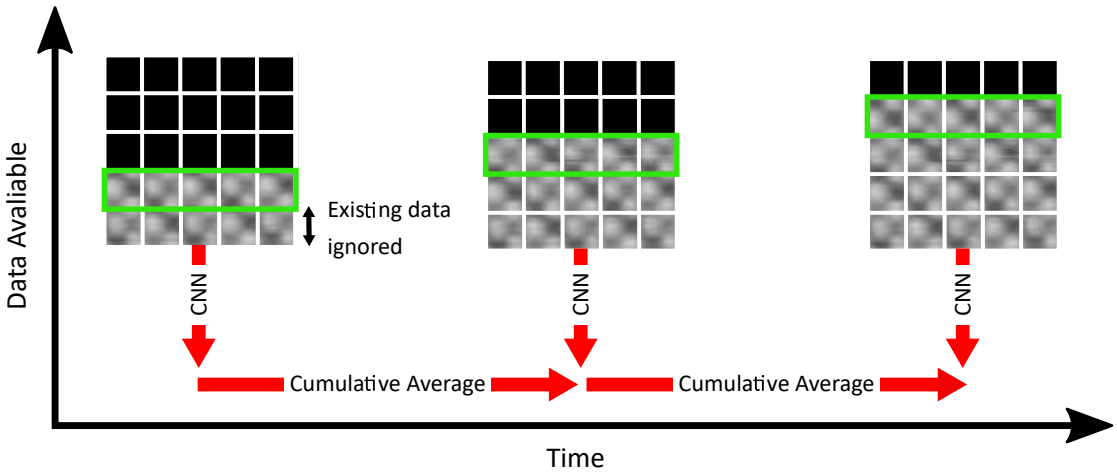

Figure 4.5.4: Figure to demonstrate a potential method to allow neural networks to predict the state of an SPM tip using incomplete scans. Instead of training/predicting with complete scans, the network (green border) was adapted to predict individual linescans. As more linescans become available during a scan, network predictions are cumulatively averaged to give context between successive linescans.

We also note that although the cumulative averaging provided context to the predictions, the actual predictive part of the network was unaware of the surrounding linescans. Whilst this averaging therefore served to reward consistent single-class output, it should be expected to have poor responsiveness to scans where the tip constantly changes shape. Furthermore, the network had little-to-no ability to distinguish between features that cannot be distinguished at the 1D level. For example, a single linescan of 'atoms' or 'rows' features in Figure 4.1.1 would appear identical with a half-rectified sinusoid. The varying scan areas of the dataset required to make predictions invariant to scan area then prevent the network from learning any spatial information to distinguish between the two states. As such, the number of tip states was simplified to just two - "generic defect" and "visible resolution".

Adaptions also had to be made to the network architecture. Because 2D convolutions cannot be performed on 1D data, the 2D layers of the network were replaced with their one-dimensional counterparts to provide the closest possible comparison between the protocols. Furthermore, because successive lines were often highly similar, only 1 in every 30 lines of each image were used during training to prevent improper training and decrease training time.

As before, performance was verified by iteratively predicting additional lines of the i images in the holdout set and calculating the cumulative predictions using Equation 4.3. This is shown in Figure 4.5.5. To compare with full-sized performance, the 1D convolutions were replaced with their 2D equivalents, and trained to recognise only the two simplified states.

Without the cumulative averaging layer, the low standard deviation demonstrated that performance remained near constant as expected, with AUROC of 0.853 ± 0.006, mean precision of 0.841 ± 0.007, and balanced accuracy of 0.780 ± 0.004. Un-averaged individual linescans therefore provide an effective means of making a basic, but accurate, assessment of the tip. Further, despite forcing the simplification of classes recognised, the decoupling of the lines

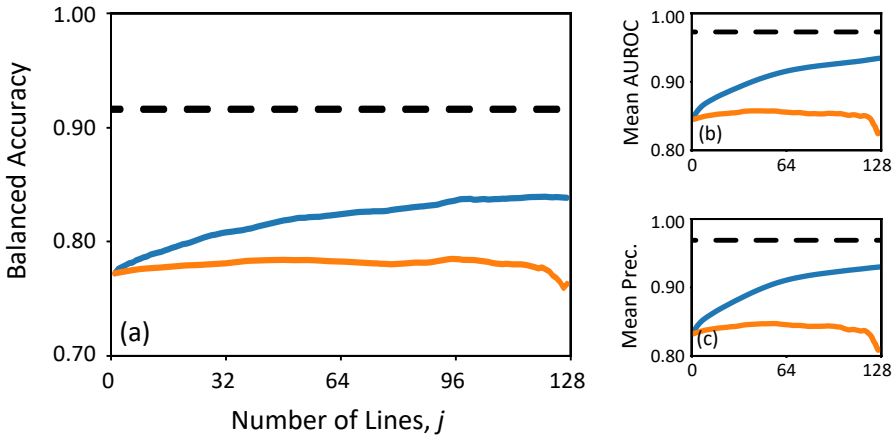

Figure 4.5.5: Figure to demonstrate the balanced accuracy **(a)**, mean area-under the receiver-operator-characteristic curve **(b)**, and mean precision **(c)**, of a neural network trained to classify the SPM tip states of the H:Si(100) surface with incomplete scans. Given that SPM data is generated one line at a time, the identical network can be trained on single linescans, instead of only on complete scans (black). In this way, the data requirements for neural network automated state detection can be reduced significantly. Because this method removes context between scans, predictions can not be influenced by prior scans. Performance is therefore only improved with the addition of new data by cumulatively averaging successive linescans in an image together (blue). Without this averaging step (yellow), performance remains roughly consistent, as expected.

meant that the network was highly sensitive to tip changes. This is visible when looking at the unaveraged output in Figure 4.5.3(b). As this network was clearly more responsive to state changes than padding, it is possible to use the single linescan network, (along with its low computational cost), solely for the purpose of detecting tip changes by looking for sharp peaks and changes in network output.

Regardless, even stronger performance was seen with single-class images after cumulatively averaging. After including the layer, performance began to improve as expected, with AUROC substantially improving by 13.2% relative to the average, mean precision by 11.8%, and balanced accuracy increasing by 9.9%. This resulted in an AUROC of over 0.9, thus demonstrating highly effective ability when full data is available. This was also found to hold true for the padding strategy with all 128 linescans. It should, however, be stressed, that relative to training only with complete scans, peak performance is still reduced. In this case, when training the 2D CNN solely with complete scans and the two simplified categories, AUROC performance was near perfect, at 0.973. Further, cumulative averaging caused predictions to be significantly less sensitive to tip changes, as expected.

4.5.5 Multiple Linescan Windows & LRCN

Whilst single linescans provide an effective method to make a basic assessment of the tip, the inability to assess the complete range of states makes it of limited use. To overcome the lack of context between linescans, a CNN could instead be trained to recognise a small "window" consisting of a fixed number, W, of linescans. As new data becomes available, the window could then be "rolled" to consist of the new line and the $(W-1)$ linescans preceding it. This window could then be iteratively rolled while an image is being generated. We therefore modify Equation 4.3, and use cumulative averaging to make predictions after each successive linescan from $j = W + 1$ to $j = N$

$$\mathbf{P}(\mathbf{A}^i_j) = \frac{\sum_{k=W+1}^{j} \mathbf{P}(\mathbf{A}^i_{(k-W):k})}{j}. \quad \{4.4\}$$

However, whilst effective at improving single-state classification performance and rewarding tip consistency, cumulative averaging does not make the predictive part of the neural network aware of the lines surrounding each window, resulting in decreased responsiveness. One recent advance in the area of video content recognition is the Long-Term Recurrent Convolutional Network (LRCN)[152], which has been shown to be highly effective at this task. Here, a second network is placed just before the final (dense) CNN layer (which reduces the output to a size equal to the number of classification categories). This second network is typically a long-short-term-memory (LSTM) network[153], which is often used for 1D sequence classification. The LSTM network then acts on the temporal domain of the data, giving context to the single CNNs which have no knowledge of how the video frames link together. This can be made analogous to SPM, where each sub-image of width W becomes a video frame. The temporal element is seen as the window rolls when j increments over time with new data. We therefore replace the cumulative averaging layer with an LSTM network with 256 hidden layers, and calculate predictions, $\mathbf{P}(\mathbf{A}^i_{(j-W):j})$, from $j = W + 1$ to $j = N$ as before, with increasing j chosen as the temporal axis. For consistency, we employ the same 2D CNN architecture as before. The resulting protocol is shown in Figure 4.5.6.

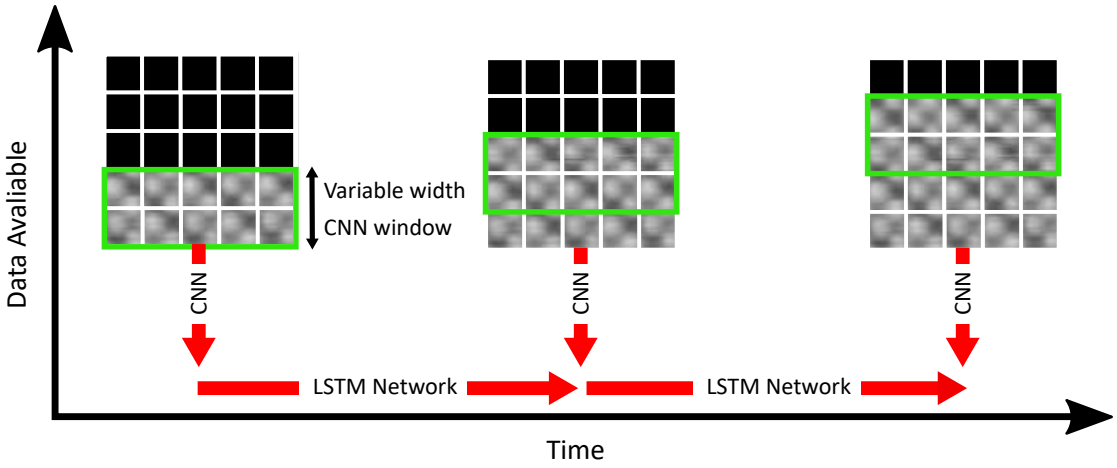

Figure 4.5.6: Figure to demonstrate a potential method to allow neural networks to predict the state of an SPM tip using incomplete scans. Rather than training/predicting with complete scans, the network (green border) can instead be allowed to predict a small group of individual linescans. This window of CNNs can then be rolled to make additional predictions as successive linescans become available over time. The outputs of these CNNs can then be fed into a second temporal neural network, to make a final prediction.

One consequence of this method is that W linescans must first be accumulated before any predictions can be made. As such, whilst larger W will give the network more data with which to make predictions, a larger number of linescans are required to be scanned before the window can be fully filled. For example, for a window of $W = 20$, predictions can only be made after the 20th, 21st, 22nd linescans, and so on. We also note that the CNN architecture used meant that predictions with $W < 20$ were not possible. Furthermore, the need to repeat data increases memory usage $N - W + 1$ times.

As can be seen in Figure 4.5.7, the inclusion of additional linescans once again resulted in improved performance, demonstrating that the LSTM component did indeed learn from the temporal evolution of the scans. Performance was also very strong regardless of j. For example, full scans with $W = 20$ yielded a near-perfect AUROC of 0.960, mean precision of 0.890, and a balanced accuracy of 0.847. This is almost identical to the AUROC, mean precision and balanced accuracy of 0.963, 0.910 and 0.856 respectively calculated when training the CNN component only on full-sized images. The wider LRCN networks were even able to exceed full-size performance, *despite using less data*. This is understandable, given that a human operator will often look not only at the scanlines, but also at how they evolve over time. Only the LRCN network takes advantage of this temporal context. It can therefore be concluded that by adding LSTM to an existing network and retraining on partial scans of fixed size, a full set of STM image classes/tip states can be correctly and accurately assessed with negligible performance impact despite using a fraction of the data. However, increasing W beyond $W = 30$ did not always improve performance. Although wider windows provided more opportunities to observe trends in the 2D convolutional domain, leading to near-baseline precision of 0.908 for $W = 30$ smaller windows provided more temporal elements for the LSTM layer to use.

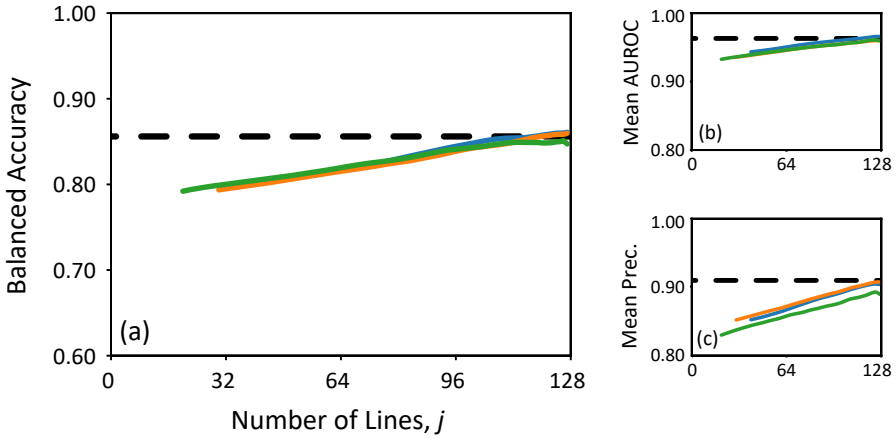

Figure 4.5.7: Figure to demonstrate the balanced accuracy **(a)**, mean area-under the receiver-operator-characteristic curve **(b)**, and mean precision **(c)** of a neural network trained to classify the SPM tip states of the H:Si(100) surface with incomplete scans. Given that SPM data is generated one line at a time, the identical network can be trained with small groups of 20 (green), 30 (yellow), or 40 (blue) linescans, for example. These predictions are then fed into a secondary LSTM network that acts temporally. This prevents the need to train (and therefore classify) only on complete scans (black). In this way, the data requirements for neural network automated state detection can be reduced significantly.

The benefit of using temporal information can also be seen by comparing LRCN to cumulative averaging. For the same $W = 20$ window, full-scan performance using cumulative averaging was calculated to have AUROC of 0.880, mean precision of 0.862 and balanced accuracy of 0.620. Not only was this slightly worse than the padding method of Figure 4.5.2, but also significantly poorer than LRCN, which scored 9.10% higher for AUROC, 3.24% for mean precision, and 36.7% for balanced accuracy. This performance disparity held true regardless of values of W and j, or when classifying variable state images. As seen in Figure 4.5.3(c), cumulative averaging was often unresponsive to both sudden changes in state. Moreover,

LRCN was more able to correctly distinguish between atoms and asymmetries, and was less likely to mistakenly see rotated surfaces as a "generic defect" compared to the baseline of full scan classification. Whilst it would seem obvious to combine both LRCN and cumulative averaging, the issues with decreased responsiveness later in a scan remain. This resulted in a small performance penalty which increased as more linescans were simulated (on the order of 1% at $j = N$). Whilst cumulative averaging was still better than guessing and is therefore another potential method for speeding up tip state recognition, LRCN is superior.

Furthermore, whilst the state of the tip was still successfully observed with $W = 20$, the size and number of convolutions used meant that window sizes below $W = 20$ were not possible to test. This meant that $j = 20$ lines must first be acquired before predictions can be made. To reduce the number of linescans further, larger images could instead be considered (which in this case would be achieved by downscaling from 512x512 to a size larger than 128x128). For example, simulating $W = 20$ with 256 points per linescan would be equivalent to 128 points per linescan with $W = 10$. However, the same number of data-points would need to be acquired before predictions could be made. There would therefore be no improvement to tip assessment speed in practice. Although the network parameters could be decreased to allow for smaller W, this would result in a different network that could not be fairly compared in this study. To allow for predictions at any j, it is trivial to create a "hybrid" network ensemble in which a basic assessment is made using the linescan/padding methods for low j, and then LRCN for the remainder of the scan.

4.6 nOmicron Python Controller for Omicron MATRIX

In order to perform any form of real-time inference, we must have access to the probes' data-streams in real time. Unfortunately, the STM market leader, Scienta Omicron, only allow for control of the probe via its MATRIX control software and MATE scripting language. These are closed, proprietary tools made for a time when automated control of the probe on a scripting level was less practical than constant manual intervention. The MATE scripting language also lacks any means to calculate complex statistics and access data files, let alone interface with the machine learning protocols developed in this thesis.

As such, a Python API was produced to programmatically interface with the Omicron VT. This was based upon Stephan Zevenhuizen's MATE-for-dummies package[154] from a few years ago. Because of its potential use to the wider STM community, the code was developed into a user-friendly package and open-sourced and released as 'nOmicron'[155]. To our knowledge, it has seen use by a small number of UK nanoscience groups, along with users in Germany and China.

At its core, nOmicron uses a sub-routine from MATE-for-dummies which allows for MATE code to be run *via Python* using the (also proprietary) Omicron MATE Remote Access API. In theory, any button or parameters in MATRIX can be called from nOmicron, including the black box controller with the right hardware upgrade. Crucially, nOmicron can intercept any image, spectroscopy measurement, or time-dependant data source being streamed from the MATRIX Control Unit to the MATRIX software. The system structure is shown in Figure 4.6.1.

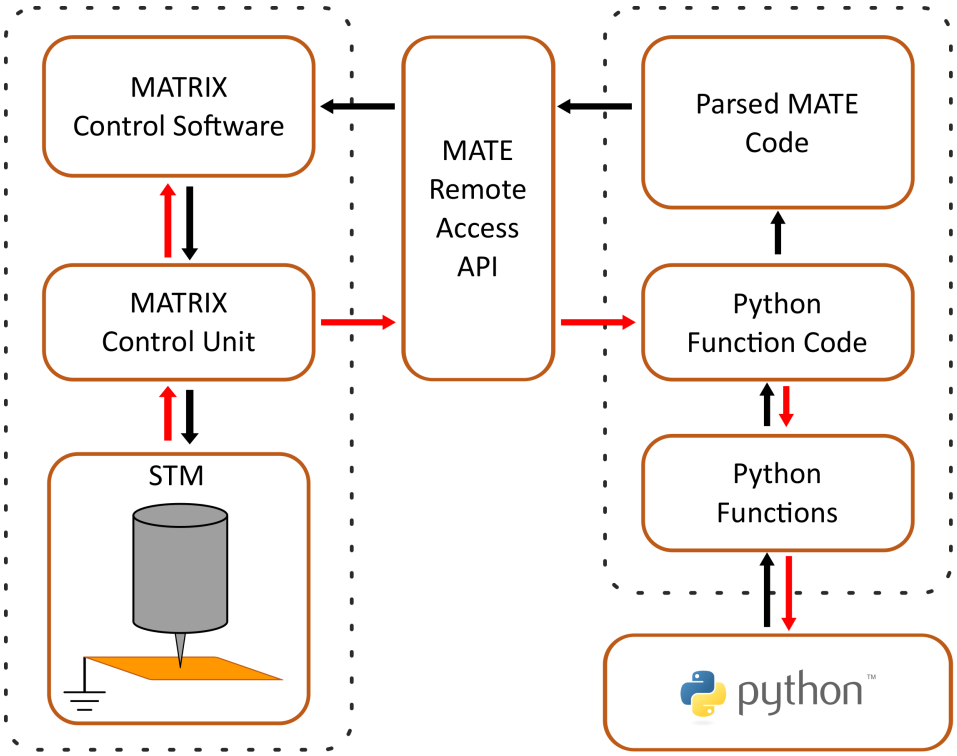

Figure 4.6.1: Figure to demonstrate the logical layout of nOmicron, the custom python package used to control and stream data from Scienta Omicron STMs using Python.

Further, MATE code can only press buttons and edit parameters in MATRIX, making the performing of basic routines (such as 'perform a tip pulse of 8V for 50ms) complex. As such, MATE code is abstracted into convenient, pythonic functions that call routines such as scanning, spectroscopy, tip positioning, tip conditioning and PLL control. These code functions contain multiple options such as conditioning positions and parameters, scan settings, and tip voltages. Further, user-friendly error messages and system hardware capability checks mean that users cannot accidentally cause hardware damage and instability via setting unphysical parameters.

4.7 Real-World Verification

4.7.1 Assessing Real Scans

To verify the protocols discussed above[96,97] with real data, the python bridge discussed in Subsection 4.6 was used to assess a series of new scans of H:Si(100) in real time. A random selection of assessments are shown in Figures 4.7.1 and 4.7.2.

As expected, correct classification for a large number of images were found (but inevitably not all), including correctly adjusting when the tip state varied mid-way through a scan. Furthermore, correct classifications were routinely achieved on images which showed creep, drift, and substantial amounts of noise. The bad/blurry classification was also only often strongly activated in only the poorest of images, as desired. Although the atomic image was misclassified by the network, experimental difficulties meant that only one good atomic image

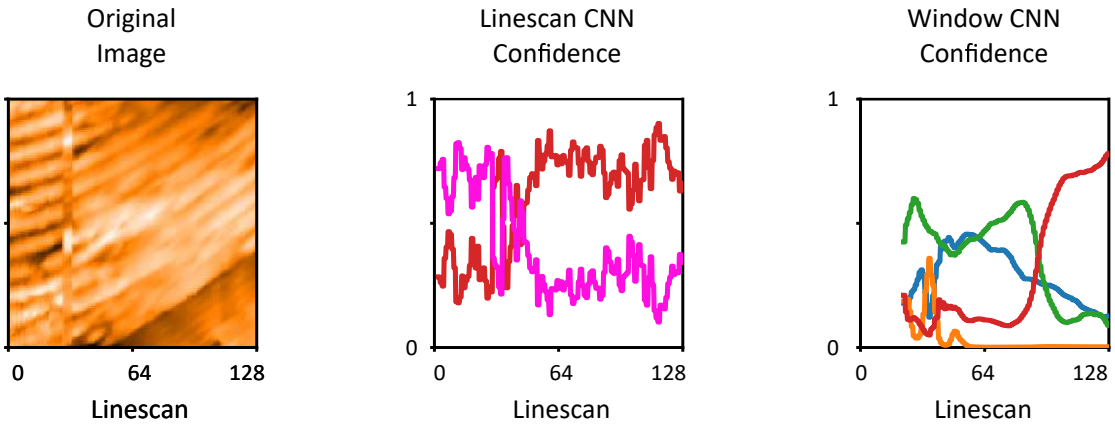

Figure 4.7.1: Figure showing the sensitivity of two different CNN protocols to detect various tip states of the H:Si(100) surface in real-time. The first protocol makes a good (pink) or bad (red) classification with single linescans as input, while the second uses a rolling temporal window of 20 linescans to assess atoms (yellow), asymmetries/dimers (orange), rows (green), and generic defect (red). Image taken over a 10x10 nm area at I=0.5nA, V=-1V, T=300K.

