Controlling the Scanning Electron Microscope with Deep Learning for Automated Acquisition and Image Quality Improvement

Sabrina Clusiau

Department of Materials Engineering

McGill University, Montreal

April 2024

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy

Abstract

Controlling image acquisition hardware with image processing software is the future of microscopy. The efficiency of microscopy techniques is currently limited by manual intervention and the need for microscopist supervision during acquisitions. More importantly, proper interpretation of image data requires quantitative analysis with computational methods. Automating simple and complex tasks on the scanning electron microscope (SEM) specifically, is feasible by replacing microscopist involvement with appropriate software algorithms and deep learning. Guided acquisitions on the SEM with hybrid methods is the solution to higher throughput and to generating better data, faster.

In this work, computational approaches to microscopy are presented. Foremost, two workflows for automated acquisitions are described, for imaging and image processing in Dragonfly, an image processing software. First, a feedforward workflow sequentially positions the beam in a predetermined grid layout, stitches the images, segments the feature of interest and computes measurements for quantitative analysis. Second, a workflow with a feedback loop between imaging and analysis is developed to enable dynamic imaging with feature segmentation and feature tracking. Guiding the hardware to image areas of interest with smart beam positioning is an optimized solution for acquiring only relevant data and minimizing acquisition duration.

Then, microscope routines, prior to acquisitions, are enhanced by integrating computational methods, focused on artificial intelligence. Microscopists spend a considerable amount of time tuning microscope parameters and aligning the beam until a desired image quality is observed. To assist with interpretation of image quality, a regression model is trained to optimize SEM

parameters thus boosting image quality for quantitative analysis. This work is presented in two parts. The first elaborates on how to generate a complete training set with MC X-ray simulations and acquired SEM data. The second describes the model architecture and steps for integrating model predictions during microscope parameter tuning routines.

Computational methods indisputably improve microscopy, as demonstrated in this thesis, with work achieved on the SEM. Progress towards integration of image processing generates smarter acquisitions, with considerably less effort, optimizing resources for analysis.

Résumé

Le contrôle de l’équipement d'acquisition d'images avec les logiciels de traitement d'images représente l'avenir dans le domaine de la microscopie. Actuellement, l’efficacité des techniques de microscopie sont limitées par des tâches manuelles et par le besoin de supervision des acquisitions par les microscopistes. Surtout, l'interprétation adéquate des données d'image nécessite une analyse quantitative avec des méthodes computationnelles. L'automatisation des tâches simples et complexes, particulièrement sur le microscope électronique à balayage (MEB), est réalisable en remplaçant des fonctions effectuées par les microscopistes, par des algorithmes logiciels appropriés et par l'apprentissage profond. Les acquisitions guidées sur le MEB, à l’aide de méthodes hybrides, forment la solution optimale vers un rendement plus élevé et pour la génération de données de qualité supérieure, plus rapidement.

Cette thèse présente des approches computationnelles à la microscopie. Tout d'abord, deux plans de travail sont décrits pour acquérir des données de manière automatisées, incluant les étapes d'imagerie et le traitement des images acquises dans Dragonfly, un logiciel de traitement d’image. Le premier est exécuté sans rétroaction : le faisceau à électrons est positionné séquentiellement dans une disposition de grille prédéterminée, les images sont recallées, les composantes d’intérêts sont segmentées, et des données sont calculées pour en effectuer l’analyse quantitative. Le deuxième est réalisé avec une boucle de rétroaction entre l'imagerie et l'analyse d’images. Cette technique est développée pour permettre des acquisitions dynamiques, d’abord avec la segmentation de composantes pour ensuite suivre ces composantes. Avec un positionnement intelligent du faisceau, le microscope est guidé vers les zones d'intérêt pour

imager. Cette solution est donc optimisée pour acquérir uniquement des données pertinentes et minimiser la durée des acquisitions.

Ensuite, les procédures de calibration du microscope, avant les acquisitions, sont améliorées en intégrant des méthodes computationnelles, axées sur l'intelligence artificielle. Les microscopistes dévouent beaucoup de temps à ajuster les paramètres du microscope et à aligner le faisceau jusqu'à ce que la qualité d'image souhaitée soit obtenue. Pour aider les microscopistes à interpréter la qualité des images, un modèle de régression est conçu afin d’optimiser les paramètres du MEB pour améliorer la qualité des images pour l'analyse quantitative. Le travail est présenté en deux parties, la première explique comment générer un ensemble de données pour l’apprentissage profond, comprenant de simulations de MC X-ray et des données acquises avec le MEB. La seconde décrit l'architecture du modèle développé et les étapes requises pour intégrer les prédictions du modèle lors des procédures de réglage des paramètres du microscope.

Les méthodes computationnelles améliorent incontestablement la microscopie, tel que démontré dans cette thèse, avec le travail réalisé sur le MEB. Les progrès vers l'intégration du traitement d'images génèrent des acquisitions plus intelligentes et nécessitant beaucoup moins d'efforts pour enfin optimiser les ressources pour l'analyse.

Contribution of Authors

The candidate is the first author for all manuscripts in the thesis, other authors contributed as references and were often consulted throughout the completion of the work. Specifically, the candidate's supervisor Professor Raynald Gauvin and co-supervisor Professor Mike Strauss, both regularly contributed ideas and resources that led to the progression of the project.

For the manuscript entitled *Workflow Automation of SEM Acquisitions for Nanoparticle Analysis* the code for workflows was developed by the candidate. Python scripts are implemented in a commercial environment, Dragonfly, with built-in libraries and frameworks. Indeed, algorithms used for image processing, such as mutual information and scale-invariant feature transform (SIFT) for stitching, the Otsu thresholding for segmentation, measurement computations from segmentations and conversion of the segmentation to a graph were all already implemented by the Dragonfly team. Presented workflows were created by the candidate by selecting appropriate algorithms, combining available tools, and applying them to the current context. The remainder of the code was designed and implemented by the candidate. Regarding all information about the SEM and the experimental set up, Nicolas Brodusch was often consulted for advice and ideas, especially for managing instrument instabilities.

For the two-part manuscript entitled *Optimizing SEM Parameters for Segmentation with AI*, the entirety of the code was developed by the candidate. The simulation data generated by the candidate used the MC X-ray plugin in Dragonfly implemented by the Dragonfly team and developed by Professor Gauvin's group. Outside the scope of this project, the candidate was one of the developers on the Dragonfly team that contributed to the integration of the MC X-ray plugin in Dragonfly. With prior knowledge on how to use the code to generate simulations, the

candidate generated simulated data autonomously. The entire training set and model were designed and built by the candidate. Benjamin Provencher and Nicolas Piché, both experts in deep learning, provided valuable feedback regarding the model architecture and model performance to improve results.

Acknowledgements

I would first like to acknowledge my thesis supervisor Prof. Raynald Gauvin, who has given me many opportunities to develop confidence in my work and present my research. Combined with the assistance of my co-supervisor Prof. Mike Strauss, both have provided the guidance and the expertise to allow me to excel in my research.

Also, I would like to thank my colleagues and the staff in the Prof. Gauvin's research group, especially for their assistance with training on the electron microscopes, with a special mention for Nicolas Brodusch.

To my colleagues at Dragonfly, who have developed image processing tools that have allowed me to have an excellent bases for creating more complex original contributions. A special mention to my employer Nicolas Piché, thank you for the constant support, advice, and laughter.

Last but not least, thank you to my mom, Simon and Lana, for being the best audience to the countless practiced presentations, giving feedback on lengthy documents and manuscripts and constant emotional support.

Thank you.

Contents

List of Figures

List of Tables

1 Introduction

Automation has transformed multiple disciplines, elevating the limits of productivity to new heights. Notably, microscopy is a field that greatly benefits from automation since routine workflows on microscopes consist of repetitive tasks. Efforts towards automation with computational microscopy, by integrating image processing tasks with image formation tasks, are increasingly relevant particularly for scanning electron microscopes (SEMs). The SEM is unequivocally a powerful tool for sample analysis, as it is extremely versatile due to the variety of signal channels that is available to collect. Interpreting the acquired SEM data requires quantitative analysis, which in turn, requires computational approaches.

Widely employed for research and industrial purposes, beam time on the SEM is a valuable commodity, in high demand. Scientists with the required expertise to manipulate SEMs can either purchase their own microscope with specific functionalities or schedule beam time on a microscope open to external users, if available. When techniques required for sample analysis are beyond their expertise, scientists also have the option of mandating more skilled microscopists to complete acquisitions on the provided sample. Regardless of the chosen method for obtaining results for analysis, all options are expensive and have limited access. The required analysis is therefore understandably executed as rapidly and efficiently as possible.

Currently, SEM workflows rely too heavily on microscopist labor and expertise, and instruments are not optimized for high throughput. Microscopists routinely book beam time for hours on the SEM, and that time is often not entirely productive. Ideally, beam time would be largely spent acquiring useful data for analysis. However, a high percentage of time is spent tuning

microscope parameters and aligning the beam, necessary steps at the start of any SEM session. Furthermore, acquisition workflows require constant microscopist supervision, microscopes therefore have a significant amount of downtime outside of working hours.

Automated image processing pipelines with the SEM, particularly with deep learning [1], pushes the boundaries of productivity and enables users to acquire more and better data for analysis. Functionalities range from simple to complex tasks and longer acquisitions can resume overnight when microscopes are normally inactive.

The objective of this work is to present developments made in computational microscopy by integrating image processing tasks and deep learning to actively control the SEM hardware to automate acquisitions and efficiently improve image quality for segmentation.

First, a thorough review of the relevant literature is presented. Generating code to automate workflows on the SEM requires expertise in both electron microscopy and in software development, specifically in deep learning and image processing algorithms. The intention is to create reliable and robust software to improve workflows by automating tasks and optimize productivity on the SEM. Then, the following chapter presents an automated workflow for acquiring images on the SEM in grid layout and with feature tracking. Subsequently, the two next chapters are Parts 1 and 2 describing workflows for generating a training set and training a regression model, respectively. The regression model is designed for optimizing parameters to set on the SEM during acquisitions with computational approaches. Finally, concluding remarks on the presented work will summarize findings, list contributions to original knowledge and present anticipated future directions.

2 Review of the relevant literature

2.1 SEM imaging

The scanning electron microscope is a complex instrument; multiple components are involved in generating high quality images. Understanding individual components and their functioning is an important first step to workflow automation on the microscope and integrating algorithms for acquisitions.

The SEM column produces a beam of accelerated electrons and scans the sample surface on a specific area. The electron beam is focused through the column by magnetic lenses with Lorentz force, to the sample inside an evacuated chamber. Once electrons hit the sample surface, distinct signals arise from the interactions between electrons and the material, collected by multiple detectors and visualized on computer screens. In fact, a multitude of signals can be acquired for analysis, which makes the SEM a very powerful tool to analyze objects from the micrometer to the nanometer scale. Sample scanning, detectors signal collection and image display are all simultaneously executed, allowing microscopists to examine their sample in real time.

2.1.1 Instrumentation

Essential SEM components include the electron gun source that produces the electron beam, the lenses, and apertures that direct, focus and filter the beam towards a specific point on the sample and the high vacuums that regulate the pressure in the chamber. The exact microscope hardware will vary from one instrument to another depending on requirements.

Electron guns are grouped into three main categories depending on the mechanism of electron emission: conventional guns by thermionic emission, hot cathode by Schottky effect and cold

cathode by tunnel effect [2]. Electrons are extracted from either a tungsten filament, a monocrystalline and oriented LaB_6 filament, a tungsten filament coated with zirconium oxide film or a monocrystalline tungsten tip [3]. Filament types are shown in Figure 2.1. Each electron source will have a different lifetime, brightness, probe current, probe diameter, temporal coherence, current stability which will impact the resulting aberrations and image quality [2]. The appropriate gun will depend on analysis requirements, sample types and desired resolution.

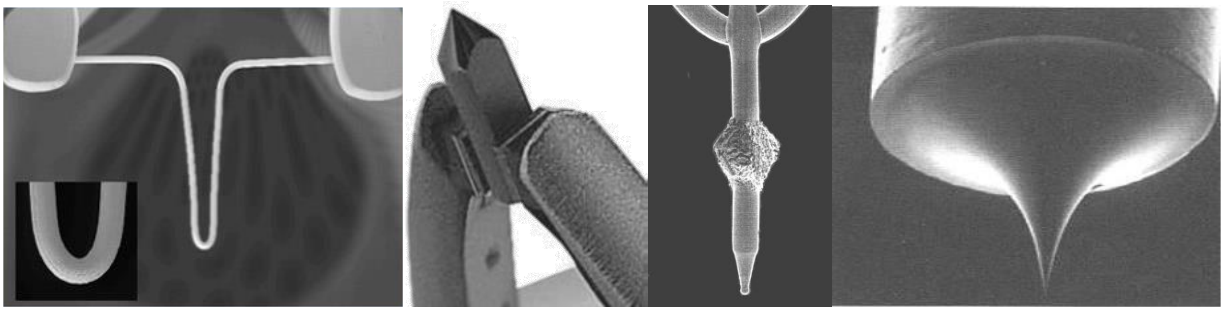


Figure 2.1 Electron emitter types from left to right: tungsten filament, LaB_6 filament, tungsten filament with ZrO_2 and tungsten tip [4].

Conventional SEM columns have condenser lenses with the purpose of reducing the beam size with cross over, and perpendicular scanning coils to deflect the beam to scan the sample surface [5]. The objective lens is the final lens encountered by the beam before reaching the sample, it will determine the focus of the beam.

2.1.2 Electron-matter interactions

The SEM column will produce and focalize the beam to the sample surface and the exposed surface is then imaged from the detected signals. In fact, the interaction of electrons with the sample surface creates a multitude of signals that can be acquired for analysis shown in Figure 2.2. Electrons travel inside the sample in a pear-shaped interaction volume, encountering sample

atoms and producing backscattered electrons (BSEs) through elastic collisions and secondary electrons (SEs) through inelastic scattering. Interaction volume size and the amount of generated signal depends on the sample material. Samples with higher atomic number atoms will generally produce more BSEs and the primary electrons will not travel as deep within the sample, shrinking the interaction volume. As seen in Figure 2.3, when comparing the interaction volume for atomic number (Z) of 13 and 79, the shape is restricted by the density of material atoms.

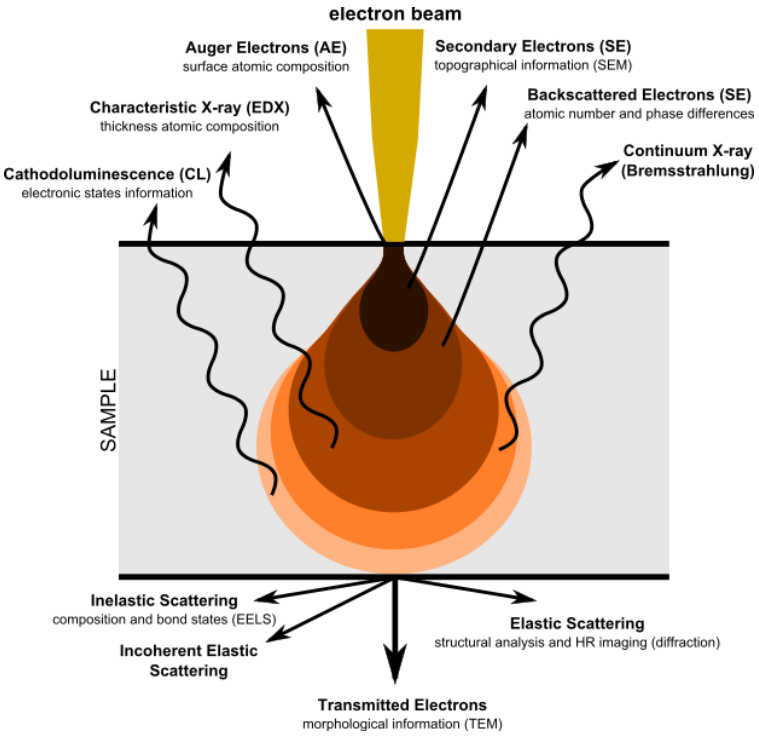


Figure 2.2 Signals generated from the interaction of the electron beam with a bulk sample. [6]

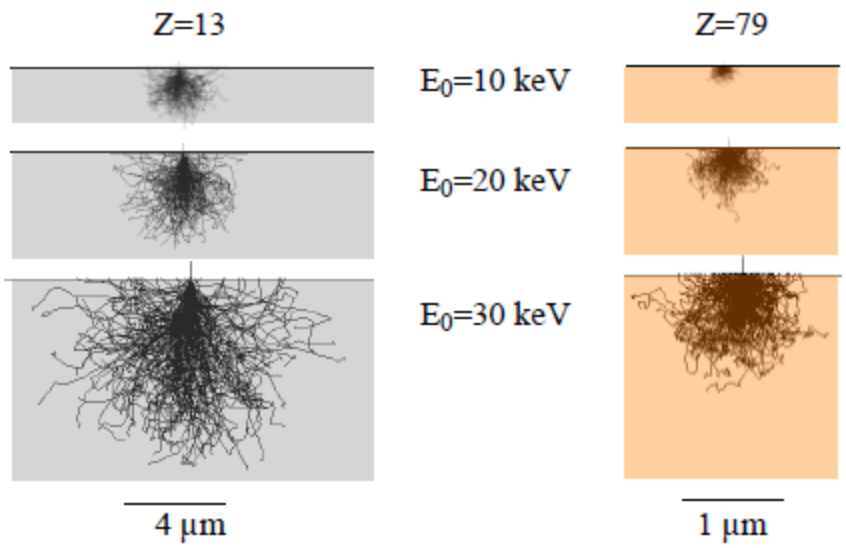


Figure 2.3 Interaction volume shape dependance on material atomic number or density [4].

Detector type and position will both determine the extent of the observed contrast of acquired signal. Normally, backscattered electrons (BSEs) are detected with solid state detectors, placed above the sample. The backscattered electron will hit the detector, a doped semiconductor material, and generate an electron-hole pair. The higher the energy of the backscattered electron, the more electron-hole pairs are generated [7].

Secondary electrons (SEs) are detected commonly with an Everhart-Thornley detector, essentially a scintillator within a positively charged Faraday cage. The positive charge will attract the negatively charged electrons and the scintillator accelerates the electrons, converting them to photons for image formation [8]. Positioning of the SE detectors will determine the type of SEs collected, leading to variable contrast.

More detectors can be attached such as energy dispersive spectroscopy (EDS) detectors for X-ray detection, contributing to image formation and maximizing the amount of collected information to analyze a sample. As seen in Figure 2.4, positioning multiple BSE and SE detectors in

different areas in the column or chamber will produce a unique contrast providing additional information for analysis.

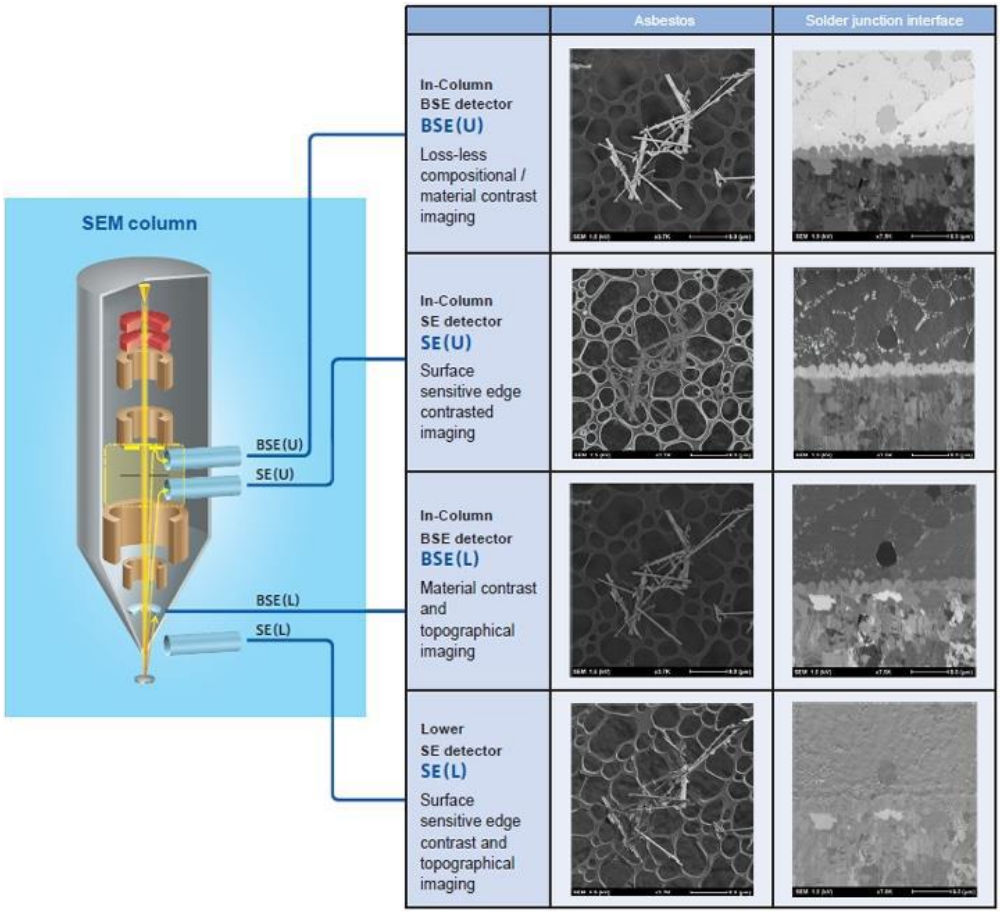


Figure 2.4 Schematic of available electron detector positioning on the NX5000, with examples of associated contrast obtained [9].

Mostly, detected BSEs will produce atomic density contrast, providing qualitative information on the sample's material composition. When the primary beam electrons are scattered, they can escape the specimen with high energy. Brighter regions in Figure 2.5 (right) designate areas with higher atomic number, where more BSE signal is detected, while darker regions designate areas with lower atomic number.

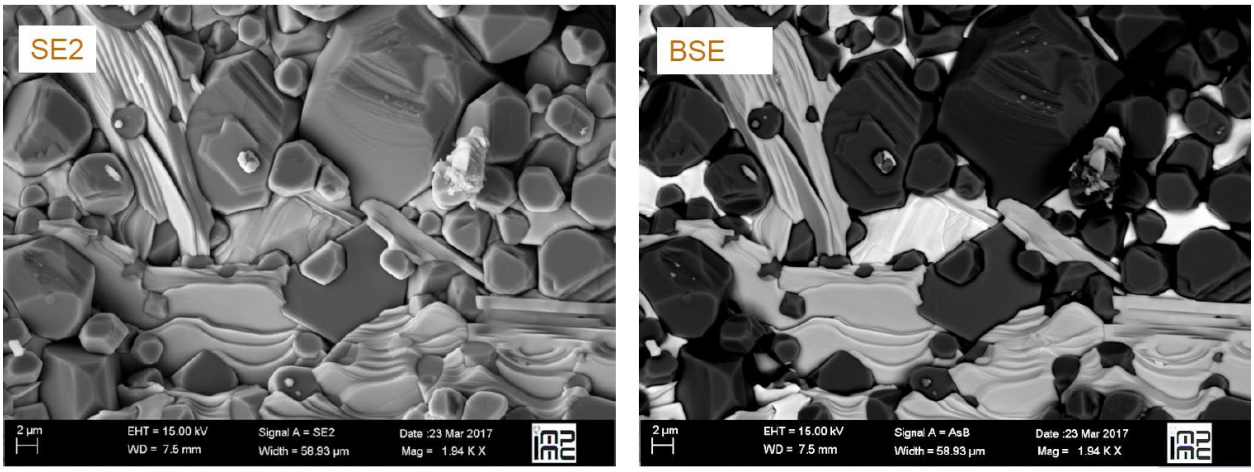


Figure 2.5 Examples of SE signal producing topographic contrast (left) and BSE signal producing compositional contrast (right) [4].

Detected SEs contain surface information generating topographic contrast shown in Figure 2.5 (left). The primary beam will dislodge specimen electrons from the surface, generating SEs. SEs will therefore have low energy, unable to escape from deep areas within the sample without being absorbed by the sample. As demonstrated in Figure 2.6, variations in sample surface topography will produce different amounts of SEs, which in turn generates variations in detected signal and high contrast.

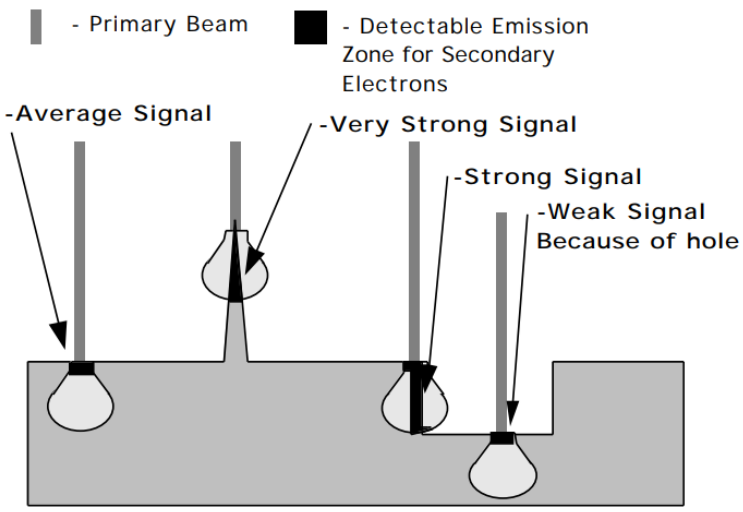


Figure 2.6 Variations of SEs produced with topography, generating signal fluctuations [10].

For the scope of the presented project, only SE and BSE signals are acquired. However, additional signals such as X-ray spectra from EDS detectors are entirely useful for analysis, giving more specific compositional information by attributing an element's characteristic peaks to areas on the sample [11]. Indeed, this occurs when electrons are dislodged from specific orbits of atoms in the specimen resulting in an emission of a characteristic photon. Moreover, electron backscatter diffraction (EBSD) detectors are available for further analysis of Kikuchi patterns with information on crystal boundaries and orientation [12].

2.1.3 Monte Carlo X-ray simulations

Simulated SEM data is extremely useful for many microscopy applications, including material characterization and parameter estimations. Mainly, simulated images are advantageous as they are simpler to generate than acquired SEM images. Simulations do not require typical sample preparation methods, there are no physical constraints when positioning the sample, column and

detectors in space and usual beam alignment routines are avoided. More importantly, simulations are a time efficient way to obtain a large amount of data for training deep learning models.

In recent years, there has been extensive development using statistical probabilities from Monte Carlo simulations to simulate electron trajectories [13]. Multiple physical models have been implemented, describing electron interactions with solids from low to high energies (0.1-30 kV) [14] and simulated results have been validated through comparison with SEM acquired results [15]. Implemented algorithms have succeeded because of the exponential progress in computational capabilities. With extremely powerful workstations, improved computational resources including more available memory and strong GPU power, large scale, realistic simulations of SEM images are possible in a timely manner.

Simulated SEM images are generated with Monte Carlo programs [14], [16] where electron trajectories into virtual samples are obtained from set microscope parameters. The beam accelerating voltage, probe size, number of simulated electrons, angle to the sample, beam direction and detector parameters are amongst the variables that are specified for simulations. Virtual samples are designed by the users, they consist of regions, with any composition from elements of the periodic table. Physical models used to describe cross sections, electron atom interactions, energy loss, absorption coefficient and more are also selected. Launched simulations produce multiple outputs, including generated and emitted x-rays, backscattered electron coefficient, maximum electron range, phi rho z distributions and more.

MC X-ray, specifically, is an extension of both Casino [17] and Win X-Ray [18], a Monte Carlo program with more features, improved to compute simulations on heterogeneous materials and a wider variety of geometries. MC X-ray was later integrated in Dragonfly [19], an image processing software, as the MC X-ray simulator plugin. Essentially, the plugin acts as a virtual

microscope. The architecture of the original code was reorganized, all the while preserving the main algorithms and core concepts. Two major improvements, specific to MC X-ray in Dragonfly, are first, the simulations run approximately 100 times faster, as electron trajectories were optimized with multi-threading. Due to the high amount of data required for training, accelerated simulations are especially beneficial for the scope of this project. Second, simulations are not limited to virtual samples with simple geometries. Virtual samples geometries were typically limited to multilayers, grain boundaries or spherical inclusions [14], whereas in Dragonfly, any virtual sample of any complexity can be designed. The interface in Dragonfly is also useful to visualize the simulation set up in 3D, positioning the beam, sample, and detectors in space.

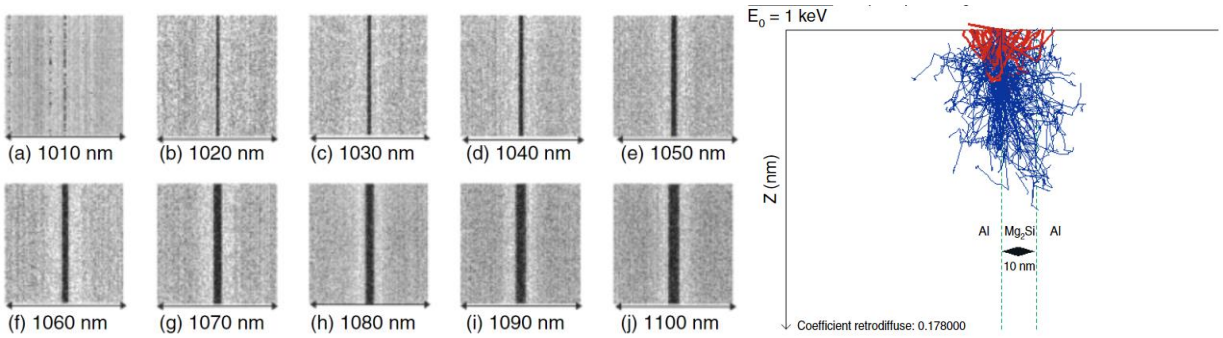


Figure 2.7 Outputs from CASINO program [14] (left) BSE images of an Al/Mg2Si/Al sample at 5 keV for different Mg2Si thicknesses and (right) electron trajectories in blue and backscattered electrons in red from simulation at 1 keV.

Electron trajectories can be simulated for specific points, line scans or complete image scans, with established resolution. Image scans will generate BSE images illustrated in Figure 2.7 (left) and EDS maps. Figure 2.7 (right) shows a lateral cross section of a sample with electron trajectories, where the interaction volume shape is available for analysis. The depth and lateral range of trajectories will depend on the simulation parameters and the sample composition. In

fact, image quality will vary depending on microscope parameters. Appropriate parameters can be evaluated with simulations to obtain the desired image quality, prior to experimenting on the SEM, minimizing sample exposure to the beam.

2.2 Microscope parameter optimization

Optimizing SEM parameters is performed iteratively with the beam alignment, where the beam must be adequately aligned at higher magnification than the intended magnification for acquisitions. Lens alignment and astigmatism correction are primordial for best image resolution and the focus is to be adjusted as well, to avoid beam distortions [20].

Tuning SEM parameters, after loading the sample and aligning the beam, is routine for improving or adjusting the image output. Highlighting features of interest with microscope parameters will not only depend on the instrumentation but also on the specimen material. Therefore, SEM parameters are optimized differently for every acquisition [21]. A collection of parameters is involved in generating the desired image quality with SEMs. Ideally, images are considered to have optimal quality when features are efficiently distinguished and identified, achieved with high contrast, best spatial resolution, high signal-to-noise ratio, and low acquisition time. However, compromises are inevitable and middle grounds depend on what material is being imaged and the type of signal to be detected. Signals available to detect are covered in the previous Section 2.1.2 on electron matter interactions (SE, BSE, EDS, EBSD).