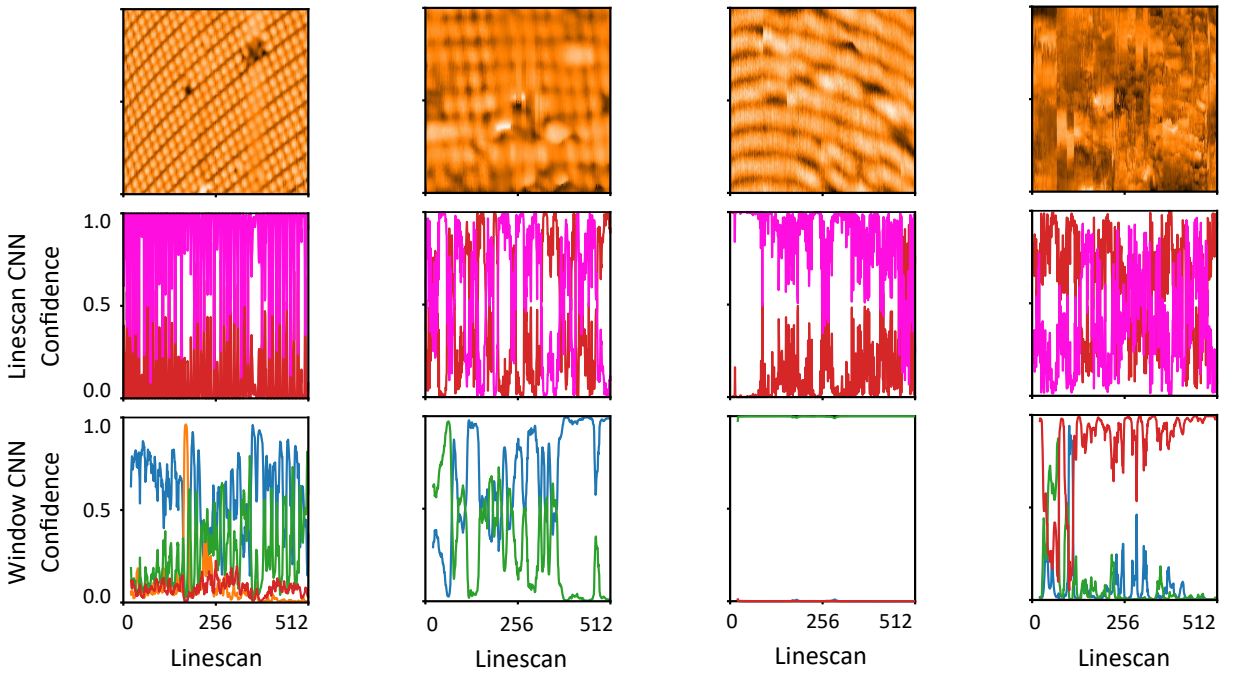

Figure 4.7.2: Figure showing the real-time assessment of several in-progress STM images with two different CNN protocols. The first protocol makes a good (pink) or bad (red) classification with single linescans as input, while the second uses a rolling temporal window of 20 linescans to assess atoms (yellow), asymmetries/dimers (orange), rows (green), and generic defect (red). All images were taken at 300K, with scan parameters (right to left) of 10x10, 7x7, 7x7, 10x10 nm, I=10pA, 150pA, 150pA, 150pA V=1, -2.1, -2.0, -2.1 V.

could be obtained to test against. This misclassification may therefore be explainable as an outlier, rather than a fundamental issue with the protocol. However, it should be noted that this particular network often tended towards making 'dimers' as misclassifications, whilst the original would tend towards 'bad/blurry', which is arguably more desirable. Further, for the single linescan assessment of this image, classification performed extremely well, especially when smoothing over several linescans.

One likely explanation for misclassifications was the fact that every incoming scan line was normalised to the be about 0 with a standard deviation of 1. Human operators, however, often look at linescan height range to determine tip quality, the current CNN protocols are having to provide an assessment with some key information missing. The network structures could be altered to accept both a linescan/image input, along with auxiliary inputs such as the range of scan heights, surface roughness, and potentially even tip voltage/setpoint if sufficient training data were available. Furthermore, misclassifications were often seen in images containing step edges and surface defects. One possibility to overcome this would be to reclassify the entire dataset (as step edges were often classified as 'bad' due to inexperience), and include images with step-edges instead of attempting to remove them. Step edges could also be found with classical statistics as in Wooley and Stirling et al[8,138], the scan frame repositioned, and the CNN polled again.

The largest issue, however, is that the CNNs are very poor at classifying large scans above 20x20nm in scale. This is likely due to the lack of training data at this scale and above, the downscaling of scans removing fine detail, and the fact that scan scale is not passed to the networks as an auxiliary input. This is to be expected, given the training data. Human scan operators also typically look at larger scan sizes when beginning to improve a tip, especially as if something large drops off the tip then a smaller scan size increases the chance of scanning back over it and potentially picking it up again. This is compounded by the issue wherein it is not possible to coarse move the tip in x/y/z programmatically (Matrix cannot do this, therefore MATE cannot do this, and therefore the Python API cannot do this either). This is only possible by bypassing the coarse movement controller and creating custom code and hardware to operate the coarse piezos in the scanner head. As such, when creating a network to improve tips, any developed system will have to employ classical scan statistics, not CNNs, to form a meaningful assessment of the tip state at larger scan sizes.

4.7.2 Limiting Temporal Memory

Additionally to verifying the current networks, the rolling window strategy was updated to allow its "memory" of linescans to be limited by restricting the maximum number of linescans could be rolled. Originally, up to all 128 compressed linescans could be rolled, limitations from 20-128 were considered. As can be seen in Figure 4.7.3, limiting memory increased sensitivity, resulting in a quicker response to classification after tip changes. It also meant that otherwise good scans were not as adversely affected when they contained defects and noise at the start of the scan. On the negative side, completely consistent images could not be as strongly differentiated from variable state images, achieving which is the main end-goal for our system.

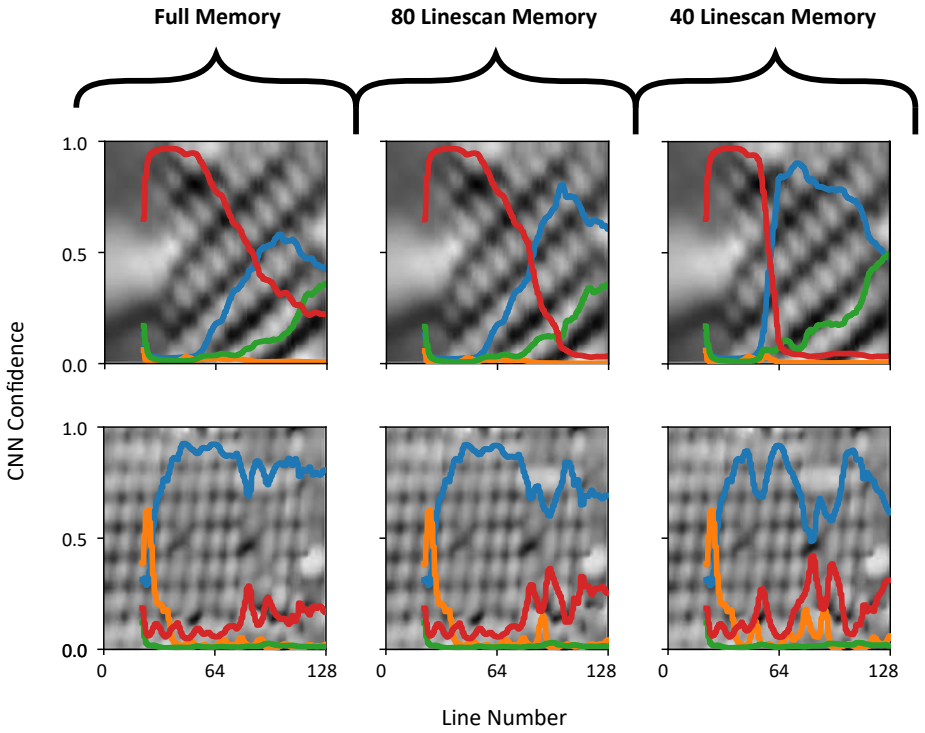

Figure 4.7.3: Figure to show how CNN assessments of STM tip state using a rolling window of multiple linescans can be affected by limiting the maximum number of linescans rolled at any one time. Each colour corresponds to a different tip state being assessed; atoms (yellow), asymmetries/dimers (blue), rows (green), generic defects (red). Scan parameters: (Top, 4x4 nm): I=10pA, V=1.6V, T=300K, (Bottom, 7x7 nm): I=150pA, V=-2.1V, T=300K

4.8 Conclusion

In Section 4.4 we have successfully trained CNNs capable of classifying numerous desirable and undesirable STM tip states for multiple surfaces. We achieve significantly greater all-round performance than other supervised learning techniques[22], and an even stronger ability to differentiate good and bad tip apices. The protocol is also likely applicable to a broad range of other SPM techniques, given the relative similarity of images produced by these methods.

In Section 4.5, by comparing a variety of methods based around a common VGG network we have successfully demonstrated that STM images of the H:Si(100) surface can be accurately assessed using partial scans. In Sections 4.6 and 4.7, we then demonstrated how this can be applied and verified using unseen data on a real system in real time. As such, only a few lines from a typical 128x128 scan are now required to assess the tip, which is a fraction of the data required by previous CNN assessment protocols. Given that the majority of the time spent maintaining SPM tips is spent acquiring data, a "hybrid" approach combining individual linescans and LRCN prediction would speed up CNN routines by approximately 100 times. This allows for state recognition in a time similar to that of current manual means, thus making it practical for everyday use. However, given that the states considered only apply to the H:Si(100) surface, new datasets and networks must be manually created and trained for each surface, making this strategy non-applicable to poorly understood surfaces.

Relative to a full-size network, we find that similar or better performance can be achieved with less data by creating a small window of multiple linescans, and adding an LSTM layer to make predictions as the window is rolled over time. Furthermore, we qualitatively demonstrate that the use of partial linescans allows tip changes to be detected without the need for a secondary network. We also show that this method allows for the detection of images in which tip changes cause multiple tip states to be present, alongside their relative position in the image.

However, there are a number of limitations to these approaches which limit its general applicability. Importantly, each trained ensemble is only applicable to a single surface (and in turn requires large amounts of training data of each surface). We also find that without significantly expanded datasets, not all surfaces are equally suitable for CNN classification. New datasets must therefore be manually created for each surface studied. This not only makes practical implementations of automatic recognition on other surfaces time-consuming and inconvenient, but also requires the surface to be well studied in advance. It would therefore be non-trivial to use this protocol to explore STM tip states of previously unexplored surfaces. Further, the low number of human classifiers was also problematic. Were more human classifiers available, the networks should have been trained on the entire multi-label dataset, and then scored based on a cross-entropy of average classifications. Performance could also be improved further with the addition of more training data.

5 Automated Selection of STM Tip State

5.1 Introduction (The Pain with Probes)

While in Chapter 4 we formed a CNN system capable of *assessing* a specific tip state, it follows that we should be able to *acquire* that state. Conventional supervised neural networks may perform incredibly well at predictive and classification tasks, but perform incredibly poorly when it comes to intelligently creating a series of actions based on its predictions. Indeed, this is a common situation; reading a textbook and passing an exam does not make a doctor fit to treat patients. They may understand the relationships between symptom and diagnosis, but they lack appreciation of the entire treatment process. The focus of this Chapter will therefore be with reinforcement learning (RL) techniques, as discussed in Section 3.5, which have been shown to perform extremely well at finding strategies not just old computer games[16,17], but also at medical treatment planning[156], minimising traffic congestion[157], and robotic manipulation[158].

Indeed, we are not the first to attempt[98,159,160] RL with SPM. Notably, in 2020 Krull et al[98]. were able to achieve molecular resolution of MgPc under liquid helium. (We note that on average, 10 'random actions' or 7 human actions were sufficient to achieve molecular resolution, whereas in our case it is effectively impossible to achieve atomic resolution through performing random actions, let alone so quickly, and so a straight RL approach is not viable in our case.) An outline of their 'DeepSPM' system is shown in Figure 5.1.1. At its core, the basic logic is not dissimilar to what we will implement; a CNN assesses tip state, which inputs to a RL agent, which picks actions to attempt to sharpen the tip.

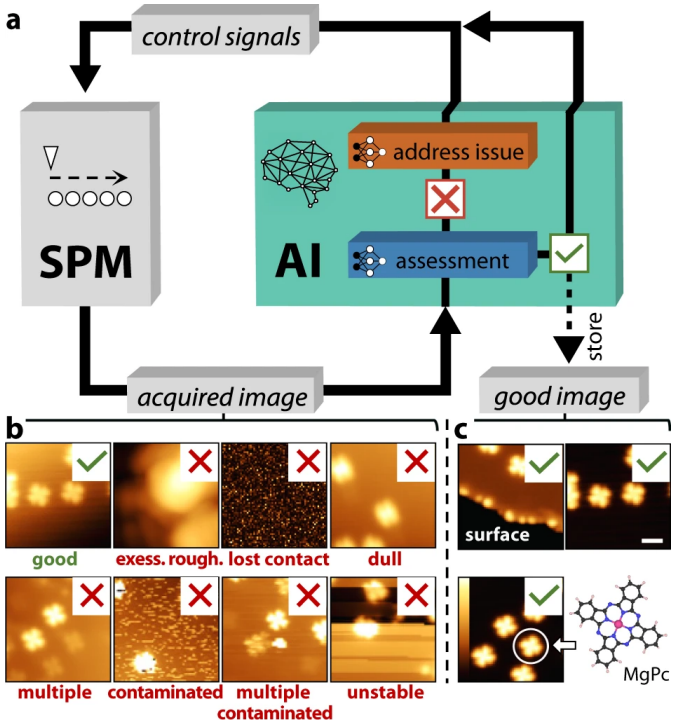

Figure 5.1.1: Krull et al.'s implementation of a deep learning agent to acquire sharp imaging resolution of MgPc. Image from Krull et al., 2020[98] is licensed under a CC BY 4.0 License.

5.2 Building a Tip-Sharpening Environment

5.2.1 Observations

To build an AI agent that can intelligently acquire an STM tip state, we must obviously first provide some input information. As discussed in Section 3.5, the Markovian assumption means that inputs should encode the entire system state. However, stochasticity and non-fully understood interactions means that we only have a partially observable markovian, and so we cannot fully meet this requirement. Complicating matters further still, we need to strike a balance between maximising exposure of the system state and keeping the learning task simple. Along with the CNN assessments from Chapter 4, we therefore supplied a minimal number of observation inputs, $\mathbf{O}$, fully detailed in Table 5.2.1. To make these raw instrumental values suitable for machine learning, we must also convert them to scales of $-1 \rightarrow \mathbf{O} \rightarrow 1$

Index	Observation Description
0	Stage of Downsizing
1	Number of Linescans Recorded
2	Scanline Range
3	Scanline Standard Deviation
4	Scanline Cosine Similarity
5	Linescan CNN (Sharp)
6	Windowed CNN (User Input Desired State)
7	Windowed CNN (Generic Defect)
8	Last Action

Table 5.2.1: Observations input to AI agent, and their related numeric encoding. These observations must encode all of the information needed to make a decision about how best to sharpen an STM tip.

Some of these actions strongly relate to the quality of the tip and are obvious, such as the range and standard deviation of the line profile, and the CNN assessments from Chapter 4. However, as it is easier to mimic human behaviour than to form a fully blind policy, we included indirect information. Typically, a human performs intense tip conditioning over a large area, and once they feel the tip is decent, they 'hone in' on a flat sub-area. Because our system downsizes the scan area in discrete steps for simplicity (as discussed below), for our agent to mimic this behaviour we therefore include the stage of downsizing.

We also further violate the Markovian assumption from Subsection 3.5.1 because of the time dependence of the system state; changes in the state should only arise from the last action, not the last state itself. Besides observing the self-similarity and range of the tip height of each individual scanline, another excellent way of determining the quality of an STM tip is to observe its stability over time. However, in RL the system state at each timestep is mathematically independent of one another. As such, we not only calculate the cosine similarity between the i^{th} and $(i-1)^{th}$ linescans, S_C, defined as

$$S_C = \frac{\mathbf{Z_i} \cdot \mathbf{Z_{i-1}}}{|\mathbf{Z_i}|^2 |\mathbf{Z_{i-1}}|^2}, \qquad \{5.1\}$$

but also encode even more time dependence into the state through the use of 'frame stacking'. To do this, we simply stitch multiple timesteps worth of observations together. While crude,

this method has been shown to result in significant performance improvements with even the most basic of tasks such as pong (as it allows deduction of momentum). As such, for a stacking of 20, we would not just input the 9 observations at time t, but also the $19 \times 9 = 171$ before it. Because this increases the complexity massively, we are limited by how many frames we can stack. We therefore opted for an arbitrary stacking of 20.

Finally, because human operators typically have an inbuilt level of patience; they will try less drastic measures first, before becoming more destructive with the tip if no change is seen. We therefore inputted the last performed action from the previous timestep, which also encourages emulating patterns of actions, such as pulsing followed by coarse moving.

5.2.2 Actions

Now that we have inputs to our agent, we must define its actionable outputs, **A**. Whilst for the previous case of supervised classification we output the confidence of the tip being in a given state, in RL we output the confidence of a given action leading to a positive outcome (i.e. maximising reward over time). While some IRL algorithms allow continuous outputs, for simplicity we attempted to opt for a small number of 11 discrete actions, as described in Table 5.2.2. It should also be particularly noted that doing nothing' is an explicitly defined action; a point we will return to later.

Index	Action Description
0	Do Nothing
1	Coarse Move Scan Window
2	Reduce Scan Area
3	4V Tip Pulse
4	8V Tip Pulse
5	0.5nm Tip Press
6	End Episode
7	Absolute Setpoint Current -0.1 nA
8	Absolute Setpoint Current +0.1 nA
9	Absolute Tip Voltage -0.1 V
10	Absolute Tip Voltage +0.1 V

Table 5.2.2: Table showing how the numerical output of an (inverse) reinforcement learning agent can be related to a discrete set of actions used to improve an STM tip.

Although some of these actions, such as performing tip pulses and nudging scanning bias, are simple and self explanatory, others, such as moving the scan window, are much more complex and routine-like. While we could increase the number of actions in the hope of the agent learning an optimum sub-routine, it is far simpler to hide this complexity from the network, and instead have the sub-routine be hard-coded as an algorithm. For example, when the agent selects action 1 to coarse move the scan window, a small sub-routine moves the scan window in a snaking zig-zag, then resets the internal counters, CNNs and scan window. The agent does not understand nor does not need to understand where it is coarse moving to and how the internal logic is preserved, as far as it is aware it is just doing one single thing,

in the same way it nudges the tip bias. Additionally, in future this would allow for other routines, such as defect avoidance neural networks[23] to be implemented, without the need for retraining the entire system.

In a similar vein, as touched upon earlier, human behaviour is to 'hone in' on a flat scan area by reducing the scan area within the bounds of the current scan. Again, we can abstract a lot of this from our agent, allowing it to step through a set of fixed scan widths; 75nm, 40nm, 20nm, and 7nm. To find the specific location to move to, we follow Wooley[8] et al., and simply move to the flattest sub-area of the current scan as determined by its standard deviation. However, there are a huge number of sub-areas to consider; reducing the scan area of a 512x512 scan from 75x75nm to 7x7nm would require $(512 \cdot \frac{75-7}{75})^2 \approx 215000$ calculations of standard deviation. Instead, we employ a Hilbert[161] space filling curve, as shown in Figure 5.2.1. By converting between real-space and pixel co-ordinates, and reducing the search space into a maximal number of overlapping boxes centred at a minimal number of co-ordinates, we can reduce the number of standard deviation calculations by 4 orders of magnitude while still finding a highly flat area. Given that the scans are normally incomplete when we/the agent elects to downscale, some of these boxes will partially lack information. While we fully ignore any boxes without any information, we weight partially filled boxes by the percentage 'filled' they are, allowing us to move to an area only partially scanned if it is much flatter than anywhere other potential sites. Again, all of this process is abstracted from the agent; it merely elects to 'downsize'. Indeed, if the coarse movement action is selected, the downsizing will be reset, and a scan size of 75×75nm set.

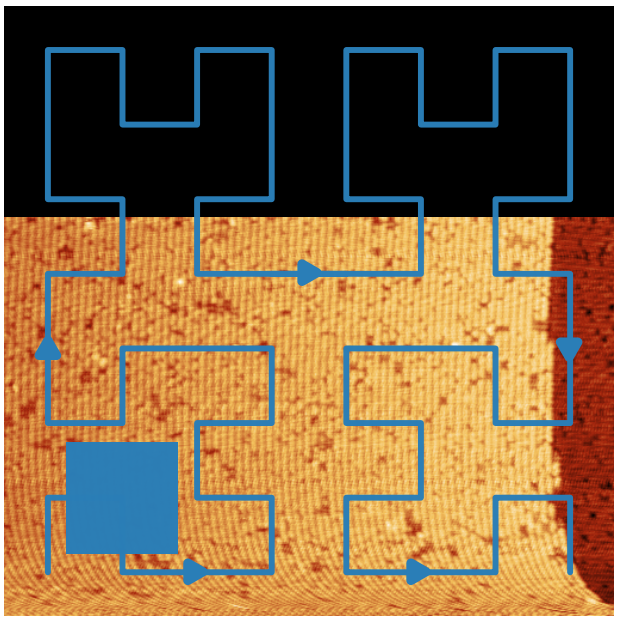

Figure 5.2.1: Reducing scan size by using a Hilbert space filling curve to find the lowest standard deviations of a minimum number of maximally overlapping sub-regions.

5.2.3 Reward

While some agents we consider later in Subsection 5.3.2 either do not use a reward, or attempt to infer one from the expert demonstrations, others base their decisions on expected, pre-calculated, reward, and so it is imperative to carefully consider how we reward and punish our network, R. The ultimate aim of RL strategies is to maximise reward over an episode, and so a nonsensical reward will lead to nonsensical behaviours. Unfortunately, we again run into difficulties. Determining reward is easy for a game such as pong ($+1$ if the agent scores, -1 if it is scored against, 0 otherwise)[16,106], but for a subjective task such as STM tip improvement we are forced to be more arbitrary and with *less clear correlation of state, chosen action, and reward*.

When forming our equation, we only use the observations $-1 \rightarrow \mathbf{O}_x \rightarrow 1$ from Table 5.2.1, as otherwise the agent has no way of deducing what is making it be rewarded or punished. This led us to arbitrarily define

$$R = \mathbf{O}_0 \left[\begin{cases} 0 & if\ \mathbf{A} = \mathbf{A}_0 \\ -1 & \text{otherwise} \end{cases} \right] + \mathbf{O}_1 + \mathbf{O}_2 + \mathbf{O}_3 + 3\mathbf{O}_4 + \mathbf{O}_5 + \left\{ \begin{matrix} \mathbf{O}_6 - \mathbf{O}_7 & \text{scan area} \leq 20\text{nm} \\ 0 & \text{otherwise} \end{matrix} \right\} \Bigg]. \quad \{5.2\}$$

While complex at first, this equation can be broken down simply: First, if the agent is performing an action that could adjust the tip, it receives a penalty of -1. This aims to discourage the agent from constantly attempting to reshape the tip, which would likely prevent any tip from being successfully optimised. We then reward or punish between -1 and 1 according to the range, standard deviation, and self-similarity of the scanline height. Finally, if the scan area has been sufficiently downsized to be smaller than 20×20nm (at which the CNNs become more reliable due to there being substantially more training data at this lengthscale), we also reward or punish depending on the linescan and window CNN confidences of the tip being in a desirable state.