Post processing approaches with algorithms that improve image quality with filtering, denoising and super-resolution methods are also available rather than dynamically tailoring microscope parameters during acquisitions. Ideally, with such powerful computational methods, adaptive

acquisitions are preferred, where computation is used synergistically with microscope parameter optimization as demonstration in Figure 2.8 [22]. The image signal is manipulated during acquisitions with the electron beam settings: the incident energy, the current, and the magnification. Additionally, settings used to capture images will contribute to image quality: acquisition time, image size and integration number.

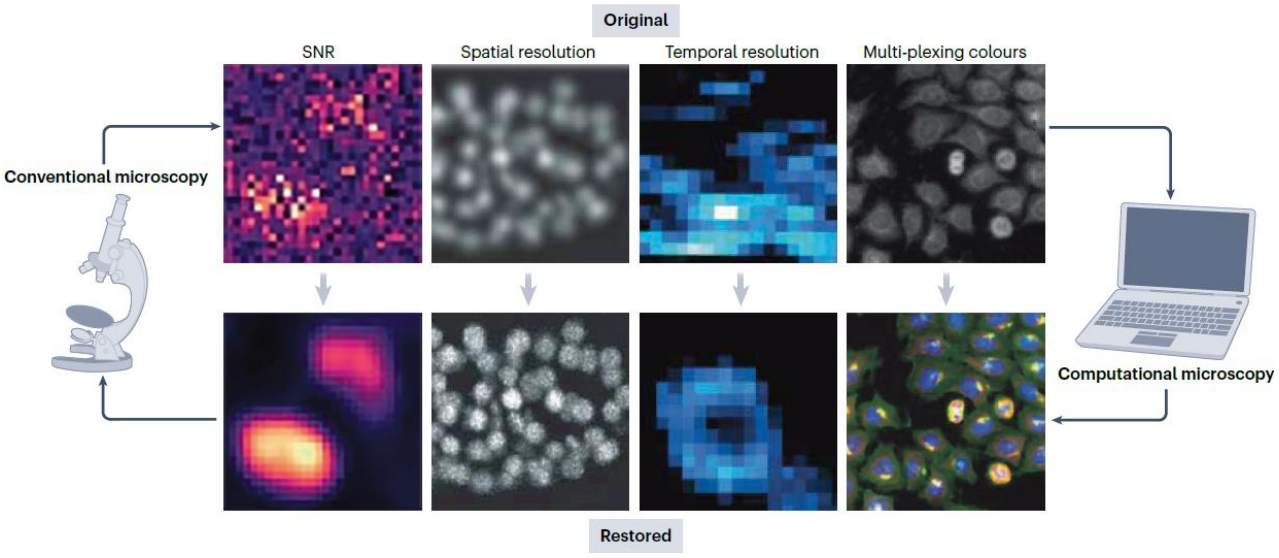


Figure 2.8 Combining conventional microscopy with computational microscopy to improve image quality [22].

2.2.1 Beam energy

The accelerating voltage determines the beam energy and the depth of penetration of electrons into the sample. SEMs operate at energies ranging from 1-30 keV [23], and samples are typically bulky rather than thin. Higher energies are favorable for thin samples since electrons will penetrate more efficiently and with less beam scattering, towards detectors underneath the sample, as illustrated in Figure 2.9b. For thin samples, increased beam energy will increase the image resolution, whereas selecting the optimal beam energy for bulk samples is not as

straightforward. There are many implications relating to the interaction volume of electrons into bulk samples, shown in Figure 2.9a.

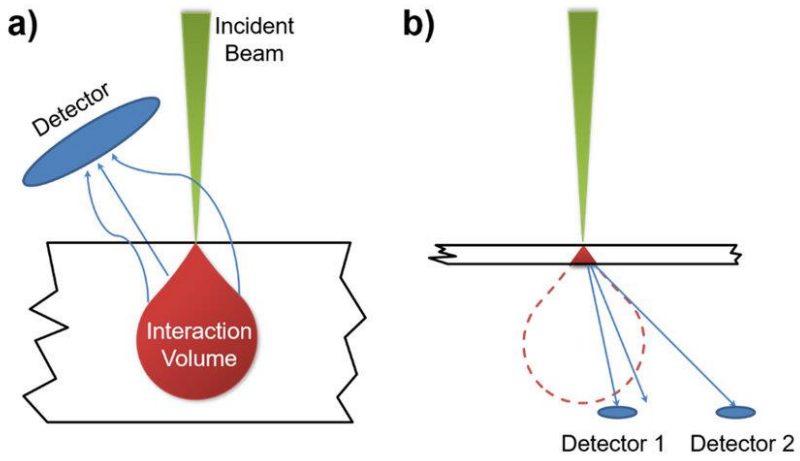


Figure 2.9 Interaction volume of electrons into a) bulk and b) thin specimen [24]

With bulk analysis, interaction volume of electrons in the sample is an important consideration since it will impact the resolution, observed contrast and charge accumulation. Simulated interaction volumes, to scale, for different energies into Cu are shown in Figure 2.10, demonstrating the decreasing lateral and depth range of the electron trajectories as the beam energy decreases. Interaction volume shape will also vary with specimen density, shown in Figure 2.3, higher density will restrict electron trajectories, while lower densities samples will permit electrons to travel larger ranges.

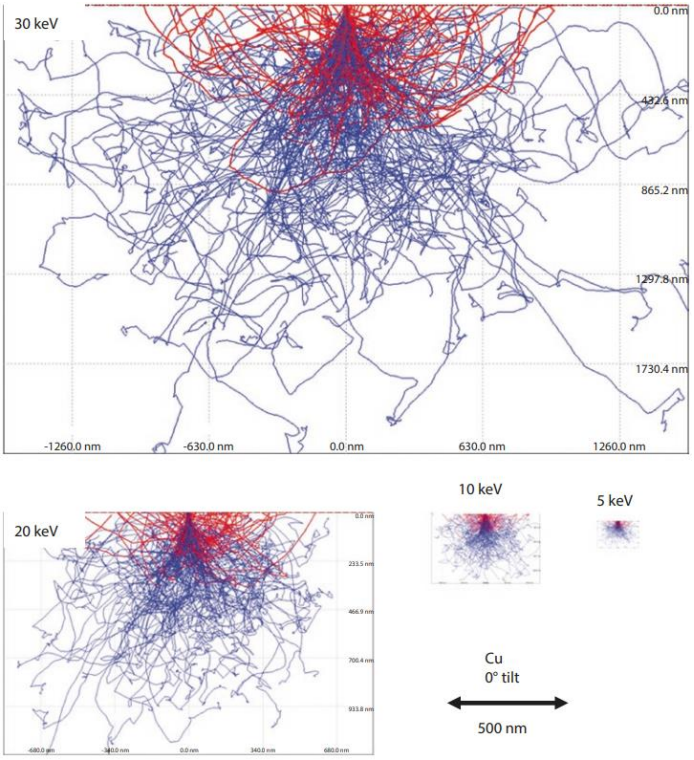


Figure 2.10 CASINO Monte Carlo simulations for different beam energies [23]

Depending on the imaged material, low SEM electron beam voltages (< 3kV) are often favorable to help avoid surface charge build up, improve lateral resolution, and reduce beam damage associated to high beam doses. Surface charge build up is observed as the charging effect, it is well exemplified in Figure 2.11 (left). Microparticles imaged at 15 kV have strong brightness with obscured surface features, whereas at 5 kV, Figure 2.11 (right), particles are well defined. However, lower beam energies generate images with lower contrast as well as providing only surface information. Optimal beam accelerating voltage is essentially achieved when images have sufficient contrast. Objects without a threshold contrast cannot be distinguished from noise of background fluctuations [20].

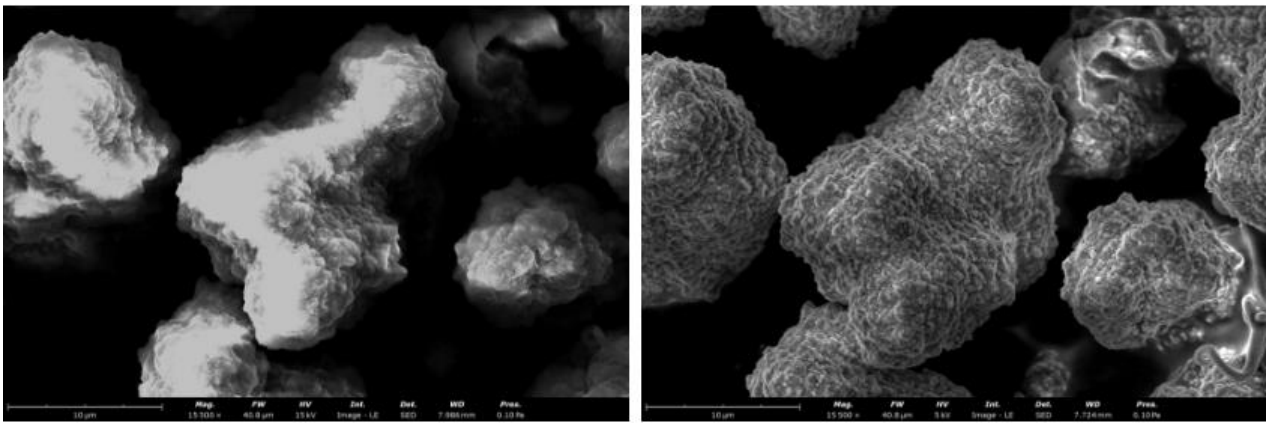


Figure 2.11 Images of the same specimen captured at 15 kV (left) and 5kV (right) with strong charging at 15 kV [25]

2.2.2 Probe current

Electron probe current is the number of electrons that hit the sample surface per unit of time, and it is directly related to the signal-to-noise ratio (SNR). Without sufficient current, images are very “noisy”, and the scan speed should be decreased, or the acquisition time prolonged, when capturing images, to compensate. However, with excessively high currents, the probe diameter increases, and the resolution is compromised leading to loss of information. When selecting the probe current, there is an implied tradeoff between the SNR and spatial resolution. The conductivity of the imaged sample also determines the relevant current to apply. Non-conducting samples require lower currents (10 to 100 pA) to avoid charge accumulation and damage, while conducting specimens can handle higher currents (100 to 1000 pA) to improve the amount of signal and contrast [21].

Noise in images is inevitable as there are multiple sources, other than lack of signal, that cannot all be managed. For instance, imaging systems introduce noise, from analogue-to-digital conversion by detectors. The appropriate probe value is therefore related to the proportion of signal to noise, quantifying the amount of useful information in images. Probe current values are

tuned to obtain sufficient SNR as the amount of signal and noise is fundamentally tied to the observed image quality [22]. Identifying features of interest requires enough signal, without lowering resolution to where features of interest can no longer be resolved. The effects of the probe current are shown in Figure 2.12 as the current is increased from left to right. The middle image is the best compromise between SNR and resolution, where ridges can be observed and are otherwise hidden in the left image by noise and in the right image by low resolution.

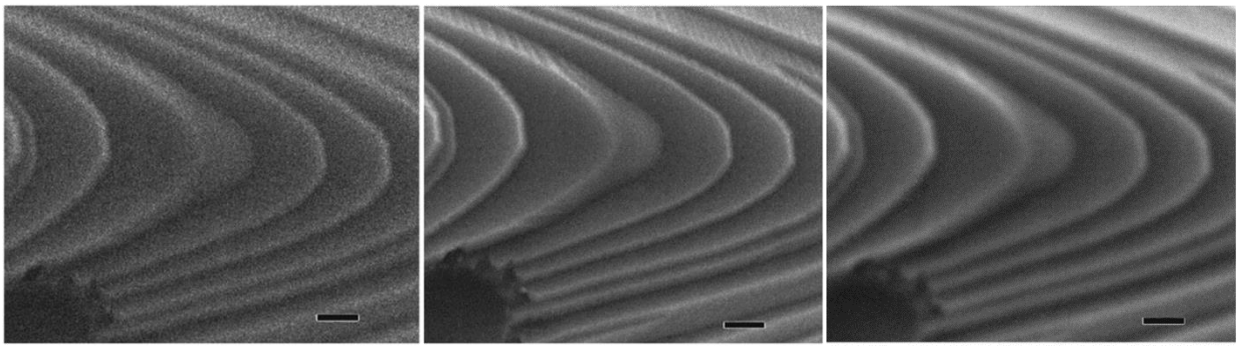


Figure 2.12 Effect of the probe size from the probe current on the image quality [2] (left) probe diameter 15 nm, probe current 1 pA, (middle) probe diameter 20 nm, probe current 5 pA and (right) probe diameter 130 nm, probe current 320 pA.

2.2.3 Magnification

The parameters selected may vary depending on the magnification used. Higher magnifications will concentrate the beam on a smaller area of the sample and a persistent beam on one region will result in a higher flux of electrons for the same parameters at lower magnifications. A high flux can cause beam damage, contamination, and sample drifting. These all contribute to reducing image sharpness and lead to charge accumulation at the surface, introducing artefacts in the image as seen in Figure 2.11. It is therefore preferable to reduce the time spent scanning the surface at high magnification or decrease the beam current and energy accordingly to reduce the electron flux.

2.2.4 Capture settings

When capturing an image, the scanning speed, image size and integration number are specified to reduce the noise quantity. Longer scanning speeds, large images and higher integration numbers will lead to clearer image, sharper edges, and smoother surfaces, at the expense of the acquisition time. Scan speed is the time spent per pixel, image size is the number of pixels in x and y per image, and the integration number is the number of frames or lines used to average the final image. Integration number can be beneficial to reduce noise, and if the sample is stable, reduce charge accumulation. However, it also introduces a time-gap, where image shift or specimen shrinking can occur. Higher integration number can ultimately lead to lower image quality or sharpness. When comparing the images in Figure 2.13, there is an obvious noise reduction when increasing the integration number.

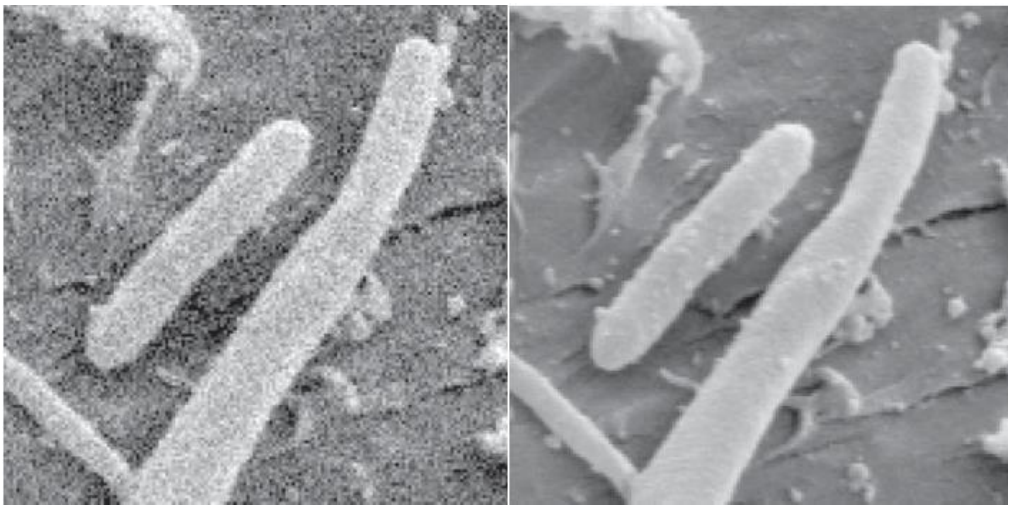


Figure 2.13 Short integration time, noisy image and longer integration, noise free image [20]

2.3 Workflow automation

Electron microscopes have evolved from optical instruments into complex systems, requiring computer interfaces due to digitization and technological advances. Automation is an area of interest in the development of applications on the SEM, where repetitive and tedious tasks should be completed through microscope control. Automation improves reproducibility by minimizing human bias and enhances data-adaptive imaging methods. Algorithms are more reliable than humans at undertaking repetitive tasks that require constant concentration over long periods of time. Controlling the SEM with hybrid methods, integrating computation software and microscope hardware for successful workflow automation, has specific requirements. Three distinct group of experts are necessary to develop optimal tools for automation: microscope hardware experts, software developers and scientists that use SEMs for their research [26]. Feedback loops between microscope hardware components and image processing software is key to automating simple to complex workflows on the SEM.

2.3.1 Requirements

Software for microscope control requires a complete API for external communication to automate acquisitions. SEM control is normally achieved through proprietary software installed on a computer (PC SEM) connected to the microscope. However, the PC SEM does not connect to the internet, and it runs in a very controlled environment to avoid corruption from external access. As such, external communication can only be realized through socket communication by TCP/IP protocol, with an external computer connected directly to the same router, shown in Figure 2.14. The external computer (EXT PC) will send commands in text format to the PC

SEM, to set or get parameters and capture images. Information is then received, and images transferred to the EXT PC for analysis. EXT PC does not run in a controlled environment and with strong computational power and sufficient data storage space, it can run image processing software with the intelligence to automate tasks.

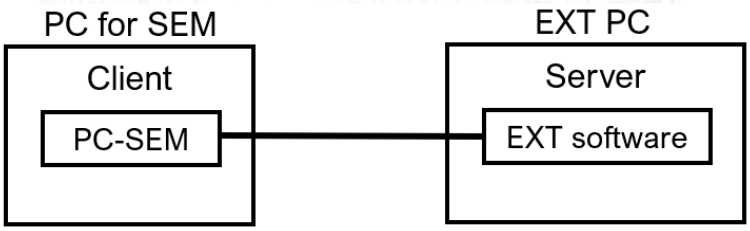


Figure 2.14 Communication between the PC-SEM with the proprietary software is done through socket communication by TCP/IP protocol with and external computer (EXT PC)

SEM workflow automation should also reduce the steep learning curve associated with using the instrument. A flexible, maintainable, open-source package with the possibility to customize protocols is a main system requirement [26]. If manual operations are easier than navigating the automation software, users will likely avoid automation. Software should be designed with object-oriented programming, where abstract classes allow for easy integration of any future microscope and in open-source language such as Python.

Robustness is another important automation requirement since many applications currently require reactiveness and unplanned interventions. Adaptive software is achieved by engineering methods to recover from system malfunctions and adjust to imaging conditions with feedback between analysis and acquisitions and reliable microscope parameter selection. Minimal microscopist involvement requires, for instance, error detection or loss of focus detection during acquisitions and subsequent seamless recovery by triggering the appropriate corrective routine [27]. Mechanical instabilities and charging can produce inconsistencies requiring drift correction

and post-processing for image registration. Drift correction is accomplished to ensure that images remain in the user-defined drift correction box [28]. For instance, realignment in the software SerialFIB is completed only with beam shifts, based on image cross-correlation. The process consists of registering a low current (10 pA) image of the current view with the initial reference image [29]. As for selecting the appropriate parameters, algorithms are available such as deep learning refocusing methods for SEM images, and proposed convoluted neural networks can be applied to accelerate acquisitions [30]. Also, a specific region or an autotune box can be used as a designated area to maintain good image quality, without exposing the entire surface to the beam [31]. The amount of time in which operations can be executed without supervision depends on the instrument itself, and its compatibility with prolonged operation times.

2.3.2 Overview of current solutions

Solutions for automation in microscopy are relatively new and often rely on commercial software and hardware, leading to challenges when attempting to develop new applications that integrate specific components [32]. Limitations have led researchers to develop custom software, in proprietary programming languages, often tailored to a specific technique and difficult to adapt to different microscopy modalities. A common framework, accessible to researchers, is necessary to break barriers to efficient microscope control [33].

Orchestrating the actions of microscope components with software is possible with µManager, a software that enables novel approaches to computer control of microscopes [34]. With a translation layer that converts code to Python functions and objects, Pycro-Manager has implemented the flexibility available with µManager in a much more user-friendly programming

environment [33]. Although μManager is well established, it does present some limitations for more complex microscopy modalities such as performing image acquisition and image analysis tasks simultaneously. MicroMator software, also in Python, incorporates reactivity with implemented Triggers and Effects during image acquisition loops [27]. Python-based software is more compliant to further development and convenient for debugging [32].

AutoscanJ is a solution that enhances acquisitions by targeting areas of interest and driving the microscope to corresponding locations. Selective imaging dramatically reduces the amount of data acquired and by consequence the acquisition duration. As a suite of ImageJ scripts, it is compatible with motorized microscopes for autonomous operation of selective imaging [35].

SerialEM [36] for TEM workflows and SerialFIB [29] for FIB workflows are user friendly software that offer interfaces for easy execution of protocols. Several modalities require reconstruction algorithms to obtain images from microscope acquired data. Software that can perform image analysis of data in real-time are highly beneficial.

2.3.3 Feature detection

Workflow automation requires software to control the microscope hardware to reposition the field of view on the sample and to adjust parameters for beam alignment. Feature detection is also a necessity for workflow automation, to reliably guide acquisitions. Repetitive repositioning tasks completed by microscopists are automated with algorithms for feature detection. Mostly, identifying imaging sites leads to capturing images of regions of interest exclusively, saving a

considerable amount of time. Also, it will substantially reduce the terabytes of data that would be collected by capturing the entire sample surface without excluding non-informative areas.

Feature detection is initially accomplished by capturing an overview image from the SEM, that serves as a reference image, to simplify subsequent stage navigation through target identification. Stage movements or beam shifts are relative to the stage coordinates in the initial reference image. The initial step is to detect features for site specific imaging. Identification of target sites can be achieved through segmentation algorithms, including segmentation with deep learning, of the overview image. A feedback loop is implemented between feature detection through model predictions and beam positioning for SEM imaging. Segmented features are translated to positions in space either by centroid identification or by conversion to a graph of vertices and edges. Centroids are useful when features are sporadic and separated, while graphs are useful to track connected features that span large areas.

Imaging sites can then be targeted sequentially by centering the SEM with stage movements, corrected for backlash and drifting. Compensating stage shifts, larger micron scaled movements, with beam shifts, smaller nanometer scale movements, is implemented during the backlash correction step for increased targeting precision. Indeed, generally, the stage movement functions are used for large distance offsets (> 1 μm) while beam shift functions are used for subtle corrections.

Once imaging sites are detected, next steps include adjusting the field of view with magnification, stage rotations, and beam shifts to keep the entirety of features contained in the image as demonstrated in Figure 2.15. Positioning the beam at the imaging site does not guarantee that the full feature is properly included, demonstrated in Figure 2.15A and verification with appropriate adjustments are necessary. Increased magnification will scan a

smaller area of the sample, potentially optimizing the acquisition box. The accuracy of the target site positioning will depend on the segmentation result.

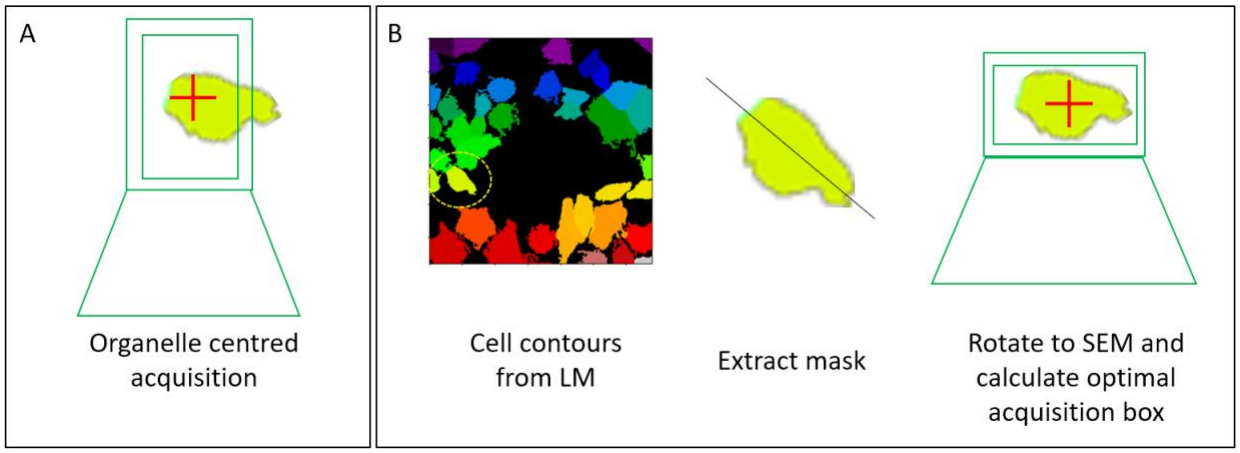


Figure 2.15 A. Centering the acquisition box to the feature of interest with part of the organelle outside the field of view. B. Optimal acquisition box computation with extracted mask, followed by stage rotation to fully contain the feature to be imaged [31]

Detecting features in SEM images regardless of noise, contrast, and intensity is much more successful with deep learning algorithms [37][38]. Other machine learning algorithms or feature point detection algorithms are sensitive to image variations such as brightness, contrast, resolution, and stigmatism. Consequently, segmentation with deep learning is prioritized, models are straightforward to train, specifically in image processing software Python environments. Workflows are implemented in Dragonfly, with the Keras TensorFlow deep learning engine, for training and applying a segmentation model [39]. Models are easily integrated during acquisition workflows for reliable feature detection. Pretrained models are provided by the software [40], but new models can be trained and tailored to specific data.

2.4 Deep learning models

Automating workflows on the SEM with computational methods improves microscopists' productivity, generating more data efficiently [22]. Additionally, it can assist microscopists in selecting the appropriate parameters for acquisitions, achieved by training artificial neural networks. AI aids in decision-making in complex situations, where there are multiple microscope parameters involved in producing high quality images and evaluating which combination is best, can be overwhelming [41]. Indeed, each parameter will have an individual impact on the output image quality and the result will depend on the sample composition. Therefore, when operating the instrument, selecting the correct parameters to obtain high quality images does require an expertise and prior knowledge on electron material interactions. Ultimately, workflows on the SEM are enhanced and much more accessible with data driven microscopy, by integrating deep learning models to automatically suggest microscope parameters to set during acquisitions.

Deep learning models for decision-making are separated into regression and classification problems. Classification will predict categorical or discrete outputs. It is applied by segmentation models, where each pixel in the image is labelled to a specific class. Regression models will predict a continuous value, or multiple continuous values from inputs [42]. A familiar example of regression is predicting the price of a house from multiple inputs, image data and numerical data: area code, square footage, number of rooms [43]. Trained models that suggest parameters will solve a regression problem, as opposed to classification. Both regression and classification algorithms that use supervised learning, are trained with available data that consists of pairs of inputs and outputs. Inputs are labelled data with desired outputs or ground truth. Building a model for a particular context, with specific data, requires an understanding of deep learning

algorithms, not only to optimize model performance, but also to have a general idea of what to change if the model is not predicting properly.

The following literature review on machine learning models is concise. The statistics and linear algebra involved in algorithms is only briefly covered and complete overview of concepts are available in other reference material [42], [44], [45].

2.4.1 Linear models

The most simplistic way of understanding how models predict outputs from input data is to explore linear models: linear regression and logistic regression. From there, more complex, and expressive models are built.

As the name states, linear regression solves for regression with a linear function that links the input to the output as shown in Equation 2.1, where *f(x)* is the output, *x* are the inputs and *w* are the weights. For one input, or with dimension (D) equal to one, the equation is reduced to Equation 2.2 and the remaining weights are w_0 and w_1. Linear regression can be applied to higher dimensions and the concept remains. The weights can have any combination of values, as shown in Figure 2.16 (left), resulting in different linear fits. The objective is to find the weight values that will best fit the data. The best fit is found by introducing a loss function, that will quantify the accuracy of the linear fit [44]. With supervised learning, the loss is computed with the provided ground truth. For linear regression, the square error loss (L2 loss) is commonly used, where for each data point value (y), the square difference from the predicted value from the linear function ($\hat{y}$) is computed as shown in Figure 2.16 (right). The error for each data point is summed and assigned to the combination of weights that generated the linear fit. During

learning, the model will attempt to minimize the loss, or the difference between the real and predicted values, by tuning the weights.

$$f_w(x) = w_0 + w_1 x_1 + \ \ldots + w_D x_D = w^T x \qquad 2.1$$

$$\boldsymbol{D = 1} \qquad f(x) = \hat{y} = w_0 + w_1 x \qquad 2.2$$

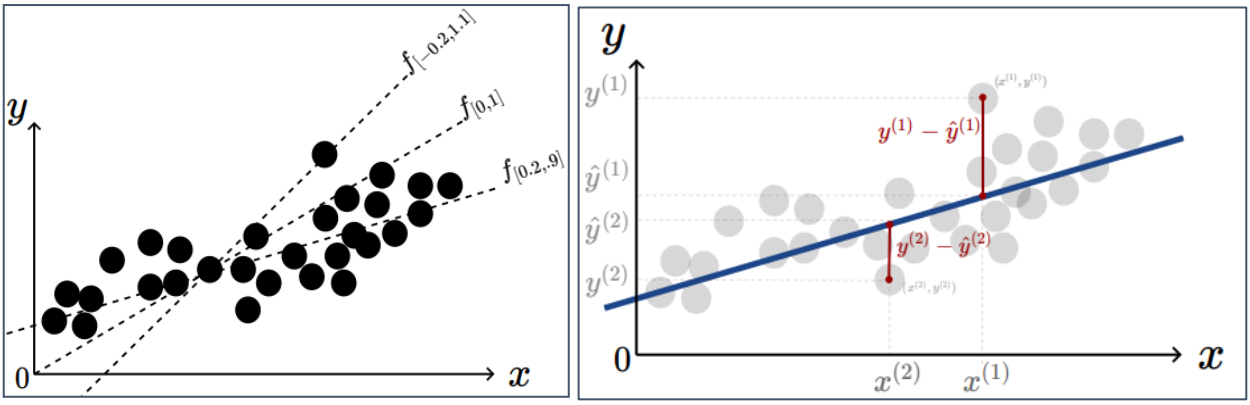


Figure 2.16 (Left) Tuning values of w_0 and w_1 will generate three different linear fits to describe the collection of datapoints. (Right) The best linear fit is found by minimizing the loss function that compares the real data point value (y) with the predicted value (ŷ). L2 loss computes the squared differences between the two values. [46]

Logistic regression adapts linear regression to classification. Linear regression will fit the data, while logistic regression will separate the data, as shown in Figure 2.17. As a linear classifier, the objective is to find the most appropriate way to separate the data into classes, achieved with the appropriate loss function.

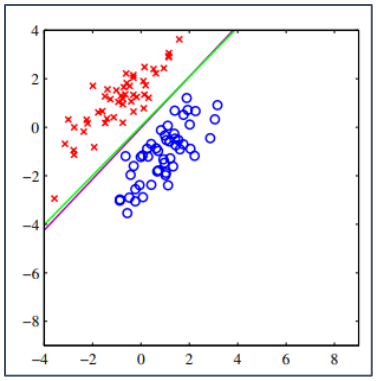


Figure 2.17 Linear classifier that correctly separates the data in red and blue [44]

Linear models are great tools that fit data efficiently and reliably, they are however, limited by linear functions. Data is rarely well defined by linear functions, and more expressive models are required for optimized model prediction accuracy. Fitting non-linear data is achieved by using nonlinear basis functions, such as polynomial, Gaussian or Sigmoid bases shown in Figure 2.18. Nonlinear bases are applied to the inputs, tweaking them for a better fit, maintaining the linear form, Equation 2.3 is the modified form.

polynomial bases $\phi_k(x) = x^k$

Gaussian bases $\phi_k(x) = e^{-\frac{(x-\mu_k)^2}{s^2}}$

Sigmoid bases $\phi_k(x) = \frac{1}{1+e^{-\frac{x-\mu_k}{s}}}$

Figure 2.18 Example of nonlinear bases used to tweak inputs x for higher expressive power of linear models.

$$f_w(x) = \sum_d w_d \phi_d(x) \qquad 2.3$$

Improvements are apparent when comparing the model fits in green in Figure 2.19. In both images the blue simulated function is the true fit, where a small dose of noise is added or

subtracted to each data point to obtain the blue data points. On the left, without nonlinear bases, the model generates a linear fit, however on the right, with applied Sigmoid bases to each input, a much more expressive fit is obtained, very similar to the true fit. Model predictions from the fit on the right will obviously generate a lower overall loss, than predictions from the linear fit on the left.