5.2.4 Ending an Episode

Also shown in Table 5.2.2 is an abstracted action called 'end episode', which warrants further discussion. As discussed in Chapter 3, an episode is simply a number of connected timesteps to attempt to maximise reward over. For Pong, an episode could end when a point is scored, or for space invaders when the player runs out of lives. However, for STM tip sharpening ending an episode is quite arbitrary. At what exact point does an STM tip become definitively 'improved'? After all, in reality tip sharpening is an ongoing process, and yet for RL or IRL to work we *must* have a *clear* and *defined* episode end point.

Typically, there are two schools of thought here. We could either end the episode depending on state, or depending on time. Stateful ending would be the simplest; we would end our episode upon reaching a threshold in the CNN assessment of a desired tip state. However, we would immediately run into issues; a human will normally wait to observe stability (i.e.

'do nothing' for several timesteps) before finding that threshold meaningful, especially when a scan may be subject to creep. This would be yet another complex sub-routine our agent would have to learn to perform, especially when recalling from Subsection 4.5.5 that the 'rolling window' CNN has a $W = 20$ linescan buffer that must first be filled before full state assessments can be made. Alternatively, we could instead have a 'time-limit' by setting an upper-limit on the number of linescans the agent has to improve the tip. This would not only make the deployed network less practical, but it has been well discussed in the literature that presence of a hidden time-dependence means this is likely to come with a performance penalty[106].

As such, we created an 'end episode' action to abstract away the subroutine of waiting for creep to subside, and waiting a number of linescans before forming a full assessment. When selecting this action, the smallest available flat area is found and moved to, an entire 512x512 scan is taken to remove creep, then another 512x512 scan is taken, all 512 averaged into one, and a reward/punishment calculated as appropriate. By including the performed action as an input observation, we effectively have our 'statewise' ending. While this admittedly makes the system not truly 'real-time', it provides a more-than-good enough approximation given the limitations of RL.

5.2.5 Closing the Loop

Finally, we need to interface our agent with the Omicron VT. Following the OpenAI Gym environment loop, we begin by performing an action ('do nothing' for the very first iteration), then acquire new observations and CNN assessments, convert them to a scale suitable for machine learning of $0 \rightarrow 1$, calculate reward, and repeat until episode end.

However, the need for optional human intervention means that at any time, we need to be able to 'cut-open' this loop and have actions chosen either by a human or an AI agent of our choice (which itself could come from a number of different community python packages), as well as use real-time data or existing data. Adding further complexity still, we not only need to abstract the actions discussed above, but also account for the variable position of the tip relative to the scan being taken. We therefore produced a system following the control logic of Figure 5.2.2.

5.3 Attempting Tip-Sharpening

5.3.1 Data Collection Strategy

Unfortunately, none of the common SPM controllers (including Scienta Omicron's Matrix, used in this work), store information about tip improvement actions, meaning that we do not have historic data to use to train our agent. We therefore used the nOmicron package from Subsection 4.6 and the GUI depicted in Figure 5.3.1 to collect data as we improved STM tips when scanning Si(100) and H:Si(100).

Ultimately, while we had multiple people, we opted for a consistent strategy, as we were hoping to start from a base of behaviour cloning. Our intention was to be as clear and

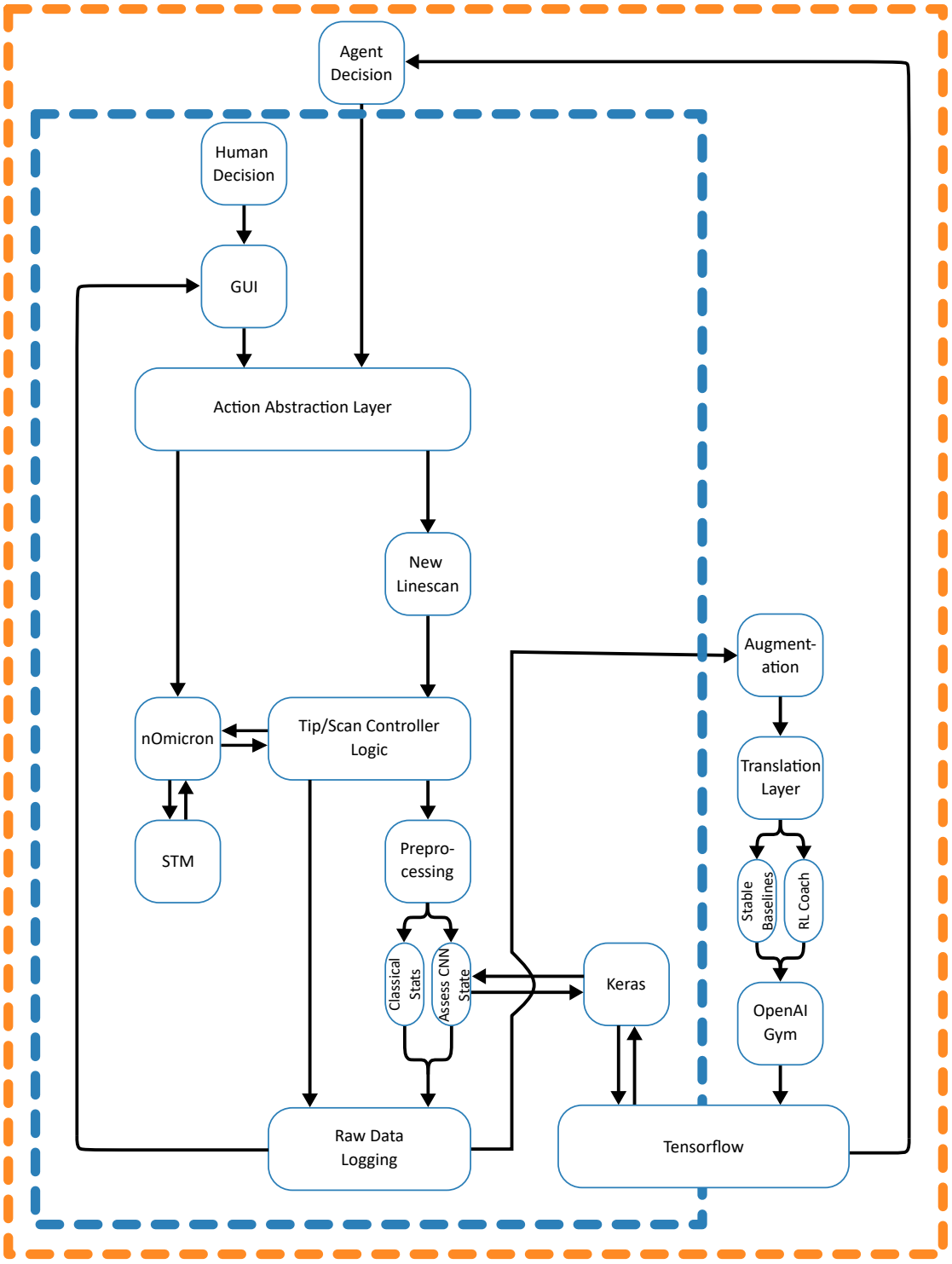

Figure 5.2.2: Control logic of a system used to recognise and alter STM tip state. Initially, input actions are chosen by a human and follows only the logic wrapped in blue. Data logged is then used to train an AI agent to run additional elements of the tool, encased in orange.

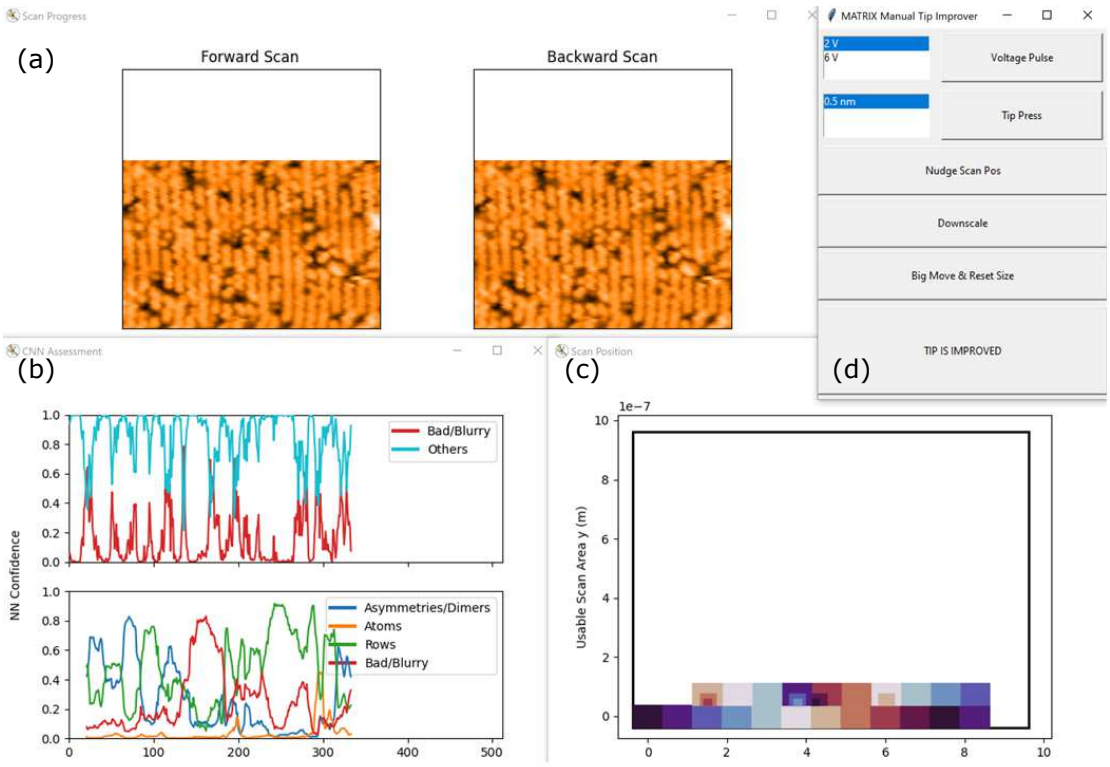

Figure 5.3.1: GUI used by a human to manually improve STM tip state. All inputs, along with various assessments of the scan, are stored, and later used to train an AI agent to do this autonomously.

repetitive in our chosen actions as possible, even if it meant oversimplifying. In particular, the highly limited temporal memory due to frame stacking meant that we had to make decisions as quickly as possible, and never wait more than a few linescans when assessing the stability of the tip. When attempting to alter the tip apex, we would pulse until the tip changed, then if it did not pulse more intensely, then move. Provided the scan height was of appropriate range, we would then immediately perform a final assessment.

However, because even simple RL agents can require on the order of 500,000 state samples (≈ 140 hours of simulated Atari gameplay)[16,162], for STM we must rely on data augmentation to significantly increase the number of transitions available to train with. Besides adding small amounts of random noise, we also changed the timing and intensity of actions, and added extra pulses/presses in clusters to further encourage the adjust-move-adjust behaviour.

5.3.2 Agent Algorithm

There are a number of different IRL algorithms, each with their own advantages and disadvantages. As this is a highly complex task with no 'known good' strategy, we considered a number of agents in an attempt to overcome different limitations of the environment.
The simplest IRL method is that of Behavioural Cloning (BC). This is effectively an expanded form of supervised learning (as in Chapter 4). It aims to reproduce the exact trajectory of

the human demonstrator, with no use of the reward function. Its aim is therefore to copy what the expert would do as closely as possible. The advantage of this method is that it can be trained fully offline. However, because it cannot explore the environment, then if the initial training data does not completely and exhaustively cover the state space, it will very quickly become ineffective.

Other IRL methods include those of DDQN (as used by Krull et al.[98]), are based off of traditional RL Learning. After initially exploring the environment with random actions, the DDQN agent picks the action that it expects will lead to maximum future reward (or explores further by picking an action randomly). It then compares expected reward to actual reward, and updates accordingly. Its aim is therefore to optimise the value of reward. In IRL, the only difference is that instead of performing random actions to initialise its expectations, it uses the expert demonstrations. This is crucial in our case, as performing random actions would quickly get the tip stuck into a state where drastic action is needed to recover it, and so it would never make progress at the task, and therefore never learn.

In future, it would be interesting to consider not only apprenticeship learning, but human-in-the-loop agents, in which an agent learns online, but can only take an action if it is approved by a human.

5.3.3 Observed Behaviours & Future Improvements

When using Behavioural Cloning (BC), in limited situations we were able to observe the agent replicating the 'pulse-pulse-move' behaviour that would eventually lead to a tip being sharpened. With both BC and DDQN, we were also routinely able to emulate the correct behaviour of 'doing nothing' most of the time, which would otherwise render the task unsolvable. However, we were unable to produce a good demonstration of tip improvement that could not be attributed to a fluke result.

This is, however, unsurprising. The massive state space meant that even with nearing 100 runs of manual tip improvement, the agent was nowhere near being exposed to the full state markovian. Further, given that hundreds of thousands of steps can be needed to train in a simulated environment, our DDQN agent was not given enough time to learn from. One good future strategy could be to approximate a simulation of an STM environment by switching between pre-scanned images, that could change depending on what action the agent has taken. After initial learning in this simulated environment, the agent may be more resilient to stochasticity in real-time datastreams.

Further, as seen in Figure 5.3.1(a) and (b), the CNN classifiers often exhibited variable performance at larger scan sizes above 7x7nm. This becomes understandable when considering the origin of the dataset from Chapter 4. While this dataset had scan sizes ranging from 3×3 nm^2 to 80×80 nm^2, the overwhelming majority were of size 4×4 nm^2. The scan window was also often manually repositioned to avoid step edges, negate creep, and have the dimer pairs running in similar directions. The networks therefore had less data provided from general scanning conditions than originally assumed, and so would inevitably perform poorly at large scan sizes of truly random direction, and with the presence of defects and step edges.

When following the old adage[10] of 'rubbish in, rubbish out', this would inevitably hamper the agent's ability to learn to solve its task. As such, the database of scans should be adapted in future. Indeed, as of the time of writing we have identified and are working the classify a new total of 17,818 complete scans of H:Si(100) from the research group dating back to 2007. Further, when resolution allows, we can also pan/crop images to artificially produce scans of different scan sizes, which can then be downsampled to a common resolution.

5.4 Envisioning A Fully Automated Experiment

While experimental issues prevented the deployment of an IRL agent able to automatically sharpen an STM tip, it is still important to be aware of how close we are to viable, fully automated STM experimentation. One area that could massively benefit from full automation is, naturally, atomic fabrication. Indeed, hydrogen lithography itself has a significant role to play in the next wave of STM-enabled discoveries, already allowing for the creation of simple structures and logic gates with atomic precision[23,129,130,163]. Here, by ramping the scanning voltage[133] over an atomic site, we can apply an energetic dose large enough to depassivate a hydrogen atom from the H:Si(100) surface (discussed extensively throughout this thesis). This leaves a highly reactive dangling bond at the depassivation site, which is imaged as a bright feature under STM. Through careful selection of parameters and rastering of the tip, it is possible to 'draw' relatively complex shapes and structures, as shown in Figure 5.4.1.

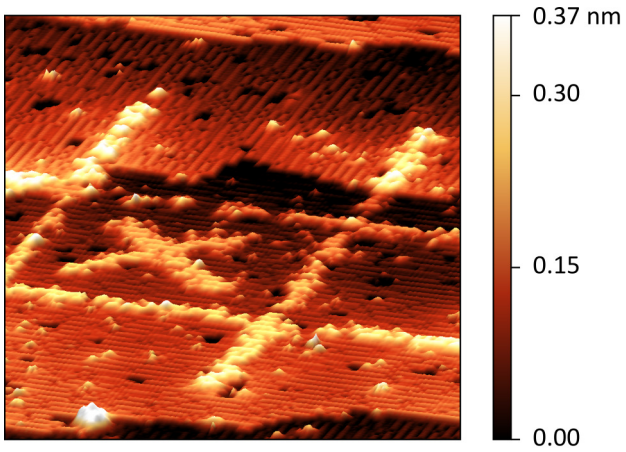

Figure 5.4.1: Faux-3D view of an atomic patterning experiment. Taken at T=300K, V=-1.63V, I=0.5nA, lithography settings of V=2V, I=1.5nA, raster time = 20ms @ 256 points.

Furthermore, not only do microscopists want to build these shapes and structures automatically, but it is easy to appreciate that they may wish to interact with the probe differently depending on the progress of their experiment. This sort of intelligent policy creation is the raison d'etre of RL, and so as a toy example of a fully automated experiment, we showcase how tic-tac-toe, a game widely studied by the ML community[10,164], can be played automatically under STM using hydrogen lithography.

To do this, we can combine several of the machine learning techniques discussed in this thesis. First and foremost, we can use the tip state assessment CNN from Chapter 4 on a

clean, reserved area of the experimental window. Ideally, such an area could be maintained (and found) through use of the tip state improvement RL agent discussed above (although limitations discussed above meant that this was not possible at this moment in time). RL can also be used to play the game aswell; by pitting two agents against each other in a simulated environment, both will eventually learn to play the game. The agent can observe the current game board, and elect to place a piece at any given position, at which point similar abstractions to those discussed in Subsection 5.2.2 programmatically draw the appropriate shape at the appropriate position. Once appropriate depassivation parameters were manually determined, games could be played, as shown in Figure 5.4.2.

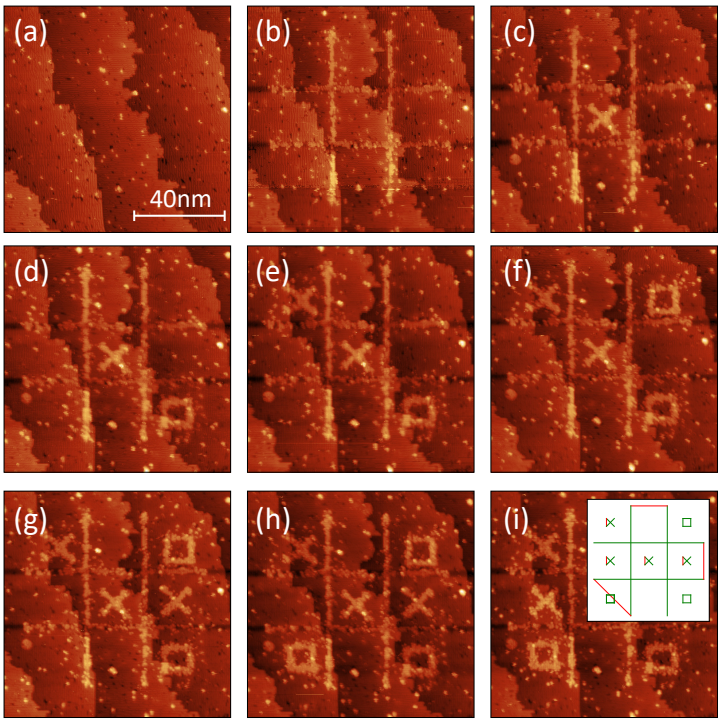

Figure 5.4.2: Playing noughts and crosses under STM in an experiment controlled by reinforcement learning. **(a)** to **(i)** show the gradual progression of the game, with the path of the tip during depassivation shown inset. Taken at T=300K, V=-2.25V, I=250pA, lithography settings of V=4.2V, I=1.5nA, raster time = 20ms @ 256 points.

While seemingly basic, there are many possibilities to extend this work. Besides using the tip assessment agent to constantly observe a non-passivated region of the surface, this could then be tied to the tip improvement RL agent. Further, at present the system assumes that the depassivated line is always drawn perfectly; in future RL agents could be combined with automated pre-processing routines to determine optimal patterning parameters.

Ultimately, it is easy to imagine how this example could be expanded to a real autonomous manufacturing environment. Indeed, these are tasks which the AI/SPM community[23,67,68,96,110,165–167], albeit disparately, have made great recent progress towards automating. Refining, and ultimately combining of, these strategies, is a clear and obvious direction for the community to take over the next few years.

5.5 Conclusion

It is clear that integrating reinforcement learning is one of the most frustrating, yet important, hurdles to be overcome in the development of automated STM. Not only does it provide a means to not just detect tip state changes, but *correct* them, and even go a step further by using additional RL agents to control more complex experiments. Looking beyond our toy example of noughts-and-crosses, it is easy to imagine performing fully automated patterning experiments, for example.

It is almost certainly a simple matter of time to refine these currently disparate systems, at which point they could be adapted, combined and applied to allow for automated SPM experimentation without any need for human babysitting.

6 Automated Identification of Self-Organized AFM Nanostructures

6.1 Introduction

There has been a recent flurry of studies applying machine learning to scanning probe microscopy (SPM), ranging from distinguishing otherwise indistinguishable data[65], segmenting and analysing surface features and defects[23,165,167–169], and assessing probe quality[96,97]. While these studies will undoubtedly allow us to collect more, higher quality data *in the future*, we are only beginning to find ways of better using[69,170] *existing* data left unanalysed on old hard drives, CDs and even floppy discs[171]. Indeed, while one of the greatest advantages of atomic force microscopy (AFM) is its ability to produce huge numbers of scans on a variety of distinct surfaces, the need to manually select data to analyse leads to much of it being under-used at best. Further, the requirement to manually pre-process data raises uncomfortable questions; at what point could our desire for aesthetic quality of publishable figures cause us to introduce unphysical information?

For example, consider the deposition of nanoparticles suspended in liquid onto a substrate. These non-equilibrium 'dewetting' experiments have a richly complex surface chemistry at their solid-liquid-air interfaces[24,172], resulting in a surprisingly distinct array of unique, self-organised nanoparticle structures being formed at multiple length-scales as the structure is driven to equilibrium by the evaporation of the liquid (hence the name 'dewetting'). While these experiments can be performed *en masse*, every image used for analysis must be hand-separated to infer the discrete layers of substrate, liquid, and nanoparticles. Indeed, the specific thresholds used can result in extremely different conclusions being formed about an individual experiment.

In this chapter, we demonstrate that it is possible to use simulated AFM images of self-organised nanoparticle assemblies[172,173] as automatically labelled training data for a neural network, which then correctly generalises to real AFM scans. We employ an automated, optimised pre-processing routine for real images of these specific structures, which is then combined with a denoising autoencoder to provide effective, automated binarisation of real images. We then combine these systems to quickly find experimental AFM images of these specific structures in an example dataset of 5519 scans of multiple structures and surfaces collected at a wide range of scan qualities and sizes from $500nm$ to $90\mu m$. We also demonstrate why this method can be readily adapted to other structures and experimental datasets.

6.2 Monte Carlo Methods

6.2.1 Basic Model

One of the most fundamental simulational methods in modern physics is that of the Monte Carlo model. Whilst randomness may at first seem problematic when looking to interrogate a complex physical problem, Monte Carlo methods inherently embrace it.

For a good example of this we can look to the origins of the technique and the Casino de

Monte Carlo (to which the technique is named after). The statistician Stanislaw Ulam in 1946 was attempting to determine the probability of one of its hardest 'games'; Demon[174], a variant of solitaire. He was trying to determine the chance that a game of Demon "laid out with 52 cards will come out successfully" [174], yet was struggling to derive such a solution analytically. Instead, he considered simulating 100 games at random (complex, given computing power at the time) and determining probability based on the number of winning hands.

This brute force approach based on the laws of large numbers was instrumental to the development of the Manhattan project Ulam was working on at the time; the creation of the world's first nuclear bomb. This technique now underpins a staggering array of fields, from medical physics[175] to financial risk management[176] and even road planning[177].

6.2.2 The Ising Model

Looking closer to nanoscience and surface science, it is extremely common to use Monte Carlo techniques to investigate time-dependent dynamics, such as the growth of structures on a surface, itself often underpinned by the Ising Model, developed in 1925 by Ernst Ising[174] to describe ferromagnetic systems. Here, we start with a grid of L points in each dimension, each with spin $\sigma = \pm 1$. The total energy of the system, E, can be found by summing all of the interaction energies between its i^{th} and adjacent j^{th} sites with the Hamiltonian[174]

$$E = \frac{-J}{2} \sum_{i,j} \sigma_i \sigma_j, \qquad \{6.1\}$$

where J represents the energetic favourability of the interaction. If the spins are aligned (i.e. $\sigma_i = \sigma_j$), the interaction is energetically favourable, so has energy $-J$. If the spins oppose (i.e. $\sigma_i = -\sigma_j$), there is an associated energetic 'cost' of $+J$.

Given that we can assume that the energy barriers of these sites, ΔE, follows the common Maxwell-Boltzmann probability distribution, it follows that the probability of a given configuration of spins at equilibrium is

$$P = \frac{e^{\frac{-\Delta E}{k_B T}}}{\sum_\sigma e^{\frac{-\Delta E}{k_B T}}}, \qquad \{6.2\}$$

where T is the temperature of the system, and k_B the Boltzmann constant.

Importantly, this system does not quite evolve continuously, there instead exists a discrete phase boundary where the macro-evolution of the system changes discontinuously. Unsurprisingly, these boundaries are often of the most experimental interest. For the Ising model, this is defined by the critical temperature, T_C, at which point the system changes from a state dominated by ferromagnetic interactions to those dominated by antiferromagnetic interactions. For the 2D model, its analytical solution (first found in 1935[178]) takes over 30 sheets of paper, while no analytical solution for the 3D model has ever been derived. However, these solutions are routinely found using Monte Carlo techniques, the coding of which now often forms part of many an undergraduate syllabus.

Regardless, the Ising model in and of itself is of no stranger to machine learning. For example, in 2019 Burzawa et al. were able to successfully use supervised machine learning of the type described in Section 3.2 to distinguish between simulations underpinned by either 2D or 3D Ising models (or a non-interacting 'percolation' model)[148].

6.2.3 Metropolis Acceptance

Another key figure, and Ulam's colleague, in the Manhattan project was Nicholas Metropolis. In his ground-breaking 1953 paper on evaluating equations of state using 'fast computing machines[179]', he described an algorithm (later generalised to any system in 1970 by Wilfred Hastings, hence this also being known as the 'Metropolis-Hastings Algorithm') to computationally evolve the system to its equilibrium state. The algorithm is as follows:

Algorithm: Metropolis-Hastings Acceptance

```
Result: System in equilibrium
Initialise lattice grid with random σ;
while not in equilibrium do
    Select random grid site;
    Invert J_0 → J_1;
    ΔE = E_1 − E_0;
    if ΔE < 0 then
        Accept J_1;
    else
        Generate random number, η;
        if η < exp(−ΔE/k_B T) then
            Accept J_1;
        else
            Reject J_1 and restore J_0;
        end
    end
end
```

Figure 6.2.1: The Metropolis-Hastings Accept/Reject algorithm, in terms of the Ising model.

For the 2D Ising model, the Metropolis algorithm at different temperatures evolves states such as those shown in Figure 6.2.1.

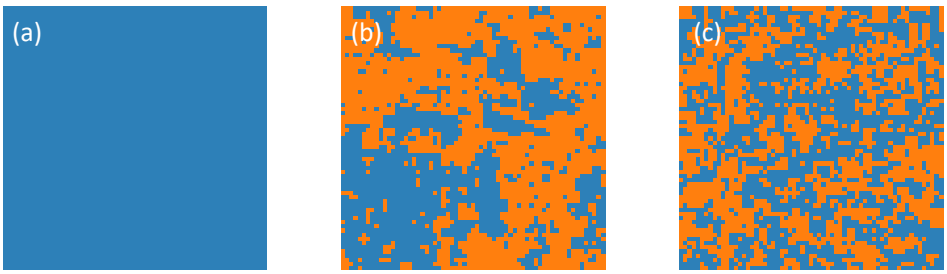

Figure 6.2.2: Equilibrium states of three different Ising model simulations at **(a)** $T << T_C$, **(b)** $T \approx T_C$, **(c)** $T >> T_C$. Each grid site can have spin $+1$ (orange), or -1 (blue).

6.3 Self-Organised Nanoparticle Simulations

6.3.1 Rabani Model

Following on from Ge and Brus[180], in 2003 Rabani et al[24], developed a simple Monte Carlo model that very accurately reproduces many solvent-evaporation experiments performed under AFM[149,172,181]. In these dewetting experiments, a controlled fraction, C, of nanoparticles of a material such as Au are deposited onto a surface containing a substrate such as Si and a liquid solvent such as alkanethiol[172]. Thermal energy allows the solvent to evaporate, leaving behind ordered structures such as that shown in Figure 6.3.1.

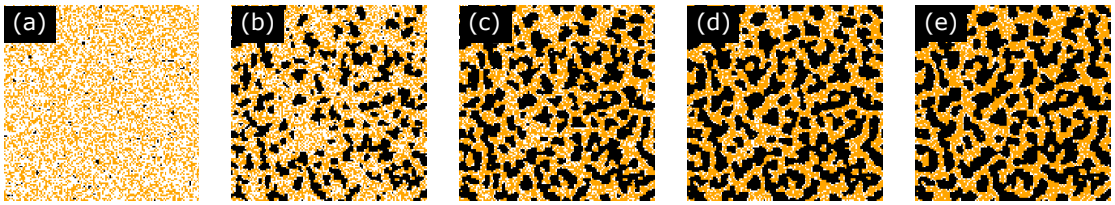

Figure 6.3.1: The evolution of a single Rabani et al. Monte Carlo simulation as it takes an initial grid of substrate (black), liquid solvent (white) and nanoparticles (orange) and grows to equilibrium after **(a)** 0 steps, **(b)** 25 steps, **(c)** 50 steps, **(d)** 75 steps, **(e)** 100 steps. This simulation began with starting parameters of $k_BT = 0.35$, $\mu = 3$, $MR = 1$, $C = 0.4$, $\epsilon_{nl} = 1.5$, $\epsilon_{nn} = 2$, $L = 128$.

Instead of beginning by randomly assigning grid sites to two cell states of 'spin up' and 'spin down', the Rabani et. al model has three cell states: 'liquid', 'substrate', and 'nanoparticle'. The interactions between grid sites are then represented by the modified Hamiltonian[24]

$$E = -\epsilon_l \sum_{i,j} l_i l_j - \epsilon_n \sum_{i,j} n_i n_j - \epsilon_{nl} \sum_{i,j} n_i l_j - \mu \sum_i l_i, \qquad \{6.3\}$$

where μ is the chemical potential responsible for determining the mean density of solvent at equilibrium, and ϵ_n, ϵ_l and ϵ_{nl} are the interaction energies between nanoparticles, liquid, and nanoparticles and liquid at neighbouring sites i and j.

As the system evolves, nanoparticles (of grid size 2x2 in the Rabani et. al paper, although this has no effect on the evolution of the system so is not implemented in our code) perform a random walk. They can move up, down, left, or right, provided their new site would be surrounded by liquid solvent (i.e. 'wet'). Metropolis acceptance/rejectance is used as before to determine if the random walk is eventually performed. However, as a modification, a 'mobility ratio', MR, is introduced, allowing for a number of 'retries' if the move was originally rejected. To negate the effect of boundaries causing non-homogenous growth, the grid is given circular boundary conditions, meaning that a nanoparticle walking off the left edge of the grid will appear on the right edge of the grid, and so on.

6.3.2 Structure Morphologies & Minkowski Statistics

As part of extensions made to the Rabani et al. model, Stannard and Martin[173,182] also more thoroughly studied the different structures that can be developed by the model. Those of most relevance to this thesis are "labyrinthine" / "finger-like" when the nanoparticle growth

is worm-like/branching, "cellular" if the nanoparticles fully enclose pockets of substrate, or conversely "islands" when the nanoparticles cluster together into isolated areas. If the solvent has not yet fully evaporated, "holes"/"pores" of substrate can form on a largely "liquid" surface[173,183]. Examples of each are shown in Figure 6.3.2.

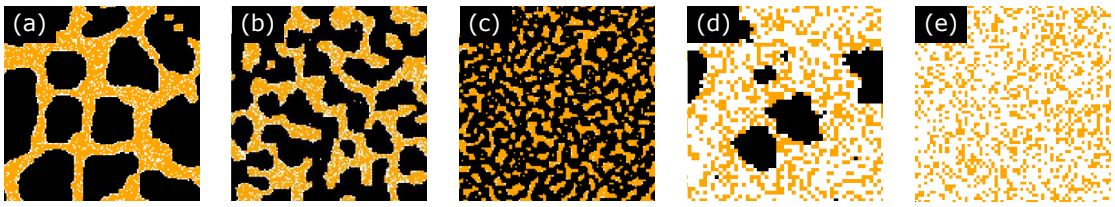

Figure 6.3.2: Examples of the distinct structures that can be formed during routine AFM dewetting imaging, with simulated materials of black (substrate), white (liquid), and orange (nanoparticle). These have been previously described as **(a)** cellular, **(b)** labyrinthine, **(c)**, islands, **(d)** holes, and **(e)** liquid.

Importantly, there is a strong relationship[173] between simulation parameters and the eventual structure developed. Particularly, by varying μ and k_BT above and below the approximate location of the spinodal line (which can be found via trial and error), we can continuously generate all of the above structures. This can be seen in Figure 6.3.3.

Further, we can combine this fact with information only deducible in simulation to reliably generate specific structures. Arbitrarily, we determined liquid structures to be those where liquid was the most abundant particle in a simulation, and holes if there were also at least a few percentage points of substrate present.

To classify cellular, labyrinthine and island structures, the Euler number of the nanoparticle layer, χ_N, in a given simulation was calculated with the equation

$$\chi_N = \frac{N - (L + S)}{N}, \qquad \{6.4\}$$

where N, L and S are the numbers of nanoparticles, liquid, and substrate in the simulation respectively. The division by N normalises χ_N to make simulations of varying size comparable to each other.

Because the Euler number quantifies the "connected-ness" of a given structure, cellular structures are more connected than labyrinths, which are more connected than islands. This can be seen in Figure 6.3.2). Each category of structure growth can therefore be mapped to a distinct range of Euler numbers, as seen in the inset of Figure 6.3.3. To prevent feature overlap and make visually training data visually distinct for the networks discussed below, an arbitrary, distinct range of Euler numbers was selected for each category.

6.3.3 Coarsening

One complication with this method of data generation is that the lengthscale/correlation length of generated features scales with simulation parameter. While the automatic labelling was scale-invariant, an image-based CNN is only scale-invariant if the training images have different feature-scales. Because the visual feature size of the simulated structures depended

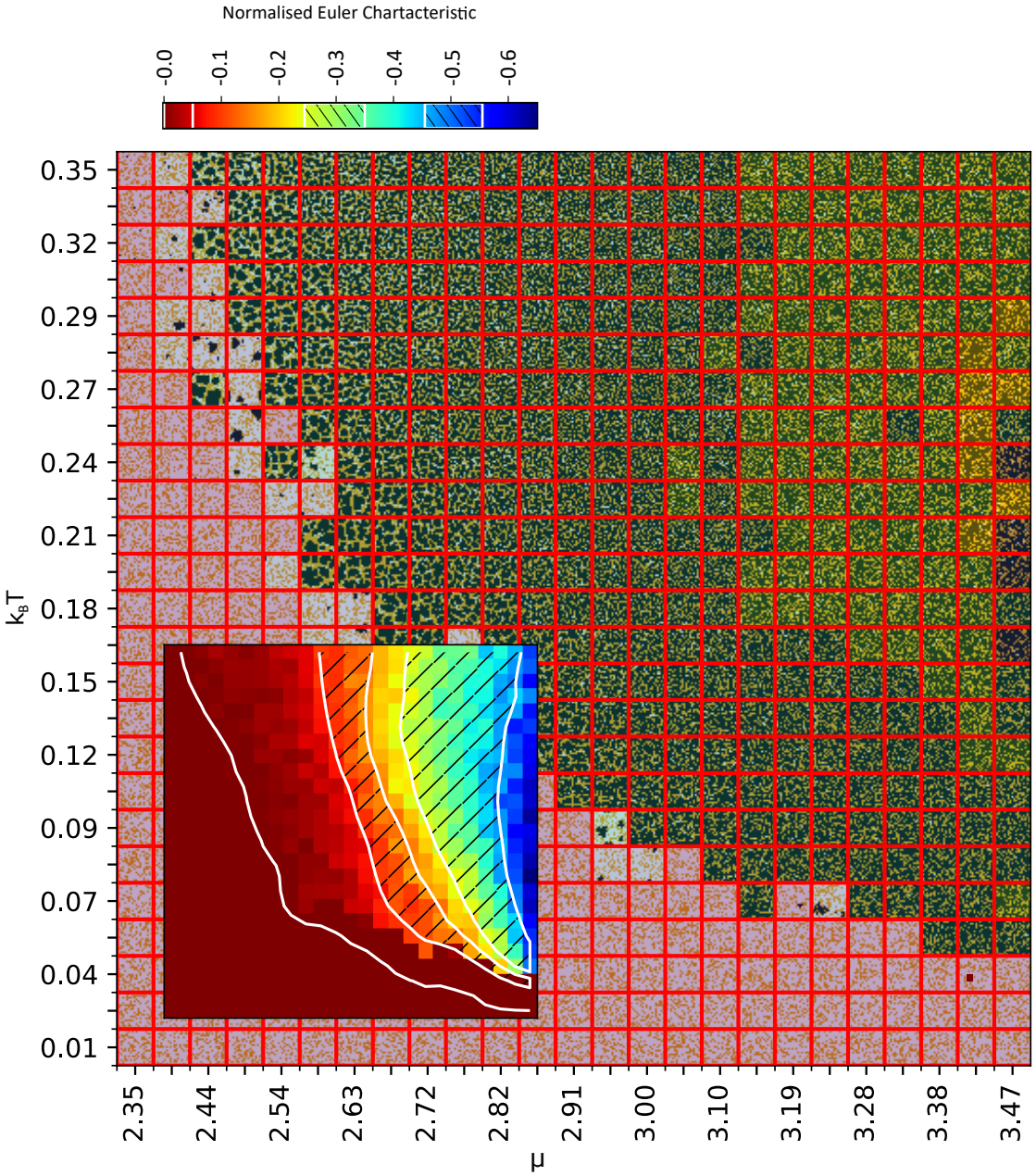

Figure 6.3.3: Figure to demonstrate how different atomic structures can be generated in a rabani Monte Carlo growth model, by varying original parameters μ and k_BT. Clear examples of these structures (except for liquid and holes, which can be determined from knowledge that the image is predominantly liquid/contains isolated blobs of substrate) can then be broadly determined from separate ranges of their normalised Euler number (inset).

on the simulation parameters, this was not the case by default. As seen in Figure 6.3.3, simulated cellular structures appeared physically larger than labyrinths, and islands larger still. This caused the classifier to often base classifications on feature size, rather than the features themselves. While this dependence on feature scale is desirable in some cases, it is not physically realistic in ours.

To overcome this issue, we exploited the coarsening of the nanoparticle structures from a kinetically hindered state towards their equilibrium configuration[24,149]. In regimes with fast liquid evaporation, simulations reach a metastable structure, and then grow in feature size with additional simulation steps as nanoparticles diffuse from regions of low average co-ordination to more highly co-ordinated sites. Larger structures therefore grow as the result of the decay of smaller features. We can therefore create different visual scales in the training data by running additional simulations for each structure type, but allowing the system to move further towards equilibrium (i.e. coarsen) by simply increasing the total simulation time. Further scale invariance was also introduced by varying simulation resolution and upscaling with nearest-neighbour interpolation to a common size of 200x200 pixels. Two examples are shown in Figure 6.3.4.

Figure 6.3.4: Examples of feature coarsening in the Rabani et al[24] monte carlo model. For a 'labyrinthine' structure as generated with the parameters $k_BT = 0.3$, $\mu = 2.9$, $MR = 3$, $C = 0.4$, $\epsilon_{nl} = 1.5$, $\epsilon_{nl} = 2$, we observe visible coarsening from **(a)** 300 to **(b)** 1500 steps. By changing to $k_BT = 0.32$, $\mu = 3.2$, and $C = 0.2$, we can generate an 'island' type image, and observe coarsening between **(c)** 100 and **(d)** 1000 steps.

6.4 Classifying Simulated Data

6.4.1 Creating Training Data

One of the biggest difficulties in employing supervised machine learning is the time-consuming task of manually labelling large training sets. Further, while scanning tunnelling microscopy (STM) and AFM scans can sometimes be simulated using density functional theory (DFT), this is only possible for relatively small system sizes, and cannot practically create data at the scale required for machine learning. While in Chapter 4, we painstakingly hand-labelled a dataset, previous authors such as Aldritt. *et al.*[65] and Burzawa[148] *et al.* used a probe-particle model involving empirical pair potentials or the Ising model, respectively, to automatically create and label orders of magnitude more training data.

For the nanoparticle assemblies of interest here, we developed a highly-optimised[184] python implementation of the Rabani *et al.*[24] Monte Carlo algorithm, which accurately reproduces experimental images of 2D nanoparticle assemblies on a variety of solid surfaces[172,173]. By using simulations for training data, we no longer need to hand label our dataset.

As a reminder, a set of distinct morphologies can be produced by the Rabani algorithm, which have previously been variously classified as[173,182] "labyrinthine" / "finger-like" when the nanoparticle growth is worm-like/branching, "cellular" if the nanoparticles fully enclose pockets of substrate, or conversely "islands" when the nanoparticles cluster together into isolated areas. If the solvent has not yet fully evaporated, "holes" / "pores" of substrate can form on a largely "liquid" surface[173,183]. Examples are shown in Figure 6.4.1.

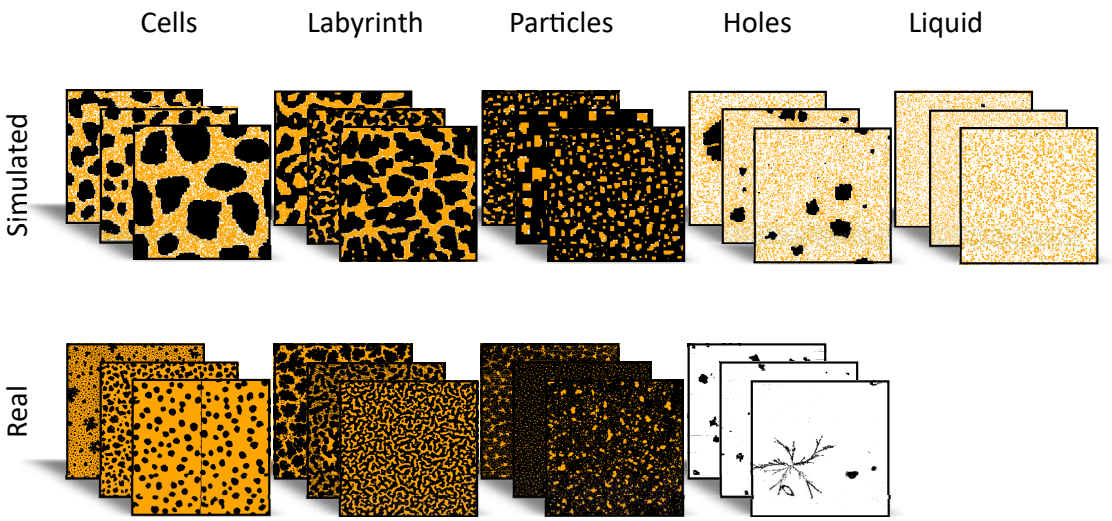

Figure 6.4.1: Examples of the distinct structures possible to be simulated and observed during routine AFM dewetting imaging, with simulated materials of black (substrate), white (liquid), and orange (nanoparticle). Because of the high similarity between simulated/real images, simulated images (top row) can be used to train a neural network, which is then capable of correctly assessing automatically binarised real images (bottom row, categories as assessed by network). This network can then be used to "hunt" for these specific structures.

To produce training data, we can also exploit the strong correlation of the initial simulation parameters with the final structure formed[173]. As such, simulations with identical input parameters will grow randomly and differently, but almost always finishing with the same type of structure.

To produce a number of structures at a scale sufficient for machine learning, code from a previous study was converted from MATLAB (as created by a previous PhD; Andrew Stannard[173]) to Python, where it could be heavily optimised using a combination of rewriting, multithreading, stopping at system equilibrium, and the Numba Python compiler[184]. On an 8 core system, the code ran approximately 200x faster, creating a suitably large dataset on the order of 10^5 images within a few hours.

6.4.2 Non-CNN Method

However, a CNN is not required to filter/classify experimental images. Given that we aimed to build a broadly applicable tool, we also considered a purely statistical classifier based on to Minkowski functions. Minkowski functions, such as perimeter, P, Euler number, χ_N, and area, A (described in Subsection 6.3.2) have been successfully used for many years for statistical analysis of a wide range of images[185–187], and can be calculated quickly by a large variety of software packages[142,188]. They are therefore a natural starting point for a generally applicable classifier. Instead of inputting an image, we instead extract scale-invariant 1D statistics about each simulated image, and use this information as regressive input to a classifier, alongside the automatic labels from before. While many regressive classifiers exist (including neural networks), for simplicity we only considered multinomial logistic regression[142].

Indeed, this method has several advantages over a visual CNN approach. Besides being computationally cheaper to train, significantly less input data is required. Given that some systems can only be accurately simulated with computationally expensive techniques such as DFT, generating data at the scale required for a CNN may not be possible. A logistic regression, meanwhile, only requires approximately $10(k-1)(m-1)$ training samples, (where k is the number of output classes and m the number of input variables)[189]. In our case, we successfully trained on simulated data with less than 500 images.

Further, this approach is naturally scale-invariant, unlike a CNN, which is only as scale-invariant as its input simulations. With statistical inputs, we do not need to produce scale-invariant simulations, only scale-invariant statistics about those simulations. This may be of benefit to situations where scale-invariant data generation is non-trivial. However, any statistics used must be very carefully selected for the given task, making the method far more sensitive when applying the model to new datasets, both simulated and real-world.