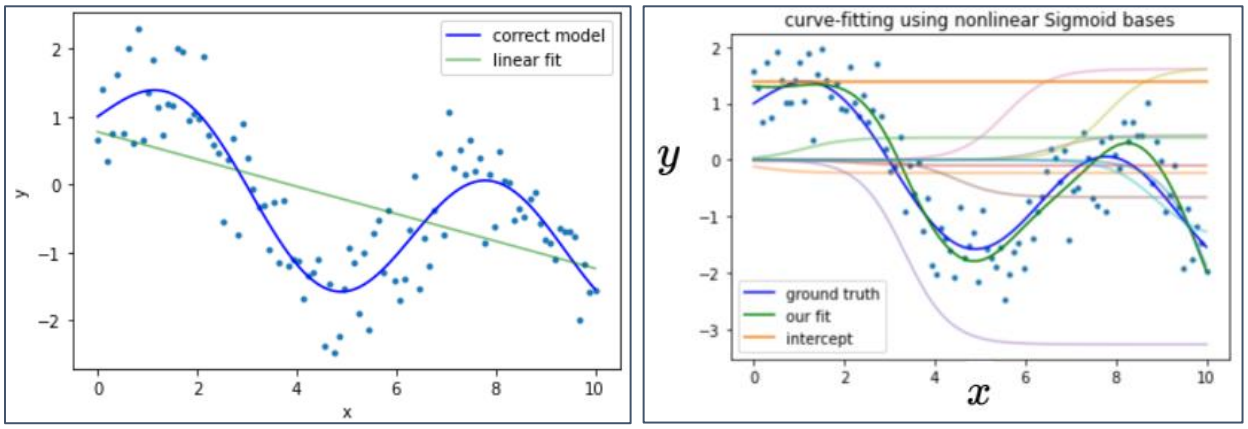


Figure 2.19 Simulated function in blue showing the true fit. Data points in blue obtained by adding and subtracting noise to simulated function data points. (Left) Best linear fit in green obtained for blue data points. (Right) Best fit with nonlinear Sigmoid bases in green for blue data points [46].

2.4.2 Multi-layer perceptron

Linear models are limiting since they do not integrate the interaction between any two inputs during training. The multi-layer perceptron (MLP) will connect multiple inputs inherently as a neural network, its architecture is biologically motivated. The model will approximate a complex function that maps an input to an output, from the provided data. The architecture is represented by composing multiple different functions shown in Equation 2.4. Each function represents a hidden layer with a series of nodes analogous to neurons illustrated in Figure 2.20, termed hidden since the output of each layer is not seen, only the output of the final layer is obtained.

The number of hidden layers will determine the depth of the model and the number of nodes in each hidden layer determines the width. A nonlinear base can be added to inputs in each layer in the network to generate a high expressive power. Ultimately, with the correct depth and complexity, MLPs can describe any smooth function with an appropriate fit.

$$f(x) = f^3(f^2(f^1(x))) \qquad 2.4$$

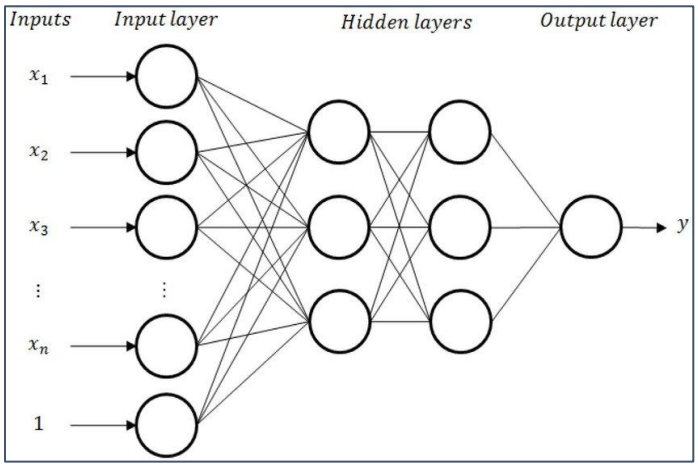
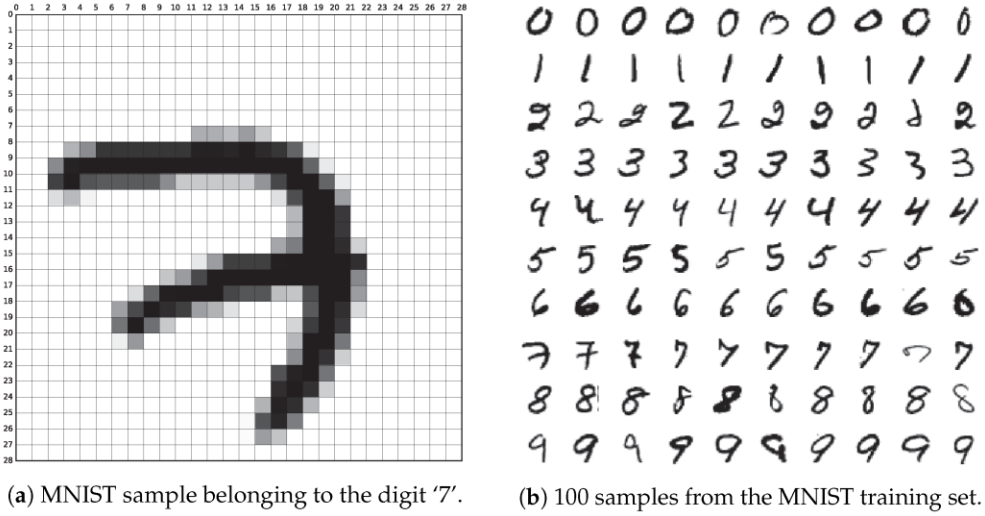


Figure 2.20 Fully connected neural network representing the MLP architecture with two hidden layers, three nodes each. [47]

The MLP is basically a series of logistic regression models at every layer, with the final layer set as either another logistic regression if the model does classification or a linear regression function if the model is solving for regression.

A popular example of applications of the MLP is with the MNIST dataset [48], where ten thousand labelled handwritten digits are available for training and classification, a training example is shown in Figure 2.21a. The input is a vector from a 2D image of the handwritten digit. All images are 28x28 pixels, vectorized into a 1D vector of 784 pixels with grey scale values. Each pixel in the vector is an input *x* connected to the hidden layers in the model and the output is the predicted digit from 0 to 9. During learning, training examples will tune the weights

w in the layers to predict the digit appropriately according to the computed loss. Only weights are changed during training; the connections, layers, and nodes are not added or removed. The weights are figuratively like knobs and dials that are tweaked during training, allowing the model to predict differently. The best weight values are found by minimizing the loss, as previously seen, and the model's prediction accuracy is computed.



(**a**) MNIST sample belonging to the digit '7'. (**b**) 100 samples from the MNIST training set.

Figure 2.21 a) Training example of an image 28x28 showing a handwritten 7 in the MNIST dataset. b) Handwritten digits range from 0 to 9 and shape will vary slightly depending on the example [49]

2.4.3 Convolutional neural networks

Although the MLP predicts quite accurately with the MNIST dataset (up to 98% accuracy [50]), the model learns nothing about the image structure. The 784 pixels in the 1D input vector could be shuffled, then used to train the model and a similar performance would be obtained [51]. Its high performance is most likely attributed to the large amount of training data available, tens of thousands of images. Also, for all training examples, digits are of similar size and positioned similarly in images (no big blank spaces). When digits are scaled and repositioned in the image as shown in Figure 2.22, the MLP algorithm does not predict accurately. In fact, MLPs are most

appropriate when inputs are numerical, whereas for image data, convolutional neural networks (CNNs) are implemented.

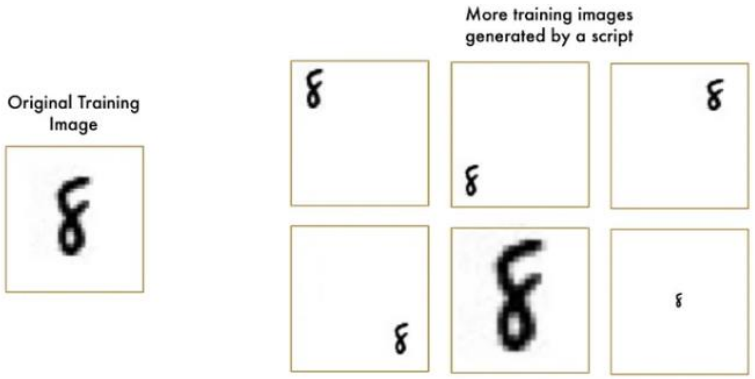


Figure 2.22 Scaling and repositioning of original digits to modify training examples and test model performance [52].

CNNs are better suited for image data because they preserve the 2D image structure, without collapsing the entire image into a 1D vector. Vectorization leads to loss of spatial information. The model will be more reliable and capable of scaling to bigger tasks when it can learn about neighboring pixels and understand how inputs are related to one another. CNNs are a specialized kind of neural network for processing data that has a known grid-like topology, such as 2D grids of pixels.

The architecture of the CNN is the same as the MLP, both models train identically. CNNs are defined by convolutional layers, with filters that apply a mathematical operation called convolution, a linear operation [45]. Filters, also known as feature extractors or kernels, will detect patterns such as shapes, edges, objects. The values in the filter matrix will determine the specific detected pattern, for example horizontal and vertical edges of digits illustrated in Figure 2.23. The convolution operation will create a 2D map, highlighting certain features found in the

input, termed a feature map. The feature map extracted from one convolutional layer is the input to the next layer.

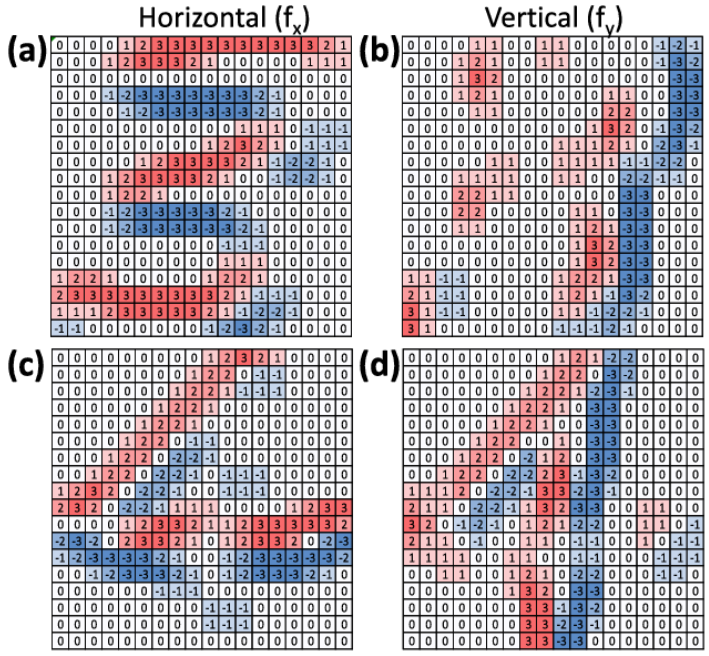


Figure 2.23 Convolution results from filters that detect horizontal and vertical edges for digits 3 and 4 [53]

Also, instead of a fully connected network like with the MLP, sliding a filter over the input will generate parameter sharing, where weights in the hidden layers are tied together. The value of a weight applied to one input is tied to the value of a weight applied elsewhere. Instead of using the full weight matrix, there is sparse connectivity, where patches of the input are connected to the hidden layers. Input patches are therefore building blocks that each describe only local interactions [45]. Much fewer parameters are stored with sparse connectivity, reducing memory requirements, improving statistical efficiency, and requiring fewer operations.

As the model depth increases, patterns detected by filters transition from abstract, or low level to detailed, or high level. For example, with the MNIST dataset, at the first layers the model would

learn to detect horizontal and vertical edges and the second layer would learn to detect shapes, like circles and lines that make up numbers.

2.4.4 Architecture design

Multiple established deep learning model architectures exist, for classification or regression and supervised, self-supervised or unsupervised learning. Model architecture depends on the input data, numerical, categorical or image, and the desired result, segmentation, denoising, super resolution, language generation and more [54]. Existing model architecture, often referred to as 'black boxes', can be trained with any individual training set to perform desired tasks successfully without any required expertise on deep learning. When training data is not available, it is even possible to skip the training step and simply use pretrained models to predict an output. Selecting the appropriate model type for training depends on the defined task and decision trees such as Figure 2.24 serve as useful guides.

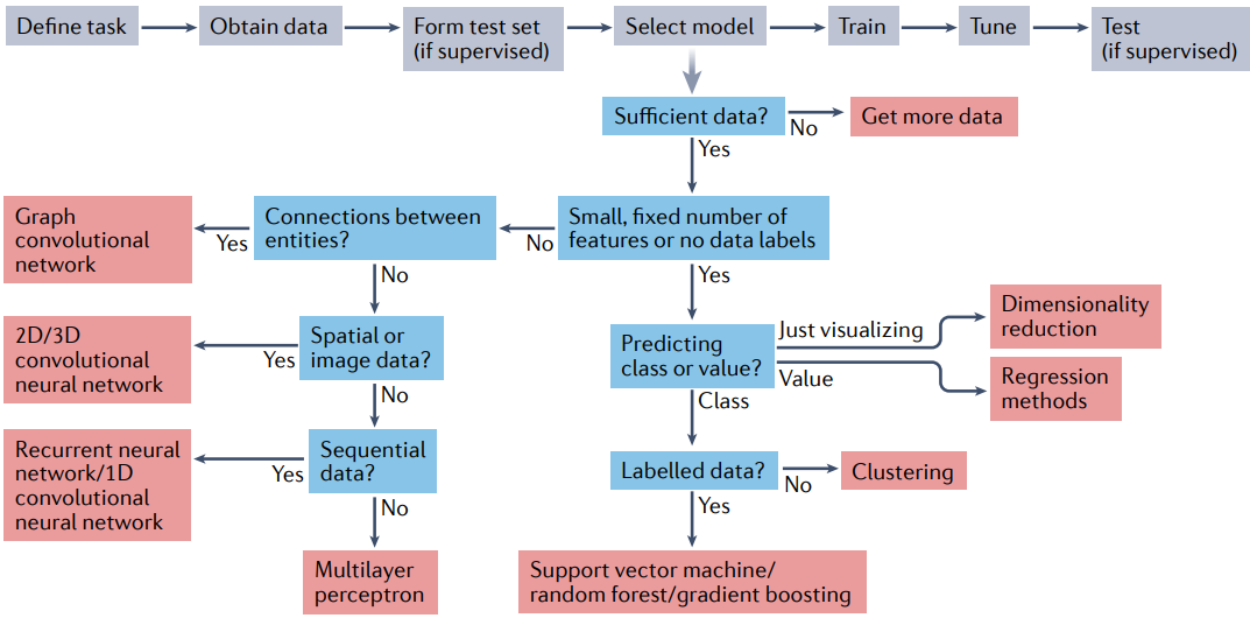


Figure 2.24 Decision tree to assist researchers in selecting the appropriate machine learning method depending on the task to complete [54].

It is often extremely useful to use existing models, especially when they apply well to collected data, since building a model from scratch is a daunting task. There are unlimited possibilities, and multiple parameters to tune to optimize model performance. Often the best network architecture is found through trial-and-error; even applying best practices may require modifications following training and testing. Using preconstructed model architectures like U-Net [55], ResNet [56] or VGG [57] for segmentation, GPT [58] for language generation or Noise2Void [59] for denoising often saves a considerable amount of time.

When nontypical tasks need to be addressed with a different combination of inputs, building a new model may be necessary. Models can be customized to any need with open-source packages, most commonly in Python, such as PyTorch [60], Keras Tensor Flow [61], scikit-learn [62] and more. These packages act as frameworks and offer tools to build a model, greatly simplifying implementation and optimize computation for users. Still, all model parameters remain to be specified. Model parameters to identify include the depth, width, optimizer, loss function, learning rate, number of epochs, batch size. For each layer the activation function, connectivity, regularization should also be defined. If the layer is a convolutional layer in a CNN, the filter sizes, number of filters, pooling, padding and more are also to be considered [63]. The perfect combination of model parameters is therefore difficult to determine, and there are certain guidelines to follow to help optimize the model. Nonetheless, building a model requires sufficient research and knowledge on the workings of deep learning algorithms.

Model performance can only truly be evaluated once training data is available. There are also certain outcomes to monitor, such as overfitting and underfitting. During training, the data is

often separated into three sets: train, validation, and test. Much of the data is kept for training, for the model to have the most examples from which to learn. A small portion of the data is set aside and used for validation during training, to diagnose model tendencies [64]. Finally, the remainder of the data, also a small portion, is kept for testing, after the training is completed, to evaluate the model's prediction accuracy. Data used to validate, or test, is never used for training.

The validation set will help determine if underfitting or overfitting is occurring. Underfitting is exemplified in Figure 2.25 (D=5) where the model fit in green is not expressive enough to represent the training set adequately. The model will not predict accurately, as previously seen with the linear fit in Figure 2.19 (left). Increasing the expressive power to remedy underfitting is possible by adding activation functions in model layers, often the ReLU activation function is implemented [65]. Conversely, when the model is too expressive, as seen in Figure 2.25 (D=200), every single data point is accounted for with the model fit in green. The model will fit the training data too well. Overfitting will give great results with the training set [66], but terrible results with the test set or with other data. The model loses its ability to generalize to unseen data, or new data.

Techniques such as regularization, pooling, dropout exist to reduce overfitting. Regularization is typically used to penalize parameters with large coefficients when the model suffers from high dimensionality [67]. Pooling reduces the size of the output from the previous layer in a CNN and dropout will randomly exclude weights in layers [68].

Also, halting training with built-in callbacks can help avoid overfitting [69]. Callbacks are mostly triggered by learning curves, plotted dynamically during training. After every epoch, a loss value is computed using the loss function with the training set and the validation set, generating a training loss and validation loss. Since the training set is used during training to tune

the model parameters (w), the loss value computed with the training set is biased. However, the validation loss computed with the validation set, provides an unbiased evaluation of the model's performance during training because validation data is unseen data. As training progresses, loss values for every epoch are plotted, ideally the loss value decreases since the objective is to minimize the loss during learning. Therefore, learning curves include the validation loss curves combined with the training loss curves, and they are extremely useful to diagnose model performance. In fact, they should always be plotted dynamically for every epoch to efficiently establish how well model architecture is built for the training set, during training.

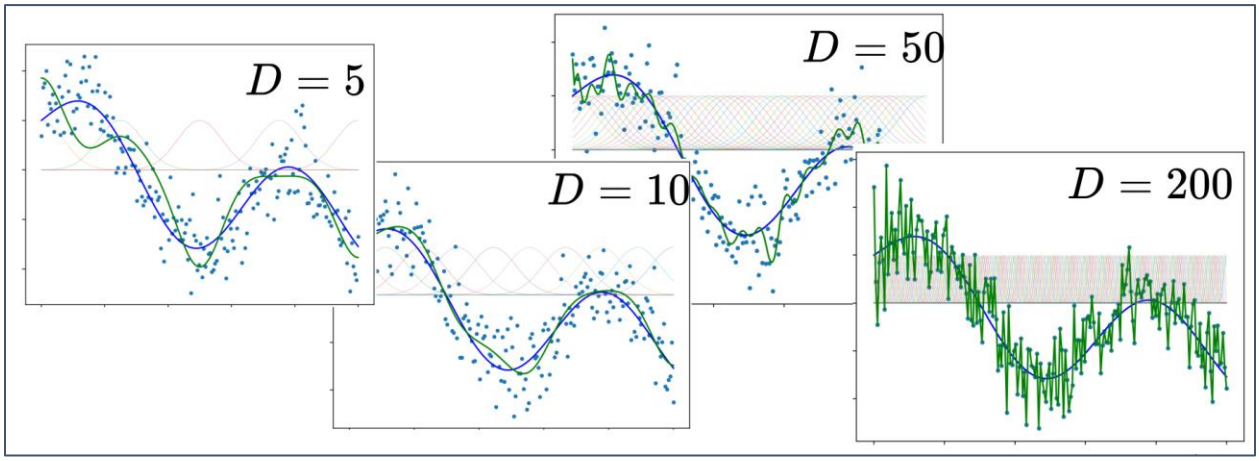


Figure 2.25 Increasing model expressive power with increasing input dimension (D) to demonstrate underfitting and overfitting [46]. Blue curves are the simulated function representing the true fit, blue dots are data points in the simulated function with added or substracted noise and green curves are model fits. At D=5, the model fit is not ideal, the data is slightly underfit, at D=10, the model fit in green is ideal, at D=50, overfitting is beginning to be observed and at D=200 there is an obvious overfitting of the model to the training data.

The collected training data is extremely important to consider for optimal results. For training data, the size of the training set, or the quantity, is crucial, as well as the training example diversity. An optimal training set will have a sufficient amount of data, that is also diversified and unbiased. The main goal of training is to create a model that is efficient at generalizing.

Obviously, the more data provided to the model, the better it will learn, with more examples from which to learn. However, the diversity of the data will impact its ability to predict to unseen data as well. The model learns from the distribution of the input data, it needs an adequate sample to represent values across all the response categories. If data is not diversified, the algorithm will work very well but only in specific cases.

3 Workflow Automation of SEM Acquisitions for Nanoparticle Analysis

Sabrina Clusiau, Nicolas Piché, Nicolas Brodusch, Mike Strauss, Raynald Gauvin

Manuscript currently under review

3.1 Preface

This chapter presents the initial work performed on the SEM, establishing a connection between the microscope hardware and the image processing software. Controlling the microscope properly requires a thorough investigation of the available commands, functionalities, and limitations of the instrument. Once settings are well understood, software algorithms can perform appropriate corrections and compensate for instrument instabilities such as drifting and backlash. Different strategies are explored throughout this chapter to manage microscope outputs and obtain reliable results, building complete automated workflows. Available microscope commands are used seamlessly throughout Python scripts in the Dragonfly developer environment to get and set parameters and capture images for quantitative analysis. A selected sample was used for experiments, to automate grid imaging, without selective positioning and to automate smart beam positioning with feature tracking on the SEM. Multiple images are captured for analysis by including software to microscopy workflows. Gathering a maximum of information on a sample, with quantitative measurements, is beneficial for fully understanding a material. Automation of full acquisition workflows is possible with methods presented in this chapter, the main objective remaining to assist microscopists, relieving them of repetitive tasks.

3.2 Abstract

Acquiring multiple high magnification, high resolution images with scanning electron microscopes (SEMs) for quantitative analysis is a time consuming and repetitive task for microscopists. We propose a workflow to automate SEM image acquisition and demonstrate its use in the context of nanoparticle (NP) analysis. Acquiring multiple images of this type of specimen is necessary to obtain a complete and proper characterization of the NP population and obtain statistically representative results. Indeed, a single high magnification image only scans a small area of sample, containing only few NPs. The proposed workflow is successfully applied to obtain size distributions from image montages at three different magnifications (20,000x, 60,000x and 200,000x) on the same area of the sample using a Python based script. The automated workflow consists of sequential repositioning of the electron beam, stitching of adjacent images, feature segmentation, and NP size computation. Results show that NPs are best characterized at higher magnifications, since lower magnifications are limited by their pixel size. Increased accuracy of feature characterization at high magnification highlights the importance of automation: many high-magnification acquisitions are required to cover a similar area of the sample at low magnification. Therefore, we also present feature tracking with smart beam positioning as an alternative to blind acquisition of very large image arrays. Feature tracking is achieved by integrating microscope tasks with image processing tasks, and only areas of interest will be imaged at high resolution, reducing total acquisition duration.

3.3 Introduction

A proper characterization of features at the nanometer scale by scanning electron microscopes (SEMs) requires high magnification, high-resolution images. Indeed, at lower magnifications the smallest features of the images are pixelated and consequently analyzed with less precision. This is demonstrated in Figure 3.1, where platinum nanoparticles (NPs) are visibly more well-defined as the magnification increases from 20,000x to 60,000x, and ultimately to 200,000x, due to the increasing number of pixels used to describe a given area. However, increasing the magnification alone is not sufficient to obtain realistic measurements from quantitative analysis. For this, high resolution is also necessary, with sufficient resolving power to distinguish individual NPs. The required high resolution at high magnification can be achieved with a suitable electron gun, well-tuned optics, and adequate beam current [1].

Yet, in the context of quantitative analysis of nanometric features, the biggest inconvenience with high magnification images is that only a small area of the sample will be scanned per image. To obtain a reliable statistical representation of quantitative results, a significant area of the sample should be considered. Therefore, a series of high magnification images must be acquired to measure an acceptable number of features to effectively describe their distributions.

Current workflows for acquiring multiple high magnification, high-resolution images with the SEM involve continuous microscopist intervention. The beam must be manually repositioned repeatedly to capture an image, or image array, in the newly selected area. These tasks are tedious, time consuming and biased to the areas that the microscopist chooses to image. In addition, image processing steps following microscope acquisitions are labor intensive, including data transfer, image stitching, feature segmentation and distribution calculations.

Existing add-on software such as SerialEM[2] and SBEMimage[3] have successfully assisted microscopists with specific workflows on the transmission electron microscope (TEM) and the serial block-face electron microscope (SBEM), respectively. SerialEM enables automated acquisitions through communication with the microscope's control software. Functionalities include tilt series for electron tomography, imaging for 3-D reconstruction from serial sections, and acquisition of datasets of macromolecules for analysis by single-particle methods. Furthermore, SBEMimage, a software compatible with SBEMs, integrates data acquisition with image post-processing. Stitching and alignment during acquisitions ensures high reliability and prevents data loss since sections are destroyed during acquisitions.

SEMs can also benefit from add on software to automate routine acquisitions, first to collect data with minimal microscopist intervention, and second, to improve workflows with integrated image analysis. This paper presents an automated, optimized workflow for acquiring high-quality data with image processing that is appropriate for quantitative analysis. The proposed workflow can run overnight and unattended when the microscope would normally be idle. The objective is not only to increase the amount of data output by the microscope but also improve the quality of the data acquired. The main workflow steps include grid imaging, stitching, segmentation, and quantitative analysis. Images of the same area on the sample are taken at three different magnifications 20 000x, 60 000x and 200 000x to compare final distributions and demonstrate the improved accuracy of quantitative analysis as magnification increases. Feature tracking is also explored to minimize duration of acquisitions by imaging only areas of interest.

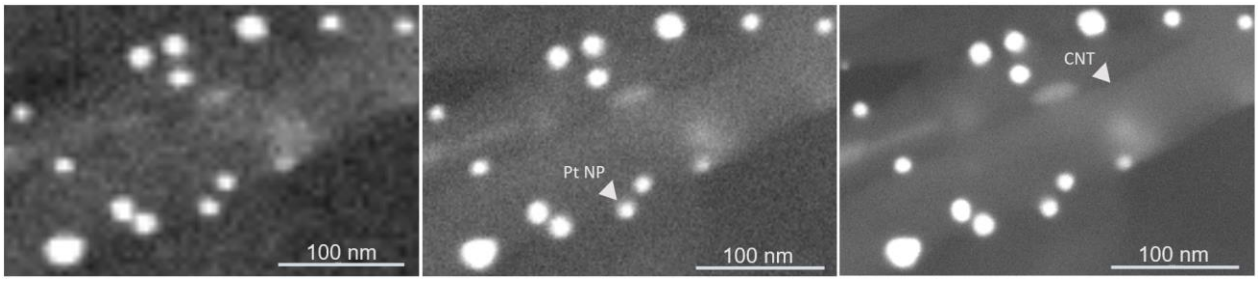


Figure 3.1 Backscattered electron (BSE) images of platinum nanoparticles on carbon nanotubes at beam energy 5keV and beam current 20 µA on the SU8230 cold field emission gun (FEG) from Hitachi. Left: 20,000x, Middle: 60,000x, Right: 200,000x.

3.4 Implementation

The proposed workflow is open-source, composed of Python-based scripts in the Dragonfly developer environment [4], soon to be a plugin. The code repository is obtained when installing the developer version of Dragonfly, a software that specializes in image processing [5]. Building the microscope control scripts in the Dragonfly environment simplifies postprocessing and quantitative analysis, since many image processing tools and packages are already implemented and available in the software to recycle. Implemented in object-oriented programming, the scripts have abstract classes, enabling Python programmers to integrate any microscope with external communication capacities. The software does not currently include standard microscope alignment routines and optimal microscope parameter selection. The correct focus, stigmatism, beam energy, probe current, etc. should be set prior to launching acquisitions.

The scripts for microscope control send commands that execute multiple tasks, limitations depend on the SEM manufacturer and external communication capabilities. With the provided

commands, the script can acquire an image externally, set the active detectors, set and get the capture settings, set and get the stage position in x, y, z, r and t, shift the beam in x and y, set and get the focus, stigmatism, magnification, current, voltage and more. This metadata can be saved and used for later analysis.

The steps in the automated workflow, illustrated in Figure 3.4, include (1) acquisition of arrays of images, *x* images wide and *y* images high, at a selected magnification on the microscope, images are then transferred to image processing software for (2) stitching adjacent images in grid layout and (3) segmentation of features of interest for quantitative analysis.

3.4.1 Grid Acquisitions

Automation of acquisitions at high magnification and high resolution is achieved through external communication with the microscope's control software. Specifically, images were collected on the SU8230 cold-field-emission SEM from Hitachi [6]. Commands are sent from an external computer (EXT PC) using socket communication by TCP/IP protocol, to the SEM computer (PC SEM) equipped with the proprietary software. The complete list of implemented commands used to acquire a grid layout is shown in Table 3.1. Commands are first converted to a manufacturer defined string format before sending to the microscope computer. An EXT PC is necessary for SEM workflow automation because PC SEMs function in a very restrictive environment to manage corruption, with no internet access, minimal GPU power and memory and the computer often runs on old operating systems (Windows 7). The EXT PC is purchased by the microscope users and will have all the requirements to run computationally expensive image processing tasks and can be upgraded as needed with changing demands. With external control of the microscope, through communication illustrated in Figure 3.2, both simple and

complex tasks can be automated, and image processing is incorporated during workflows to optimize and guide acquisitions. This is exemplified in Section 3.4.4 when feature tracking is presented.

Python command	Description
set_capture_settings	Sets the following capture settings: - scan mode: Rapid, Fast, Slow, CSS, Slow1Integration - resolution (pixels): 640x480, 1280x960, 2560x1920, 5120x3840 - scan time (seconds): 10, 20, 40, 80, 160, 320 - integration number (frames): 8, 16, 32, 64, 128, 256, 512, 1024 Values are specific to the SU8230 and included in a Python dictionary.
set_magnification *get_magnification*	Sets the magnification to specified value in range for High Mag mode or Low Mag mode. Gets the current magnification set on the microscope, from magnification, pixel size is computed.
set_capture_and_save	Runs image capturing and saves the captured image(s), one or all screens are captured and saved depending on how many detectors are set up. The image(s) is saved with fixed file names in fixed folder on the microscope computer. If command is repeated, files are overwritten to avoid exhausting memory resources on the microscope computer. Files are therefore directly transferred to the external computer after being captured to avoid data loss.
set_image_shift_X *set_image_shift_Y*	Sets the image shift in X and Y to move the image horizontally or vertically.
get_stage_position	Gets the current stage position for the 5 axes: x, y, z, r, t
set_stage_XY	Sets the x and y coordinates only, keeps all other axes identical

Table 3.1 Python commands used to perform microscope task in the proposed automated workflow including grid layout acquisitions and smart beam positioning for tracking.

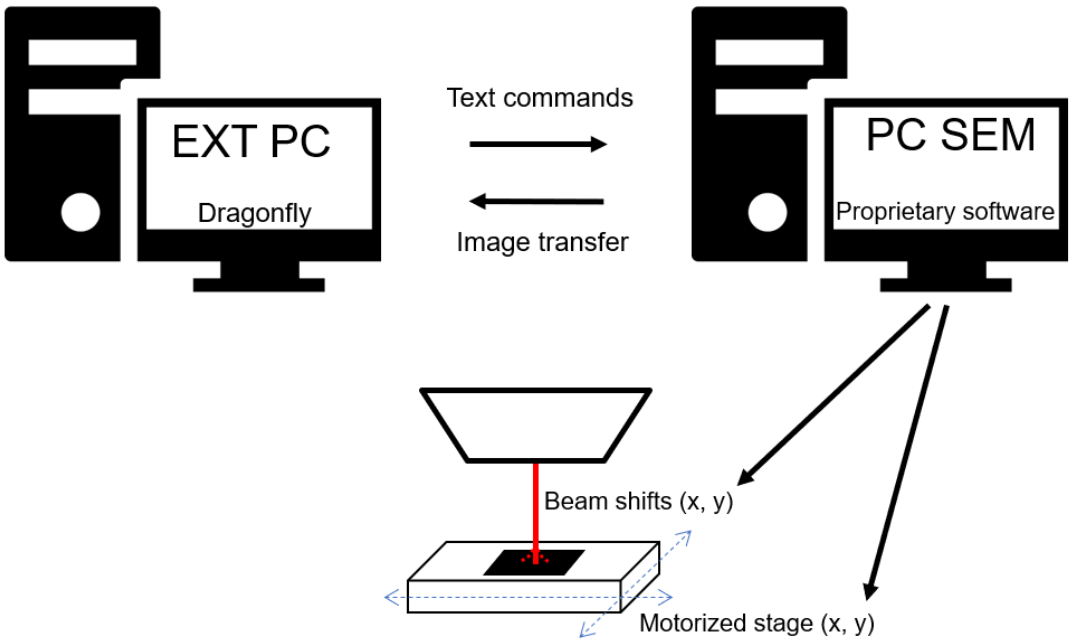


Figure 3.2 Microscope control is achieved through external communication between the PC SEM and EXT PC. EXT PC will send text commands to the PC SEM, which will execute the commands with the microscope's proprietary software. Commands include stage movements and beam shifts. Once images are captured, they are transferred from the PC SEM to the EXT PC for image analysis with image processing software (Dragonfly).

Grid imaging is a relatively straightforward operation, consisting of sequential repositioning of the beam or the sample stage with 10% overlap between images. Stage shift or image shift commands are sent to the microscope to change the image view, followed by a capture command to acquire the image. Stage shifts are achieved by specifying stage coordinates, while beam shifts require a specific value within a range, included in the command. The given value is equivalent to a certain distance depending on the magnification. Generally, for higher magnifications (90,000x or more) the beam is repositioned using beam shifts for more precise movements. Whereas at lower magnification (90,000x or less) the stage is shifted since stage movements are suited to longer distances and tend to have courser steps driven by motors, while beam shift range is limited. Through testing, it was found that stage movements are not precise below

approximately 1 micron on the SU8230, which is larger than the image size at high magnification. For instance, at 200,000x, with the microscope used for experiments, the image size is approximately 650x480 nm. Therefore, 1 micron stage shifts are not appropriate and smaller shifts are required at higher magnifications for 10% overlap between images. Capturing sufficient overlap between neighboring images in the grid layout improves the success of stitching algorithms used to correct imprecise beam or stage movements, covered in Section 3.4.2.

There are multiple possible image acquisition patterns, a few are shown in Figure 3.3: row-by-row, column-by-column, snake-by-row or snake-by-column. Imaging pattern has an important impact on the assembly of the grid layout, since larger stage movements are motorized and imprecise, leading to stage drift and backlash. Snake patterns require fewer corrections, as movements are smaller with less mechanical instabilities, the motor stops and repositions the stage with more precision. Drifting occurs when there is torque remaining at the transmission between the motor and the stage mechanism. Drifting is more prominent when the stage is moved at high magnification. The image will drift slowly in the direction of the movement of the stage after the stage stops moving. The distance traveled is not the only issue, the direction of the movement will contribute to imprecisions. Backlash is the result of moving the stage in one direction, followed by movement in the reverse direction. With backlash, the stage does not move the same degree in each direction, and the actual stop position varies depending on the direction of stage movement, even if the same coordinates are specified. For row-by-row and column-by-column, large distances are traveled in opposite directions, when returning to the initial row or column, and drifting and backlash are heightened. Images will not be aligned perfectly when assembled side by side in the grid layout, even if the correct coordinates are

requested during acquisitions. These undesired effects are minimized by acquiring images in a snake pattern, where one forward direction is used throughout acquisitions.

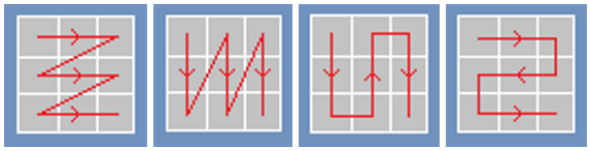


Figure 3.3 Grid imaging patterns: row-by-row, column-by-column, snake-by-column or snake-by-row.

In our imaging scheme, an overview image is first captured at low magnification (20,000x), then, grid layouts of the same area on the sample are acquired at higher magnifications (60,000x and 200,000x). These magnifications can be tailored to the type of sample being acquired and the size scale of the information desired. The purple, red and green rectangles in Figure 3.4.1 (left) represent images taken at 20,000x, 60,000x and 200,000x respectively. Although the rectangles are not to scale, the figure is meant to illustrate that images at different magnifications are not equivalent in size. Higher magnification acquisitions require more images to cover a comparable area of the sample at lower magnification.

Additionally, the microscope is equipped with both secondary electron (SE) detectors and backscattered electron (BSE) detectors, enabling a multi-signal approach. Acquiring multiple signals at every position shown in Figure 3.4.1 (right) does not cost extra time, will simplify image processing tasks that follow, and is encouraged in all contexts. Indeed, the different contrasts obtained from signals can serve different purposes during analysis. This is in fact the case for the sample used throughout this paper, platinum nanoparticles (Pt NPs) on carbon nanotubes (CNTs) identified in Figure 3.4.1 (right). The presented automation workflow, however, is not limited to this material nor to spherical NPs.

In this context, qualitative analysis of images in Figure 3.4.1 (right) demonstrates that secondary electron (SE) images are feature-rich, making them ideally suited for stitching algorithms, however they lack sufficient contrast between the NPs and other components for feasible segmentation. Backscattered electron (BSE) images present the opposite challenge, contrast between platinum and carbon is perfect for segmentation, while images do not contain enough features for proper stitching by algorithms.

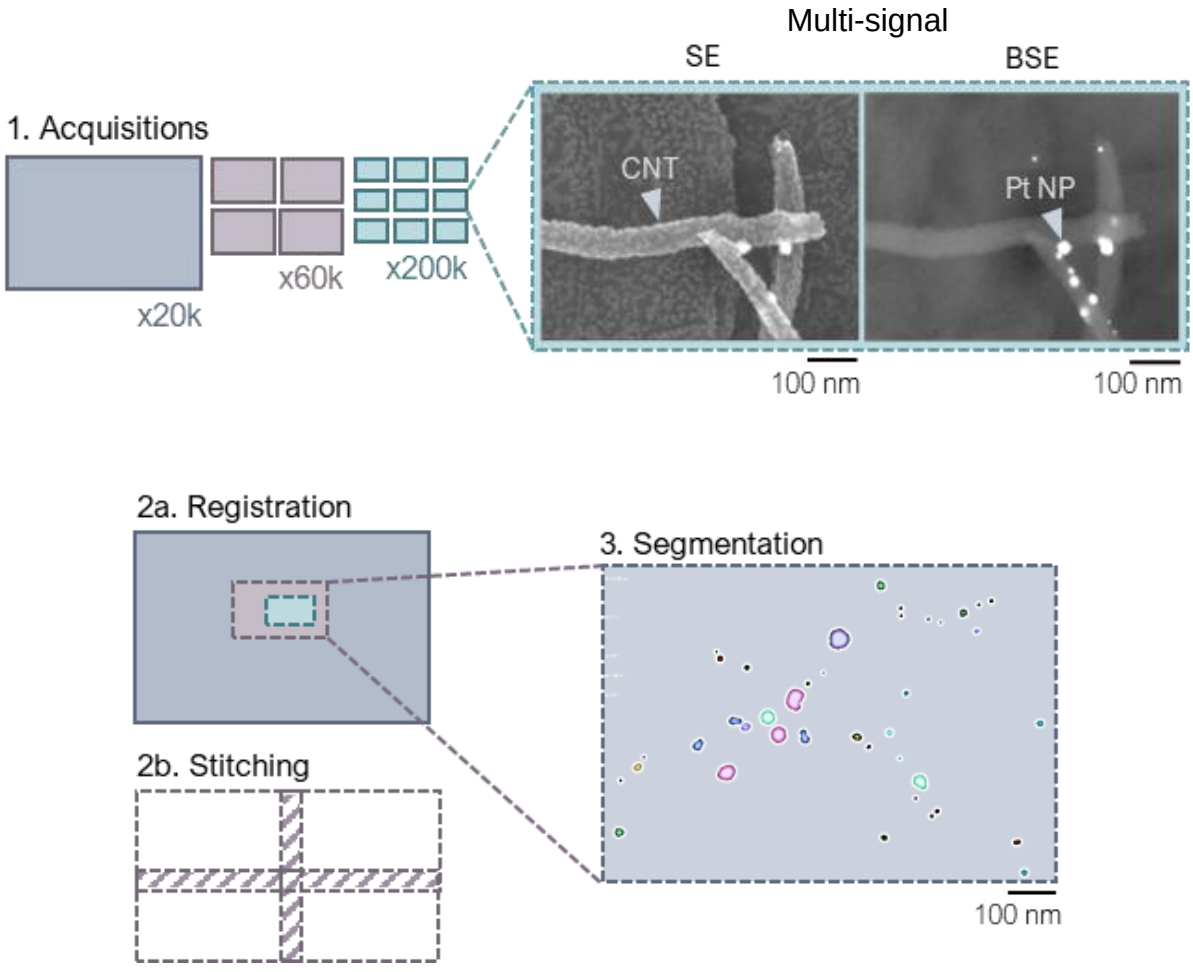


Figure 3.4 Proposed automated workflow for automated acquisitions and quantitative analysis. 1. Grid acquisitions at selected magnification, (left) three magnifications (20,000x, 60,000x and 200,000x) are used for comparing size distributions and (right) both SE and BSE images are acquired at every position to benefit from a multi-signal approach. 2. Grid SE images are either

a. registered with mutual information algorithm or b. stitched with SIFT algorithm. 3. Stitched BSE image is segmented with Otsu thresholding for quantitative analysis of NPs.

3.4.2 Stitching

Grid imaging, especially at higher magnification, requires postprocessing corrections such as stitching, due to imprecise beam or stage movements, explained in the previous section. Added intelligence in scripts will guide acquisitions, ensuring high reliability and facilitating the correction achieved by stitching algorithms. For example, the presented acquisition workflow selects either beam shift or stage shift depending on the scale of step movements and ensures that sufficient overlap between neighboring images is captured in the grid layout, preventing gaps in the final image. Overlapping regions are prioritized because of the features they contain. Detected features will determine the performance of stitching algorithms during post processing.

Two different stitching approaches are proposed, demonstrated in Figure 3.4.2a and Figure 3.4.2b, to ensure that if one algorithm fails, a second option is available to compensate. The first option, in Figure 3.4.2a, is to register high magnification SE images to a low magnification SE image with the mutual information algorithm [7][8]. The second option in Figure 3.4.2b is to stitch high magnification SE images with overlapping regions using feature points with the scale-invariant feature transform (SIFT) algorithm [9][10]. The performance of either algorithm depends on the data acquired, and especially the contrast achieved. The translation and rotation transformations found by stitching algorithms with the SE images are then applied to the BSE images since as mentioned previously, they are more easily segmented for analysis of the Pt NPs. But, in principle, the accurate stitching of the signal type giving the best contrast can seamlessly be applied to all other signal types.

3.4.3 Segmentation

Next, NP segmentation is implemented for quantitative analysis. Manual segmentation of one image with multiple NPs can take hours. With grid acquisitions, 50 images or more are to be expected, requiring users to perform precise and accurate manual segmentation for days or months. For this reason, automation of the segmentation process will greatly improve workflow duration.

Again, the appropriate segmentation algorithm depends on the contrast in the acquired images, highlighting the importance of tuning microscope parameters before acquiring data, for successful segmentation. Microscope parameters can be tuned to reduce noise, improve resolution, and emphasize contrast on features of interest, simplifying image processing tasks to avoid manual labelling. In this case, BSE images of Pt NPs present such high contrast on carbon background that Otsu thresholding is easily applied. A portion of the segmented NPs are illustrated in Figure 3.4.3. Features of interest are, however, rarely easily segmented with simple thresholding and more sophisticated algorithms are required. Multiple segmentation algorithms including deep learning are available in Dragonfly [11] and workflows can be flexibly adapted to data requirements. Once the segmentation is complete, NP size distributions are computed on the final segmentation, with the 2D Equivalent Diameter [12], shown in Equation 3.1. With an adequate segmentation of features of interest, several measurements are available to apply in image processing software. For spherical NPs, the diameter, eccentricity, roundness, and surface area are all useful for analysis. For other features, with varying shapes, the orientation, ferret box, perimeter, longest length, anisotropy, roughness and more can all be of interest.

$$d = \frac{4 \times Area}{Perimeter} \quad 3.1$$

3.4.4 Tracking

Feature tracking was explored as an alternative to grid layout imaging, to propose a solution for imaging only areas surrounding features of interest. The main objective is to reduce total acquisition time, which is accomplished by avoiding regions devoid of features. The workflow for feature tracking also integrates image processing tasks, including segmentation, smart beam positioning and stitching, in a feedback loop with microscope tasks, specifically, data acquisition. Combining data acquisition with image processing is beneficial for saving time and resources, optimizing imaging to acquire more and better data, faster.

The feature tracking workflow consists of the following steps summarized in Figure 3.5:

A. Acquire an overview, low magnification image (20,000x) with the SEM.
B. Transfer data to an image processing software (Dragonfly) for CNT segmentation (in red).
C. Convert the segmentation to a graph of vertices (spheres) and edges (segments) with boost library [13] implemented in Dragonfly.
D. Use the graph vertex positions to establish imaging sites (outlined in dashed lines).
E. Position the beam at each imaging site through established referential with Dragonfly coordinate system, acquiring high magnification (100,000x) images.

The resulting 100,000x images acquired are shown in Figure 3.5E, after stitching, on top of the initial 20,000x image.

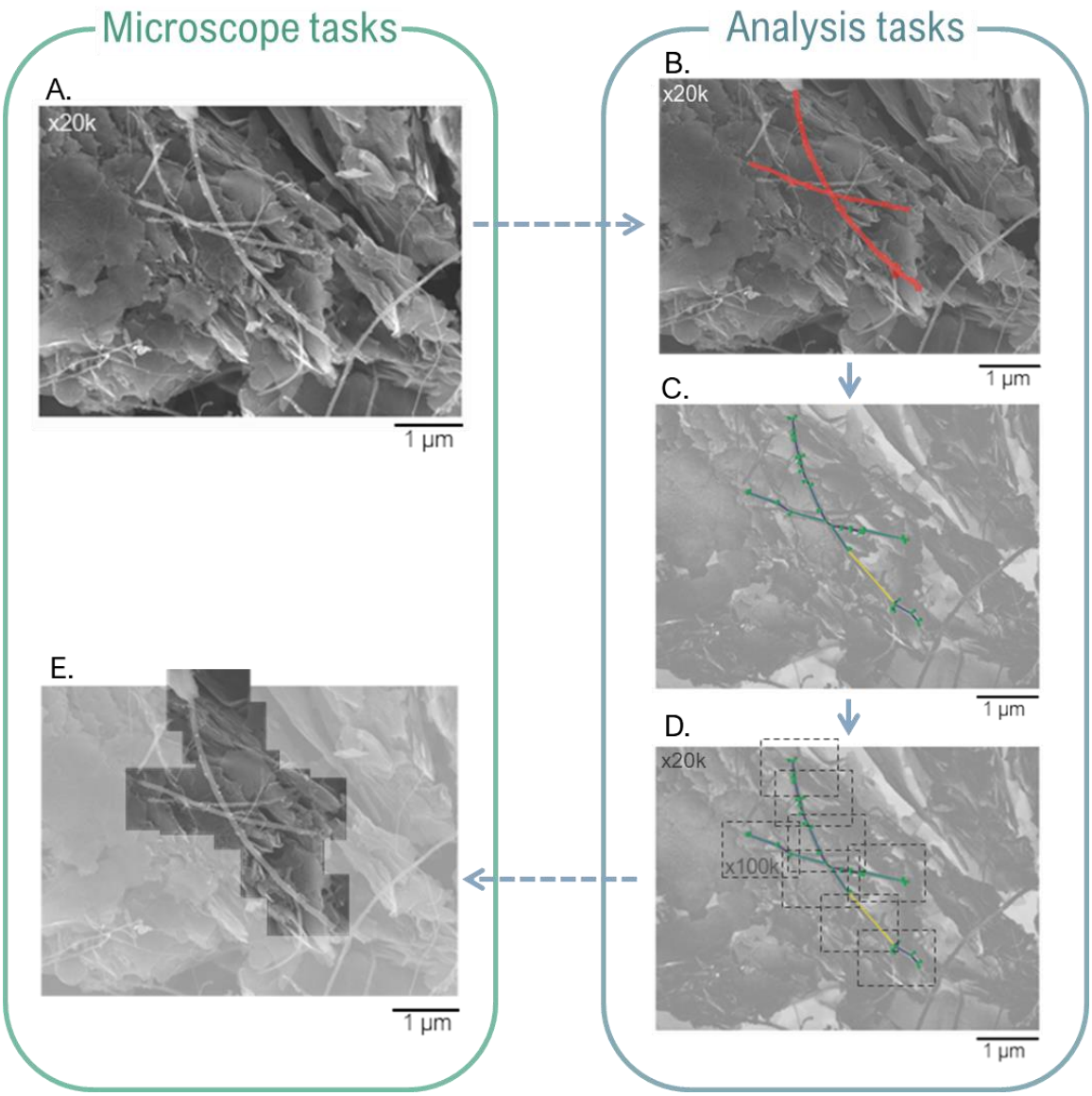


Figure 3.5 Proposed automated workflow for feature tracking. A. Acquire low magnification (20,000x) image containing features of interest. B. Import image to image processing software and segment feature to track. C. Convert segmented pixels to a graph of vertices and edges. D. Establish imaging sites from the vertex positions. E. Position beam at imaging sites for high magnification (100,000x) acquisitions.

The tracking workflow is also open-source in the Dragonfly code repository, composed of Python-based scripts in the Dragonfly developer environment. The complete list of commands used for acquiring images is identical to the one for grid imaging in Table 3.1. However, an extra

command, to change the scan status, was found useful for minimizing image drifting during image analysis tasks. The scan status can be set either to RUN, where the beam will scan the sample surface continuously, or to FREEZE, where the scanning is paused, and the beam is diverted elsewhere in the chamber, away from the sample surface. Maintaining continuous scanning of the sample at a specific location for a certain period, leads to charge accumulation which then leads to image drifting. The tracking workflow transitions from the microscope to the software and back to the microscope. The scan status is therefore set to FREEZE while image processing tasks are carried out, while the microscope is waiting for repositioning, to minimize negative effects of charge accumulation. Pausing the sample scanning is especially important for identifying the high magnification imaging sites correctly. The imaging site coordinates are identified by the software on the low magnification image used as a reference, and they will be inaccurate if image pixel positions vary in time. If image drift does occur inadvertently, it should be compensated for with image registration using correlation techniques or key point detection algorithms [14]. The low magnification image used as reference is automatically imported from the microscope computer to Dragonfly software, with real image size found in the image's associated metadata file, and a specified position in the coordinate system. Once in Dragonfly, all required image processing algorithms are available to include in the script.

The segmentation algorithm used in Figure 3.5B depends on the feature that is being tracked. For this specific CNT segmentation, simple thresholding was used, the CNT to be tracked was then isolated by separating connected components. Thresholding is often appropriate for tracking workflows since the segmentation does not need to be perfectly defined, it will be converted to a graph, where segmentation details are lost. The objective of feature segmentation is simply to highlight the features of interest, to then identify high magnification imaging sites.

Also, only a portion of vertices in the graph are selected to center the imaging sites, as seen in Figure 3.5D, where there are more vertices than imaging sites. The schematic in Figure 3.6 demonstrates that for a graph in blue, vertices contained within a certain range of the vertex (circled in green) used to position the image area sites (orange rectangle) are excluded, to avoid imaging the same area multiple times. Excluded vertices are marked with a yellow circled X. Overlapping regions are considered for stitching (dashed line in imaging site). Smart beam positioning is therefore accomplished by first obtaining a rough segmentation of the features of interest, converting the segmentation to a graph, and then selecting the vertices to position imaging sites. Selected vertices will be far enough apart to prevent multiple images of the same region, but close enough for sufficient overlapping regions for stitching without gaps.

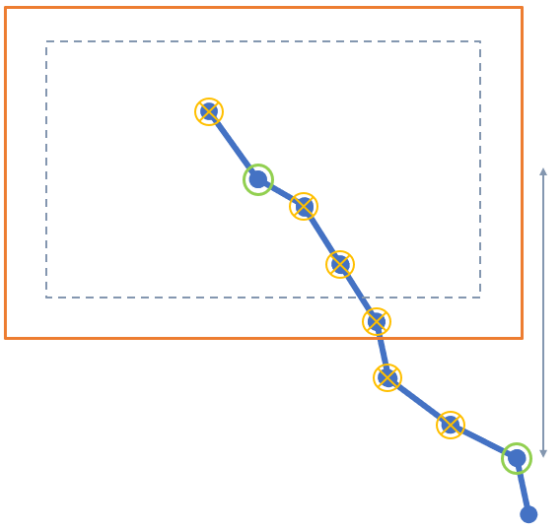


Figure 3.6 Schematic of selected vertices to position imaging sites. Graph of vertices and edges is in blue, vertices circled in green will determine the center position of imaging sites, vertices within a certain range are excluded and circled in yellow with an X and imaging site is delimited by an orange rectangle with the dashed rectangle smaller than the imaging site to accommodate for any imprecise stage or beam movements.

Converting the segmentation to a graph is a useful approach when features are elongated and connected. If features are small, disconnected and scattered, like nanoparticles, each individual feature centroid should be used for smart beam positioning. Nanoparticles would be segmented and separated by connected components, for which the centroid is computed. The centroid is a position vector (x, y, z) in the coordinate system, and therefore determines where high magnification imaging sites should be located, analogous to graph vertices. The appropriate technique, conversion to graph or centroid computation, also depends on the magnification required to adequately analyze features of interest. Generally, low magnifications need only one image to include the entire feature and the centroid can be used to position the beam. Higher magnifications require multiple images to cover the entire features, and the graph vertices will more adequately determine the multiple imaging sites to fully image the feature of interest.

3.5 Results

3.5.1 Nanoparticle size distributions

The proposed automation workflow was tested on three separate magnifications (20,000x, 60,000x and 200,000x) and the results are presented in Figure 3.7 and Table 3.2. NP size distributions are computed using 2D equivalent diameter, from final segmentations with Otsu thresholding.

Results are presented side-by-side in Figure 3.7 for quick visual comparison. Distributions follow similar trends with highest frequency at approximately 10 nm diameter. The pixel size for each magnification, statistics describing the distribution including the minimum, maximum, and mean values, as well as the total number of NPs are shown in Table 3.2. The results demonstrate

a few key points: minimum NP size at lower magnification is restricted by the associated pixel size, resulting in gaps in the 20,000x size distribution, and the total number of detected NPs decreases with magnification.

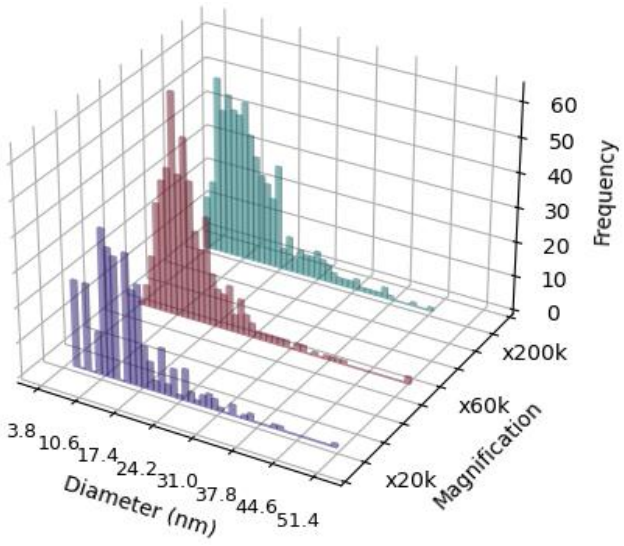


Figure 3.7 NP size distribution based on segmentation at three different magnifications (20,000x, 60,000x, 200,000x) for visual comparison.

Magnification	Pixel size (nm)	Min size (nm)	Max size (nm)	Mean (nm)	Total detected NPs
20,000x	4.961	5.60	51.3	15.26	350
60,000x	1.654	4.17	52.4	13.88	505
200,000x	0.496	3.80	45.4	12.75	554

Table 3.2 NP statistics at three different magnifications (20,000x, 60,000x, 200,000x) with associated pixel size.

3.5.2 Total Acquisition time

For NP analysis, only high-resolution images of areas surrounding CNTs were considered because most Pt NPs are latched onto or located near CNTs in the sample. Tracking CNTs will

considerably reduce the number of acquired images needed, therefore abridging a significant amount of time, and avoiding empty images i.e., areas without NPs.

Table 3.3 compares the amount of time required to acquire a grid layout and to track CNTs at 100,000x based on the number of images taken for the same area on the sample, shown in Figure 3.5. The grid layout consists of 25 images, a 5x5 grid containing the entire area on the sample as seen in the 20,000x image in Figure 3.5A, whereas the CNT tracking considers the 10 images required to cover the segmented CNT in Figure 3.5E. The total time is considerably reduced as the number of images acquired decreases by more than half, where 65 minutes are required for a grid layout and only 20 minutes for CNT tracking.

Acquisition type (100,000x)	Number of images	Total acquisition time (minutes)
Grid	25	65
CNT tracking	10	20

Table 3.3 Comparison of total acquisition time based on number of images taken for grid acquisitions and CNT tracking.

3.6 Discussion

Firstly, the study carried out for NP quantitative analysis evaluated size distributions at low (20,000x) and high (60,000x and 200,000x) magnification from an automated workflow consisting of grid acquisitions, stitching and segmentation. The presented results show that (1) the proposed automated workflow achieves reliable NP quantitative analysis for low to high magnification acquisitions, (2) multi-signal approaches are beneficial, providing different image contrasts useful for image processing tasks, and (3) acquiring multiple high magnification

images of a sample area, comparable to the area covered by a low magnification image, results in a more accurate representation of NP distributions.

The appropriate magnification depends on the size of the features to be analyzed and should be established prior to launching acquisition. The highest magnification may be unnecessary if features are larger, since similar results will be accomplished at lower magnification. Consistently selecting the highest magnification for workflows, without accommodating to feature size, will waste time and resources, acquiring an excessive number of images to cover the same sized region as a lower magnification grid layout. Additionally, the number of total images required to adequately describe distributions should be determined prior to acquisitions. A grid of 5x5 for a total of 25 images is four times quicker to acquire than a 10x10 grid for a total of 100 images. However, increasing the number of images leads to increased data for quantitative analysis. The appropriate grid size should generate enough data for a statistically representative analysis without exhausting an unnecessary number of resources. For NPs, with a mean size of 10 nm, a grid size of 5x5 was acquired and higher magnifications (60,000x and 200,000x) resulted in an improved size distribution and more detected NPs, emphasizing the need for automation. However, perhaps a lower magnification would have been sufficient to adequately represent the sample. With half the magnification, half the images need to be acquired, considerably reducing acquisition time.

Acquiring multiple high magnification images, side by side, for representative statistics, is repetitive and time consuming. By combining grid acquisitions on the SEM and image processing tasks into a complete automated workflow, automation enables scientists to spend time on more productive, complex tasks. Acquisitions can now run unattended and overnight

when microscopes would otherwise be idle. Further analysis by examining results for multiple grid sizes would help determine the best grid size for accurate description of NP distributions.

Secondly, our feature tracking results demonstrate that (1) the proposed automated workflow successfully finds imaging sites for high resolution imaging of features of interest, (2) integrating microscope tasks with analysis tasks leads to more intelligent and efficient acquisitions, and (3) tracking features considerably reduces the total acquisition time when compared to grid acquisitions.

Feature tracking can be especially beneficial when a feature spans a large, nonlinear area of the sample, for instance a crack, where grid imaging would not be appropriate and quite time consuming. Imaging large grid arrays, without selecting favored areas manually or without smart beam positioning will almost certainly result in the acquisition of images without features of interest. Acquiring these empty regions in grid layouts impairs total acquisition speed, since many images will be discarded, as they are not useful for analysis. By tracking features of interest with smart beam positioning, time and resources are better spent by imaging only relevant areas that contain the features of interest.

The proposed algorithms for stitching and segmentation in both workflows may not apply to all contexts, and we have designed these procedures so they can be adapted to the imaged sample, where multiple options are available and implemented in Dragonfly. The workflows are implemented in the Dragonfly environment specifically for this reason, where other algorithms can be easily swapped in the Python script, to accommodate any image processing need. As a future direction, we intend to include a graphical user interface (GUI) as a plugin in Dragonfly. This would greatly simplify the user experience, improve ease of use, and eliminate the need for coding skills to apply the workflow. Users could more easily tailor the workflow to their sample,

select the appropriate algorithms from a list, set up acquisitions and visualize acquisition progress.

3.7 References

[1] J. I. Goldstein, D. E. Newbury, J. R. Michael, N. W. M. Ritchie, J. H. J. Scott, and D. C. Joy, “Scanning Electron Microscope (SEM) Instrumentation,” in Scanning Electron Microscopy and X-Ray Microanalysis, J. I. Goldstein, D. E. Newbury, J. R. Michael, N. W. M. Ritchie, J. H. J. Scott, and D. C. Joy, Eds., New York, NY: Springer, 2018, pp. 65–91. doi: 10.1007/978-1-4939-6676-9_5.