Further, while it is possible to directly classify simulations using the normalised Euler number, this was not an effective strategy on real-world images. This is likely due to the extremely sensitivity to instrumental noise and heavy dependence on the arbitrary ranges of Euler number used to define each category discussed in Subsection 6.3.2. Also, the "holes" cannot be described by Euler numbers, so had to be labelled using information only fully known in simulations (if a simulation had an abundance of liquid and isolated pockets of substrate), and so we cannot label "holes" in this manner for real images using these Minkowski functions as-is. This would require knowledge of all three discrete particle levels of substrate/liquid/nanoparticle, which is impractical as it is often difficult to trinarise real-world images, so we can only infer what material a pixel belongs to, unlike in a simulation where we explicitly know this. While correct classification is broadly possible on high quality images by assuming that all liquid in a given real-world image has evaporated and binarising instead, a significant number of false-positive results would be produced on poor quality scans/structures that should otherwise be filtered out.

Given that applying this method to new situations requires increasingly sophisticated particle statistics which an investigation may not possess. This becomes harder still in a mixed dataset containing unwanted images, and un-applicable in cases where labelling information is only accessible in simulations. As a result, regression is far less generalisable than visual classification with a CNN, which is applicable to almost any situation that can be accurately simulated. While we therefore based our final implementation and future discussion on a CNN, optimal data-mining comes from thoughtfully combining machine learning and alternative statistics.

6.4.3 Training Method

To train the classifier network, a simple CNN classifier based on the Oxford VGG[76] model was used. This model features heavily in many common machine learning applications, and a simpler implementation was used in our case as our images are discrete and lack fine detail, making the extra learning capacity unnecessary. This implementation consisted

of two 2D convolutional layers with a kernel size of 32 and 3x3 strides and ReLu activation, followed by max pooling with 2x2 strides. This was then repeated with kernel sizes of 64, drop-out regularisation added, and then final classification outputted with softmax activation and categorical cross entropy loss. For simplicity, we used the common Adam optimiser[79], and standard hyper-parameters[79] of $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-7}$.

To massively augment the ≈75,000 images in the training set, the periodic boundaries of the simulations allowed random circular shifts in the x and y axes. To simplify the learning task further, the 3-level simulations were flattened to 2 levels, firstly as ideal scans have a negligible amount of liquid, and secondly because it is easier to binarise rather than trinarise real images. This was done by replacing the least common level randomly with the other two, producing speckle noise in the process. Additional random speckle noise, level randomising, and vertical and horizontal flips and rotations were also applied. Scale invariance was also introduced through the coarsening methods discussed in 6.3.3.

We note that because of the relatively simple task and large variety of augmentations, only a fraction of the data was required, making the method more applicable to more intensive simulations. Because multiple model structures and hyper-parameters were not compared or tuned, performance could be significantly improved by hyper-parameter and structure tuning an ensemble of multiple CNNs. Because training progress was sensitive to random initialisation, we did not train for a fixed number of epochs, but instead until the loss on a testing set of ≈2,500 images plateaued. Further scale invariance was also introduced by varying simulation resolution and upscaling with nearest-neighbour interpolation to a common size of 200x200 pixels.

After training, over 95% accuracy was rapidly found for a validation set of 2290 additional simulations.

6.5 Historic Experimental Dataset

Thus far, we have used of data only available in simulations to avoid the need to hand-label actual scans. However, our trained network must still be able to correctly classify real AFM images.

For this purpose, we employ a historic dataset of 5519 scans collected in the mid 2000's. Not only are there multiple examples of each of the types of structures desired, but crucially there are many examples of other scans of irrelevant surfaces and of generally poor images. A small number of examples are shown in Figure 6.5.1.

In this sense this dataset is typical of experimental AFM datasets; a large, sporadic collection too large to filter through by hand, which was therefore left forgotten and untouched for over a decade. While the methods mentioned here are somewhat specific to the target structures and dataset, we aimed to use methods that are easily adaptable to other structures and datasets. If this were not the case, the proposed routine would be highly specific to this one individual example dataset of no specific significance, and therefore of little worth.

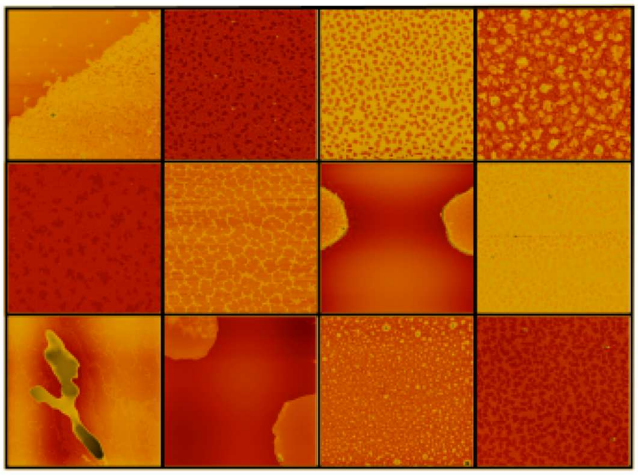

Figure 6.5.1: Random examples of (lightly preprocessed) AFM solvent evaporation images from a dataset of over 5000 historic scans. Researchers often ignore significant amounts of data in datasets such as this due to their need to manually preprocess and search through them for images of desired structures. Through a combination of automatic preprocessing, classical statistics, and neural networks, these image sets can be datamined for specific growth structures, without the need for manual human searching.

6.6 Assessing & Filtering Real Scans

6.6.1 Preprocessing

After training the network on automatically labelled simulations (rather than manually labelled data), the network can then be used to find similar structures in real AFM scans, even if the wanted scans are mixed with unwanted scans in a large, messy dataset. The idea of using target images to find similar data is becoming of increasing interest, such as when performing reverse image searches[83] or other content-based-retrieval tasks. However, for AFM, non-identical scanning conditions means that pre-processing/binarising data is first required before classification can take place. Because automating this process is not trivial, pre-processing is typically done manually[190]. However, unlike other automated methods[168,191] which maximise aesthetic quality for a large visual variety of images, we are not interested in aesthetic quality, and only need a broadly reasonable pre-processing method. By optimising the routine for the target structures, we can also assume that images unable to be processed are of low quality and/or not the structures being searched for, and so can be discarded.

Before classifying each image, we first applied several layers of pre-processing and filtering. After converting[192] from proprietary file format into Python (and discarding any corrupt files), we normalize and median of difference align the image data from the scan. Noisy scans were arbitrarily discarded if over 5% of rows had over 95% of data single-valued and/or with row mean more than 2σ from the overall image mean. A 5^{th} order polynomial plane fit was then used for plane removal along both x and y axes. We discarded images that were not self-similar by testing the stability of Minkowski numbers as a sub-sampling window was moved around the image frame. Binarisation was then performed with a multiple-layered Otsu threshold[193], which has been previously shown to be broadly optimal for AFM images[168].

We note that the shape of the structures can be significantly altered by the binarisation routine[168], resulting in the network correctly assessing the binarised image, but not the actual image. However, we also note that a given image may have multiple valid binarisations, and therefore multiple valid structure classifications (which itself often leads to subconscious human bias, an issue avoided by this automated approach). Regardless, classification is limited by pre-processing quality. It is also for this reason that we used multiple filters.

6.6.2 Denoising

Further, because of pre-processing artefacts, the susceptibility of CNNs to noise[194], and the inherent lack of instrumental noise in simulated training data, we found poor classification performance without denoising. To this end, we employed an autoencoder as described in Subsection 3.3.2. Here, we input simulated data plus speckle noise (in this case $40 \pm 5\%$) as $\mathbf{x}$, and teach it to output the original, noiseless simulations as $\hat{\mathbf{y}}$. The network therefore learns to remove noise. For simplicity, we used the above network structure (minus the final dense classification layer) for encoding, inversed this structure for decoding, and employed identical optimiser and hyper-parameters. Not only did denoising improve aesthetic quality dramatically (as seen in Figure 6.6.1, in which scanning artefacts such as noise and short tip crashes were removed), but classification performance also improved as a result.

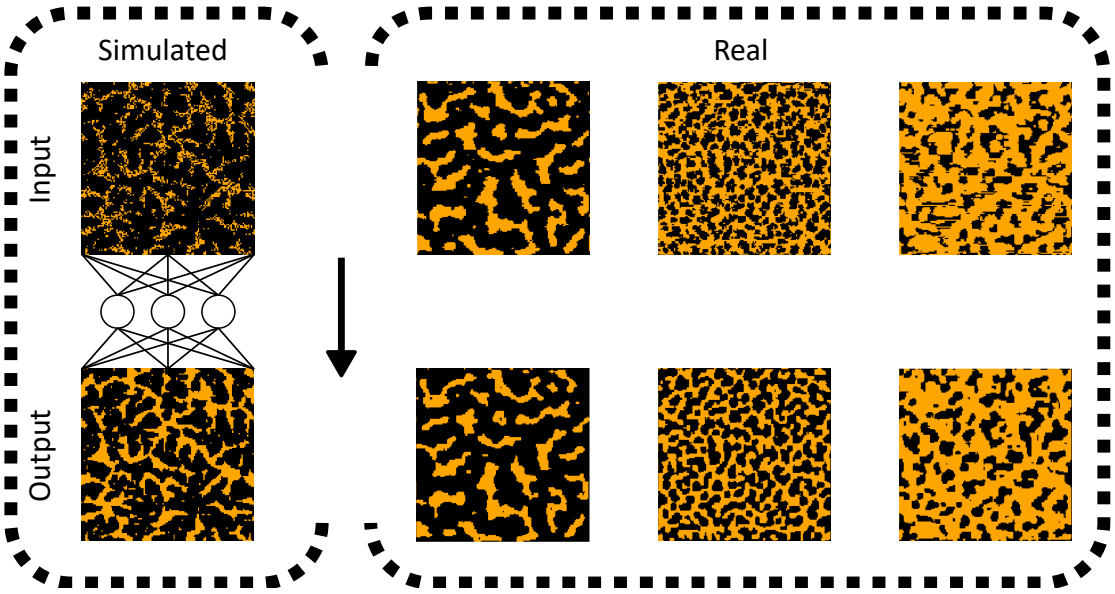

Figure 6.6.1: Results of a denoising autoencoder used to remove noise and artefacts from subsections of AFM dewetting images. Starting with simulated scans and adding speckle noise, the autoencoder removes noise by learning to recover the original simulated scan before noise was applied. This network can then be applied to real, pre-processed/binarised scans, where it is able to remove noise and artefacts from a wide variety of images.

6.6.3 CNN Assessment

At this stage, any remaining images could then be classified by the CNN. Because the network input was smaller than the total resolution of each image, multiple assessments for each image could be made with a sliding sub-window. This is key, as it allows for images of any size to be classified. Because neural networks require fixed image resolution, new simulations and networks would otherwise be required for every different scan resolution.

Additionally, this method gives a distribution of network confidences over the entire image, which can then be used as another filtering stage to remove non-homogenous images and/or those containing different structures. Examples of this process are shown in Figure 6.6.2. To further improve filtering, receiver-operator-characteristic (ROC) curves were used to tune the threshold value used to filter/pass each image. This could be expanded upon by using sub-images as an additional input dimension when training the CNN[97].

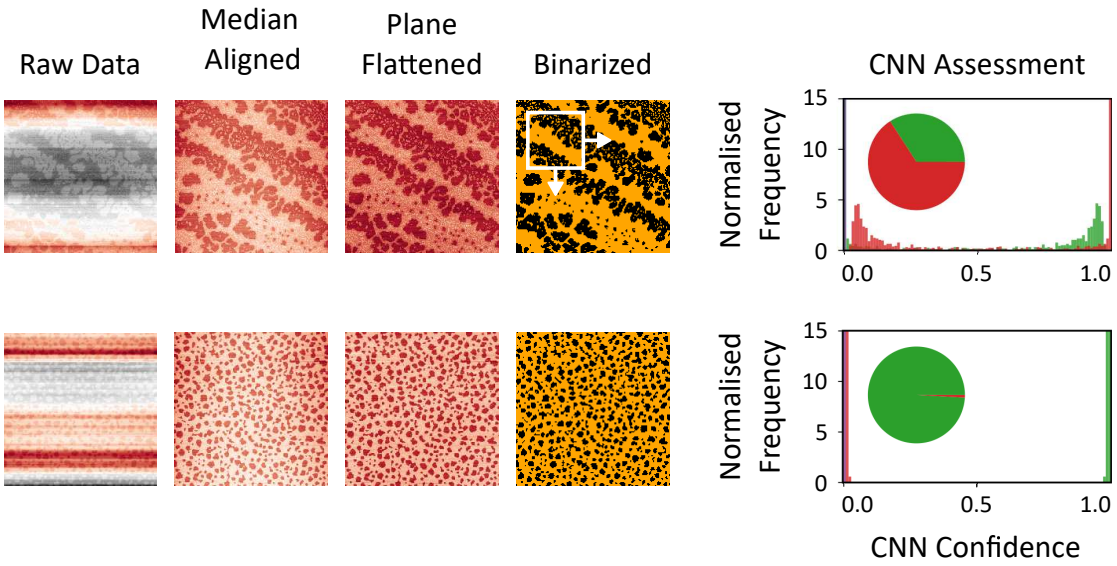

Figure 6.6.2: Automated pre-processing and CNN assessment of two AFM dewetting images of micro- and nanostructured nanoparticle assemblies on silicon substrates. Each image is subsampled and assessed with a CNN trained on simulated images of desirable structures. For the top, undesirable image, the network cannot confidently pick a classification, and is unsure if the structure is 'cellular' (green) or 'labyrinthine' (red) (amongst others). The bottom, desirable image clearly contains cells, and is correctly assessed as such by the CNN. This network, alongside other statistics, forms a system of anomaly detection and classification to find specific structures in a large, variable database.

6.6.4 Filtering Performance

To assess filtering performance, we first consider if a reasonable number of images in total are discarded by the filter. From a manual assessment of 100 random, un-processed images (so as to avoid analysis being skewed by incorrect processing), we expected approximately 60% of images to be discarded by the filter. Despite the difficulty of the task, this was approximately the case, with 70% discarded. Examples of final classifications are shown in Figure 6.4.1.

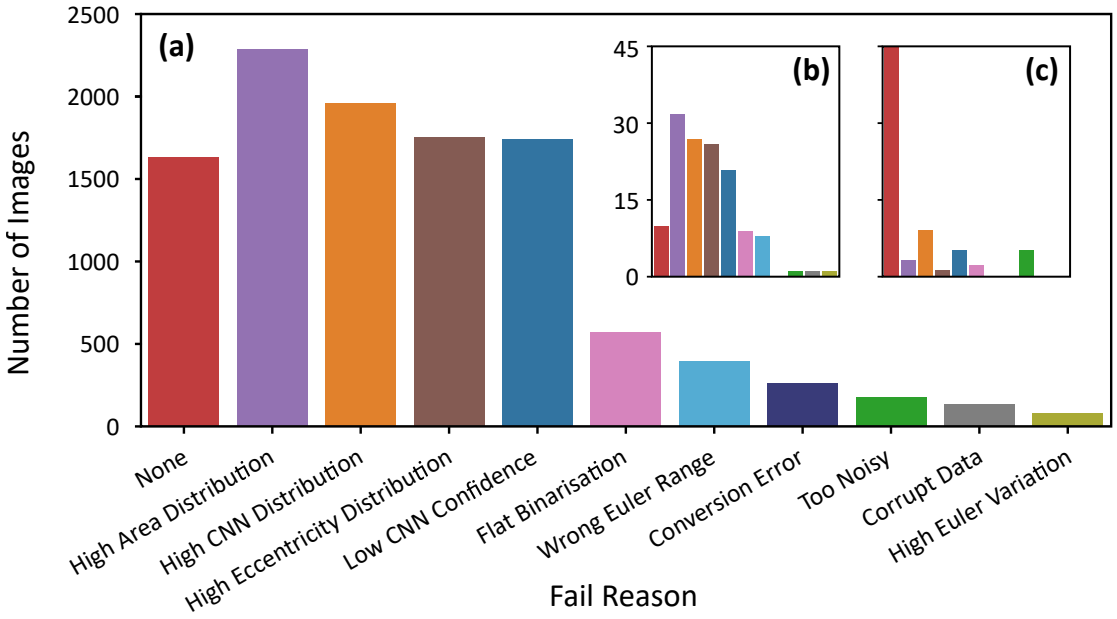

Figure 6.6.3: Figure to show filtering results on **(a)** a large, varied dataset of 5517 scans, **(b)**, 59 hand-selected 'bad' images, and **(c)**, 61 hand-selected 'good' images. Ideally, all of the 'good' images should pass the filter ("None", red), and vice versa for the 'bad' images. This system combines general statistical analysis with a CNN trained on Monte-Carlo simulations of the target scans, and each image can fail multiple filters.

To determine if the correct images were discarded (i.e. not just the correct number), two datasets of 61 "good" and 59 "bad" images were then hand-selected from the final dataset (without denoising). Ideally, all of the 'good' images should not be filtered out, while all the 'bad' images should be. As expected, 49/59 (83% specificity) bad images were correctly filtered out, while 45/61 (74% sensitivity) of the good images were correctly passed. Many false cases were due to poor pre-processing, often due to defects or tip changes. Comparing Figures 6.6.3(b) and (c), the filters clearly differentiated the data correctly. Additionally, many failing "good" images only failed a single filter, while many "bad" images failed multiple. While we chose to balance between the two, sensitivity and specificity could be adjusted by adjusting filters and their thresholds.

To assess final classification performance, we considered if a classification for an image by the entire filter/classier system was 'broadly correct'. This was as classification is extremely subjective and can have multiple valid outcomes. From the final output, we manually verified 20 random images from each category, finding that 15/20 holes, 14/20 cells, 13/20 labyrinths/fingers and 10/20 islands were broadly correct. We note that as of the filtering specificity, $\approx 20\%$ of these misclassifications were images incorrectly passing the filtering. This was also far superior to random selection from the original, unfiltered dataset, in which 2/20 holes, 2/20 cells, 3/20 labyrinths and 0/20 islands were correct. Other misclassifications were due to misleading pre-processing, were correct but non-homogenous, or were between visually similar categories (e.g. cells and labyrinths).

Finally, while classification was clearly scale invariant, many misclassifications were made at very small feature scales. This is understandable, as at this scale the CNN convolutions

would break up connected regions, effectively turning them into isolated islands and being classified as such. Furthermore, the CNNs are only as scale invariant as their training data, and our method of data simulation was unable to produce tiny feature scales while still being low-resolution enough to allow windowing. This issue is seen in areas such as painting recognition[195], and could be improved by creating an ensemble of networks with different convolution sizes trained on progressively smaller sections of each image. While outputs from the filtering/classifying system require manual verification, it can still clearly find areas of interest in datasets too large to search manually, making new analysis viable.

6.7 Conclusion

We have shown that it is possible to use simple simulations to correctly find specific structures in a dataset of mixed AFM images without the need for manually labelled training data. The method is heavily reliant on good pre-processing and binarisation. While noise removal is also particularly vital, a denoising autoencoder trained on the simulated data performed extremely well at this task, and may be applicable elsewhere to smooth features and remove processing artefacts. Additional improvements could be made with alternative network structures, fewer target structures, and/or a machine learning approach to binarisation[89,196], defect segmentation, and anomaly detection[197,198].

The method we have developed for automated identification of nanostructured patterns is an effective first stage of file search, capable of isolating files and locations of interest. It very significantly reduces the time spent searching datasets to perform additional analysis. Provided a simple model of structure growth and a method to broadly categorise simulations is available, this CNN protocol can be easily applied to other forms of SPM, datasets and target structures. When rapid simulation is not possible, strategies such as regression using scale-invariant statistics could also be effective, with a full comparison the subject of future work.

7 Water in a C_{60} Cage

7.1 Introduction

One of the most commonly studied molecules in soft matter physics is of course the fullerene molecule C_{60}, otherwise known as the buckyball, discovered by Kroto et al. in 1985[199], for which they were later awarded the 1996 Nobel Prize in Chemistry. By then encapsulating a molecule such as La[200], HF or H_2O within it, we can produce endofullerenes. Because buckyballs potentially influence electrostatic interactions with entrapped molecules, we can in effect create a test tube at the nanoscale for all number of experiments, from qubit candidates[201], nanoelectronics[202] and the production of molecular markers[203].

However, the extent of this chemical and electrostatic screening is very much an open topic of debate. In particular, it has been suggested that the C_{60} acts as a faraday cage, fully screening out electric fields from the molecule contained within. One of the most well studied examples of this is that of H_2O @ C_{60}, which can be manufactured through a process dubbed 'molecular surgery'[204].

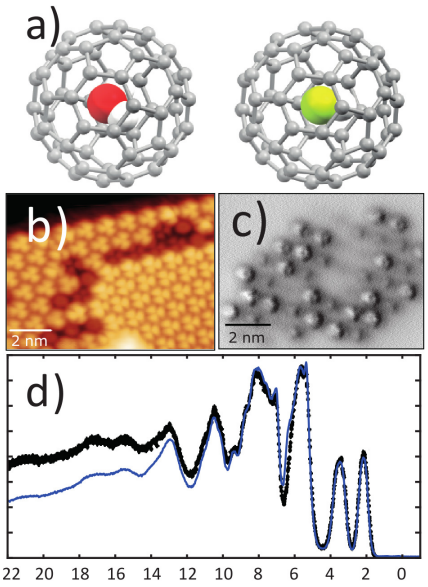

Figure 7.1.1: Demonstration of how HF and H_2O (theoretically shown in **(a)**) can be substantially screened by encapsulating them in C_{60}. As a result, filled and unfilled cages cannot be distinguished in either **(b)** STM or **(c)** non-contact AFM, and only at high binding energies (95% pure H_2O @ C_{60} in blue, empty C_60 in black) for **(d)** valence band synchrotron photoemission. From Jarvis et al.[205]

As a result of the electrostatic shielding, tunnelling techniques such as STM, AFM and photoemission spectroscopy are unable to distinguish between H_2O filled and unfilled C_{60} (after depositing in a 70:30 ratio), as seen in Figure 7.1.1[205]. We therefore sought to use NIXSW, as discussed in Section 2.3, to locate the position of the water within the cage. While a full discussion of this experiment is outside the bounds of this thesis, we also attempted to observe any potential temperature-dependence of the coherent position and fraction, given the suggestion that the encapsulated water can still significantly influence its surrounding environment due to the presence of temperature-dependant spin conversion[206].

While Figure 7.1.2 shows a clear O 1S peak, indicating the presence of H_2O @ C_{60}, and Figure 7.1.3 indicates that beam damage was kept to a minimum, we found no evidence for a temperature dependence within the error bars of our measurements.