[2] “The SerialEM Home Page.” Accessed: Jan. 02, 2024. [Online]. Available: https://bio3d.colorado.edu/SerialEM/

[3] B. Titze, C. Genoud, and R. W. Friedrich, “SBEMimage: Versatile Acquisition Control Software for Serial Block-Face Electron Microscopy,” Front. Neural Circuits, vol. 12, 2018, Accessed: Jan. 02, 2024. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fncir.2018.00054

[4] “Setup for development with PyCharm — Dragonfly 4.1 documentation.” Accessed: Jan. 10, 2023. [Online]. Available: http://dev.theobjects.com/dragonfly_4_1_release/Documentation/SetupForDevelopmentWithPyCharm/setupfordevelopmentwithpycharm.html

[5] “Dragonfly | 3D Visualization and Analysis Solutions for Scientific and Industrial Data | ORS.” Accessed: Apr. 10, 2022. [Online]. Available: https://www.theobjects.com/dragonfly/index.html

[6] “Equipment | McGill Electron Microscopy Research Group,” memrg. Accessed: Jan. 10, 2023. [Online]. Available: https://www.memrg.com/our-equipment

[7] “Mutual information as an image matching metric — Tutorials on imaging, computing and mathematics.” Accessed: Jan. 10, 2023. [Online]. Available: https://matthew-brett.github.io/teaching/mutual_information.html

[8] G. Egnal and K. Daniilidis, “Image Registration Using Mutual Information,” no. 117, Jan. 2000, Accessed: Mar. 26, 2024. [Online]. Available: https://repository.upenn.edu/handle/20.500.14332/7014

[9] “OpenCV: Introduction to SIFT (Scale-Invariant Feature Transform).” Accessed: Jan. 10, 2023. [Online]. Available: https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

[10] Z. Min, Z. Jiguo, and X. Xusheng, “Panorama Stitching Based on SIFT Algorithm and Levenberg-Marquardt Optimization,” Phys. Procedia, vol. 33, pp. 811–818, Jan. 2012, doi: 10.1016/j.phpro.2012.05.139.

[11] R. Makovetsky, N. Piche, and M. Marsh, “Dragonfly as a Platform for Easy Image-based Deep Learning Applications,” Microsc. Microanal., vol. 24, no. S1, pp. 532–533, Aug. 2018, doi: 10.1017/S143192761800315X.

[12] F. Latief, “Analysis and Visualization of 2D and 3D Grain and Pore Size ofFontainebleau Sandstone Using Digital Rock Physics,” J. Phys. Conf. Ser., vol. 739, p. 012047, Aug. 2016, doi: 10.1088/1742-6596/739/1/012047.

[13] “Boost C++ Libraries.” Accessed: Jan. 10, 2023. [Online]. Available: https://www.boost.org/

[14] N. Marturi, S. Dembélé, and N. Piat, “Fast Image Drift Compensation in Scanning Electron Microscope using Image Registration.,” in IEEE International Conference on Automation Science and Engineering, CASE’13., United States, Jan. 2013, pp. 1–6. Accessed: Feb. 08, 2024. [Online]. Available: https://hal.science/hal-00876194

4 Optimizing SEM Parameters for Segmentation with AI – Part 1: Generating a Training Set

Sabrina Clusiau, Nicolas Piché, Benjamin Provencher, Mike Strauss, Raynald Gauvin

Manuscript currently under review

4.1 Preface

This chapter presents the first of two parts on optimizing microscope parameters for image quality improvement. Acquiring images automatically as presented in the previous chapter is extremely beneficial for quantitative analysis of samples, however collecting multiple images without the appropriate image quality is a waste of time. Establishing the right microscope parameters prior to automating acquisitions will facilitate quantitative analysis that follows acquisitions. Segmentation is the primary concern for quantitative analysis, and parameters such as the beam energy and the probe current should be selected for the purpose of generating image quality where features of interest are easily segmented. Microscope parameter selection, depending on the sample imaged, is accomplished by training deep learning models. Training a model to predict accurately requires appropriate data, namely, a complete training set with sufficient examples from which to learn. As demonstrated in this chapter, obtaining numerous training examples is possible by launching simulations with MC X-ray and including real images acquired on the SEM.

4.2 Abstract

Extracting significant quantitative results from SEM images requires feature segmentation with image processing software. The efficiency of segmentation algorithms depends on the image quality, determined largely by the parameters set on the microscope during acquisitions. By integrating AI within SEM acquisition workflows, it is possible to suggest microscope parameters that will produce images where the features to quantify will be easily segmented. Specifically, a model is trained to automatically suggest the beam energy and probe current to set on the microscope during acquisitions. This paper is the first of two parts, describing workflows for generating a complete training set. The training set is carefully designed, consisting of both simulated data and real data acquired on the SEM by varying the energy and current. Separate workflows are required for generating simulated and acquired training examples. Simulated data generation is accomplished with the MC X-ray simulator in Dragonfly, where multiple virtual samples are created to intentionally diversify the training set. Acquiring data on the SEM for training is a time-consuming process when compared to generating simulations and would ideally be avoided but is included here to determine the degree to which it is required. Using only simulated data for adequate training, we show that our data generation workflow can be fully automated and produces a considerable amount of high quality data rapidly and with minimal effort.

4.3 Introduction

Acquiring high quality images with scanning electron microscopes (SEMs) has led to a comprehensive understanding of a variety of phenomena from the micrometer to nanometer scale. Often, analysis of acquired images leads to quantification; qualitative analysis is not sufficient to thoroughly describe processes [1]. A variety of quantitative measurements from SEM images can be of interest, including preferential orientation of precipitates, dislocation structures of additively manufactured materials and nanoparticle distributions. Currently, computing these measurements from SEM images requires segmentation of features of interest.

In fact, improving segmentation results is possible by selecting the appropriate parameters during acquisitions. Indeed, the set parameters on the SEM will determine the observed image quality, including the resolution, noise, and contrast. Segmentation algorithms are limited by the signal-to-noise ratio (SNR), resolution and contrast in images. Determining the appropriate values for parameters, to optimize image quality, is a daunting task that often takes microscopists a significant amount of time prior to acquisitions. To assist microscopists, the proposed solution is to train a regression model, through supervised learning, to automatically suggest microscope parameter values to set on the microscope during acquisitions.

Multiple microscope parameters are involved in acquiring high quality images on the SEM [2]. As a proof of concept, the regression model is trained to predict two parameters that are both important for image quality: the beam energy and the probe current. Varying both parameters results in an apparent difference in image quality and the optimized values to select depend on the features to analyze and the sample being imaged.

The beam energy will determine the shape of the interaction volume within the sample [3]. At higher energies the interaction volume is wider, including more lateral signal from the sample, altering the image's lateral resolution. Electrons will also have more energy to travel deeper within the sample resulting in increased depth information, and features that are further from the sample surface are detected. Conversely, at lower energies, the interaction volume is reduced both laterally and in depth and only signal from or near the surface of the sample is collected.

The probe current will determine the size of the probe diameter at the sample surface, adjusting the signal-to-noise ratio (SNR) and resolution [3]. At low currents, the SNR is low, producing noisy images. Increasing the SNR by increasing the current can reveal features previously hidden by noise. However, increasing the probe current excessively will also increase the probe size which can lead to loss of resolution, masking features, since they can no longer be resolved.

Ultimately, the objective is for the model to suggest the appropriate beam energy and probe current to obtain images where features of interest are easily segmented for quantification.

This paper is the first of two parts, providing a complete description of the methods used to generate data for training. Model performance relies greatly on the data it is trained with, therefore generating an adequate training set is a crucial step to successful predictions. A thorough explanation of workflows for collecting both simulated data and real data is relevant, more importantly for simulations that require creating virtual samples and setting up the virtual microscope. Training examples are also diversified, as it is an important consideration when developing a well-balanced training set.

4.4 Data generation workflows

By varying both the beam energy and the probe current, multiple images are obtained to generate a training set. The specific sample used for training is platinum nanoparticles (Pt NPs) on carbon nanotubes (CNTs), shown in Figure 4.1. The features of interest in this sample to be segmented and quantified are the Pt NPs. There is a high atomic composition contrast between the carbon and platinum that is especially evident in the backscattered electron (BSE) images shown in Figure 4.1 (middle). Here, a noticeably higher signal (brighter) is evident where the Pt NPs are located, resulting in an easier segmentation of NPs with BSE images when compared with secondary electron (SE) images in Figure 4.1 (left). Facilitated segmentation of Pt NPs is demonstrated when comparing the SE and BSE images, where some of the NPs are clearly detected in the BSE image but are not visible in the SE images. For these reasons, the BSE images are more appropriate, in this context, to use as training examples for training the model.

In Figure 4.1, the SE and BSE images are both real images acquired with the SEM. It is also possible to create virtual samples for simulations, illustrated in Figure 4.1 (right). The virtual samples are 3D MultiROI objects, designed to contain multiple spheres of different sizes and at different depths in the sample. The MultiROI consists of two classes, the white spheres are labelled as platinum and the dark background is labelled as carbon. From virtual samples, BSE images are simulated with the MC X-ray simulator, explained in Section 4.4.1, and included during training for a complete, extensive training set.

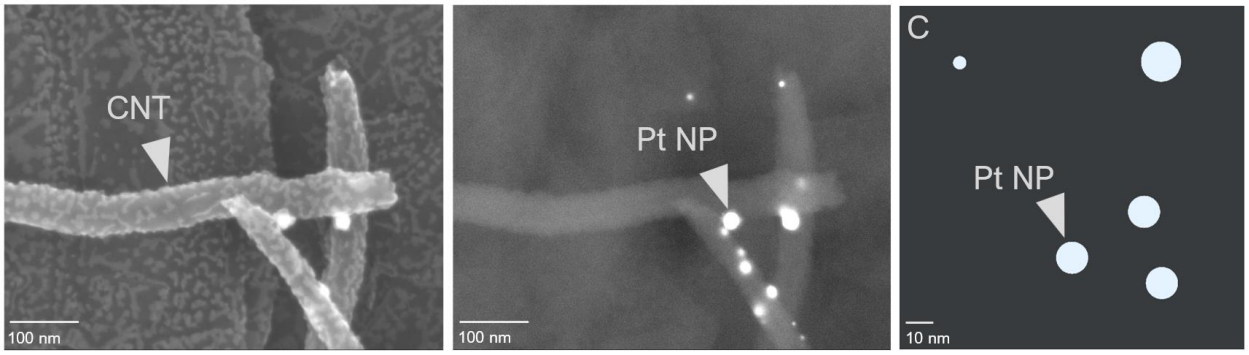


Figure 4.1 (Left) Secondary electron (SE) image and (Middle) back scattered electron (BSE) image of platinum nanoparticles on carbon nanotubes and (Right) Surface of volumetric virtual sample, a labelled 3D MultiROI, background labelled as carbon (C) and white spheres labelled as platinum (Pt).

Each training example consists of a BSE image of the sample, the beam energy and the probe current used to generate the image and an associated Dice score, listed in Table 4.1. All BSE images are 130x130 pixels and all pixel values are normalized between 0 and 255. Image normalization is commonly implemented prior to training, it improves the model's convergence and generalization efficiency [4]. The beam energy will range from 1 to 20 keV and the probe current will range from 1.2 to 614 pA. The Dice score, or Dice similarity coefficient, is a common metric used to evaluate segmentation accuracy [5] and is obtained by comparing two segmentations. The BSE image segmentation is compared to a ground truth segmentation and its accuracy is quantified with the Dice score; segmentation procedures are explained in Section 4.4.1.

All training examples are stored in a Python Pandas DataFrame [6], a two-dimensional data structure, where the columns are the four inputs, and every row is a training example instance. The general structure is illustrated in Table 4.1, including 2 training examples, additional training examples are presented in Section 4.6. Training example 1 in Table 4.1 is a BSE image simulated

at 7 keV, 246 pA and 77% Dice score, training example 2 is an acquired BSE image at 12 keV, 30 pA and 82% Dice score. Workflows for generating training examples simulated and acquired are presented in this section.

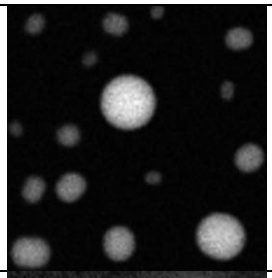
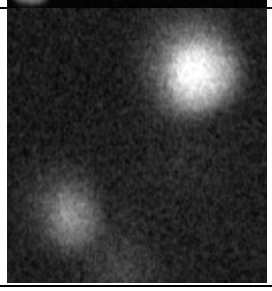
	BSE image (130x130 pixels)	Beam energy (1-20 keV)	Probe current (1.2 – 614 pA)	Dice score (%)
Training example 1		7	246	77
Training example 2		12	30	82

Table 4.1 Representation of how training instances are stored in a Python Pandas DataFrame, a 2D array. Columns are inputs: the BSE image, the beam energy, the probe current and the Dice score. Rows are training examples. Two training examples are shown: training example 1 is simulated and training example 2 is acquired on the SEM.

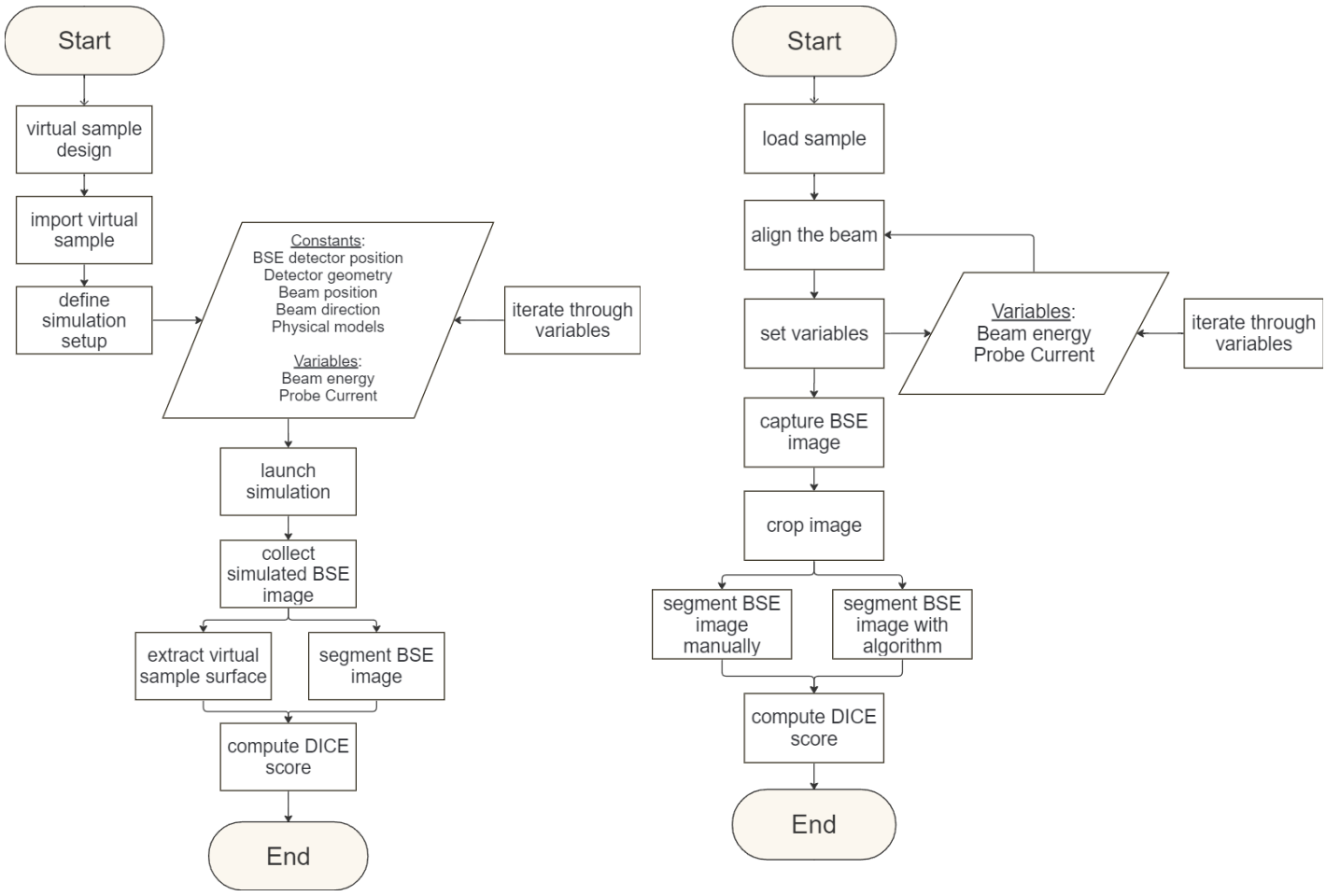


Figure 4.2 (Left) Flow chart diagram of worklow for generating simulated training examples. (Right) Flow chart diagram of worklow for generating acquired training examples.

4.4.1 Simulated training examples

The procedure for generating simulated training examples is entirely virtual. A Python based script in the Dragonfly environment [7] will import a virtual sample, define the simulation set up, generate a BSE image, segment the BSE image, and compute a Dice score. The complete workflow is summarized as a flow chart diagram in Figure 4.2 (left).

Dragonfly is an image processing software with a virtual microscope, or MC X-ray plugin and it provides a variety of options to create virtual samples as well. Virtual sample design is the first step in the flowchart, with the tools available in Dragonfly, samples with features of any complexity can be designed. These components can then be assigned an elemental composition from the periodic table or a chemical composition. The virtual sample is formatted as a MultiROI, a 3D array of values belonging to a class. In Figure 4.3, one of the virtual samples is presented in a 3D view, with spheres (white) labelled as platinum, or class value 1, and the background (grey), labelled as carbon, or class value 2. Images from left to right show the progression, as cross sections are taken from deeper within the virtual sample and new nanoparticles appear, while others disappear. Spheres are intentionally positioned at different depths within the sample to observe the effect of using different parameters in the simulated data.

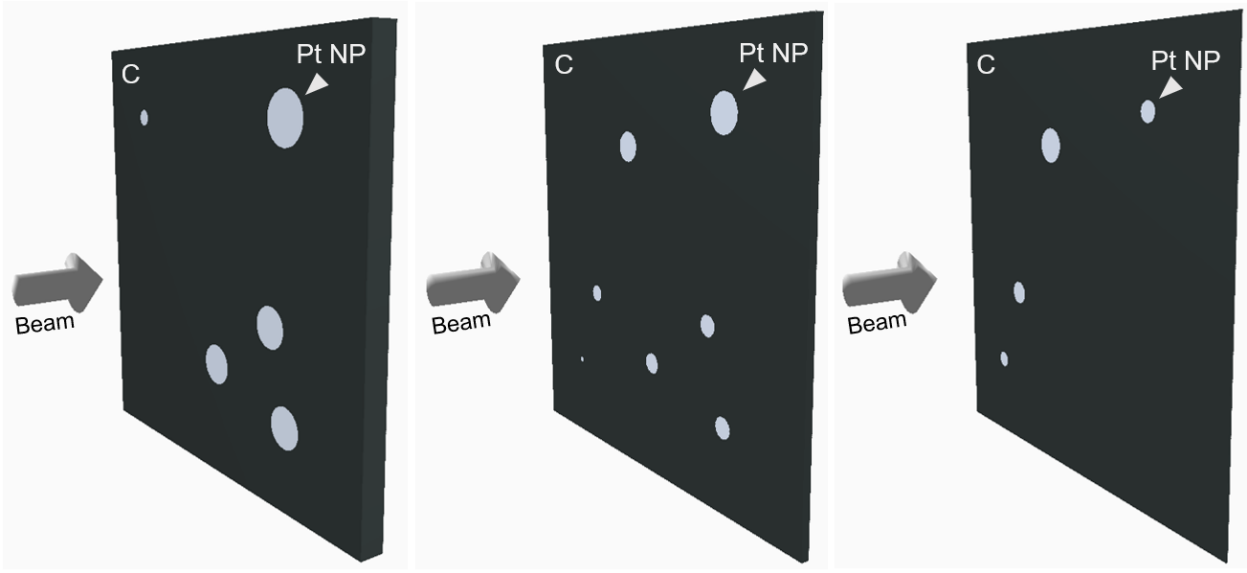


Figure 4.3 Cross sections of MultiROI visualized in 3D, in Dragonfly, with arrow illustrating the direction of electron beam. From left to right, depth into MultiROI increases, new nanoparticles are revealed, while others disappear.

By coding in the Dragonfly environment, the script can easily import the virtual sample, define the entire simulation set up, and implement the virtual microscope functionalities. Indeed, all tasks are executed using the MC X-ray plugin integrated in the Dragonfly software. The simulation set up is first defined by selecting the surface to image on the virtual sample. Selecting the surface will determine the direction of the electron beam, illustrated in Figure 4.3 by an arrow. Next, the detectors are positioned anywhere in space, relative to the virtual sample. For the presented simulations, BSE images are generated and typically, BSE detectors are located under the electron column, at a normal incident angle to the sample. Then, default physical models and microscope and detector parameters are used, such as the working distance, scanning pattern and detector geometry. Any parameters can be modified depending on requirements. Lastly, the simulation parameters are set, with all parameters being kept constant for simulations, except the beam energy and the probe current, or number of simulated electrons. The complete list of setup parameters is summarized in Table 4.2.

Detector geometry	Ring	
	Inner radius: 1 mm	Outer radius: 4 mm
Detector position	3 mm under beam	
Beam direction	0° specimen normal	
Probe size	1 nm	
Beam energy	1 – 20 keV	
Probe current	1.2 – 614 pA	
Acquisition time	100 s	
Physical models	Atom collision	Browning [8][9][10]
	Atom cross section	Browning [8][11]
	Atom energy loss	Bethe [12]
	Atom mean ionization potential	Joy & Luo [13]

	Atom screening	Henoc & Maurice [14]
	Region energy loss	Bethe [12]

Table 4.2 Parameters used in MC X-ray plugin for simulations launched to generate BSE images.

The probe current is defined for simulations by specifying the number of simulated electrons. Increasing the number of electron trajectories increases the number of detected BSEs, resulting in more signal in images. Normally, when working on the SEM, modifying the probe current will also modify the probe size. As previously mentioned, increasing the probe current does increase the SNR, but with an increased probe size, the resolution is negatively impacted. This is not depicted in the simulations, only the number of electrons used for simulations is modified when adjusting the current, without adjusting the probe size. Additional work towards establishing the relationship between the probe size and probe current is required to accurately include the probe size in simulations. Incorporating the probe size would lead to more realistic simulation results and should definitely be considered in future work. The relationship between current and number of electrons used for simulations is shown in Equation 4.1, with the unit for probe current Amperes (A) and the seconds (s) are replaced by the acquisition time.

$$A = \frac{C}{s} = \frac{6.24 \times 10^{18}\ electrons}{s} \qquad 4.1$$

Once the setup is complete, simulations are launched, with multiple combinations of beam energies and probe current. MC X-ray is a simulator, essentially a virtual microscope, scanning the virtual sample surface, where the beam energy and probe current can be specified for every

simulation launched. The plugin is based on Monte Carlo simulations, more specifically, where numerous electron trajectories into the sample are simulated [15]. With a different combination of beam energy and probe current defined for every launched simulation, the virtual microscope will generate multiple BSE images of varying image quality. BSE images are 2D arrays of pixels with grey scale values obtained from the virtual sample (MultiROI), a 3D array of pixels labelled to a class, 1 or 2. The MC X-ray plugin has previously been used, notably to generate synthetic data to train a segmentation model for nanoporous structures [16]. MC X-ray was integrated in Dragonfly and optimized to run over 100x faster, through the use of multithreading. Quicker simulations are extremely beneficial, especially in the context of training a model, since a substantial amount of data is required for optimal model performance.

Generating these simulated BSE images requires minimal effort, and thus it is possible to automatically iterate through multiple beam energies and probe currents without routine beam alignments. The script can run overnight and unattended, and a little under 7000 BSE simulated images were generated for training. Obtaining this substantial amount of data, often required for proper model training, with little effort and in minimal time, is not possible with a real microscope.

For each instance of the simulated BSE image, the script automatically segments pixels as Pt NPs or carbon with a segmentation algorithm of choice. For our sample, Pt NPs on CNTs, the segmentation is performed with Otsu thresholding, a suitable algorithm since a simple binary classification is required and platinum on carbon exhibits a high contrast. Other segmentation algorithms are available in the Dragonfly environment, and they could potentially classify pixels or features of interest more accurately than thresholding.

The final step is to compute the Dice score by comparing the Otsu segmented BSE image with the surface of the virtual sample. The surface of the virtual sample is used as a ground truth, from which the BSE simulated image is generated. The surface represents the true segmentation, where pixels are correctly classified as Pt NPs or carbon. The Dice score will be higher when the Otsu segmentation correctly classifies BSE image pixels.

4.4.2 Acquired training examples

The workflow for acquiring data on the SEM to generate training examples is not fully automated: a few first steps in the flowchart diagram in Figure 4.2 (right) are executed on the microscope and the rest is implemented in the script. The image acquisition process is completed on the SU8230, a cold field emission gun Hitachi microscope [17].

First the sample is loaded and then, the beam is aligned, as a routine procedure. Beam alignment is a manual task, and it is known to take a significant amount of time as it must be repeated every time a parameter is modified. As illustrated in Figure 4.2 (right), as users iterate through the acquisition parameters being varied, there is a constant loop between variable modification and beam alignment. This additional step alone makes collecting a large amount of data on the microscope much more time consuming and labor intensive than launching simulations.

The probe current is determined by selecting an emission current, a probe mode (High, Medium, Low) and a condenser lens current. Modifying any of these values will have an impact on the probe current, which is the current of the beam as it hits the sample surface. The resulting probe current is measured by inserting a Faraday cup in the vacuum chamber [18] following the beam alignment routine. This instrument is a Pico-ammeter allowing for precise measurement of the

current as the beam exits the column and hits the sample. Indeed, it is important to measure and collect the probe current of each BSE acquired image for training, as the beam current will undergo a large reduction of several orders of magnitude, from µA to pA, as the beam traverses the column, experiencing deviations by the lenses and filtering from the apertures.

Also, when acquiring images at lower energies (< 3keV) sample surface cleaning is often required as an additional step. Before loading the sample into the microscope, it should be inserted into an ozone cleaner, for example, to remove any surface contaminants. Lower energies generate small interaction volumes, close to the surface, and if the sample is not cleaned, images will contain signal from the contamination instead of the sample surface.

Once the beam energy and probe current are set, and the beam is aligned, the BSE image can be captured. Acquired BSE images are then imported to the Python script in the Dragonfly environment. BSE images in training examples are all 130x130 pixels and therefore captured images need to be cropped since microscopes typically have set image sizes. The cropped regions are then segmented both manually and with Otsu thresholding. For the simulations, the ground truth is the initial virtual sample segmentation, however, there exists no ground truth segmentation for acquired BSE images. Instead, a manual segmentation is used as ground truth and the Dice score is computed by comparing the two segmentations: the Otsu segmentation and a manual segmentation.

Both beam alignment after every parameter change, and manual segmentation of multiple images for ground truth data, are bottleneck steps that prevent workflow automation. Manual tasks are required for acquiring training examples, including sample cleaning, sample loading, beam alignment, probe current measurement and ground truth segmentation, whereas the workflow for generating simulated training examples is fully automated. Ideally, including real data to the

training set should be avoided, as only 462 training examples were collected in significantly more time than it took to generate the 6722 simulated training examples. In Part 2 of this paper, we show that acquired data is beneficial for model predictions and the importance of virtual sample design is explained. It is demonstrated that models trained with only simulated training examples, generated from poor virtual sample design do not produce best results, necessitating acquired training examples for improved model accuracy.

4.5 Diversifying the training set

When building a training set, it is important to consider not only the quantity, the number of training examples, but also the diversity of the training examples. The model learns from the distribution of the input data, and without sufficient variety, it will only perform well for specific cases. By diversifying the virtual samples used to generate BSE images, the regression model is intentionally trained with as much variety as possible therefore improving its prediction accuracy [19]. A total of twenty different virtual samples were used for simulations.

Virtual samples with nanoparticles of different sizes and at different depths were added, illustrated in Figure 4.3. Also, nanoparticles of different shapes are added (non-spherical) and virtual samples at different magnifications are used for simulations: 20,000x, 60,000x, 80,000x, 100,000x, 200,000x, shown in Figure 4.4.

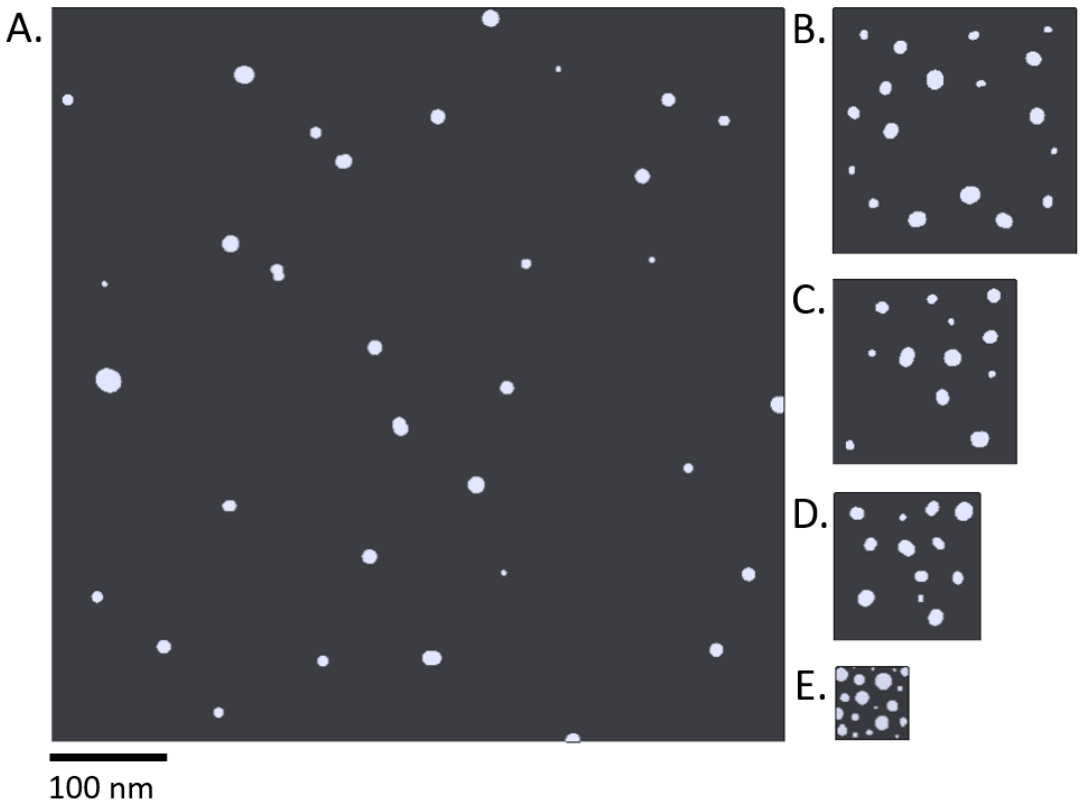


Figure 4.4 Surface of virtual samples used to simulate BSE images with nanoparticles of different shapes and sizes and at multiple magnifications A. 20,000x B. 60,000x C. 80,000x D. 100,000x and E. 200,000x to diversify the training set.

Virtual samples with nanoparticles positioned at various depths within the sample will produce different BSE simulated images, particularly when varying the beam energy. Higher beam energies will detect nanoparticles further from the surface, whereas lower beam energies will only detect nanoparticles on or near the surface. Indeed, at lower beam energies, the interaction volume is limited to the surface.