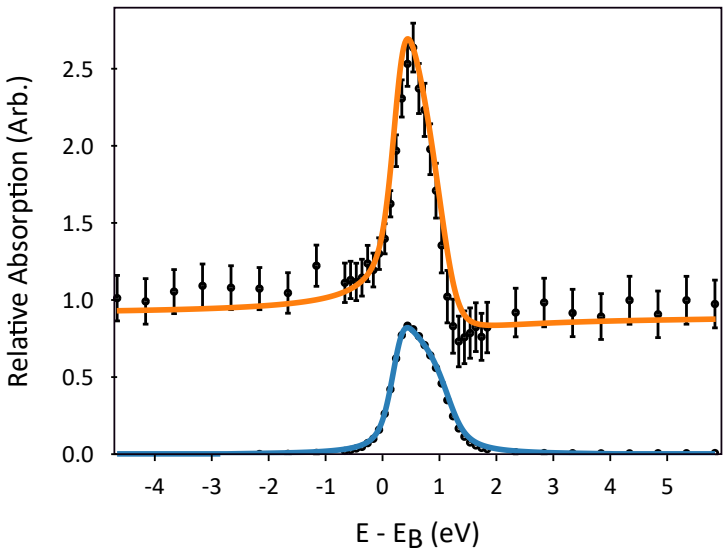

Figure 7.1.2: XSW intensity vs energy (orange), and fit to reflectivity (blue), for the O1S peak of the H_2O @ C_{60} surface at 200K. This fit is based on the Bragg energy (2631eV) coherent position (0.385 ± 0.100) and coherent fraction (0.225 ± 0.120).

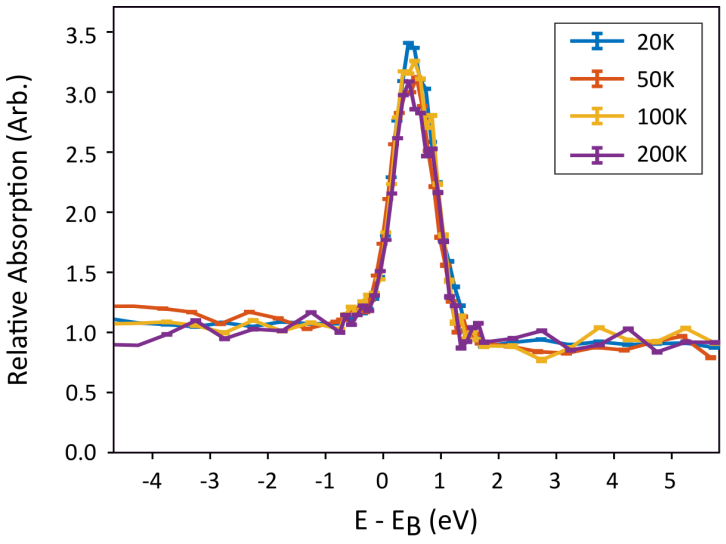

Figure 7.1.3: XSW spectra for H_2O @ C_{60}, normalised by temperature. As the sample temperature was varied throughout the experiment, the intensity of the O 1S peak remains relatively constant, indicating that any results found during the experiment were not a result of sample damage caused by the x-ray beam.

7.2 Distinguishing Noisy Spectra with Unsupervised k-Means

7.2.1 Simulating Spectra

One of the reasons for the large error bars that made finding the temperature dependence of $H_2O@C_{60}$ difficult was that of poor fitting onto noisy data. The same can also be seen in STM; an earlier analysis[207] by a previous member of the group showed no human-discernable difference between $\mathrm{d}F/\mathrm{d}z$ STM spectra of C_{60} and $H_2O@C_{60}$, even though in theory this should still be possible. However, Wahl et al. were recently able to use a slightly modified k-means unsupervised classifier to distinguish between two types of human-undistinguishable STM spectra[72], without prior knowledge of the form of each spectrum.

When we discussed the application of the Lennard-Jones potential to AFM in Section 2.2 potential[46], U, of the form

$$U(z) = 4\epsilon\left[\left(\frac{r_0}{z}\right)^{12} - \left(\frac{r_0}{z}\right)^{6}\right], \qquad \{7.1\}$$

only Pauli exclusion and Van-Der-Waals attraction are considered. Here, ϵ is a measure of how strongly the tip and sample interact, and r_0 is z at the point $U = 0$.

For C_{60} - C_{60} interactions (and AFM) in reality, this is not exactly the case, and we should instead consider small modifications. For a pair of C_{60} molecules, the high degree of rotational freedom means that the Lennard-Jones potential is an oversimplification. Instead, Girifalco[208] found that by approximating each C_{60} as a uniform sphere,

$$U(z) = -2\alpha\left[\frac{1}{s(s-1)^3} - \frac{1}{s(s+1)^3} - \frac{2}{s^4}\right] + 2\beta\left[\frac{1}{s(s-1)^9} - \frac{1}{s(s+1)^9} - \frac{2}{s^{10}}\right], \quad \{7.2\}$$

where

$$s = \frac{z}{2a}, \qquad \{7.3\}$$

$$\alpha = \frac{N^2 A}{12(2r_0)^6}, \qquad \{7.4\}$$

$$\beta = \frac{N^2 B}{90(2r_0)^{12}}. \qquad \{7.5\}$$

Here, N is the number of atoms per molecule (60 for C_{60}), A and B are constants, and a is the radius of C_{60} He went on to find that[208] for C_{60}), $a = 0.35$nm, $A = 0.0019266$ kJ mol^{-1}nm^{-6}, and $B = 3.357787 \times 10^{-6}$ kJ mol^{-1}nm^{-12} (though there is some debate amongst the literature as to the precise experimental values[209,210]).

As such, by calculating the Girifalco potential[208] for a pair of C_{60} cages, differentiating $U(z)$ twice, and adding noise, we can simulate spectra that could be recorded under STM, and use unsupervised clustering to attempt to separate them.

To simulate different types of interaction between encapsulated molecule with its surrounding cage, the repulsive exponent in Equation 7.2 was altered. While 100,000 spectra containing

512 data points were generated with the theoretical repulsive exponent of 12, 50,000 were generated with a repulsive exponent of 8, producing a slightly differently shaped spectra, as shown in Figure 7.2.1. This 2:1 ratio added complexity to the clustering process, and also made the separation task more realistic as not all dual-molecule surfaces are equally distributed. To make the simulation more realistic, random gaussian noise of $\sigma = 0.1$ was added to the repulsive exponent in Equation 7.2. Instead of analytically differentiating Equation 7.2, we numerically differentiated, adding additional noise of $\sigma = 0.1$ to $\phi(z)$, $F(z)$, and $\mathrm{d}F/\mathrm{d}z$ before each stage of differentiation.

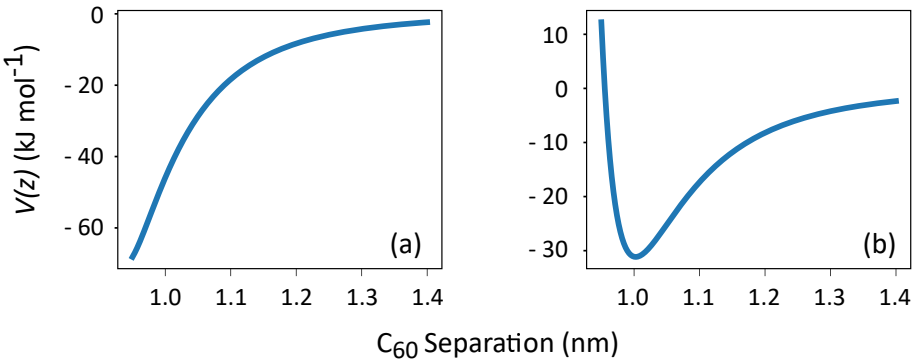

Figure 7.2.1: Figure to simulate the Girifalco potential of C_{60}. The system was modified to have a repulsive exponent of 8 (a), instead of the normal potential of 12 **(b)**. These curves can them be differentiated twice to simulate $\mathrm{d}F/\mathrm{d}z$ spectra that could be produced by STM, and unsupervised clustering used to automatically distinguish them.

At this point, the 150,000 spectra were randomly shuffled into training/testing sets in an 80:20 ratio (to hide the actual classification from the clustering algorithm). By learning off the training set, an unmodified k-means classifier (Wahl. et al used k-means, but with a small, single modification) was then used to attempt to distinguish the otherwise unseen testing spectra back into their original categories. A mean can then be taken to reproduce the underlying spectrum of each category.

7.2.2 Output & Performance

Remarkably, the basic, unmodified k-means algorithm was able to correctly cluster incredibly noisy $\mathrm{d}F/\mathrm{d}z$ and $\mathrm{d}^2F/\mathrm{d}z^2$ spectra, albeit it with some caveats. As expected, the clustering began to fail when increasing the noise, reducing the difference between the exponents, or adding extra regimes of spectra. Further, the classifier frequently performed perfectly on the otherwise unseen $\mathrm{d}F/\mathrm{d}z$ test set.

While this perfect performance may be due to the small shoulder in the lower exponent dataset (as seen in Figure 7.2.2), reasonable (albeit not perfect) performance was seen with $\mathrm{d}^2F/\mathrm{d}z^2$ spectra, where the shoulder was not visible. Interestingly, with $\mathrm{d}^2F/\mathrm{d}z^2$ spectra in a 2:1 ratio, a significant percentage of $(2a)^8$ exponents were misclassified as $(2a)^{12}$, but not the other way around. This is likely as the classifications were incorrectly in a broadly 1:1 ratio. When generated in a 1:1 ratio with $\sigma = 0.05$, the human-indistinguishable spectra could be distinguished with an accuracy of 61%, with similar rates of misclassifications between the two exponents. This is shown in Figure 7.2.3.

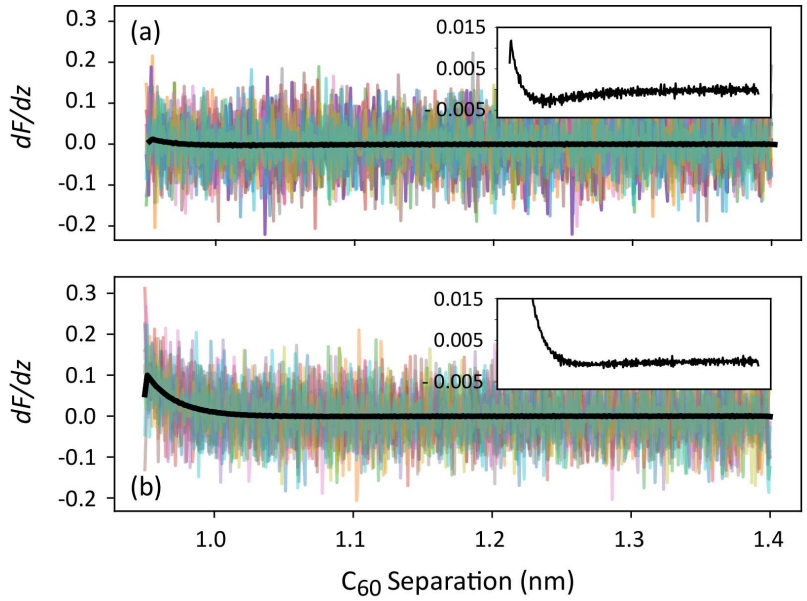

Figure 7.2.2: Figure to show simulated $\mathrm{d}F/\mathrm{d}z$ spectra based on a noisy Girifalco potential of C_{60}. The system was modified to have a repulsive exponent of 8 (a), instead of the normal potential of 12 **(b)**. 150,000 noisy spectra (shown in colour), were provided to an unmodified k-means clusterer, which was able to perfectly distinguish the two potentials with no knowledge of which spectra belonged to which category. These classified spectra were then averaged (black), to perfectly reproduce the "correct" average $\mathrm{d}F/\mathrm{d}z$ spectra (red). This protocol could potentially be applied to real STM spectra to attempt to distinguish between C_{60} and $\mathrm{H_2O@C_{60}}$ molecules.

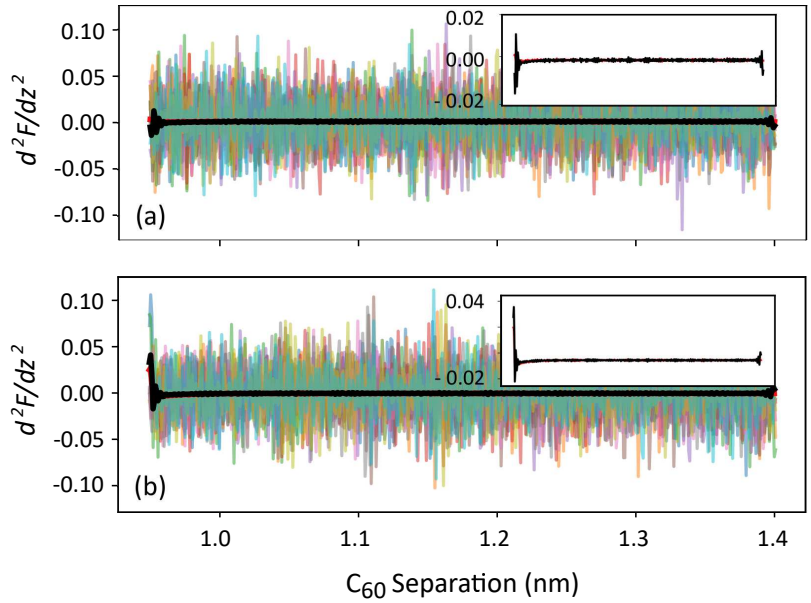

Figure 7.2.3: Figure to show simulated $\mathrm{d}^2F/\mathrm{d}z^2$ spectra based on a noisy Girifalco potential of C_{60}. The system was modified to have a repulsive exponent of 8 (a), instead of the normal potential of 12 **(b)**. 150,000 noisy spectra (shown in colour), were provided to an unmodified k-means clusterer, which was able to distinguish the two potentials with no knowledge of which spectra belonged to which category with an accuracy of over 60%, despite the spectra being indistinguishable to the human eye. These classified spectra were then averaged (black), in an attempt to reproduce the "correct" average $\mathrm{d}^2F/\mathrm{d}z^2$ spectra (red). This protocol could potentially be applied to real STM spectra to attempt to distinguish between C_{60} and $\mathrm{H_2O@C_{60}}$ molecules.

Regardless, modifications to the clustering routine, combined with additional analysis such as PCA, could allow for large numbers of otherwise indistinguishable spectra to be separated and averaged. A confidence metric based on the distance to each cluster centroid could also be used to improve performance at the expense of wasted data. This improved protocol could then be applied to the noisy XSW data, or potentially even the $\mathrm{d}F/\mathrm{d}z$ C_{60} and $H_2O@C_{60}$ data collected previously.

7.3 Conclusion

Given the intense debate surrounding the appearance of interactions of water in its C_{60} cage, there is significant scope to look to unsupervised classification methods to attempt to extract features from noisy data to attempt to interrogate otherwise indistinguishable spectra. Indeed, it is very common for signal-to-noise to only become tolerable after a significant number of repeat experiments, and so combining multiple signals is incredibly commonplace in the world of noisy data; the only question is therefore if the separation itself is trustable.

There are, however, significant limitations to this approach when working with real-world spectra. Firstly, the sensitivity of k-means to noise means that we must make a major assumption that any noise is consistent between the two types of spectra. More importantly, k-means often struggles when presented with imbalanced data (e.g. if 70% of spectra should belong to one class, and 30% to another). We would therefore have to deposit in a 50:50 ratio, and then identify meaningful features in the average spectra for each cluster. Otherwise, even if the spectra were correctly separated (which, in turn, assumes that there is something to distinguish, which may not even necessarily be the case), we could not know which belongs to empty C_{60}, and which to $H_2O@C_{60}$.

8 Conclusion

Probe microscopy at times astounds, but mostly infuriates. For all of the exciting potential of next generation materials science, computing regimes and methods to control the very building blocks of our universe, there is a cold, disillusioned, disgruntled microscopist trying to take back control of their equipment.

Throughout this thesis, we have explored how machine learning methods, and particularly neural networks, may soon finally provide an end to this frustration. After introducing core SPM and machine learning theory in Chapters 2 and 3, in Chapter 4, we were successfully able to use CNNs to detect a variety of surface states in real-time, under generic scanning conditions, and on live instrumentation through the development of the nOmicron Python API, which is mature enough to the point where it is being applied to other STM experiments in research groups around the world. Besides incorporating scans from much larger scan-areas into the networks and direct input of parameters such as scanning bias and range of tip height, the most significant issues come from the need to hand-label data, and having models that only apply to specific surfaces. Further, alternative data structures could be considered, such as training models with both forward and backward traces combined, instead of the current method of classifying them separately. Besides considering alternative methods of tip state recognition such as anomaly detection, it would also likely be fruitful to consider aspects of 'explainable-AI' to better understand what features are being used to base classifications on.

Following on in Chapter 5, we then took the next logical step of not simply assessing blunted tips, but attempting to intelligently sharpen them in response, using human examples as a basis for imitation learning. While we sadly can only report limited success thus far, our system did exhibit some broadly reasonable behaviours, such as avoiding the random behaviour of constantly performing actions which would render the task effectively impossible. We have also captured a significant amount of live data for use in future attempts. One significant issue encountered when attempting to capture live data was quite poor classification performance from the CNN networks. On closer inspection, the original 2015 dataset had low variety in scan sizes, little-to-no presence of step edges, and had the direction of visual features adjusted to be near constant. Following the classic phrase "rubbish in, rubbish out", this will inevitably harm performance. Significant performance improvements will likely be seen by expanding and reclassifying the dataset to include all H:Si(100) images available to the research group. Indeed, such a task is currently being undertaken as of the time of writing. Regardless, given that this is an extremely complex, stochastic task, we also believe that we are limited by the state of machine learning research today, and so would certainly benefit from closer collaborations not just with equipment manufacturers, but with state-of-the-art, theoretical machine learning researchers. Improvements may also be seen by attempting to produce a vaguely realistic STM scan simulator by pulling existing scan data, therefore bypassing the need for human based samples.

In Chapter 6, we then explored the concept of using simulations and machine learning to datamine experimental AFM data. Particular success was seen when using autoencoders to denoise experimental data, especially when removing single-pixel high artefacts. It is impressive that such a striking impact was seen with a basic network trained with bina-

rised images and speckle noise. As such, it would likely be fruitful to research into this further. Future attempts could seek to combine together forward/backward height and current traces, simulated artefacts, and more complex network architectures into a system that could output a single denoised image. Besides structure category, such a system could also be applied to segmentation of defects and step edges, or counting/detecting deposits and other atomic scale markers on a surface. Finally, in terms of file search, further research into adapting anomaly detection and reverse-image searching methods may also be highly lucrative, particularly when trying to analyse highly noisy data such as that of Chapter 7.

Ultimately, while the dream of an SPM autopilot has been considered *"not far away"* for the best part of half a century now, the potential to refine, and combine, recent breakthroughs mean that automated SPM experimentation is no longer a mere 1980's pipe dream, but an extremely realistic, *imminent*, prospect.

References

[1] Martin Ruivenkamp and Arie Rip. Entanglement of imaging and imagining of nanotechnology. *NanoEthics*, 5(2):185–193, jul 2011. doi:10.1007/s11569-011-0122-2.

[2] Ray Kurzweil. Keynote address. In *Eighth Foresight Conference on Molecular Nanotechnology*, 2000.

[3] R. Bollongino, J. Burger, A. Powell, M. Mashkour, J.-D. Vigne, and M. G. Thomas. Modern taurine cattle descended from small number of near-eastern founders. *Molecular Biology and Evolution*, 29(9):2101–2104, mar 2012. doi:10.1093/molbev/mss092.

[4] Taleana Huff, Hatem Labidi, Mohammad Rashidi, Lucian Livadaru, Thomas Dienel, Roshan Achal, Wyatt Vine, Jason Pitters, and Robert A Wolkow. Binary atomic silicon logic. *Nature Electronics*, 1(12):636, 2018. doi:10.1038/s41928-018-0180-3.

[5] M. F. Crommie, C. P. Lutz, and D. M. Eigler. Confinement of electrons to quantum corrals on a metal surface. *Science*, 262(5131):218–220, oct 1993. doi:10.1126/science.262.5131.218.

[6] D. M. Eigler and E. K. Schweizer. Positioning single atoms with a scanning tunnelling microscope. *Nature*, 344(6266):524–526, apr 1990. doi:10.1038/344524a0.

[7] Leo Gross, Fabian Mohn, Nikolaj Moll, Peter Liljeroth, and Gerhard Meyer. The chemical structure of a molecule resolved by atomic force microscopy. *Science*, 325 (5944):1110–1114, 2009.

[8] Richard A. J. Woolley, Julian Stirling, Adrian Radocea, Natalio Krasnogor, and Philip Moriarty. Automated probe microscopy via evolutionary optimization at the atomic scale. *Applied Physics Letters*, 98(25):253104, jun 2011. doi:10.1063/1.3600662.

[9] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, dec 1943. doi:10.1007/bf02478259.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[11] François Chollet et al. Keras. `https://keras.io`, 2015.

[12] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, November 2016. USENIX Association. ISBN 978-1-931971-33-1. URL `https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi`.

[13] Navin Sharma, Pranshu Sharma, David Irwin, and Prashant Shenoy. Predicting solar generation from weather forecasts using machine learning. In *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, oct 2011. doi:10.1109/smartgridcomm.2011.6102379.

[14] Adam Byerly, Tatiana Kalganova, and Ian Dear. A branching and merging convolutional network with homogeneous filter capsules. *arXiv*, January 2020.

[15] Ilias Maglogiannis. *Emerging artificial intelligence applications in computer engineering : real word AI systems with applications in eHealth, HCI, information retrieval and pervasive technologies*. IOS Press, Amsterdam Washington, DC, 2007. ISBN 9781586037802.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiV preprint: arXiv:1312.5602v1*, December 2013.

[17] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, June 2016.

[18] Christopher J.C. Burges Yann LeCun, Corinna Cortes. Mnist dataset. Website, 1999. URL `http://yann.lecun.com/exdb/mnist/`. Accessed 6/2/2020.

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi:10.1109/5.726791.

[20] Jack C Straton, Taylor T Bilyeu, Bill Moon, and Peter Moeck. Double-tip effects on scanning tunneling microscopy imaging of 2d periodic objects: unambiguous detection and limits of their removal by crystallographic averaging in the spatial frequency domain. *Crystal Research and Technology*, 49(9):663–680, 2014. doi:10.1002/crat.201300240.

[21] Adam Sweetman, Sam Jarvis, Rosanna Danza, and Philip Moriarty. Effect of the tip state during qplus noncontact atomic force microscopy of si (100) at 5 k: Probing the probe. *Beilstein journal of nanotechnology*, 3:25, 2012. doi:10.3762/bjnano.3.3.

[22] Mohammad Rashidi and Robert A Wolkow. Autonomous scanning probe microscopy in situ tip conditioning through machine learning. *ACS Nano*, 2018. doi:10.1021/acsnano.8b02208.

[23] Mohammad Rashidi, Jeremiah Croshaw, Kieran Mastel, Marcus Tamura, Hedieh Hosseinzadeh, and Robert A Wolkow. Deep learning-guided surface characterization for autonomous hydrogen lithography. *Machine Learning: Science and Technology*, 1 (2):025001, 2020. doi:10.1088/2632-2153/ab6d5e.

[24] Eran Rabani, David R Reichman, Phillip L Geissler, and Louis E Brus. Drying-mediated self-assembly of nanoparticles. *Nature*, 426(6964):271–274, 2003.

[25] Gerd Binnig, Heinrich Rohrer, Ch Gerber, and Edmund Weibel. Surface studies by scanning tunneling microscopy. *Physical Review Letters*, 49(1):57, jul 1982. doi:10.1103/physrevlett.49.57.

[26] Franz J Giessibl. The qplus sensor, a powerful core for the atomic force microscope. *Review of Scientific Instruments*, 90(1):011101, 2019. doi:10.1063/1.5052264.

[27] C. Julian Chen. *Introduction to Scanning Tunneling Microscopy*. Oxford University Press, sep 2007. doi:10.1093/acprof:oso/9780199211500.001.0001.

[28] Roland Wiesendanger and Wiesendanger Roland. *Scanning probe microscopy and spectroscopy: methods and applications*. Cambridge university press, 1994.

[29] J. Tersoff and D. R. Hamann. Theory of the scanning tunneling microscope. *Physical Review B*, 31(2):805–813, jan 1985. doi:10.1103/physrevb.31.805.

[30] J. Bardeen. Tunnelling from a many-particle point of view. *Physical Review Letters*, 6

(2):57–59, jan 1961. doi:10.1103/physrevlett.6.57.

[31] Julian Stirling, Ioannis Lekkas, Adam Sweetman, Predrag Djuranovic, Quanmin Guo, Brian Pauw, Josef Granwehr, Raphaël Lévy, and Philip Moriarty. Critical assessment of the evidence for striped nanoparticles. *PLoS ONE*, 9(11):e108482, nov 2014. doi:10.1371/journal.pone.0108482.

[32] Werner A. Hofer. Heisenberg, uncertainty, and the scanning tunneling microscope. *Frontiers of Physics*, 7(2):218–222, mar 2012. doi:10.1007/s11467-012-0246-z.

[33] Morten Møller, Samuel P Jarvis, Laurent Guérinet, Peter Sharp, Richard Woolley, Philipp Rahe, and Philip Moriarty. Automated extraction of single h atoms with STM: tip state dependency. *Nanotechnology*, 28(7):075302, jan 2017. doi:10.1088/1361-6528/28/7/075302.

[34] P. Moriarty, Y.R. Ma, M.D. Upward, and P.H. Beton. Translation, rotation and removal of c60 on si(100)-2 x 1 using anisotropic molecular manipulation. *Surface Science*, 407 (1-3):27–35, jun 1998. doi:10.1016/s0039-6028(98)00082-x.

[35] M. J. Humphry, R. Chettle, P. J. Moriarty, M. D. Upward, and P. H. Beton. Digital scanning probe microscope controller for molecular manipulation applications. *Review of Scientific Instruments*, 71(4):1698–1701, apr 2000. doi:10.1063/1.1150522.

[36] D.L Keeling, M.J Humphry, P Moriarty, and P.H Beton. Attractive mode manipulation of covalently bound molecules. *Chemical Physics Letters*, 366(3-4):300–304, dec 2002. doi:10.1016/s0009-2614(02)01588-9.

[37] D. L. Keeling, M. J. Humphry, R. H. J. Fawcett, P. H. Beton, C. Hobbs, and L. Kantorovich. Bond breaking coupled with translation in rolling of covalently bound molecules. *Physical Review Letters*, 94(14):146104, apr 2005. doi:10.1103/physrevlett.94.146104.

[38] Maohua Li, Ruiqi Lv, Shengchao Huang, Yinzhen Dai, Zhicong Zeng, Lei Wang, and Bin Ren. Electrochemical fabrication of silver tips for tip-enhanced raman spectroscopy assisted by a machine vision system. *Journal of Raman Spectroscopy*, 47(7):808–812, 2016. doi:10.1002/jrs.4898.

[39] Moh'd Rezeq, Jason Pitters, and Robert Wolkow. Tungsten nanotip fabrication by spatially controlled field-assisted reaction with nitrogen. *The Journal of chemical physics*, 124(20):204716, 2006. doi:10.1063/1.2198536.

[40] Vt spm system. URL `https://scientaomicron.com/en/system-solutions/SPM/VT-SPM-Lab`.

[41] Gerd Binnig, Calvin F Quate, and Ch Gerber. Atomic force microscope. *Physical review letters*, 56(9):930, 1986. doi:10.1007/978-94-011-1812-5_4.

[42] Mohammad Rashidi, Wyatt Vine, Thomas Dienel, Lucian Livadaru, Jacob Retallick, Taleana Huff, Konrad Walus, and Robert A Wolkow. Initiating and monitoring the evolution of single electrons within atom-defined structures. *Physical review letters*, 121(16):166801, 2018. doi:10.1103/physrevlett.121.166801.

[43] L Bartels, G Meyer, and K-H Rieder. Controlled vertical manipulation of single co molecules with the scanning tunneling microscope: A route to chemical contrast. *Applied Physics Letters*, 71(2):213–215, 1997. doi:10.1063/1.119503.

[44] Leo Gross, Bruno Schuler, Niko Pavliček, Shadi Fatayer, Zsolt Majzik, Nikolaj Moll, Diego Peña, and Gerhard Meyer. Atomic force microscopy for molecular structure

elucidation. *Angewandte Chemie International Edition*, 57(15):3888–3908, 2018. doi:10.1002/anie.201703509.

[45] Niko Pavliček and Leo Gross. Generation, manipulation and characterization of molecules by atomic force microscopy. *Nature Reviews Chemistry*, 1(1):0005, 2017. doi:10.1038/s41570-016-0005.

[46] J. E. Jones. On the determination of molecular fields.—i. from the variation of the viscosity of a gas with temperature. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 106(738):441–462, oct 1924. doi:10.1098/rspa.1924.0081.

[47] T. R. Albrecht, P. Grütter, D. Horne, and D. Rugar. Frequency modulation detection using high-qcantilevers for enhanced force microscope sensitivity. *Journal of Applied Physics*, 69(2):668–673, jan 1991. doi:10.1063/1.347347.

[48] Sangmin An, Mun heon Hong, Jongwoo Kim, Soyoung Kwon, Kunyoung Lee, Manhee Lee, and Wonho Jhe. Quartz tuning fork-based frequency modulation atomic force spectroscopy and microscopy with all digital phase-locked loop. *Review of Scientific Instruments*, 83(11):113705, nov 2012. doi:10.1063/1.4765702.

[49] Boris W. Batterman. Effect of dynamical diffraction in x-ray fluorescence scattering. *Physical Review*, 133(3A):A759–A764, feb 1964. doi:10.1103/physrev.133.a759.

[50] D P Woodruff. Surface structure determination using x-ray standing waves. *Reports on Progress in Physics*, 68(4):743–798, mar 2005. doi:10.1088/0034-4885/68/4/r01.

[51] Robert G Jones, A S Y Chan, M G Roper, M P Skegg, I G Shuttleworth, C J Fisher, G J Jackson, J J Lee, D P Woodruff, N K Singh, and B C C Cowie. X-ray standing waves at surfaces. *Journal of Physics: Condensed Matter*, 14(16):4059–4074, apr 2002. doi:10.1088/0953-8984/14/16/301.

[52] D P Woodruff, B C C Cowie, and A R H F Ettema. Surface structure determination using x-ray standing waves: a simple view. *Journal of Physics: Condensed Matter*, 6 (49):10633–10645, dec 1994. doi:10.1088/0953-8984/6/49/007.

[53] Jörg Zegenhagen and Alexander Kazimirov. *The X-Ray Standing Wave Technique*. WORLD SCIENTIFIC, 2013. doi:10.1142/6666. URL `https://www.worldscientific.com/doi/abs/10.1142/6666`.

[54] Jörg Zegenhagen. X-ray standing waves technique: Fourier imaging active sites. *Japanese Journal of Applied Physics*, 58(11):110502, nov 2019. doi:10.7567/1347-4065/ab4dec.

[55] F.C. Bocquet, G. Mercurio, M. Franke, G. van Straaten, S. Weiß, S. Soubatch, C. Kumpf, and F.S. Tautz. Torricelli: A software to determine atomic spatial distributions from normal incidence x-ray standing wave data. *Computer Physics Communications*, 235:502–513, feb 2019. doi:10.1016/j.cpc.2018.06.009.

[56] D P Woodruff and D A Duncan. X-ray standing wave studies of molecular adsorption: why coherent fractions matter. *New Journal of Physics*, 22(11):113012, nov 2020. doi:10.1088/1367-2630/abc63a.

[57] Christian Szegedy and Vincent O Vanhoucke. Processing images using deep neural networks, July 25 2017. US Patent 9,715,642.

[58] Sameer Samat. Android p: Packed with more smarts and simpler than ever. Web, May

2018. https://www.blog.google/products/android/android-p/. Accessed 10/03/21.

[59] John Byabazaire, Cristian Olariu, Mohit Taneja, and Alan Davy. Lameness detection as a service: Application of machine learning to an internet of cattle. In *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, jan 2019. doi:10.1109/ccnc.2019.8651681.

[60] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]*, 2, 2010. URL `http://yann.lecun.com/exdb/mnist`.

[61] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[62] Ernst Kussul and Tatiana Baidyk. Improved method of handwritten digit recognition tested on mnist database. *Image and Vision Computing*, 22(12):971–981, 2004. doi:10.1016/j.imavis.2004.03.008.

[63] Thomas Grote and Philipp Berens. On the ethics of algorithmic decision-making in healthcare. *Journal of medical ethics*, 46(3):205–211, 2020.

[64] Scott Mayer McKinney, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, Trevor Back, Mary Chesus, Greg S. Corrado, Ara Darzi, Mozziyar Etemadi, Florencia Garcia-Vicente, Fiona J. Gilbert, Mark Halling-Brown, Demis Hassabis, Sunny Jansen, Alan Karthikesalingam, Christopher J. Kelly, Dominic King, Joseph R. Ledsam, David Melnick, Hormuz Mostofi, Lily Peng, Joshua Jay Reicher, Bernardino Romera-Paredes, Richard Sidebottom, Mustafa Suleyman, Daniel Tse, Kenneth C. Young, Jeffrey De Fauw, and Shravya Shetty. International evaluation of an AI system for breast cancer screening. *Nature*, 577(7788):89–94, jan 2020. doi:10.1038/s41586-019-1799-6.

[65] Benjamin Alldritt, Prokop Hapala, Niko Oinonen, Fedor Urtev, Ondrej Krejci, Filippo Federici Canova, Juho Kannala, Fabian Schulz, Peter Liljeroth, and Adam S Foster. Automated structure discovery in atomic force microscopy. *Science advances*, 6(9):eaay6913, 2020. doi:10.1126/sciadv.aay6913.

[66] Rama K. Vasudevan, Kamal Choudhary, Apurva Mehta, Ryan Smith, Gilad Kusne, Francesca Tavazza, Lukas Vlcek, Maxim Ziatdinov, Sergei V. Kalinin, and Jason Hattrick-Simpers. Materials science in the artificial intelligence age: high-throughput library generation, machine learning, and a pathway from correlations to the underpinning physics. *MRS Communications*, 9(3):821–838, sep 2019. doi:10.1557/mrc.2019.95.

[67] Maxim Ziatdinov, Ondrej Dyck, Artem Maksov, Xufan Li, Xiahan Sang, Kai Xiao, Raymond R. Unocic, Rama Vasudevan, Stephen Jesse, and Sergei V. Kalinin. Deep learning of atomically resolved scanning transmission electron microscopy images: Chemical identification and tracking local transformations. *ACS Nano*, 11(12):12742–12752, dec 2017. doi:10.1021/acsnano.7b07504.

[68] Nikolay Borodinov, Sabine Neumayer, Sergei V. Kalinin, Olga S. Ovchinnikova, Rama K. Vasudevan, and Stephen Jesse. Deep neural networks for understanding noisy data applied to physical property extraction in scanning probe microscopy. *npj Computational Materials*, 5(1), feb 2019. doi:10.1038/s41524-019-0148-5.

[69] Yi Zhang, A. Mesaros, K. Fujita, S. D. Edkins, M. H. Hamidian, K. Ch'ng, H. Eisaki, S. Uchida, J. C. Séamus Davis, Ehsan Khatami, and Eun-Ah Kim. Machine learning in electronic-quantum-matter imaging experiments. *Nature*, 570(7762):484–490, jun 2019. doi:10.1038/s41586-019-1319-8.

[70] Prokop Hapala, Georgy Kichin, Christian Wagner, F. Stefan Tautz, Ruslan Temirov, and Pavel Jelínek. Mechanism of high-resolution STM/AFM imaging with functionalized tips. *Physical Review B*, 90(8):085421, aug 2014. doi:10.1103/physrevb.90.085421.

[71] Ondrej Krejčí, Prokop Hapala, Martin Ondráček, and Pavel Jelínek. Principles and simulations of high-resolution STM imaging with a flexible tip apex. *Physical Review B*, 95(4):045407, jan 2017. doi:10.1103/physrevb.95.045407.

[72] P. Wahl, U. R. Singh, V. Tsurkan, and A. Loidl. Nanoscale electronic inhomogeneity in $\mathrm{fese}_{0.4}\mathrm{te}_{0.6}$ revealed through unsupervised machine learning. *ArXiV: preprint:2002.10004v1*, 2020. doi:10.1103/PhysRevB.101.115112.

[73] Sendhil Mullainathan and Jann Spiess. Machine learning: An applied econometric approach. *Journal of Economic Perspectives*, 31(2):87–106, may 2017. doi:10.1257/jep.31.2.87.

[74] Oliver Gordon and Philip Moriarty. Machine learning at the (sub) atomic scale: next generation scanning probe microscopy. *Machine Learning: Science and Technology*, 2020. doi:10.1088/2632-2153/ab7d2f.

[75] Thomas Hofmann, Bernhard Schölkopf, and Alexander J Smola. Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220, 2008. doi:10.1214/009053607000000677.

[76] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[77] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, jan 2015. doi:10.1016/j.neunet.2014.09.003.

[78] Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki*, pages 6–7, 1970.

[79] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiV preprint: arXiV:1412.6980v9*, 2014.

[80] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009. doi:10.1145/1541880.1541882.

[81] Mohiuddin Ahmed, Abdun Naser Mahmood, and Md. Rafiqul Islam. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55:278–288, feb 2016. doi:10.1016/j.future.2015.01.001.

[82] Qi Wei, Bibo Shi, Joseph Y. Lo, Lawrence Carin, Yinhao Ren, and Rui Hou. Anomaly detection for medical images based on a one-class classification. In Kensaku Mori and Nicholas Petrick, editors, *Medical Imaging 2018: Computer-Aided Diagnosis*. SPIE, feb 2018. doi:10.1117/12.2293408.

[83] Alex Krizhevsky and Geoffrey E Hinton. Using very deep autoencoders for content-based image retrieval. In *ESANN*, volume 1, page 2. Citeseer, 2011.

[84] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global k-means cluster-

ing algorithm. *Pattern Recognition*, 36(2):451–461, feb 2003. doi:10.1016/s0031-3203(02)00060-2.

[85] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR, 2015.

[86] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size. *arXiv*, February 2016.

[87] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196, 2018.

[88] Taking the pain out of probes, 2019. URL `https://muircheartblog.wordpress.com/2019/09/24/taking-the-pain-out-of-probes/`.

[89] Steff Farley, Jo E A Hodgkinson, Oliver M Gordon, Joanna Turner, Andrea Soltoggio, Philip J Moriarty, and Eugenie Hunsicker. Improving the segmentation of scanning probe microscope images using convolutional neural networks. *Machine Learning: Science and Technology*, 2(1):015015, dec 2020. doi:10.1088/2632-2153/abc81c.

[90] John Kelleher. *Fundamentals of machine learning for predictive data analytics : algorithms, worked examples, and case studies*. The MIT Press, Cambridge, Massachusetts, 2015. ISBN 9780262029445.

[91] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3):e0118432, mar 2015. doi:10.1371/journal.pone.0118432.

[92] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[93] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[94] Itai Caspi, Gal Leibovich, Gal Novik, and Shadi Endrawis. Reinforcement learning coach, December 2017. URL `https://doi.org/10.5281/zenodo.1134899`.

[95] Jae Won Lee. Stock price prediction using reinforcement learning. In *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*, volume 1, pages 690–695. IEEE, 2001.

[96] O Gordon, P D'Hondt, L Knijff, SE Freeney, F Junqueira, P Moriarty, and I Swart. Scanning tunneling state recognition with multi-class neural network ensembles. *Review of Scientific Instruments*, 90(10):103704, 2019. doi:10.1063/1.5099590.

[97] Oliver M Gordon, Filipe LQ Junqueira, and Philip J Moriarty. Embedding human heuristics in machine-learning-enabled probe microscopy. *Machine Learning: Science and Technology*, 1(1):015001, 2020. doi:10.1088/2632-2153/ab42ec.

[98] A Krull, P Hirsch, C Rother, A Schiffrin, and C Krull. Artificial-intelligence-driven scanning probe microscopy. *Communications Physics*, 3(1):1–8, 2020. doi:10.1038/s42005-020-0317-3.

[99] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, jan 2016. doi:10.1038/nature16961.

[100] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pages 663–670, 2000.

[101] Chenyang Shen, Yesenia Gonzalez, Peter Klages, Nan Qin, Hyunuk Jung, Liyuan Chen, Dan Nguyen, Steve B Jiang, and Xun Jia. Intelligent inverse treatment planning via deep reinforcement learning, a proof-of-principle study in high dose-rate brachytherapy for cervical cancer. *Physics in Medicine & Biology*, 64(11):115013, 2019. doi:10.1088/1361-6560/ab18bf.

[102] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[103] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[104] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *arXiv*, February 2016.

[105] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv*, September 2015.

[106] Matthijs T. J. Spaan. Partially observable markov decision processes. In *Adaptation, Learning, and Optimization*, pages 387–414. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-27645-3_12.

[107] Andrew A Gewirth and Brian K Niece. Electrochemical applications of in situ scanning probe microscopy. *Chemical reviews*, 97(4):1129–1162, 1997. doi:10.1021/cr960067y.

[108] Bert Voigtländer. *Scanning Probe Microscopy.* Springer, 2016.

[109] Noël Bonnet, Samuel Dongmo, Philippe Vautrot, and Michel Troyon. A mathematical morphology approach to image formation and image restoration in scanning tunnelling and atomic force microscopies. *Microscopy Microanalysis Microstructures*, 5(4-6): 477–487, 1994. doi:10.1051/mmm:0199400504-6047700.

[110] Mohammad Rashidi, Jeremiah Croshaw, Kieran Mastel, Marcus Tamura, Hedieh Hosseinzadeh, and Robert A Wolkow. Autonomous atomic scale manufacturing through machine learning. *arXiv preprint arXiv:1902.08818*, 2019.

[111] Dan C Cireşan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 411–418. Springer, 2013. doi:10.1007/978-3-642-40763-5_51.

[112] Michelle Lochner, Jason D McEwen, Hiranya V Peiris, Ofer Lahav, and Max K Winter. Photometric supernova classification with machine learning. *The Astrophysical Journal Supplement Series*, 225(2):31, 2016. doi:10.3847/0067-0049/225/2/31.

[113] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. doi:10.1145/3065386.

[114] John Rumble. *CRC handbook of chemistry and physics : a ready-reference book of chemical and physical data*. CRC Press/Taylor & Francis Group, Boca Raton, 2020. ISBN 9780367417246.

[115] R. J. Hamers, R. M. Tromp, and J. E. Demuth. Scanning tunneling microscopy of si(001). *Physical Review B*, 34(8):5343–5357, oct 1986. doi:10.1103/physrevb.34.5343.

[116] Sam Jarvis, Adam Sweetman, Joseph Bamidele, Lev Kantorovich, and Philip Moriarty. Role of orbital overlap in atomic manipulation. *Physical Review B*, 85(23):235305, jun 2012. doi:10.1103/physrevb.85.235305.

[117] Adam Sweetman, Sam Jarvis, Rosanna Danza, Joseph Bamidele, Subhashis Gangopadhyay, Gordon A. Shaw, Lev Kantorovich, and Philip Moriarty. Toggling bistable atoms via mechanical switching of bond angle. *Physical Review Letters*, 106(13):136101, mar 2011. doi:10.1103/physrevlett.106.136101.

[118] D. J. Chadi. Atomic and electronic structures of reconstructed si(100) surfaces. *Physical Review Letters*, 43(1):43–47, jul 1979. doi:10.1103/physrevlett.43.43.

[119] Robert A. Wolkow. Direct observation of an increase in buckled dimers on si(001) at low temperature. *Physical Review Letters*, 68(17):2636–2639, apr 1992. doi:10.1103/physrevlett.68.2636.

[120] Lev Kantorovich and Chris Hobbs. Probing theSi(001)surface with a si tip: Anab initiostudy. *Physical Review B*, 73(24):245420, jun 2006. doi:10.1103/physrevb.73.245420.

[121] R. J. Hamers and U. K. Köhler. Determination of the local electronic structure of atomic-sized defects on si(001) by tunneling spectroscopy. *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, 7(4):2854–2859, jul 1989. doi:10.1116/1.576158.

[122] R. M. Tromp, R. J. Hamers, and J. E. Demuth. Si(001) dimer structure observed with scanning tunneling microscopy. *Physical Review Letters*, 55(12):1303–1306, sep 1985. doi:10.1103/physrevlett.55.1303.

[123] Jing Wang, T. A. Arias, and J. D. Joannopoulos. Dimer vacancies and dimer-vacancy complexes on the si(100) surface. *Physical Review B*, 47(16):10497–10508, apr 1993. doi:10.1103/physrevb.47.10497.

[124] O. Warschkow, S. R. Schofield, N. A. Marks, M. W. Radny, P. V. Smith, and D. R. McKenzie. Water on silicon (001):cdefects and initial steps of surface oxidation. *Physical Review B*, 77(20):201305, may 2008. doi:10.1103/physrevb.77.201305.

[125] Masamichi Yoshimura, Izumi Ono, and Kazuyuki Ueda. Initial stages of ni reaction on si(100) and h-terminated si(100) surfaces. *Applied Surface Science*, 130-132:276–281, jun 1998. doi:10.1016/s0169-4332(98)00070-1.

[126] John J. Boland. Scanning tunnelling microscopy of the interaction of hydrogen with silicon surfaces. *Advances in Physics*, 42(2):129–171, apr 1993. doi:10.1080/00018739300101474.

[127] T. C. Shen, C. Wang, G. C. Abeln, J. R. Tucker, J. W. Lyding, Ph. Avouris, and R. E. Walkup. Atomic-scale desorption through electronic and vibrational excitation mecha-

nisms. *Science*, 268(5217):1590–1592, jun 1995. doi:10.1126/science.268.5217.1590.

[128] Taleana R. Huff, Thomas Dienel, Mohammad Rashidi, Roshan Achal, Lucian Livadaru, Jeremiah Croshaw, and Robert A. Wolkow. Electrostatic landscape of a hydrogen-terminated silicon surface probed by a moveable quantum dot. *ACS Nano*, 13(9): 10566–10575, aug 2019. doi:10.1021/acsnano.9b04653.