Additionally, virtual samples with few nanoparticles are included, shown in Figure 4.5A, to penalize noise. Indeed, when high nanoparticle density virtual samples are used (Figure 4.5C), it is more probable that noise will fall correctly on where a nanoparticle is located and contribute to the Dice score. With virtual samples that have low nanoparticle density (Figure 4.5A), noise is

more likely to fall incorrectly on background pixels and penalize the Dice score. Generated BSE images with 3 keV and 1.23 pA from virtual samples in Figure 4.5A and Figure 4.5C are shown in Figure 4.5B and Figure 4.5D, respectively. With such a low probe current, simulated BSE image are very noisy, the low density nanoparticle virtual sample in Figure 4.5A, is therefore highly penalized with a 3.7% Dice score compared to the high density nanoparticle virtual sample in Figure 4.5C with a 70.9% Dice score.

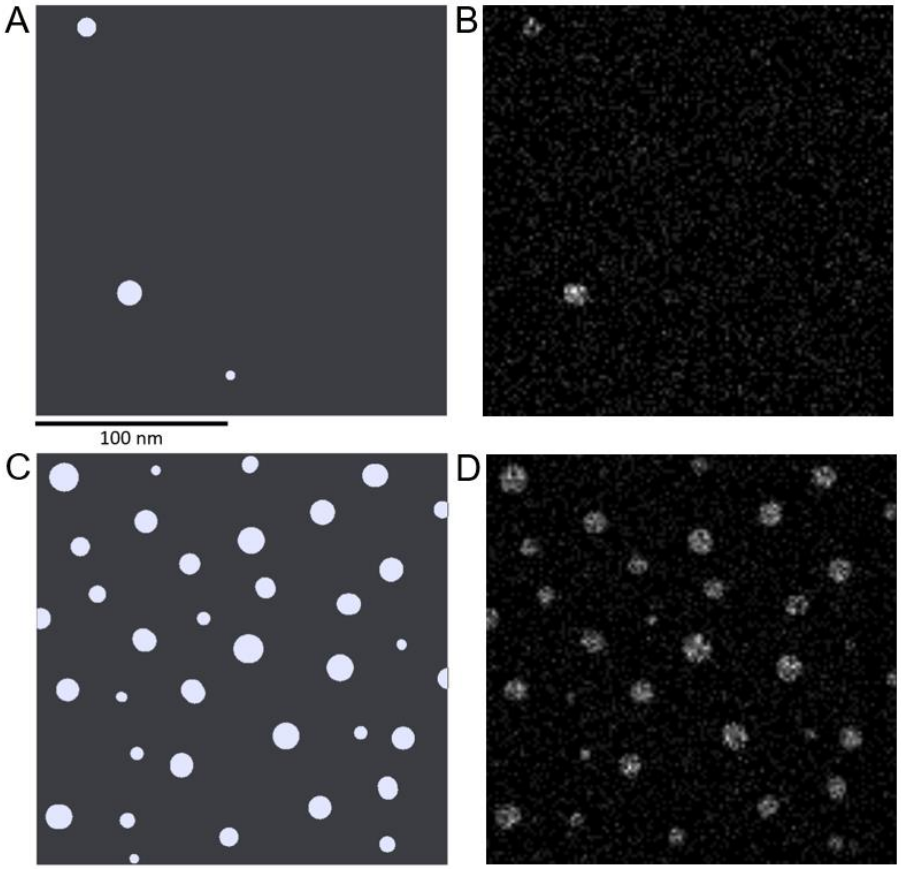


Figure 4.5 A. Surface of virtual samples with low density nanoparticles. B. BSE simulated image from low density nanoparticle virtual sample with 3 keV and 1.23 pA producing a Dice score of 3.7% C. Surface of virtual samples with high-density nanoparticles. D. BSE simulated image from high density nanoparticle virtual sample with 3 keV and 1.23 pA producing a Dice score of 70.9%.

The training set is further diversified by including real images acquired on the SEM. Including a considerable amount of real data for adequate training is, however, as mentioned, a non-automated, time-consuming process. Ideally, the training set would consist of only simulations since they require minimal effort to generate, seeing as data generating scripts can run overnight and unattended, while SEM acquisitions involve microscopist intervention and repetitive tasks. The complete list of characteristics considered when generating diversified virtual samples is summarized in Table 4.3.

Characteristics	Values
Nanoparticle size	Sphere diameter 2 – 20 nm
Nanoparticle shape	Spherical Non spherical Elliptical
Nanoparticle density	Low: 0.1% High: 25%
Nanoparticle depth	Surface: 0 nm Thickness: 15 nm
Sample magnification	20,000x 60,000x 80,000x 100,000x 200,000x
Type of sample	Simulated Acquired

Table 4.3 Virtual sample characteristics included to diversify generated BSE images in the training set.

4.6 Results

Workflow results for both simulated and acquired training examples are presented in Figure 4.6 and Figure 4.7, respectively.

4.6.1 Simulated training examples

Four selected examples of BSE simulated images are shown in Figure 4.6 (middle). Images contain grey level values, generated from scanning the surface of the virtual sample, the 3D MultiROI, shown Figure 4.6 (left). Each simulation is launched with a different combination of parameters, the highest and lowest beam energies with the highest and lowest probe currents are selected for presentation, to emphasize the effects of parameters on the simulation output. The segmentation results are shown in the right column in Figure 4.6, pixels classified as belonging to nanoparticles are white and pixels classified as background are dark grey.

Figure 4.6A was simulated at 1 keV and 1.2 nA. It evidently produces a very noisy image with low SNR due to the low probe current and the BSE image only contains surface information due to the low voltage. The Otsu segmentation is shown to the right of the BSE image and when compared visually with the surface of the virtual sample, it is obvious that a lot of noise is classified as nanoparticles. The parameters are not appropriate for imaging this sample, the segmentation results in a low Dice score of 22%. Figure 4.6B was simulated at 1 keV and 614 nA. This image is simulated at low voltage but high current, resulting in much more signal and therefore a much better segmentation than the first example in A. Nanoparticles under the sample surface are faintly detected but the Otsu segmentation does not include them entirely. With these parameters, the segmentation is satisfactory, with a relatively high Dice score of 93%. Figure 4.6C was simulated at 20 keV and 1.2 pA. Low current will necessarily lead to low signal as

clearly observed in the BSE simulated image. However, with a high voltage, electrons travel much deeper within the sample, including information behind the nanoparticles. With these parameters, the computed Dice score is low at 25%. Figure 4.6D was simulated with 20 keV and 614 pA. The high current produces an image with sufficient signal, the nanoparticles are clearly defined. With high voltage, nanoparticles further from the surface are now fully detected. These nanoparticles are not present in the initial virtual sample surface. The Dice score is 83% with these parameters. The score is penalized by the additional segmented nanoparticles that are now detected with higher energies.

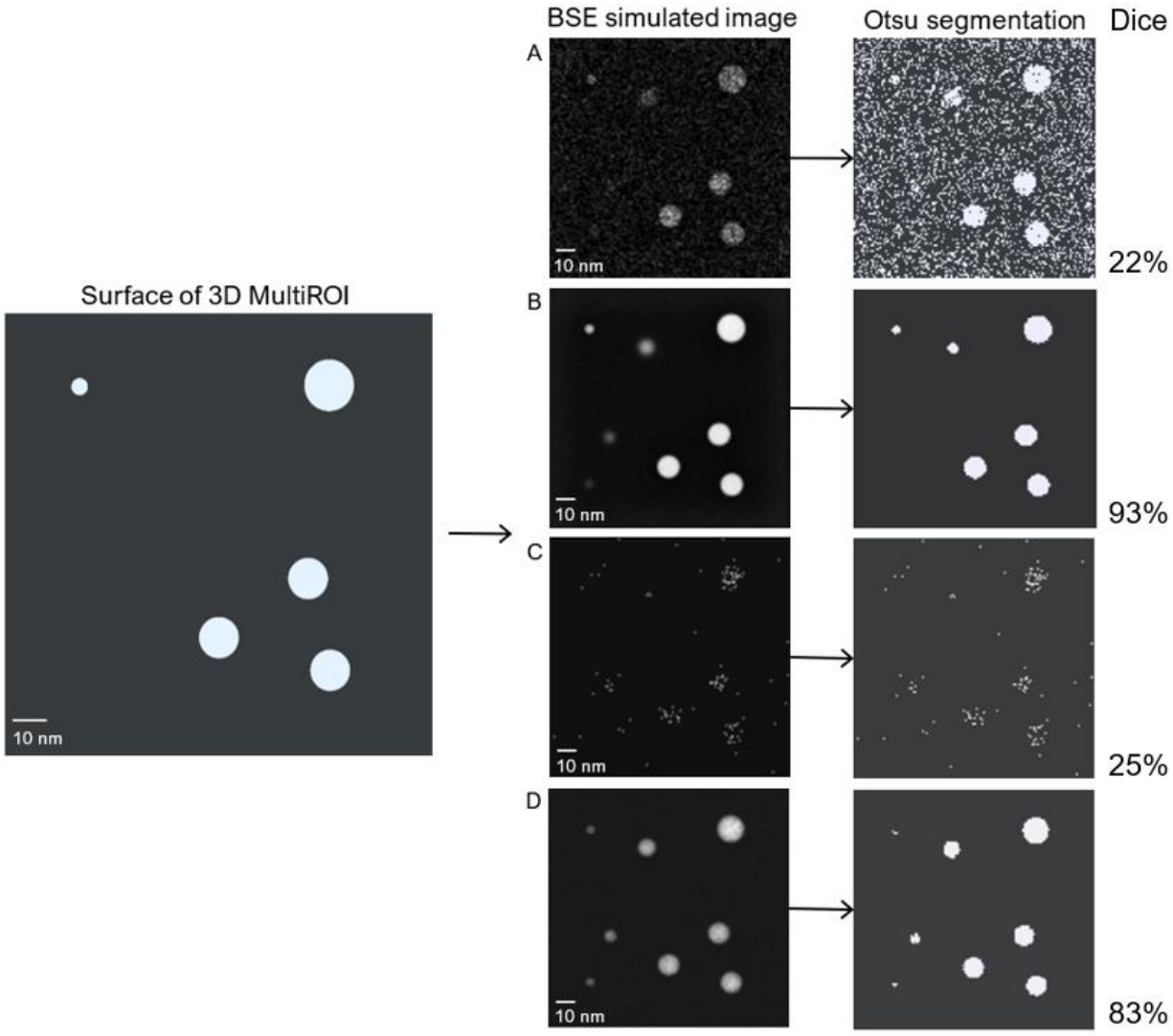


Figure 4.6 (Left) Virtual sample surface used to compare with final segmentation. (Middle) Selected examples of BSE simulated images (grey scale values) from the surface of the virtual. (Right) Associated Otsu segmentation. Images are simulated with different beam energies and probe currents and segmentations are attributed a Dice score listed next to Otsu segmentation A. 1 keV, 1.2 pA, Dice: 22% B. 1 keV, 614 pA, Dice: 93% C. 20 keV, 1.2 pA, Dice: 25% D. 20 keV, 614 pA, Dice: 83%.

4.6.2 Acquired training examples

The process of acquiring real data was included for experiments, to test model performance when trained with and without real images. As shown in Figure 4.7, two regions (A and B) are cropped of 130x130 pixels on acquired BSE images and extracted for training. Both regions provide the model with useful information on the sample with observed variations in image quality for different combinations of beam energies and probe currents. For instance, the first area Figure 4.7A) highlights the relationship between the current and the SNR. The second area (Figure 4.7B) reveals that new features appear as the energy increases, features that were located deeper within the sample, which is comparable to positioning nanoparticles at different depths in the virtual samples.

The beam energy and probe current used during acquisitions are listed above each cropped region and Otsu segmentations are presented under each cropped regions in Figure 4.7. When compared to a manual segmentation, the first row, representing the cropped region in Figure 4.7A produced segmentations with higher Dice scores than the second region in Figure 4.7B. Associated Dice scores are noted on the bottom right of all segmentations in Figure 4.7. For region A, at 3 keV and 17 pA, a 88% score is obtained, at 5 keV and 6 pA, the score is 65%, at 10 keV and 104 pA, the score is 84% and at 20 keV and 70 pA, the score is 89%. For region B,

at 3 keV and 17 pA, a 74% score is obtained, at 5 keV and 6 pA, the score is 36%, at 10 keV and 104 pA, the score is 50% and at 20 keV and 326 pA, the score is 55%.

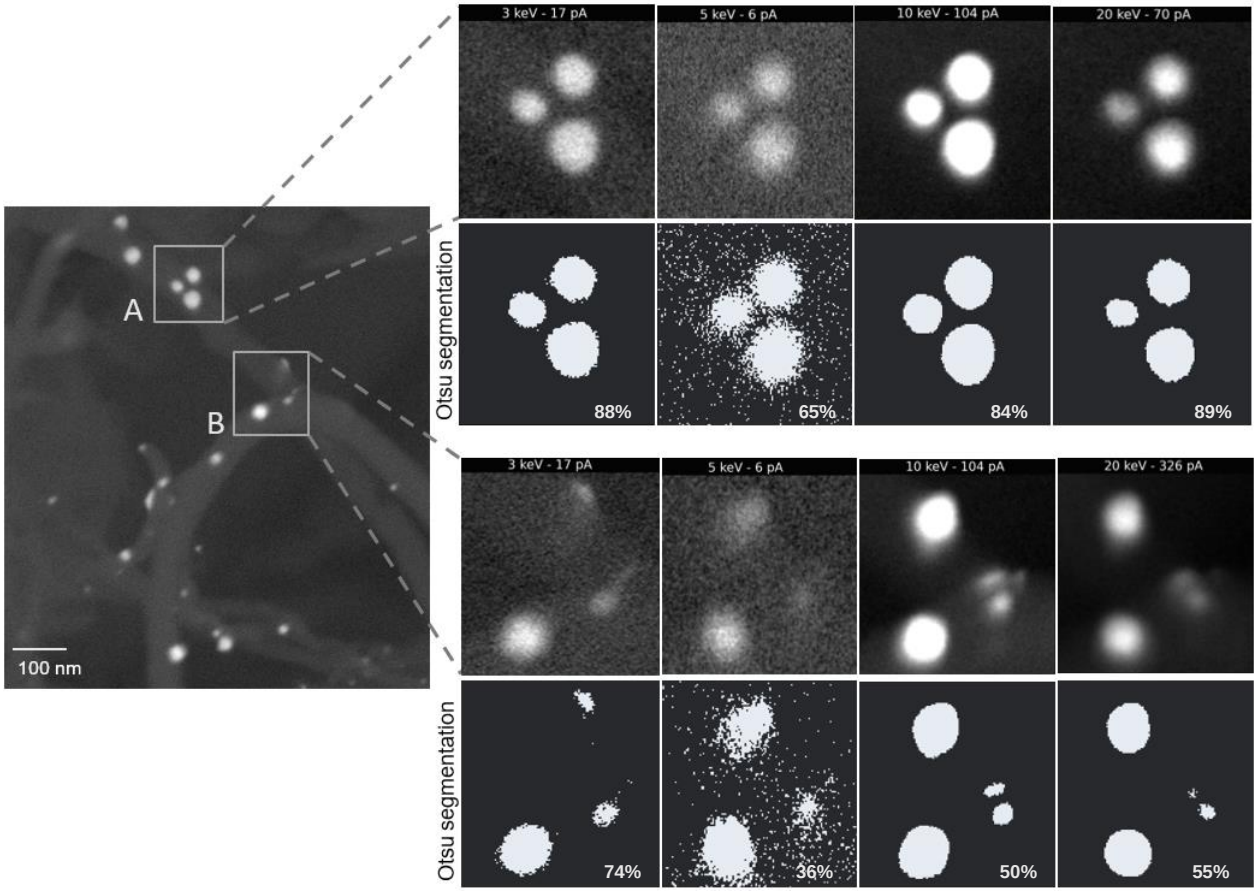


Figure 4.7 Cropped areas on SEM acquired BSE images for training A) obvious increase in SNR with increased current and B) Pt NPs are revealed as energy increases. Combinations of beam energy and probe current lead to observed variations in image quality. Segmentation results obtained with Otsu thresholding are shown under each cropped BSE image with computed Dice score in bottom right of segmentation.

4.7 Discussion

4.7.1 Virtual sample design

Virtual sample design will have a huge impact on resulting training examples and therefore on the model's prediction tendencies. Including virtual samples with Pt NPs located deeper within the sample will penalize higher energies as shown in Figure 4.6, where the training example in Figure 4.6B has a higher Dice score than the training example in Figure 4.6D. The training example in Figure 4.6B is generated with a lower beam energy, while training example in Figure 4.6D is generated with a higher beam energy for the same current. Higher beam energies will detect Pt NPs farther from the surface and the Otsu segmentation will classify them as NPs, although they are not present in the surface segmentation. The surface segmentation is used as ground truth, to compare with the Otsu segmentation to obtain a Dice score. With this design, higher energies are penalized by receiving a lower score, and the model will learn it is not always favorable to use the maximum energy during acquisition. Depending on the context, and the material being imaged, it can be beneficial to use lower energies, as they could avoid or reduce some forms of beam damage. If higher energies are favorable, slices further from the surface can be used as ground truth to compute the Dice score.

4.7.2 Extending workflow to other contexts

The training set is presented in a very specific context, with images from one sample and with a particular segmentation algorithm. The proposed model can be extended to any other context by generating a training set with any sample and any segmentation algorithm. More complex algorithms are available to implement as an alternative, and they could potentially produce better segmentation results than thresholding.

Virtual samples presented in this paper are designed manually and are relatively straightforward to create, by adding multiple spheres throughout the volume. Not all samples have simple, spherical nanoparticles to be analyzed, and when more complex shapes are to be constructed, algorithms should be used to automatically generate the virtual samples. Available libraries such as OpenPNM[20] and PuMA[21] are implemented in Dragonfly and generate a variety of samples, a few are shown in Figure 4.8. Sample geometries range from spheres to cylinders to lattices to Voronoi and more. Sample size, density or porosity and size of features is specified, and positioning and orientation of features are randomized.

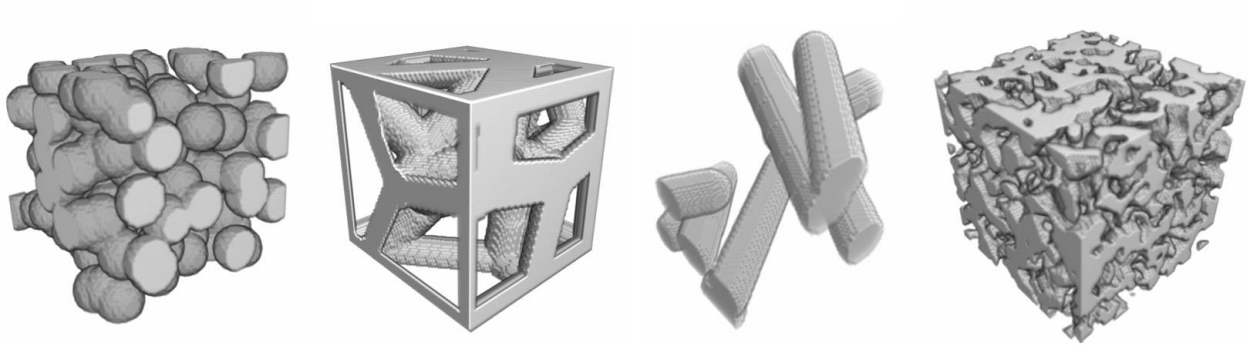


Figure 4.8 Examples of automatically generated samples with OpenPNM or PuMA in Dragonfly, from left to right, random spheres, Voronoi edges, random fibers and porous structures.

Furthermore, the current workflow assumes that the sample compositions are known prior to imaging. To generate a training set with MC X-ray simulations, virtual samples must be labelled with the appropriate composition. The workflow can be extended to integrate an energy dispersive spectroscopy (EDS) analysis as a first step for samples with unknown composition. With an EDS map, sample composition is identified, and it can be automatically attributed to features in the virtual samples.

4.8 Conclusion

Generating a training set for deep learning applications requires careful consideration since model performance depends heavily on the data with which it is trained. Often a significant quantity of training examples is necessary, therefore the time and effort required to collect data is important to prioritize. Workflows for generating simulated and acquired training examples are presented and simulated data is clearly beneficial in this context and should be favored over acquiring real data for training. With a generalized workflow, where users can generate virtual samples, set the material composition, and then launch simulations, training examples are easily generated for model training. An excellent training set will not only have numerous training examples but also diversified ones, both of which may be accomplished by creating virtual samples that have multiple, different characteristics to help the model generalize properly.

4.9 References

[1] “The quest for quantitative microscopy,” Nat Methods, vol. 9, no. 7, Art. no. 7, Jul. 2012, doi: 10.1038/nmeth.2102.

[2] S. Chapman, “Understanding & Optimising Scanning Electron Microscopy Performance,” Infocus, no. 14, Jun. 2009, doi: 10.22443/rms.inf.1.45.

[3] J. I. Goldstein et al., “Electron Beam–Specimen Interactions,” in Scanning Electron Microscopy and X-ray Microanalysis: Third Edition, J. I. Goldstein, D. E. Newbury, P. Echlin, D. C. Joy, C. E. Lyman, E. Lifshin, L. Sawyer, and J. R. Michael, Eds., Boston, MA: Springer US, 2003, pp. 61–98. doi: 10.1007/978-1-4615-0215-9_3.

[4] J. Shao, K. Hu, C. Wang, X. Xue, and B. Raj, “Is normalization indispensable for training deep neural network?,” Advances in Neural Information Processing Systems, vol. 33, pp. 13434–13444, 2020.

[5] A. A. Taha and A. Hanbury, “Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool,” BMC Med Imaging, vol. 15, p. 29, Aug. 2015, doi: 10.1186/s12880-015-0068-x.

[6] “pandas documentation — pandas 2.2.0 documentation.” Accessed: Feb. 08, 2024. [Online]. Available: https://pandas.pydata.org/docs/index.html

[7] “Dragonfly | 3D Visualization and Analysis Solutions for Scientific and Industrial Data | ORS.” Accessed: Sep. 25, 2023. [Online]. Available: https://www.theobjects.com/dragonfly/index.html

[8] R. Browning et al., “Empirical forms for the electron/atom elastic scattering cross sections from 0.1 to 30 keV,” Journal of Applied Physics, vol. 76, no. 4, pp. 2016–2022, Aug. 1994, doi: 10.1063/1.357669.

[9] D. Drouin, R. Gauvin, and D. C. Joy, “Computation of polar angle of collisions from partial elastic mott cross-sections,” Scanning, vol. 16, no. 2, pp. 67–77, 1994, doi: 10.1002/sca.4950160202.

[10] Mott N. F., The Theory Of Atomic Collisions. 1949. Accessed: Feb. 09, 2024. [Online]. Available: http://archive.org/details/in.ernet.dli.2015.3748

[11] Z. Czyżewski, D. O. MacCallum, A. Romig, and D. C. Joy, “Calculations of Mott scattering cross section,” Journal of Applied Physics, vol. 68, no. 7, pp. 3066–3072, Oct. 1990, doi: 10.1063/1.346400.

[12] H. Bethe et al., Eds., Quantentheorie. Berlin, Heidelberg: Springer, 1933. doi: 10.1007/978-3-642-52619-0.

[13] D. C. Joy and S. Luo, “An empirical stopping power relationship for low-energy electrons,” Scanning, vol. 11, no. 4, pp. 176–180, 1989, doi: 10.1002/sca.4950110404.

[14] K. F. J. Heinrich, D. E. Newbury, and H. Yakowitz, Use of Monte Carlo Calculations in Electron Probe Microanalysis and Scanning Electron Microscopy: Proceedings of a Workshop Held at the National Bureau of Standards, Gaithersburg, Maryland, October 1-3, 1975. U.S. Department of Commerce, National Bureau of Standards, 1976.

[15] R. Gauvin and P. Michaud, “MC X-Ray, a New Monte Carlo Program for Quantitative X-Ray Microanalysis of Real Materials,” Microsc Microanal, vol. 15, no. S2, pp. 488–489, Jul. 2009, doi: 10.1017/S1431927609092423.

[16] T. Sardhara et al., “Training Deep Neural Networks to Reconstruct Nanoporous Structures From FIB Tomography Images Using Synthetic Training Data,” Frontiers in Materials, vol. 9, 2022, Accessed: Nov. 13, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fmats.2022.837006

[17] H. H.-T. in Canada, “Scanning Electron Microscope FlexSEM 1000 II,” Hitachi High-Tech in Canada. Accessed: Jul. 18, 2023. [Online]. Available: https://www.hitachi-hightech.com/ca/en/products/microscopes/sem-tem-stem/sem/flexsem1000.html

[18] “PELCO® Faraday Cup.” Accessed: Feb. 12, 2024. [Online]. Available: https://www.tedpella.com/calibration_html/Faraday_Cup_for_Electron_Beam_Current_Measurement.aspx

[19] Z. Gong, P. Zhong, and W. Hu, “Diversity in Machine Learning,” IEEE Access, vol. 7, pp. 64323–64350, 2019, doi: 10.1109/ACCESS.2019.2917620.

[20] J. Gostick et al., “OpenPNM: A Pore Network Modeling Package,” Computing in Science & Engineering, vol. 18, no. 4, pp. 60–74, Jul. 2016, doi: 10.1109/MCSE.2016.49.

[21] J. C. Ferguson, F. Panerai, A. Borner, and N. N. Mansour, “PuMA: the Porous Microstructure Analysis software,” SoftwareX, vol. 7, pp. 81–87, Jan. 2018, doi: 10.1016/j.softx.2018.03.001.

5 Optimizing SEM Parameters for Segmentation with AI – Part 2: Designing and Training a Regression Model

Sabrina Clusiau, Nicolas Piché, Benjamin Provencher, Mike Strauss, Raynald Gauvin

Manuscript currently under review

5.1 Preface

This chapter presents the second part of optimizing microscope parameters to improve image quality. While the first part describes how a training set is generated, this second part explains the intricacies behind designing and testing a model. A new model architecture is created to perform tasks adapted to the specific context of improving the image quality for segmentation. From the generated training set, the model is trained to predict the optimal beam energy and probe current to set during acquisitions on the SEM. The predicted microscope parameters will modify output image quality to simplify the segmentation of designated features in Part 1.

With new model architectures, model parameters need to be optimized for best performance, achieved by training and testing. First, learning curves are plotted during training with the generated training set to supervise the model as learning progresses. Then, model accuracy is obtained by testing model predictions with a portion of the generated data for training (split into training, validation and testing sets). Once high accuracy is achieved, the model is tested in real time, by integrating predictions into a routine acquisition on the SEM. Model testing in this context will ultimately provide insight on how to proceed in the future to generate an adequate training set for other samples in other contexts.

5.2 Abstract

Selecting the best microscope parameters for optimal image quality currently relies on microscopists; there exist no procedures or guidelines for tuning parameters to ensure the desired image quality is achieved. More importantly, for quantitative analysis purposes, adequate image quality for segmentation should be prioritized. This paper is the second of two parts, describing a regression model, mixed input, multiple output with Keras TensorFlow, trained to predict the beam energy and probe current, two important parameters for image quality. Specifically, parameters are predicted to optimize the image quality for segmentation, using a generated training set, as described in Part 1 of this paper. Model performance is then tested on models trained with multiple different training sets, and with different proportions of simulated and acquired data. First, to examine the impact of the training set on the prediction accuracy and then, to evaluate the importance of including real data during training. The model successfully predicted the beam energy and probe current to set on the microscope to improve image quality for segmentation. Models trained with both simulated and acquired data performed the best, as evaluated by their efficacy at improving the image quality for feature segmentation.

5.3 Introduction

Segmentation is a necessary postprocessing step for quantitative analysis, however, it is not always a simple task. Although image processing software offer multiple segmentation algorithms to avoid manual labelling, these algorithms do not always perform well on provided data. The choice of microscope parameters set during acquisitions contribute to image quality for segmentation. The wrong set of parameters can completely hide features to quantify, making it difficult for segmentation algorithms to perform well; or conversely, the appropriate selection of parameters can produce images that highlight features of interest and allow them to be segmented easily. Suitable microscope parameter selection will generally determine the outcome of segmentation success.

Optimizing microscope parameters for quantitative microscopy currently relies on microscopist expertise. It is not reproducible, since microscopists will select the parameters such as the beam energy, the beam current, the magnification, and the capture settings depending on their personal insights, prior knowledge, and competencies. Appropriate parameter selection requires an understanding of how electrons interact with the sample, depending on the sample composition. Furthermore, it is time consuming to manually cycle through beam alignment and parameter tuning while searching for the desired image quality, even for more experienced microscopists. More importantly, even if the selected combination of parameters produces impressive, high-quality images, there is no guarantee that they will be segmented easily by algorithms during postprocessing steps. Without successful segmentation algorithms, users must resort to the alternative, manual labelling, which is known to be extremely time consuming and tedious, and consequently cost ineffective.

The goal is to ensure that acquired images will be segmented easily, more specifically, that the features to quantify are easily segmented. The proposed solution is to integrate artificial intelligence (AI) within SEM acquisition workflows to consistently obtain images that are optimized for efficient segmentation. A regression model is trained to automatically suggest microscope parameters to set during acquisitions, allowing microscopists to spend their time acquiring data instead of tuning parameters.

By training a regression model with supervised learning to automatically suggest what microscope parameters to select, we can improve and simplify data acquisition with SEMs for quantitative analysis by:

- Ensuring the success of segmentation algorithms, by producing images for that purpose (efficiency).
- Providing an unbiased selection of microscope parameters with observed consistency throughout images (reproducibility).
- Eliminating the need for an expertise on electron material interactions, to produce desirable images (accessibility).

The following paper will first present the regression model architecture, explaining its design. Then, the model is integrated into a usual SEM workflow, demonstrating how predictions are applied for testing. Finally, the model performance is summarized.

5.4 Model architecture

5.4.1 Overview

The overview of the model architecture is shown in Figure 5.1, built in Python with Keras TensorFlow. The model is a Functional API, a highly flexible model structure that handles any type of input, numerical or image, combined to predict multiple or single outputs [1]. Acquired or simulated training examples include a BSE image, Dice score, beam energy and probe current. These are imported and contained in a Python Pandas DataFrame [2]. BSE images are simulated using MC X-ray or acquired on a SU8230 SEM from Hitachi, Dice scores are computed from the BSE image segmentation, and the beam energy and probe current are the parameters used when simulating or acquiring the BSE image.

The inputs to the model are the BSE images, and their associated Dice score. The BSE image is a 2D array of pixel values normalized between 0 and 255, and the Dice score is numerical data, a value between 0 and 1. The outputs are the parameters to predict, the beam energy and the probe current. Combining image data and numerical data as inputs with two outputs, makes the model mixed input, multiple output. A complete description of the training set and how it is generated is covered in Part 1 of this paper.

As shown in Figure 5.1, the BSE image is the input to a convolutional neural network (CNN) and the Dice score is the input to a multi-layer perceptron (MLP). A CNN is generally used to train image data, it is preferred since the model will learn information about neighboring pixels during training, whereas a MLP for image data requires vectorization and collapsing the 2D image into a 1D vector will result in loss of spatial information [3]. CNNs will maintain the 2D

structure of the input, and layers will pass filters on inputs to extract features. Following the CNN and MLP layers the inputs are merged with a concatenate layer in Keras TensorFlow [4] to predict the outputs.

The model is designed as such since the goal is for the model to suggest acquisition parameters to set on the microscope. The concept behind including the BSE image as input is for the model to analyze the image and suggest parameters that will improve the image quality. More importantly, the Dice score is included as input as well, for the model to suggest parameters that will successfully improve the image quality for segmentation. For instance, if the input BSE image has a high amount of noise, or low SNR, and the input Dice score is high, the model should suggest a higher probe current.

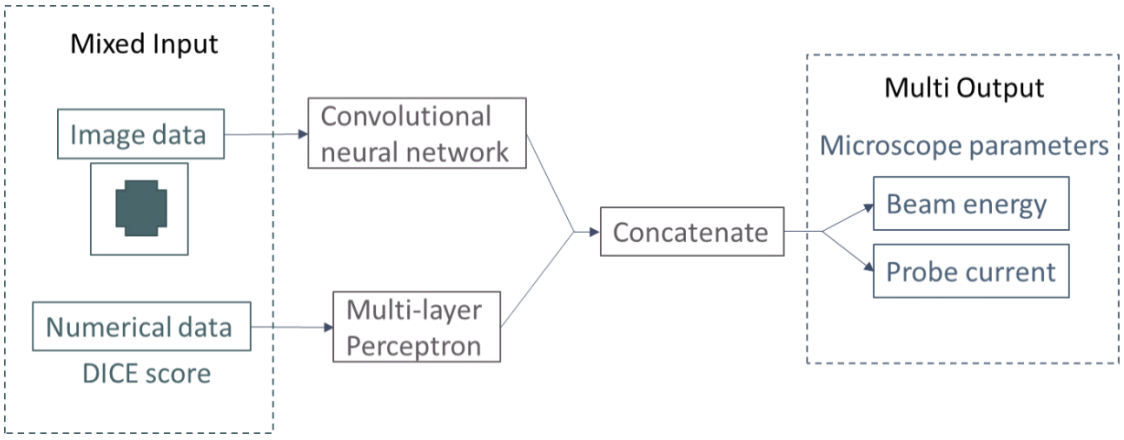


Figure 5.1 Regression model architecture.

5.4.2 Model parameters

A more detailed description of the model architecture is illustrated in Figure 5.2, with all model hyperparameters summarized in Table 5.1 and training details in Table 5.2. There are many parameters to consider when creating model layers, including the depth or number of layers, the number of weights for the MLP layers, the filter sizes, and the number of filters for CNN layers and the input dimensions. Then, there are also techniques to implement at each layer to optimize model performance such as regularization, max pooling, dropout, padding or adding an activation function. Finally, training parameters like the learning rate, the optimizer and the loss need to be selected [5].

Increasing the depth of the model will generally help improve the performance, the exact number of layers was selected depending on the performance during training [6]. The resulting architecture consists of four convolutional layers, two layers of weights for the MLP and two dense layers after the concatenate layer. The last layer, for each output has a linear activation function since outputs are continuous values, as the model solves a regression problem, not classification.

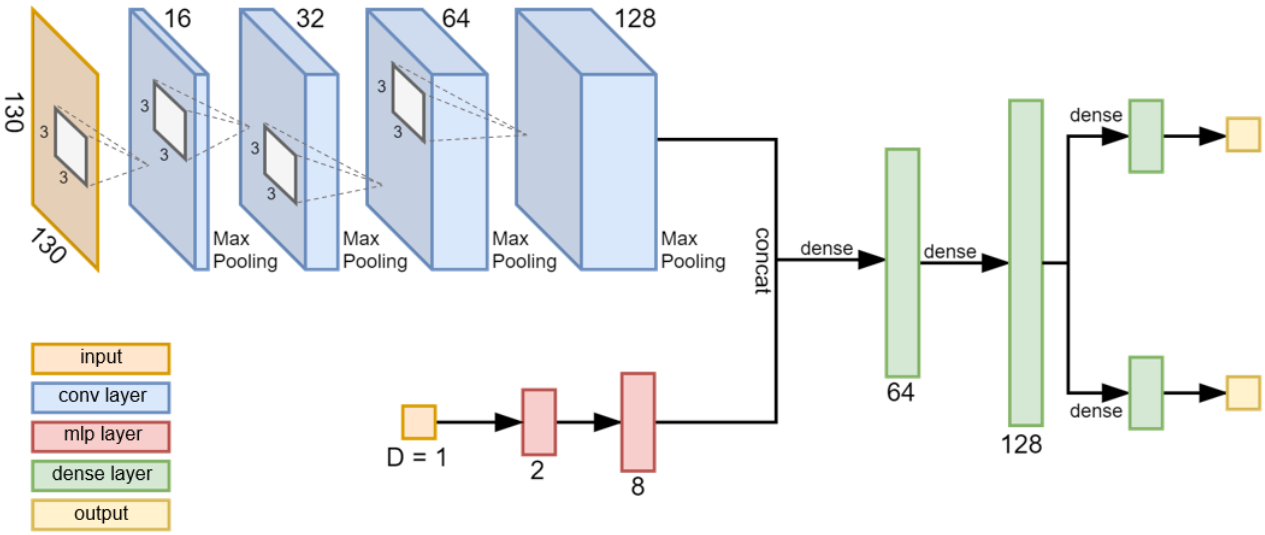


Figure 5.2 Detailed schematic of model architecture.

Following best practices, the number of weights used per layer increases as the layer is deeper in the model [7]. The best depth and width of layers was found by trial and error, evaluating model performance during training with a validation set and testing set. For MLP layers, the first starts with 2 weights and increases to 8 for the second layer. Similarly, the first dense layer after the concatenate layer starts with 64 weights and increases to 128 weights for dense layer 2. For convolutional layers, a similar concept is implemented. The number of filters used per layer increases as the layers are deeper within the model, conv 1 has 16 filters, conv 2 has 32, conv 3 has 64 filters and conv 4 has 128 filters, and all filter sizes are the same at 3x3. Filter size, also known as the receptive field size, are selected to be as small as possible since the input image size is relatively small. Larger filter sizes (5x5 or 7x7) are typically used on larger image sizes [7].

Often, regularization is a technique applied to layers with a high number of weights to manage overfitting [8]. As listed in Table 5.1, L2 regularization is applied to layers Dense 2, Conv 3 and

Conv 4. Max pooling is applied after every convolutional layer to down sample the feature map and emphasize strong features in the images [9].

Model parameters are listed in Table 5.2, the input dimension of images is 130x130x1 and the ReLU activation function [10] is used at every layer, except the last output layers that have a linear activation function. Adding an activation function to layers will increase the model's expressiveness, a technique used to avoid underfitting. A standard loss function for regression models is used, the mean squared error (MSE) [11], although other loss functions (mean absolute error) were also tested. The metric used to evaluate the model's performance on the test set is the adjusted R squared score and explained variance score [12], where results are expressed as percentages and more easily interpreted for performance than a loss value. Commonly, the optimizer that implements the Adam algorithm is applied, since it is computationally efficient and converges well [13]. A typically low learning rate of 0.001 is used as an initial value and the entire data is split randomly into 90% training data, 5% validation data and 5% testing data. Data augmentation was not included for the current context since data generation scripts can simply launch more simulations and generate more training examples, if ever more training data is required.

Finding the appropriate amount of data for training is a known uncertainty with deep learning models. Splitting the data with multiple percentages of train, validation, and test sets, can help establish what amount of training data is necessary for satisfactory model performance. The model should be trained often to evaluate if prediction behavior is as anticipated, or if any corrections are needed.

Layer	Regularization	Weights	Number of filters
MLP 1	-	2	-
MLP 2	-	8	-
Dense 1	-	64	-
Dense 2	L2	128	-
Conv 1	-	-	16
Conv 2	-	-	32
Conv 3	L2	-	64
Conv 4	L2	-	128

Table 5.1 Layer parameters

Input dimension	130x130x1
Layer activation function	ReLU
Loss function	Mean squared error
Metric	Adjusted R^2 score
Optimizer	Adam
Learning rate	0.001
Training data percentage	90
Validation data percentage	5
Test data percentage	5

Table 5.2 Summary of model variables set for training.

The model is designed diligently for best prediction accuracy, and it can be trained with any training set, from any sample. Indeed, the model should be recycled for any context, while the inputs and outputs provided for training may vary, the model architecture remains.

5.5 Model training

The listed hyperparameters and training details in Section 5.4.2 were mainly selected based on review of literature of basic deep learning principles and trial and error during training. Once a considerable amount of data is generated or collected, it is relevant to start training the model to evaluate the performance and make modifications to the model or the training set iteratively. Specifically, the portion of data set aside for validation was heavily used to tune hyperparameters and train the model adequately.

5.5.1 Callbacks

Keras Tensorflow offers built-in callbacks that are easily included in scripts, to optimize training. More specifically, three callbacks are implemented during training. The first callback notifies when an epoch is complete, providing the current loss value. The model loss during training can therefore be plotted dynamically to monitor training progress and determine if the algorithm is converging adequately. Plotted learning curves are shown in Figure 5.3.

The second callback included is ReduceLROnPlateau [14], to monitor the slope of the plotted model loss curves. Parameters used for the callback are listed in Table 5.3, the monitored loss is specified, often the validation loss is preferred. When the loss no longer improves (or plateaus) for a specified number of epochs, defined by the patience, the callback signals to decrease the learning rate by a specified factor. The learning rate will decrease up to a minimum value, also specified when instantiating the callback. Decreasing the learning rate at the right moment will stabilize the training and help the algorithm converge more easily. Without this callback the learning rate will remain the same as the initially specified value throughout training.

EarlyStopping [15] is the third implemented callback that also monitors the slope of the plotted model loss curve. Parameters used for the callback are also listed in Table 5.3, the metric to monitor is selected, often the validation loss, and the callback will signal a halt in training when the loss no longer improves for a number of epochs (patience) within the range of a given delta. The loss function is considered to improve when decreasing, however other metrics can improve by increasing. The selected mode, minimum or maximum, will designate what to consider as an improvement. Without this callback, the model will continue to train for the defined number of epochs prior to beginning training, with no consideration for how the model is performing. Training for an unnecessary number of epochs can lead to overfitting.

		Beam energy	Probe current
ReduceLROnPlateau	Monitor	Validation loss	Validation loss
	Factor	0.06	0.08
	Patience	50	90
	Minimum LR	0.0001	0.0001
EarlyStopping	Monitor	Validation loss	Validation loss
	Minimum delta	0.00001	0.00001
	Mode	Minimum	Minimum
	Patience	70	70

Table 5.3 Parameters used to initialize callbacks ReduceLROnPlateau and EarlyStopping during training.

5.5.2 Learning curves

During training, model performance is tracked, as mentioned, by plotting dynamic validation curves, orange curves in Figure 5.3 and training curves, blue curves in Figure 5.3, of the MSE loss and R squared score. Learning curves are frequently used to diagnose model performance,

they are helpful for tuning model parameters listed in Table 5.1 and Table 5.2 including the number of layers (model depth), the CNN filter sizes, the learning rate and more [16]. Models that overfit or underfit the training set will have training and validation curves with typical features [17]. Interpreting and comparing training and validation curves is extremely helpful for optimizing model parameters and it is common practice during training.

The final validation curves obtained, after model parameter optimization and including callbacks, are shown in Figure 5.3. The losses are plotted over epochs, shown on the x axis, and after about 100 epochs, an obvious stabilization of the curves is observed due to the ReduceLROnPlateau callback. Also, the number of epochs specified for training was 300, however the model stops training after about 120 epochs, the result of the EarlyStopping callback.

Once the training and validation show typical features of a good fit model, the trained model is ready for testing. The portion of data kept for testing is used to predict with the model and evaluate model performance. With this architecture and these parameters, the model performs relatively well on the test set with a 93% R squared score for predicting the beam energy and 98.8% R squared score for predicting the probe current. With satisfactory results, the trained model is saved and easily integrated to data acquisition workflows to predict values.

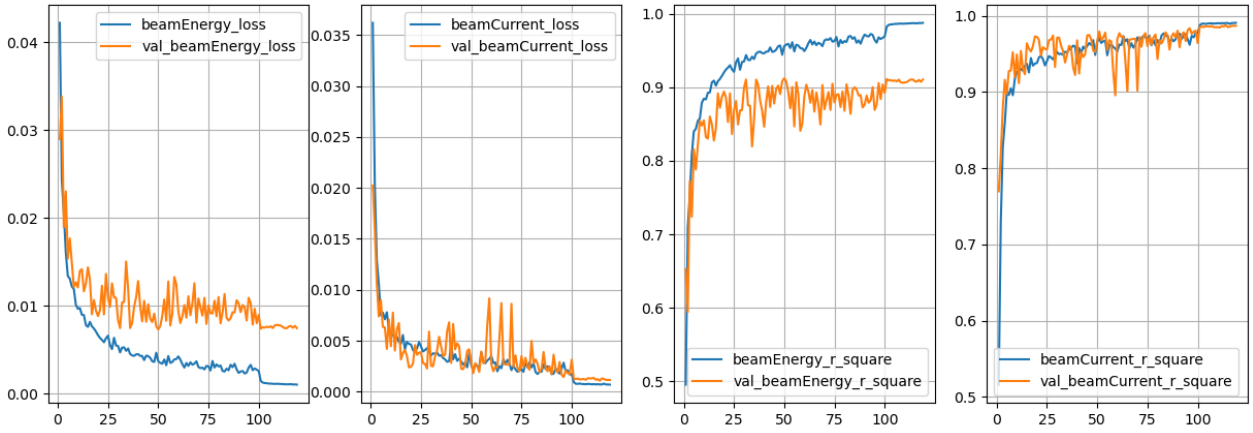


Figure 5.3 Training and validation curves for the MSE beam energy loss, MSE beam current loss, adjusted R squared score for beam energy and adjusted R squared score for beam current.

5.6 Model testing

5.6.1 Workflow

The trained model is integrated into a SEM acquisition workflow for testing, as illustrated in Figure 5.4A. The workflow steps include first loading the sample, and aligning the beam at any two arbitrary parameters, in Figure 5.4A, the beam energy is initially set to 8 keV and the probe current to 13 pA. With the parameters set, the input image is obtained with a BSE detector, shown in Figure 5.4A. The input image contains visible Pt NPs, however, there is no way to determine how easily the NPs will be segmented in this image with Otsu thresholding.

The image is then sent to the model, requesting a relatively high Dice score of 90%, to get a satisfactory segmentation. The trained model will suggest parameters, for this specific case, the suggested beam energy is 3 keV and the suggested probe current is 130 pA. The new parameters are set on the microscope and the beam is aligned to produce the resulting image in Figure 5.4A.

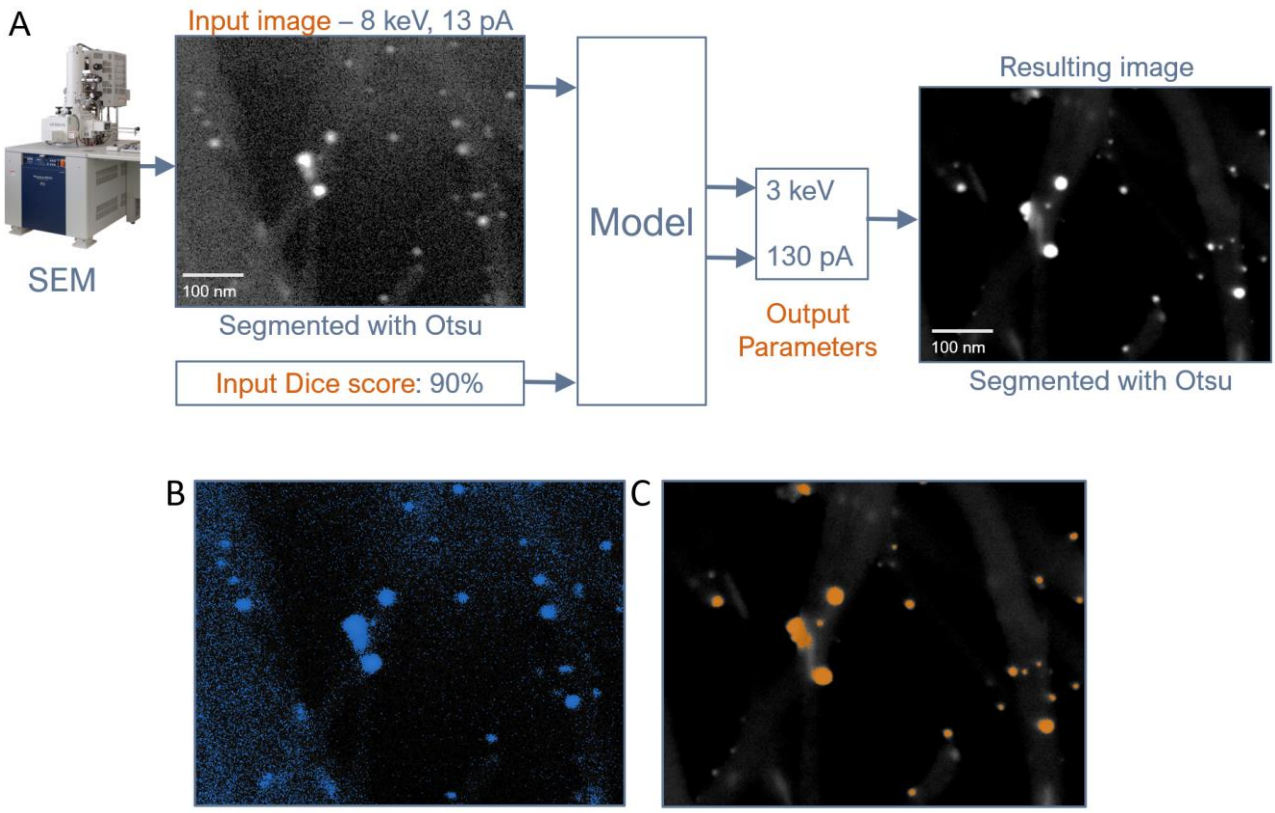


Figure 5.4 A. SEM data acquisition workflow with integrated model for parameter prediction. With the input image and input desired Dice score, the model suggests parameters to set to improve image quality for segmentation. Results of segmentation with Otsu thresholding B. of initial input image in blue and C. of resulting image in orange.

At first glance, the image quality of the resulting image seems improved from the image quality of the input image, facilitating the Pt NPs segmentation. The blue segmentation of the input image includes a lot of noise and some of the CNT pixels are misclassified as well. The orange segmentation of the resulting image is an obvious improvement, labelling only Pt NPs. The noticeable improvement can be validated by computing the Dice score for both images. The nanoparticles are segmented with Otsu thresholding for the input image shown in Figure 5.4B in blue and for the resulting image shown in Figure 5.4C in orange. The Dice score for both images is calculated by comparing each segmentation with the same manual segmentation. The initial

and final Dice scores calculated from the segmentations of the initial and final image respectively, are used to evaluate the model performance. Indeed, parameters are selected to optimize the quality of the segmentation. The objective is that the model suggests parameters that will improve the segmentation of features of interest. The presented workflow, simplified in Figure 5.5, provides a quantitative metric of model performance.

Multiple acquired BSE input images, with a constant requested Dice score of 90%, are used as inputs to the trained model. All input images are acquired with different combinations of beam energies (X_i), from highest energy to lowest, and probe currents (Y_i), from highest current to lowest. Testing a variety of initial combinations of parameters can help determine the model's prediction tendencies and how the model reacts to different input image quality. The model will predict a beam energy (X_f) and a probe current (Y_f), and the new parameters are set on the microscope to obtain a resulting image. With an initial input image and a final resulting image, the image quality is compared using the Dice score as previously mentioned. By comparing the Dice score of the input image and the Dice score of the resulting image, it is possible to determine model performance. Indeed, if the Dice final is higher than the Dice initial in Figure 5.5, then the model successfully suggested parameters that improved the image quality for segmentation.

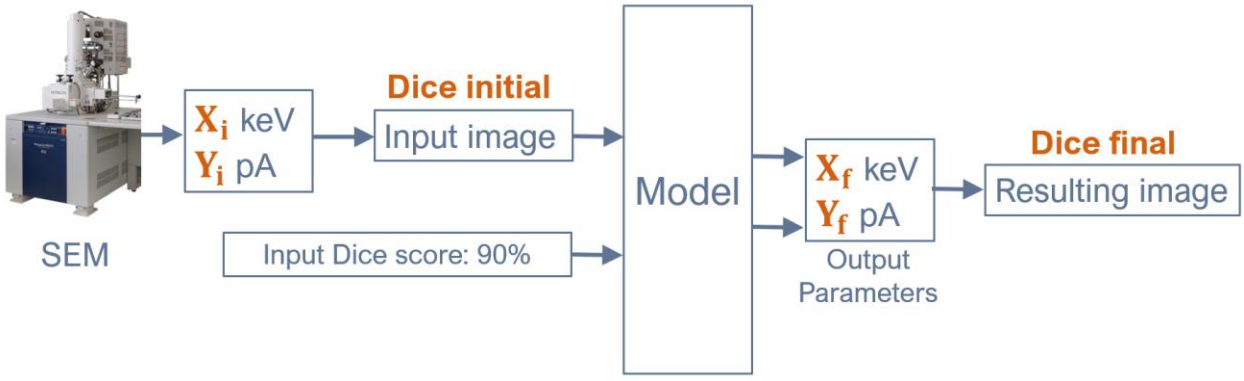


Figure 5.5 Generalized depiction of workflow to test model performance. Multiple images with different combinations of beam energy (X_i) and probe current (Y_i) are sent to the model to evaluate output parameters suggested by the model. Performance is quantified comparing the Dice final and Dice initial.

5.6.2 Test data

For testing, a total of 34 input images are acquired with the SU8230 SEM from Hitachi, using the BSE detector. Images are taken at different beam energies and probe current, but also at different magnifications, 60,000x, 80,000x, 100,000x and 200,000x. Table 5.4 summarizes the parameters used for each of the 34 input images, including the associated Dice score computed by comparing the segmentation with Otsu thresholding of the input image with a manual segmentation. The mean Dice score of all input images in Table 5.4, or mean Dice initial, is 66.6%.

Magnification (1000x)	Beam energy (keV)	Probe current (pA)	Dice score (%)
60	1	381	53.2
60	5	90	72.2
60	10	104	79.8
60	10	34	85.8
60	7	7	83.2
60	12	9	74.5
60	12	57	83.2
60	20	12	58.5
60	20	326	81.2
60	15	51	84.4
60	2	73	72.8
80	10	9	66.3
80	10	4	24.9
80	2	106	73.5
80	3	81	78.8
80	7	22	69.3
80	5	66	60.6

Magnification (1000x)	Beam energy (keV)	Probe current (pA)	Dice score (%)
80	1	27	14
80	12	102	86.3
80	12	9	76.6
100	5	6	41.8
100	5	90	58.9
100	20	114	76.1
100	10	153	75.2
100	2	10	28.8
100	1	175	66
100	12	18	76.1
100	1	88	52.4
200	3	4	47.1
200	7	116	67.8
200	3	17	66.7
200	15	8	72.3
200	15	151	74.5
200	2	106	82.9

Table 5.4 Parameters used to acquire input images for model performance testing.

5.6.3 Training sets

With all data combined, there are 6722 simulated BSE images and 462 acquired BSE images. The training set is notably disproportioned, simulations are much more abundant than real SEM data. As mentioned in Part 1 of this paper, simulations are easier to generate for many reasons, notably the workflow for generating simulated training examples is fully automated, whereas the workflow for generating acquired training examples has multiple manual steps. Consequently, the training set is imbalanced. Using different proportions of the simulated training examples with all the acquired training examples produces multiple training sets, as illustrated in Figure 5.6. The objective is to train the model separately with different training sets, to then determine which model performs best. The two main observations to be made are: first, the effect of the training set size or the number of required training examples for acceptable model performance; second, the contribution of the amount of acquired data included in the training set on model performance.

Reducing the number of simulations will increase the weight of the real data for training, summarized in Table 5.5. Training set 1 consists of 100% of the BSE simulated images only (no real data is included). Training set 2 consists of all the available data, 100% of the BSE simulated images and all the real data, the real data represents 6.4% of the training set. Training set 3 is half of the simulated images, randomly selected, including all the real data, the real data represents 12% of the training set. Training set 4 is a quarter of the simulated images, also randomly selected, including all the real data, the real data represents 21% of the training set. Training set 5

is reduced to 15% of the simulations with all the real data, the real data represents about a third of the training set. Finally, training set 6 consists of only the real data (no simulated images are included).

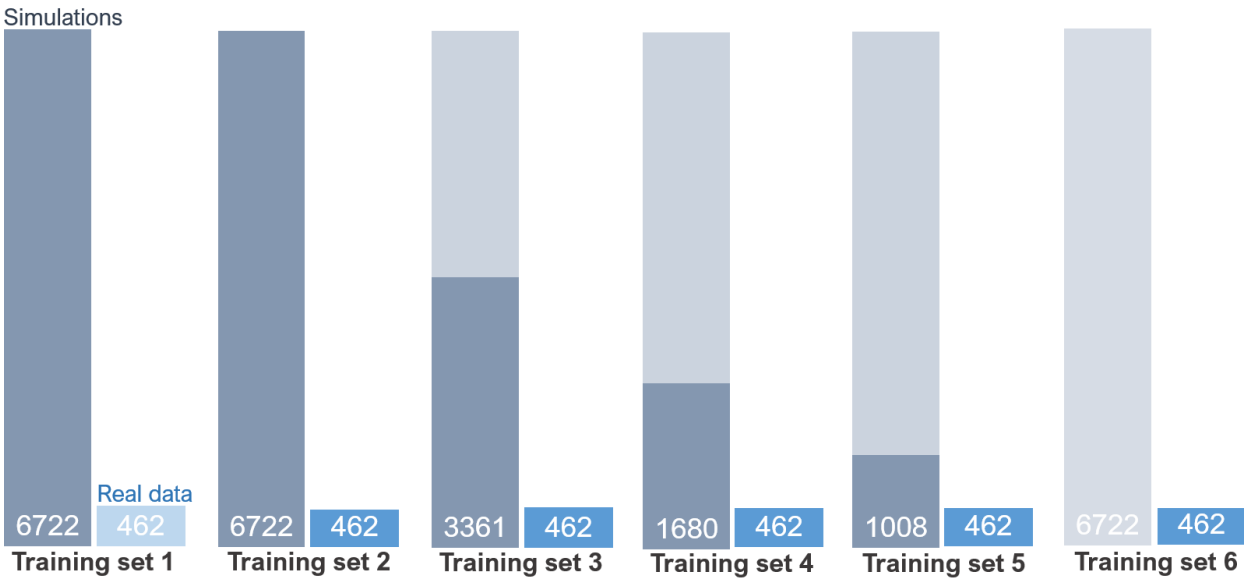


Figure 5.6 Training sets with varying proportions of simulated and acquired data.

Training set	Percentage simulations (%)	Weight of real data (%)
1	100	0
2	100	6.4
3	50	12
4	25	21
5	15	31
6	0	100

Table 5.5 Training set proportions of simulated and acquired data.

5.7 Model performance

Once the model is trained with one of the above listed training sets, it is saved, to be used to predict parameters with the 34 input images. All models receive the same input images, to then

assess if the Dice score of the resulting image is higher than the Dice score of the input image. Therefore, for a total of six models, performance is tested by evaluating their efficiency at suggesting parameters that improve image quality for segmentation, or increase the Dice score, as previously described. The mean Dice score of the 34 resulting images from suggested parameters (mean Dice final) is computed for all 6 models. The model performance is determined by evaluating the mean Dice final and the mean Dice initial from the input images.

The results are summarized in Figure 5.7, the model trained with training set 1, simulations only, performed the least well. It predicted parameters that produced images with a mean Dice score of 62%. Its success rate was low, at only 26.5%, meaning that the model only successfully increased the Dice score for a little over a quarter of the images. When the real data is included in the training set, the models perform much better. The training set 2, that includes the entirety of the available data, predicts parameters that produce images with a mean Dice score of 82.4% and it successfully increased the Dice score for 85.3% of the input images. The mean Dice score is 80.5%, with a success rate of 79.4% for the model trained with training set 3, with 50% of the simulated data and all the real data. The mean Dice score computed for the model trained with training set 4, with 25% of the simulated data and all the real data, is 82.5% with the highest success rate of 94%. The last mixed training set, training set 5, with 15% simulated data and all the real data, has a mean Dice score of 81.6% with a success rate of 88.2%. Finally, the training set 6, with real data only, predicted parameters that produced images with a mean Dice score of 74.2% and has a success rate of 73.5%.

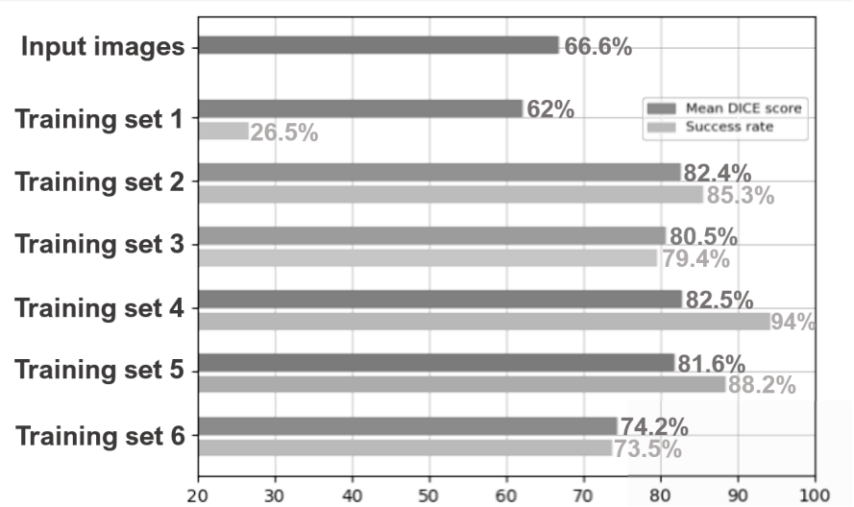


Figure 5.7 Resulting mean Dice score and success rate for models trained with all training sets.

5.8 Discussion

5.8.1 Training sets

The results show that model performance depends on the data used for training. As anticipated, models that included both simulated and acquired data (mixed training sets 2, 3, 4, 5), perform similarly and much better than models trained with only simulated data or only acquired data (training sets 1 and 6). Even with a small percentage of real data included to the training set, the model performance is notably improved, suggesting that acquired training examples provide information to the model that the simulated training examples do not.

Mixed training sets are likely to produce better results for three main reasons. First, the data is more diversified when mixed. The model can learn from examples that vary significantly, from simulated data produced with controlled parameters to acquired data influenced by microscopist contributions for stigmatism correction, focus, contrast, brightness selection and more.

Second, with the mixed training set, the model can benefit from a greater amount of training examples. The simulations only training set size is relatively large enough, however, the real data only training set size is the smallest, and its trained model could potentially produce better results with more data. Acquiring real data, aligning the beam after every parameter change, followed by necessary post processing, including manual segmentation, is time consuming, cost ineffective and tedious. The 462 training examples collected on the SEM are sufficient to demonstrate that including real images for training has a significant impact on model performance, although insufficient for best performance alone.

Third, the model trained with the simulations only training set has a questionable performance, especially since the objective is to avoid including real data for training. It is most important to note that the model is tested with real data (testing set), 34 images acquired on the SEM, as described in Figure 5.5. For successful predictions, the training set should reflect the testing set as much as possible. When creating virtual samples, some unintentional discrepancies between the real sample and the virtual samples were introduced, altering the model's prediction performance. As seen in Figure 5.8, the real sample includes CNTs, whereas the virtual samples do not, the current design, to label the background as carbon, is evidently oversimplified. In some cases, especially for lower energies, the CNTs become very apparent in the SEM acquired images as shown in Figure 5.8 (left) and are classified with the nanoparticles because Otsu thresholding is not a specific algorithm. The Otsu segmentation of an acquired image at 2 keV and 169 pA is shown in Figure 5.8 (right) in yellow, demonstrating the issue. For the model to successfully suggest parameters that differentiate CNTs from Pt NPs in the testing set, improving the performance, CNTs should be included in the virtual samples.