[129] Joshua B. Ballard, James H. G. Owen, William Owen, Justin R. Alexander, Ehud Fuchs, John N. Randall, James R. Von Ehr, Stephen McDonnell, Don D. Dick, Robert M. Wallace, Yves J. Chabal, Maia R. Bischof, David L. Jaeger, Richard F. Reidy, Joseph Fu, Pradeep Namboodiri, Kai Li, and Richard M. Silver. Pattern transfer of hydrogen depassivation lithography patterns into silicon with atomically traceable placement and size control. *Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena*, 32(4):041804, jul 2014. doi:10.1116/1.4890484.

[130] Joshua B. Ballard, Thomas W. Sisson, James H. G. Owen, William R. Owen, Ehud Fuchs, Justin Alexander, John N. Randall, and James R. Von Ehr. Multimode hydrogen depassivation lithography: A method for optimizing atomically precise write times. *Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena*, 31(6):06FC01, nov 2013. doi:10.1116/1.4823756.

[131] B. Weber, S. Mahapatra, H. Ryu, S. Lee, A. Fuhrer, T. C. G. Reusch, D. L. Thompson, W. C. T. Lee, G. Klimeck, L. C. L. Hollenberg, and M. Y. Simmons. Ohm's law survives to the atomic scale. *Science*, 335(6064):64–67, jan 2012. doi:10.1126/science.1214319.

[132] Martin Fuechsle, Jill A. Miwa, Suddhasatta Mahapatra, Hoon Ryu, Sunhee Lee, Oliver Warschkow, Lloyd C. L. Hollenberg, Gerhard Klimeck, and Michelle Y. Simmons. A single-atom transistor. *Nature Nanotechnology*, 7(4):242–246, feb 2012. doi:10.1038/nnano.2012.21.

[133] S. R. Schofield, P. Studer, C. F. Hirjibehedin, N. J. Curson, G. Aeppli, and D. R. Bowler. Quantum engineering at the silicon surface using dangling bonds. *Nature Communications*, 4(1), apr 2013. doi:10.1038/ncomms2679.

[134] Martin Fuechsle and Michelle Simmons. *Using Scanning Tunneling Microscopy to Realize Atomic- Scale Silicon Devices*. Pan Stanford Publishing, apr 2013. doi:10.1201/b14792-5.

[135] A. Bellec, D. Riedel, G. Dujardin, O. Boudrioua, L. Chaput, L. Stauffer, and Ph. Sonnet. Nonlocal activation of a bistable atom through a surface state charge-transfer process onSi(100)-(2x1):h. *Physical Review Letters*, 105(4):048302, jul 2010. doi:10.1103/physrevlett.105.048302.

[136] Zhixiang Sun, Mark P. Boneschanscher, Ingmar Swart, Daniel Vanmaekelbergh, and Peter Liljeroth. Quantitative Atomic Force Microscopy with Carbon Monoxide Terminated Tips. *Phys. Rev. Lett.*, 106(4), JAN 27 2011. ISSN 0031-9007. doi:10.1103/PhysRevLett.106.046104.

[137] Peter H Jacobse, Adri van den Hoogenband, Marc-Etienne Moret, Robertus JM Klein Gebbink, and Ingmar Swart. Aryl radical geometry determines nanographene formation on au (111). *Angewandte Chemie International Edition*, 55(42):13052–13055, 2016. doi:10.1002/ange.201606440.

[138] Julian Stirling, Richard A. J. Woolley, and Philip Moriarty. Scanning probe image wizard: A toolbox for automated scanning probe microscopy data analysis. *Review of Scientific Instruments*, 84(11):113701, nov 2013. doi:10.1063/1.4827076.

[139] FN Iandola, S Han, MW Moskewicz, K Ashraf, WJ Dally, and K Keutzer. Squeezenet v1. 1 model, 2017.

[140] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[141] Kaibo Duan, S Sathiya Keerthi, Wei Chu, Shirish Krishnaj Shevade, and Aun Neow Poo. Multi-category classification by soft-max combination of binary classifiers. In *International Workshop on Multiple Classifier Systems*, pages 125–134. Springer, 2003. doi:10.1007/3-540-44938-8_13.

[142] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[143] DP Kingma and J Ba. Dp kingma and j. ba, adam: A method for stochastic optimization,. *arXiv: 1412.6980.*, 2014.

[144] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv:1212.5701*, 2012.

[145] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul): 2121–2159, 2011.

[146] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*, September 2016.

[147] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000. doi:10.1007/3-540-45014-9_1.

[148] L. Burzawa, S. Liu, and E. W. Carlson. Classifying surface probe images in strongly correlated electronic systems via machine learning. *Physical Review Materials*, 3(3): 033805, mar 2019. doi:10.1103/physrevmaterials.3.033805.

[149] M. O. Blunt, C. P. Martin, M. Ahola-Tuomi, E. Pauliac-Vaujour, P. Sharp, P. Nativo, M. Brust, and P. J. Moriarty. Coerced mechanical coarsening of nanoparticle assemblies. *Nature Nanotechnology*, 2(3):167–170, feb 2007. doi:10.1038/nnano.2007.25.

[150] Peter Siepmann, Christopher P Martin, Ioan Vancea, Philip J Moriarty, and Natalio Krasnogor. A genetic algorithm approach to probing the evolution of self-organized nanostructured systems. *Nano letters*, 7(7):1985–1990, 2007. doi:10.1021/nl070773m.

[151] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[152] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE*

conference on computer vision and pattern recognition, pages 2625–2634, 2015.

[153] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

[154] Stephan J. M. Zevenhuizen. Mate-for-dummies, August 2018. URL `https://pypi.org/project/MATE-for-Dummies/`.

[155] Oliver Gordon. nomicron, 2018. URL `https://github.com/OGordon100/nOmicron`.

[156] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. A guide to deep learning in healthcare. *Nature Medicine*, 25(1):24–29, jan 2019. doi:10.1038/s41591-018-0316-z.

[157] Baher Abdulhai, Rob Pringle, and Grigoris J. Karakoulas. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, 129(3): 278–285, may 2003. doi:10.1061/(asce)0733-947x(2003)129:3(278).

[158] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, may 2019. doi:10.1109/icra.2019.8794127.

[159] Shenkai Wang, Junmian Zhu, Raymond Blackwell, and Felix R. Fischer. Automated tip conditioning for scanning tunneling spectroscopy. *The Journal of Physical Chemistry A*, 125(6):1384–1390, feb 2021. doi:10.1021/acs.jpca.0c10731.

[160] Philipp Leinen, Malte Esders, Kristof T Schütt, Christian Wagner, Klaus-Robert Müller, and F Stefan Tautz. Autonomous robotic nanofabrication with reinforcement learning. *arXiv preprint arXiv:2002.11952*, 2020.

[161] David Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. In *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes*, pages 1–2. Springer, 1935. doi:10.1007/978-3-662-25726-5_1.

[162] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, feb 2015. doi:10.1038/nature14236.

[163] Roshan Achal, Mohammad Rashidi, Jeremiah Croshaw, David Churchill, Marco Taucer, Taleana Huff, Martin Cloutier, Jason Pitters, and Robert A. Wolkow. Lithography for robust and editable atomic-scale silicon devices and memories. *Nature Communications*, 9(1), jul 2018. doi:10.1038/s41467-018-05171-y.

[164] D.B. Fogel. Using evolutionary programing to create neural networks that are capable of playing tic-tac-toe. In *IEEE International Conference on Neural Networks*. IEEE, 1993. doi:10.1109/icnn.1993.298673.

[165] Nikolay Borodinov, Wan-Yu Tsai, Vladimir V. Korolkov, Nina Balke, Sergei V. Kalinin, and Olga S. Ovchinnikova. Machine learning-based multidomain processing for texture-based image segmentation and analysis. *Applied Physics Letters*, 116(4):044103, jan 2020. doi:10.1063/1.5135328.

[166] Maxim Ziatdinov, Artem Maksov, and Sergei V. Kalinin. Learning surface molecular structures via machine vision. *npj Computational Materials*, 3(1), aug 2017. doi:10.1038/s41524-017-0038-7.

[167] Maxim Ziatdinov, Ondrej Dyck, Xin Li, Bobby G. Sumpter, Stephen Jesse, Rama K. Vasudevan, and Sergei V. Kalinin. Building and exploring libraries of atomic defects in graphene: Scanning transmission electron and scanning tunneling microscopy study. *Science Advances*, 5(9):eaaw8989, sep 2019. doi:10.1126/sciadv.aaw8989.

[168] Steff Farley, Jo Hodgkinson, Oliver Gordon, Joanna Turner, Andrea Soltoggio, Philip Moriarty, and Eugenie Hunsicker. Improving segmentation of scanning probe microscope images using convolutional neural networks. *arXiv*, 2008.12371, August 2020.

[169] Mark P Oxley, Junqi Yin, Nikolay Borodinov, Suhas Somnath, Maxim Ziatdinov, Andrew Lupini, Stephen Jesse, Rama K Vasudevan, and Sergei V Kalinin. Deep learning of interface structures from simulated 4d STEM data: cation intermixing vs. roughening. *Machine Learning: Science and Technology (Just Accepted)*, jul 2020. doi:10.1088/2632-2153/aba32d. URL `10.1088/2632-2153/aba32d`. 10.1088/2632-2153/aba32d.

[170] Maxim Ziatdinov, Artem Maksov, Li Li, Athena S Sefat, Petro Maksymovych, and Sergei V Kalinin. Deep data mining in a real space: separation of intertwined electronic responses in a lightly doped BaFe2as2. *Nanotechnology*, 27(47):475706, oct 2016. doi:10.1088/0957-4484/27/47/475706.

[171] Peter M. Hoffmann. *Life's Ratchet*. Basic Books, New York, US, October 2012. ISBN 0465022537. URL `https://www.ebook.de/de/product/18694981/peter_m_hoffmann_life_s_ratchet.html`.

[172] Andrew Stannard. Dewetting-mediated pattern formation in nanoparticle assemblies. *Journal of Physics: Condensed Matter*, 23(8):083001, feb 2011. doi:10.1088/0953-8984/23/8/083001.

[173] Andrew Stannard, Christopher P Martin, Emmanuelle Pauliac-Vaujour, Philip Moriarty, and Uwe Thiele. Dual-scale pattern formation in nanoparticle assemblies. *The Journal of Physical Chemistry C*, 112(39):15195–15203, 2008.

[174] STEPHEN G. BRUSH. History of the lenz-ising model. *Reviews of Modern Physics*, 39(4):883–893, oct 1967. doi:10.1103/revmodphys.39.883.

[175] Habib Zaidi. Relevance of accurate monte carlo modeling in nuclear medical imaging. *Medical Physics*, 26(4):574–608, apr 1999. doi:10.1118/1.598559.

[176] Peter Jäckel. *Monte Carlo Methods in Finance*. WILEY, April 2002. ISBN 047149741X. URL `https://www.ebook.de/de/product/1767289/peter_jaeckel_monte_carlo_methods_in_finance.html`.

[177] Seungwoo Jeon and Bonghee Hong. Monte carlo simulation-based traffic speed forecasting using historical big data. *Future Generation Computer Systems*, 65:182–195, dec 2016. doi:10.1016/j.future.2015.11.022.

[178] Lars Onsager. Crystal statistics. i. a two-dimensional model with an order-disorder transition. *Physical Review*, 65(3-4):117–149, feb 1944. doi:10.1103/physrev.65.117.

[179] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines.

The Journal of Chemical Physics, 21(6):1087–1092, jun 1953. doi:10.1063/1.1699114.

[180] Guanglu Ge and Louis Brus. Evidence for spinodal phase separation in two-dimensional nanocrystal self-assembly. *The Journal of Physical Chemistry B*, 104(41):9573–9575, oct 2000. doi:10.1021/jp002280a.

[181] Christopher P. Martin, Matthew O. Blunt, and Philip Moriarty. Nanoparticle networks on silicon: self-organized or disorganized? *Nano Letters*, 4(12):2389–2392, dec 2004. doi:10.1021/nl048536w.

[182] P. Moriarty, M. D. R. Taylor, and M. Brust. Nanostructured cellular networks. *Physical Review Letters*, 89(24):248303, nov 2002. doi:10.1103/physrevlett.89.248303.

[183] Pamela C. Ohara and William M. Gelbart. Interplay between hole instability and nanoparticle array formation in ultrathin liquid films. *Langmuir*, 14(12):3418–3424, jun 1998. doi:10.1021/la971147f.

[184] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, 2015.

[185] Klaus R. Mecke. Additivity, convexity, and beyond: Applications of minkowski functionals in statistical physics. In *Statistical Physics and Spatial Statistics*, pages 111–184. Springer Berlin Heidelberg, 2000. doi:10.1007/3-540-45043-2_6.

[186] David Legland, Kiên Kiêu, and Marie-Françoise Devaux. Computation of minkowski measures on 2d and 3d binary images. *Image Analysis & Stereology*, 26(2):83–92, 2007. doi:10.5566/ias.v26.p83-92.

[187] Hubert Mantz, Karin Jacobs, and Klaus Mecke. Utilizing minkowski functionals for image analysis: a marching square algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(12):12015, dec 2008. doi:10.1088/1742-5468/2008/12/p12015.

[188] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, Jean-Yves Tinevez, Daniel James White, Volker Hartenstein, Kevin Eliceiri, Pavel Tomancak, and Albert Cardona. Fiji: an open-source platform for biological-image analysis. *Nature Methods*, 9(7):676–682, jun 2012. doi:10.1038/nmeth.2019.

[189] Frank E. Harrell, Kerry L. Lee, Robert M. Califf, David B. Pryor, and Robert A. Rosati. Regression modelling strategies for improved prognostic prediction. *Statistics in Medicine*, 3(2):143–152, apr 1984. doi:10.1002/sim.4780030207.

[190] David Nečas and Petr Klapetek. Gwyddion: an open-source software for SPM data analysis. *Open Physics*, 10(1):181–188, jan 2012. doi:10.2478/s11534-011-0096-2.

[191] A Delvallée, N Feltin, S Ducourtieux, M Trabelsi, and J F Hochepied. Direct comparison of AFM and SEM measurements on the same set of nanoparticles. *Measurement Science and Technology*, 26(8):085601, jun 2015. doi:10.1088/0957-0233/26/8/085601.

[192] Suhas Somnath, Chris R. Smith, Nouamane Laanait, Rama K. Vasudevan, Anton Ievlev, Alex Belianinov, Andrew R. Lupini, Mallikarjun Shankar, Sergei V. Kalinin, and Stephen Jesse. Usid and pycroscopy – open frameworks for storing and analyzing spectroscopic and imaging data. *arXiv*, 1903.09515v2, March 2019.

[193] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image

processing in python. *PeerJ*, 2:e453, jun 2014. doi:10.7717/peerj.453.

[194] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv*, 1412.6572, December 2014.

[195] Nanne van Noord and Eric Postma. Learning scale-variant and scale-invariant features for deep image classification. *Pattern Recognition*, 61:583–592, jan 2017. doi:10.1016/j.patcog.2016.06.005.

[196] Wesley Tatum, Diego Torrejon, Patrick O'Neil, Jonathan W Onorato, Anton Resing, Sarah Holliday, Lucas Q. Flagg, David S Ginger, and Christine K. Luscombe. A generalizable framework for algorithmic interpretation of thin film morphologies in scanning probe images. *Journal of Chemical Information and Modeling*, 60(7):3387–3397, jun 2020. doi:10.1021/acs.jcim.0c00308.

[197] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis 14*. ACM Press, 2014. doi:10.1145/2689746.2689747.

[198] Chong Zhou and Randy C. Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2017. doi:10.1145/3097983.3098052.

[199] H. W. Kroto, J. R. Heath, S. C. O'Brien, R. F. Curl, and R. E. Smalley. C60: Buckminsterfullerene. *Nature*, 318(6042):162–163, nov 1985. doi:10.1038/318162a0.

[200] Yan Chai, Ting Guo, Changming Jin, Robert E. Haufler, L. P. Felipe Chibante, Jan Fure, Lihong Wang, J. Michael Alford, and Richard E. Smalley. Fullerenes with metals inside. *The Journal of Physical Chemistry*, 95(20):7564–7568, oct 1991. doi:10.1021/j100173a002.

[201] Alexey Popov, editor. *Endohedral Fullerenes: Electron Transfer and Spin*. Springer-Verlag GmbH, May 2017. ISBN 9783319470498. URL `https://www.ebook.de/de/product/29215117/endohedral_fullerenes_electron_transfer_and_spin.html`.

[202] Chengbo Zhu and Xiaolin Wang. Tuning the conductance of h 2 o@ c 60 by position of the encapsulated h 2 o. *Scientific reports*, 5(1):1–7, 2015. doi:10.1038/srep17932.

[203] Jianyuan Zhang, Youqing Ye, Ying Chen, Christopher Pregot, Tinghui Li, Sharavanan Balasubramaniam, David B. Hobart, Yafen Zhang, Sungsool Wi, Richey M. Davis, Louis A. Madsen, John R. Morris, Stephen M. LaConte, Gordon T. Yee, and Harry C. Dorn. Gd3n@c84(OH)x: A new egg-shaped metallofullerene magnetic resonance imaging contrast agent. *Journal of the American Chemical Society*, 136(6):2630–2636, feb 2014. doi:10.1021/ja412254k.

[204] K. Kurotobi and Y. Murata. A single molecule of water encapsulated in fullerene c60. *Science*, 333(6042):613–616, jul 2011. doi:10.1126/science.1206376.

[205] Samuel Jarvis, Hongqian Sang, Filipe Janqueira, Oliver Gordon, Jo E. A. Hodgkinson, Alex Saywell, Philipp Rahe, Salvatore Mamone, Simone Taylor, Adam Sweetman, Jeremy Leaf, david Duncan, Tien-Lin Lee, Pardeep K. Thakur, Gabriella Hoffman, Richard Whitby, Malcolm levitt, George Held, Lev Kantorovich, Philip Moriarty, and Robert G. Jones. Intramolecular water divining: Locating h2o inside a c60 cage. *Nature*

Communications, July 2021. ISSN 2041-1723.

[206] Benno Meier, Salvatore Mamone, Maria Concistrè, Javier Alonso-Valdesueiro, Andrea Krachmalnicoff, Richard J. Whitby, and Malcolm H. Levitt. Electrical detection of ortho–para conversion in fullerene-encapsulated water. *Nature Communications*, 6(1), aug 2015. doi:10.1038/ncomms9112.

[207] Samuel P Jarvis, Filipe Junqueira, Alex Saywell, Philipp Rahe, Salvatore Mamone, Simon Taylor, Adam Sweetman, Jeremy Leaf, Hongqian Sang, Lev Kantorovich, David Duncan, Tien-Lin Lee, Pardeep Kumar, Richard Whitby, Philip Moriarty, and Robert G Jones. Intramolecular water divining: Locating h2o inside a fullerene cage. *In Production*, 2019.

[208] L. A. Girifalco. Interaction potential for carbon (c60) molecules. *The Journal of Physical Chemistry*, 95(14):5370–5371, jul 1991. doi:10.1021/j100167a002.

[209] D. M. Edmunds, P. Tangney, D. D. Vvedensky, and W. M. C. Foulkes. Free-energy coarse-grained potential for c60. *The Journal of Chemical Physics*, 143(16):164509, oct 2015. doi:10.1063/1.4932591.

[210] M. C. Abramo, C. Caccamo, D. Costa, G. Pellicane, and R. Ruberto. Atomistic versus two-body central potential models ofC60:a comparative molecular dynamics study. *Physical Review E*, 69(3):031112, mar 2004. doi:10.1103/physreve.69.031112.

List of Figures

List of Tables

List of Symbols & Nomenclature

Abbreviations

SPM	Scanning Probe Microscopy
STM	Scanning Tunnelling Microscopy
VT	Variable Temperature
AFM	Atomic Force Microscopy
NIXSW	Normal Incidence X-ray Standing Wave
UHV	Ultra High Vacuum
PID	Proportional–Integral–Derivative Controller
CNN	Convolutional Neural Network
RL	Reinforcement Learning
IRL	Inverse Reinforcement Learning
LSTM	Long Short-Term Memory
LRCN	Long-term Recurrent Convolutional Network
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
TPR	True Positive Rate (i.e. sensitivity)
FPR	False Positive Rate (i.e. recall)
PPV	True Positive Rate (i.e. precision)
ROC	Receiver-Operator-Characteristic
PR	Precision-Recall
AUC	Area-Under-Curve

Scanning Tunneling Microscopy

e	Charge of the electron
I	Transmission current
i	Number of image in dataset
V	Tip bias
z	Tip-sample separation
j	Number of scanlines recorded in a partial scan
M	Marker value
N	Total number of scanlines in a complete scan
W	Number of scanlines forming a subsampling window
$\mathbf{A}_j^i$	An incomplete scan of the i^{th} image array with j lines taken
$P(\mathbf{A})$	Tip state prediction of image array **A**

Atomic Force Microscopy

F	Tip-sample force
U	Tip-sample potential
z	Tip-sample separation
r_0	Tip-sample separation at which U=0
k	Cantilever effective spring constant
Q	Cantilever quality factor
f	Tip oscillation frequency
f_{res}	Tip oscillation frequency at resonance
A	Tip oscillation amplitude
A_{drive}	Driving amplitude of tip oscillation

Normal Incidence XSW

n	Mode of oscillation
λ	Wavelength of oscillation
θ_B	Angle of incidence
d_H	Lattice constant
z	Normal distance from the crystal
E_B	Bragg energy of material
E_0	Energy of incident beam
E_H	Energy of reflected beam
I	Standing wave intensity
R	Standing wave reflectivity
D	Coherent position
f_{co}	Coherent fraction

Machine Learning Theory

$\mathbf{x}$	Input vector to a network
$\mathbf{y}$	Output vector from a network
$\hat{\mathbf{y}}$	Manually defined 'correct' output vector from a network
k	Number of outputs at $\mathbf{y}$ (i.e. number of categories in classifier)
m	Number of data samples in batch
l	Layer number
t	Timestep
L	Total number of layers in network
M	Total number of data samples in dataset
T	Total time

$\mathbf{a}^{(l)}$	Input to the l^{th} layer of nodes
$\mathbf{z}^{(l)}$	Intermediary output from the l^{th} layer of nodes
$\mathbf{w}^{(l)\mathsf{T}}$	Weights of the l^{th} layer of nodes, transposed
$\mathbf{b}^{(l)}$	Biases of the l^{th} layer of nodes
S_t	State at time t
A_t	Action at time t
R_t	Reward at time t
G_t	Total return at time t
α	Loss rate
γ	Discount rate
$f()$	Entire network function
$g()$	Transfer function
$J()$	Loss (i.e. cost) function

Monte Carlo Theory

i	Index of a grid site
j	Index of a neighbouring grid site
L	Size of the system along each dimension
E	Total energy of the system
ΔE	Energy change due to (proposed) change of state
$\pm J$	Energetic favourability
σ	Spin of an ising model system
P	Probability of a given system state
C	Starting nanoparticle concentration
μ	Chemical potential

MR	Nanoparticle mobility ratio
ϵ_l	Liquid-liquid interaction energy
ϵ_n	Nanoparticle-nanoparticle interaction energy
ϵ_{nl}	Nanoparticle-liquid interaction energy
T	Temperature of the system
T_c	Critical temperature of the system
k_B	Boltzmann constant