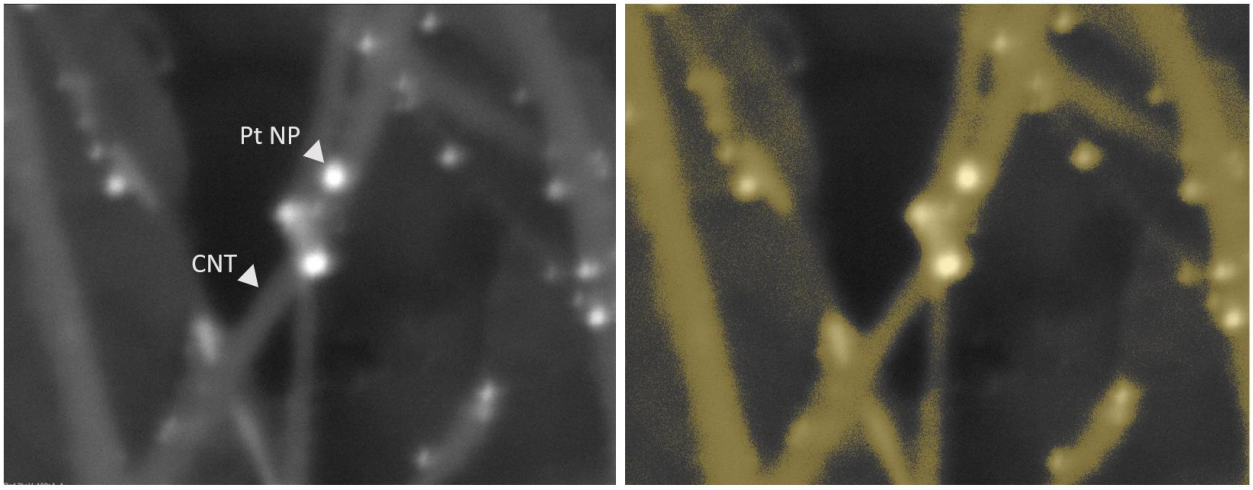


Figure 5.8 (Left) Acquired BSE image at 2 keV and 169 pA of CNTs and Pt NPs. (Right) Segmentation obtained with Otsu thresholding, in yellow, of acquired BSE image.

Also, the real sample is much thicker, not electron transparent, as compared to the virtual samples, designed relatively thin. Sample thickness has an impact on the resulting BSE images, especially when varying the beam energy. At higher energies, for thin virtual samples, electrons pass through and lower the number of detected BSEs, resulting in less signal. Consequently, the model trained with simulation-only data typically suggested lower beam energies (1-2 keV) because BSE simulated images at lower energies produced more signal. However, during testing, when very low beam energies were set on the microscope, images did not have the desired image quality for segmentation, impacting the model's prediction accuracy.

The model learns practical information during training when real data is included in the training set. Therefore, two major improvements in virtual sample design should be considered in the future for best model performance: virtual sample design that is much more representative of the real sample, include CNTs and appropriate sample thickness, and use of more specific segmentation algorithms, rather than thresholding, to differentiate nanoparticles from nanotubes.

5.8.2 Model performance testing

A few important points are to be mentioned about how the testing was conducted to evaluate model performance. First, the model was only tested with an input Dice score of 90%, selected because it is a relatively high score. However, model performance should be tested with multiple different Dice scores, higher and lower than 90% to assess prediction tendencies. Theoretically, requesting a Dice score of 100% should increase the model performance, and requesting a Dice score of 80% should lower the model performance, but this is yet to be confirmed experimentally.

Second, four out of the six training sets used to evaluate model performance are mixed, including portions of the simulated data, selected at random. Further testing should be conducted, first by splitting the simulation data into smaller steps to find the ideal ratio for best model performance. Currently, the simulated data is split at 15%, 25%, 50% and 100%. The optimal split can be found by evaluating model performance with training sets consisting of more gradual portions of simulated data combined with real data. Also, data is selected randomly, which could impact the outcome of model performance. Some portions of the simulated data could lead to improving the model's prediction accuracy, while other training examples could impair it. Model performance should be evaluated multiple times for a specific split, but with different random portions of the simulated data, to ensure that performance is not correlated to the data included in the training set.

Lastly, model performance was evaluated using the mean Dice score and the success rate, or how often the model successfully suggested parameters that increased the final Dice score from the

initial Dice score. However, the success rate only confirms that the model increased the Dice score, but it does not specify by how much. For instance, trained models that suggest parameters that increase the Dice score by 20% should be selected over trained models that only increase the Dice score by 1%. Therefore, success rate efficiency should also be considered, as it is valuable information to include when evaluating the model's prediction accuracy.

5.9 Conclusion

A carefully tailored mixed input, multiple output model was trained to successfully suggest the beam energy and probe current on the SEM for improved image quality for segmentation. Results depend on training sets used to train the model and further testing could potentially help improve the model. Eventually, when a model is sufficiently trained, users can choose to integrate pretrained models to acquisition workflow or train their own model with their own dataset to set microscope parameters.

5.10 References

[1] “The Functional API | TensorFlow Core,” TensorFlow. Accessed: Sep. 15, 2023. [Online]. Available: https://www.tensorflow.org/guide/keras/functional_api

[2] “pandas documentation — pandas 2.2.0 documentation.” Accessed: Feb. 08, 2024. [Online]. Available: https://pandas.pydata.org/docs/index.html

[3] K. P. Murphy, Machine learning: a probabilistic perspective. in Adaptive computation and machine learning. Cambridge, Mass.: MIT Press, 2012.

[4] K. Team, “Keras documentation: Concatenate layer.” Accessed: Sep. 15, 2023. [Online]. Available: https://keras.io/api/layers/merging_layers/concatenate/

[5] T. Yu and H. Zhu, “Hyper-Parameter Optimization: A Review of Algorithms and Applications.” arXiv, Mar. 12, 2020. doi: 10.48550/arXiv.2003.05689.

[6] S. Ghosh and S. Ghosh, “Exploring the Ideal Depth of Neural Network when Predicting Question Deletion on Community Question Answering.” arXiv, Dec. 07, 2019. doi: 10.48550/arXiv.1912.03585.

[7] J. Brownlee, “Crash Course in Convolutional Neural Networks for Machine Learning,” MachineLearningMastery.com. Accessed: Feb. 16, 2024. [Online]. Available: https://machinelearningmastery.com/crash-course-convolutional-neural-networks/

[8] K. Pykes, “Fighting Overfitting With L1 or L2 Regularization: Which One Is Better?,” neptune.ai. Accessed: Feb. 16, 2024. [Online]. Available: https://neptune.ai/blog/fighting-overfitting-with-l1-or-l2-regularization

[9] “Max Pooling,” DeepAI. Accessed: Feb. 16, 2024. [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/max-pooling

[10] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. in Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016. Accessed: Apr. 14, 2023. [Online]. Available: http://www.deeplearningbook.org/

[11] “Mean Squared Error : Overview, Examples, Concepts and More | Simplilearn,” Simplilearn.com. Accessed: Feb. 16, 2024. [Online]. Available: https://www.simplilearn.com/tutorials/statistics-tutorial/mean-squared-error

[12] G. Casella and R. L. Berger, Statistical inference, 2nd ed. in Duxbury advanced series in statistics and decision sciences. Australia: Thomson Learning, 2002. Accessed: Feb. 16, 2024. [Online]. Available: http://catdir.loc.gov/catdir/enhancements/fy1302/2001025794-t.html

[13] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization.” arXiv, Jan. 29, 2017. doi: 10.48550/arXiv.1412.6980.

[14] K. Team, “Keras documentation: ReduceLROnPlateau.” Accessed: Sep. 15, 2023. [Online]. Available: https://keras.io/api/callbacks/reduce_lr_on_plateau/

[15] K. Team, “Keras documentation: EarlyStopping.” Accessed: Sep. 15, 2023. [Online]. Available: https://keras.io/api/callbacks/early_stopping/

[16] J. Brownlee, “How to use Learning Curves to Diagnose Machine Learning Model Performance,” MachineLearningMastery.com. Accessed: Sep. 15, 2023. [Online]. Available: https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/

[17] K. S. V. Muralidhar, “Learning Curve to identify Overfitting and Underfitting in Machine Learning,” Medium. Accessed: Feb. 17, 2024. [Online]. Available: https://towardsdatascience.com/learning-curve-to-identify-overfitting-underfitting-problems-133177f38df5

6 Concluding remarks

6.1 Conclusions

This thesis demonstrates in Chapters 3, 4 and 5 how computational microscopy, specifically with SEMs, contributes to higher throughput and optimized workflows. In this study, software is designed to control the SEM, a complex instrument, through external communication. The practical implications of automation are clear, where optimizing workflows creates new opportunities to assist microscopists during acquisitions. Microscopist should spend their time on complex, innovative analysis rather than on of repetitive, tedious tasks. The proposed workflows will minimize acquisition time, and consequently costs and beam exposure.

Automation with software enables new approaches to microscopy. Automated acquisitions can run overnight and unattended when microscopes would normally be idle. Increasing the microscope's output will increase the amount of available data for analysis and more importantly, guided acquisitions generate superior data. Indeed, by integrating segmentation algorithms to find features of interest and guide acquisitions, more comprehensive data is obtained, faster. The two main objectives of automation are to (1) track features throughout acquisitions and (2) produce images with proper image quality for quantitative analysis, specifically for segmentation. For this purpose, the proposed workflows in this thesis use artificial intelligence, multiple detector signals and feedback loops between the instrument and software, to dictate microscope positioning and tune microscope parameters.

Image processing software offer a strong basis for development of integrated solutions with implemented frameworks, programmers consistently maintaining code and implementing new

computationally efficient algorithms. Microscopists are not software developers; they do not have the capacity and resources to build powerful image processing software on their own time. Building methods for acquiring data in collaboration with existing image processing software is key to the efficient and robust progress of optimization in microscopy. For this reason, workflows presented in this thesis were developed as scripts in the Dragonfly environment, a proficient image processing software. The approach is to import images to Dragonfly, rather than creating an entirely new platform from scratch, where multiple image processing algorithms are available and implemented. Many tools were found useful during the development of presented workflows, including the virtual microscope, segmentation algorithms, stitching and registration algorithms and data storage. Built in frameworks eliminate the need for recreating existing code, leaving room for innovation. Image processing software is an important resource for the development of new computational methods in microscopy. Once merged with acquisitions, workflows have the potential to fill important gaps in existing systems.

As for contributions related to optimizing microscope parameter selection, multiple findings provide a new perspective in quantitative microscopy. Images are not necessarily acquired for esthetical purposes but rather, for obtaining quantitative results. More accurate data is acquired by suggesting parameters that generate the best image quality for segmentation. Parameters should be tuned according to algorithms, thereby eliminating any inherent microscopist bias. When parameter selection relies on microscopists, workflows are not reproducible, but also depend on individual expertise to correctly set up the instrument.

Segmentation is the known bottleneck step during quantitative analysis that can be overcome by optimizing microscope parameters during acquisitions. Effectively, enhanced acquisitions will produce images that are successful for obtaining quantitative results including the number, shape,

3D distribution, connectivity, and integrity of structures. Quantitative microscopy also becomes more accessible to scientists, without the need for prior knowledge on electron-material interactions for obtaining desired image quality. Acquired images are guaranteed to be easily segmented, presenting evident practical implications.

6.2 Contributions to original knowledge

- Automation of repetitive tasks through microscope control and integrating microscope tasks with image processing tasks are not original ideas. However, the methods used for automation are unique to this study. By coding in the Dragonfly software environment, all image processing algorithms are available to implement seamlessly for quicker development of innovative acquisition routines.
- This is the first study to automate a complete workflow on the SEM, with added intelligence, from acquisitions to image processing to quantitative analysis. Automation has important practical implications in microscopy. Strategies used to accomplish automation include guiding the beam or stage to specific positions on the sample, collecting information from multiple signals, importing images to Dragonfly, applying stitching and segmentation algorithms and computing measurements.
- Feature tracking by converting a segmented reference image to a graph is an innovative technique. The beam is relocated at imaging sites with the graph vertex positions, using an established referential between the imaging processing software and the microscope stage. This technique is termed smart beam positioning for guided acquisition, described in Chapter 3.
- The deep learning model designed for suggesting microscope parameters to set during acquisitions is an original contribution. This is the first study to train a model to suggest a beam energy and probe current for improved image quality. A properly trained model can replace parameter tuning routines.

- This is the first study to create scripts for automatically generating high volumes of training examples with MC X-ray. Simulations are essential to obtaining the required quantity of data for training.
- This is the first study to train a model that suggests parameters during acquisitions to improve image quality for segmentation. Quantitative analysis of images relies on segmentation and segmentation results depend on image quality. Optimizing microscope parameters for segmentation has important practical implications in microscopy.

6.3 Future work

This study contributes to the work performed towards computational microscopy. Specifically, towards combining microscopy techniques with image processing algorithms. The work completed on the SEM for workflow automation and parameter selection requires a user-friendly interface for adaptability purposes. Scripts developed in Dragonfly should be translated to a plugin for microscope control, available in the software. In the form of a plugin, computational microscopy is much more accessible. Users do not need to know how to read and modify code snippets. Users should potentially customize and build their workflows with selected algorithms and monitor acquisition progression. For this, future work on developing a user interface is required, creating generalized solutions that users can customize individually. The code can be extended to other microscopes as it was not designed for a specific microscope system.

Furthermore, the work would be extended to the FIB-SEM, a dual beam system, controlling both the FIB and the electron beam. Having already accomplished a considerable amount with one of the two beams, the SEM, functionalities can be added to tackle the FIB. Although the level of complexity is increased with the addition of the FIB, workflows can certainly be automated with integrated image processing algorithms. Tracking features and optimizing parameters is especially significant when performing 3D imaging, as the FIB slices through layers and the SEM images newly exposed surfaces.

Only two microscope parameters were selected for optimization in Chapters 4 and 5, the beam energy and the probe current. As a future direction, more parameters will be incorporated into predictions, namely the acquisition time. Minimizing the acquisition time, all the while

conserving the appropriate image quality for segmentation can be beneficial for multiple reasons. It generates quicker results for analysis and reduces the exposure time to the sample. Although images may be noisier and less esthetically pleasing with lower acquisition times, image quality will be proper for segmentation and therefore for quantitative analysis.

Automation of alignment routines is an additional prospective goal to simplify microscope tasks and assist microscopist, specifically, by sending commands to adjust the focus and stigmatism settings with artificial intelligence and by measuring the sharpness of images. Automated beam alignment combined with automated parameter selection presents the ideal solution to creating an acquisition set up efficiently and quickly on the SEM.

7 Bibliography

[1] E. Moen, D. Bannon, T. Kudo, W. Graf, M. Covert, and D. Van Valen, “Deep learning for cellular image analysis,” *Nat Methods*, vol. 16, no. 12, Art. no. 12, Dec. 2019, doi: 10.1038/s41592-019-0403-1.

[2] J. I. Goldstein *et al.*, “Electron Beam–Specimen Interactions,” in *Scanning Electron Microscopy and X-ray Microanalysis: Third Edition*, J. I. Goldstein, D. E. Newbury, P. Echlin, D. C. Joy, C. E. Lyman, E. Lifshin, L. Sawyer, and J. R. Michael, Eds., Boston, MA: Springer US, 2003, pp. 61–98. doi: 10.1007/978-1-4615-0215-9_3.

[3] M. I. Szynkowska, “MICROSCOPY TECHNIQUES | Scanning Electron Microscopy,” in *Encyclopedia of Analytical Science (Second Edition)*, P. Worsfold, A. Townshend, and C. Poole, Eds., Oxford: Elsevier, 2005, pp. 134–143. doi: 10.1016/B0-12-369397-7/00385-X.

[4] “Contact - Institut de minéralogie, de physique des matériaux et de cosmochimie.” Accessed: Feb. 21, 2024. [Online]. Available: https://impmc.sorbonne-universite.fr/fr/plateformes-et-equipements/plate-forme-de-microscopie-electronique-meb-fib/contact.html

[5] “Scanning Electron Microscopy | Nanoscience Instruments.” Accessed: Feb. 18, 2024. [Online]. Available: https://www.nanoscience.com/techniques/scanning-electron-microscopy/

[6] A. Nanakoudis, “SEM: Types of Electrons and the Information They Provide,” Advancing Materials. Accessed: Jun. 20, 2024. [Online]. Available: https://www.thermofisher.com/blog/materials/sem-signal-types-electrons-and-the-information-they-provide/

[7] “SEM working principle: the detection of backscattered electrons - CA.” Accessed: Mar. 22, 2024. [Online]. Available: https://www.thermofisher.com/ca/en/home/global/forms/industrial/backscattered-electrons-sem.html

[8] “Different Types of SEM Imaging – BSE and Secondary Electron Imaging,” AZoM. Accessed: Mar. 22, 2024. [Online]. Available: https://www.azom.com/article.aspx?ArticleID=14309

[9] “Focused Ion and Electron Beam System Ethos NX5000 Series.” Accessed: Feb. 22, 2022. [Online]. Available: https://www.hitachi-hightech.com/global/science/products/microscopes/focused-ion-beam-systems/nx5000.html

[10] M. Dunlap and Dr. J. E. Adaskaveg, “Introduction to the Scanning Electron Microscope Theory, Practice, & Procedures.” FACILITY FOR ADVANCED INSTRUMENTATION, U. C. Davis, 1997.

[11] “Energy-dispersive detector (EDS),” Geochemical Instrumentation and Analysis. Accessed: Feb. 18, 2024. [Online]. Available: https://serc.carleton.edu/research_education/geochemsheets/eds.html

[12] A. Winkelmann, T. B. Britton, and G. Nolze, “Constraints on the effective electron energy spectrum in backscatter Kikuchi diffraction,” *Phys. Rev. B*, vol. 99, no. 6, p. 064115, Feb. 2019, doi: 10.1103/PhysRevB.99.064115.

[13] D. C. Joy, *Monte Carlo Modeling for Electron Microscopy and Microanalysis*. Oxford University Press, 1995.
[14] P. Hovington, D. Drouin, and R. Gauvin, “CASINO: A new monte carlo code in C language for electron beam interaction —part I: Description of the program,” *Scanning*, vol. 19, no. 1, pp. 1–14, 1997, doi: 10.1002/sca.4950190101.
[15] E. Lifshin, B. Thiel, and R. Gauvin, “The Validation of Monte Carlo Methods for Scanning Electron Microscopy and Electron Microprobe Analysis,” *Microscopy and Microanalysis*, vol. 11, no. S02, pp. 1346–1347, Aug. 2005, doi: 10.1017/S1431927605506792.
[16] K. F. J. Heinrich, D. E. Newbury, and H. Yakowitz, *Use of Monte Carlo Calculations in Electron Probe Microanalysis and Scanning Electron Microscopy: Proceedings of a Workshop Held at the National Bureau of Standards, Gaithersburg, Maryland, October 1-3, 1975*. U.S. Department of Commerce, National Bureau of Standards, 1976.
[17] D. Drouin, P. Hovington, and R. Gauvin, “CASINO: A new monte carlo code in C language for electron beam interactions—part II: Tabulated values of the mott cross section,” *Scanning*, vol. 19, no. 1, pp. 20–28, 1997, doi: 10.1002/sca.4950190103.
[18] R. Gauvin, E. Lifshin, H. Demers, P. Horny, and H. Campbell, “Win X-ray: A New Monte Carlo Program that Computes X-ray Spectra Obtained with a Scanning Electron Microscope,” *Microscopy and Microanalysis*, vol. 12, no. 1, pp. 49–64, Feb. 2006, doi: 10.1017/S1431927606060089.
[19] “Dragonfly | 3D Visualization and Analysis Solutions for Scientific and Industrial Data | ORS.” Accessed: Apr. 10, 2022. [Online]. Available: https://www.theobjects.com/dragonfly/index.html
[20] O. Meckes and N. Ottawa, “Part 1: Optimizing the Image Output: Tuning the SEM Parameters for the Best Photographic Results,” in *Biological Field Emission Scanning Electron Microscopy*, John Wiley & Sons, Ltd, 2019, pp. 625–636. doi: 10.1002/9781118663233.ch30_1.
[21] “Comment pouvez-vous optimiser les paramètres d’imagerie MEB pour différents échantillons ?” Accessed: Feb. 19, 2024. [Online]. Available: https://fr.linkedin.com/advice/1/how-can-you-optimize-sem-imaging-parameters-mdrnc?lang=fr
[22] H. Shroff, I. Testa, F. Jug, and S. Manley, “Live-cell imaging powered by computation,” *Nat Rev Mol Cell Biol*, pp. 1–21, Feb. 2024, doi: 10.1038/s41580-024-00702-6.
[23] J. I. Goldstein, D. E. Newbury, J. R. Michael, N. W. M. Ritchie, J. H. J. Scott, and D. C. Joy, “Scanning Electron Microscope (SEM) Instrumentation,” in *Scanning Electron Microscopy and X-Ray Microanalysis*, J. I. Goldstein, D. E. Newbury, J. R. Michael, N. W. M. Ritchie, J. H. J. Scott, and D. C. Joy, Eds., New York, NY: Springer, 2018, pp. 65–91. doi: 10.1007/978-1-4939-6676-9_5.
[24] L. Tian and C. A. Volkert, “Measuring Structural Heterogeneities in Metallic Glasses Using Transmission Electron Microscopy,” *Metals*, vol. 8, no. 12, Art. no. 12, Dec. 2018, doi: 10.3390/met8121085.
[25] “How to Combat Electric Charge Buildup in Scanning Electron Microscopy | Nanoscience Instruments.” Accessed: Feb. 19, 2024. [Online]. Available:

https://www.nanoscience.com/blogs/how-to-combat-electric-charge-buildup-in-scanning-electron-microscopy/
[26] R. Chhetri, S. Preibisch, and N. Stuurman, “Software for Microscopy Workshop White Paper.” arXiv, Apr. 30, 2020. doi: 10.48550/arXiv.2005.00082.
[27] Z. R. Fox *et al.*, “Enabling reactive microscopy with MicroMator,” *Nat Commun*, vol. 13, no. 1, Art. no. 1, Apr. 2022, doi: 10.1038/s41467-022-29888-z.
[28] T. Zachs *et al.*, “Fully automated, sequential focused ion beam milling for cryo-electron tomography,” *Elife*, vol. 9, p. e52286, Mar. 2020, doi: 10.7554/eLife.52286.
[29] S. Klumpe *et al.*, “A Modular Platform for Streamlining Automated Cryo-FIB Workflows.” bioRxiv, p. 2021.05.19.444745, May 20, 2021. doi: 10.1101/2021.05.19.444745.
[30] J. Na, G. Kim, S.-H. Kang, S.-J. Kim, and S. Lee, “Deep learning-based discriminative refocusing of scanning electron microscopy images for materials science,” *Acta Materialia*, vol. 214, p. 116987, Aug. 2021, doi: 10.1016/j.actamat.2021.116987.
[31] J. M. Serra Lleti, “Automated Correlative Light and Electron Microscopy using FIB-SEM as a tool to screen for ultrastructural phenotypes.” Accessed: Feb. 21, 2022. [Online]. Available: https://archiv.ub.uni-heidelberg.de/volltextserver/26154/
[32] X. Casas Moreno, M. M. Silva, J. Roos, F. Pennacchietti, N. Norlin, and I. Testa, “An open-source microscopy framework for simultaneous control of image acquisition, reconstruction, and analysis,” *HardwareX*, vol. 13, p. e00400, Mar. 2023, doi: 10.1016/j.ohx.2023.e00400.
[33] H. Pinkard *et al.*, “Pycro-Manager: open-source software for customized and reproducible microscope control,” *Nat Methods*, vol. 18, no. 3, Art. no. 3, Mar. 2021, doi: 10.1038/s41592-021-01087-6.
[34] A. Edelstein, N. Amodaj, K. Hoover, R. Vale, and N. Stuurman, “Computer control of microscopes using μManager,” *Curr Protoc Mol Biol*, vol. Chapter 14, p. Unit14.20, Oct. 2010, doi: 10.1002/0471142727.mb1420s92.
[35] S. Tosi *et al.*, “AutoScanJ: A Suite of ImageJ Scripts for Intelligent Microscopy,” *Frontiers in Bioinformatics*, vol. 1, 2021, Accessed: Feb. 25, 2024. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fbinf.2021.627626
[36] “The SerialEM Home Page.” Accessed: Jan. 02, 2024. [Online]. Available: https://bio3d.colorado.edu/SerialEM/
[37] N. O. Mahony *et al.*, *Deep Learning vs. Traditional Computer Vision*, vol. 943. 2020. doi: 10.1007/978-3-030-17795-9.
[38] Y. Lai, “A Comparison of Traditional Machine Learning and Deep Learning in Image Recognition,” *J. Phys.: Conf. Ser.*, vol. 1314, no. 1, p. 012148, Oct. 2019, doi: 10.1088/1742-6596/1314/1/012148.
[39] R. Makovetsky, N. Piche, and M. Marsh, “Dragonfly as a Platform for Easy Image-based Deep Learning Applications,” *Microscopy and Microanalysis*, vol. 24, no. S1, pp. 532–533, Aug. 2018, doi: 10.1017/S143192761800315X.
[40] *How to use a pretrained deep learning model for additive manufacturing porosity*, (Oct. 26, 2022). Accessed: Jan. 25, 2024. [Online Video]. Available: https://www.youtube.com/watch?v=enkLJhnlG5k

[41] A. Durand *et al.*, “A machine learning approach for online automated optimization of super-resolution optical microscopy,” *Nat Commun*, vol. 9, no. 1, Art. no. 1, Dec. 2018, doi: 10.1038/s41467-018-07668-y.
[42] K. P. Murphy, *Machine learning: a probabilistic perspective*. in Adaptive computation and machine learning. Cambridge, Mass.: MIT Press, 2012.
[43] A. Rosebrock, “Keras: Multiple Inputs and Mixed Data,” PyImageSearch. Accessed: Jan. 29, 2024. [Online]. Available: https://pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data/
[44] C. Bishop, “Pattern Recognition and Machine Learning,” Jan. 2006, Accessed: Jan. 26, 2024. [Online]. Available: https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/
[45] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. in Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016. Accessed: Apr. 14, 2023. [Online]. Available: http://www.deeplearningbook.org/
[46] R. Rabbany, “CS551 McGill.” Accessed: Apr. 14, 2023. [Online]. Available: http://www.reirab.com/Teaching/AML23/index.html
[47] “Building a Regression Multi-Layer Perceptron (MLP).” Accessed: Apr. 14, 2023. [Online]. Available: https://kaggle.com/code/mejbahahammad/building-a-regression-multi-layer-perceptron-mlp
[48] “MNIST handwritten digit database | BibSonomy.” Accessed: Jan. 29, 2024. [Online]. Available: https://www.bibsonomy.org/bibtex/2935bad99fa1f65e03c25b315aa3c1032/mhwombat
[49] A. Baldominos, Y. Saez, and P. Isasi, “A Survey of Handwritten Character Recognition with MNIST and EMNIST,” *Applied Sciences*, vol. 9, no. 15, Art. no. 15, Jan. 2019, doi: 10.3390/app9153169.
[50] N. Manral, “nipunmanral/MLP-Training-For-MNIST-Classification.” Jan. 30, 2024. Accessed: Mar. 05, 2024. [Online]. Available: https://github.com/nipunmanral/MLP-Training-For-MNIST-Classification
[51] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, “Measuring Catastrophic Forgetting in Neural Networks.” arXiv, Nov. 09, 2017. doi: 10.48550/arXiv.1708.02072.
[52] R. JAIN, “Convolutional neural networks(Part-1),” Medium. Accessed: Feb. 01, 2024. [Online]. Available: https://medium.com/@rajatjain0807/machine-learning-6ecde3bfd2f4
[53] L. Gao, P.-Y. Chen, and S. Yu, “Demonstration of Convolution Kernel Operation on Resistive Cross-Point Array,” *IEEE Electron Device Lett.*, vol. 37, no. 7, pp. 870–873, Jul. 2016, doi: 10.1109/LED.2016.2573140.
[54] J. G. Greener, S. M. Kandathil, L. Moffat, and D. T. Jones, “A guide to machine learning for biologists,” *Nat Rev Mol Cell Biol*, vol. 23, no. 1, Art. no. 1, Jan. 2022, doi: 10.1038/s41580-021-00407-0.
[55] “Papers with Code - U-Net Explained.” Accessed: Feb. 25, 2024. [Online]. Available: https://paperswithcode.com/method/u-net
[56] “ResNet: The Basics and 3 ResNet Extensions,” Datagen. Accessed: Feb. 25, 2024. [Online]. Available: https://datagen.tech/guides/computer-vision/resnet/

[57] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition.” arXiv, Apr. 10, 2015. doi: 10.48550/arXiv.1409.1556.
[58] “OpenAI Platform.” Accessed: Feb. 25, 2024. [Online]. Available: https://platform.openai.com
[59] A. Krull, T.-O. Buchholz, and F. Jug, “Noise2Void - Learning Denoising from Single Noisy Images.” arXiv, Apr. 05, 2019. doi: 10.48550/arXiv.1811.10980.
[60] “PyTorch,” PyTorch. Accessed: Feb. 25, 2024. [Online]. Available: https://pytorch.org/
[61] “Keras | TensorFlow Core,” TensorFlow. Accessed: Feb. 25, 2024. [Online]. Available: https://www.tensorflow.org/guide/keras?hl=fr
[62] “scikit-learn: machine learning in Python — scikit-learn 1.4.1 documentation.” Accessed: Feb. 25, 2024. [Online]. Available: https://scikit-learn.org/stable/
[63] J. Brownlee, “Your First Deep Learning Project in Python with Keras Step-by-Step,” MachineLearningMastery.com. Accessed: Feb. 01, 2024. [Online]. Available: https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
[64] K. S. V. Muralidhar, “Learning Curve to identify Overfitting and Underfitting in Machine Learning,” Medium. Accessed: Feb. 17, 2024. [Online]. Available: https://towardsdatascience.com/learning-curve-to-identify-overfitting-underfitting-problems-133177f38df5
[65] A. F. Agarap, “Deep Learning using Rectified Linear Units (ReLU).” arXiv, Feb. 07, 2019. doi: 10.48550/arXiv.1803.08375.
[66] T. Chen, Z. Zhang, S. Liu, S. Chang, and Z. Wang, “Robust Overfitting may be mitigated by properly learned smoothening,” presented at the International Conference on Learning Representations, May 2021. Accessed: Feb. 25, 2024. [Online]. Available: https://research.ibm.com/publications/robust-overfitting-may-be-mitigated-by-properly-learned-smoothening
[67] K. Pykes, “Fighting Overfitting With L1 or L2 Regularization: Which One Is Better?,” neptune.ai. Accessed: Feb. 16, 2024. [Online]. Available: https://neptune.ai/blog/fighting-overfitting-with-l1-or-l2-regularization
[68] H. Wu and X. Gu, “Max-Pooling Dropout for Regularization of Convolutional Neural Networks,” arXiv.org. Accessed: Feb. 25, 2024. [Online]. Available: https://arxiv.org/abs/1512.01400v1
[69] K. Team, “Keras documentation: Callbacks API.” Accessed: Feb. 25, 2024. [Online]. Available: https://keras.io/api/callbacks/