Automated Analysis of Heterogeneous Catalyst Materials using Deep Learning



Kevin Patrick Treder

Kellogg College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2022

For a long time I have wanted to give something back to the world. This is my humble contribution to the knowledge of humanity and the quest to understand our place in the universe.

Dedicated to

Elisabeth and Andreas Treder

and

Alicja Duda and Krystyna Danuta Treder

who made everything possible.

Declaration of Authorship

I, Kevin Patrick Treder, declare that this thesis entitled ‘Automated Analysis of Heterogeneous Catalyst Materials using Deep Learning’ and the work presented in it are my own.

I confirm that:

- I have not used any sources other than those listed in the references or explicitly mentioned in the text.
- It is clearly stated where presented results are based on work done in collaboration.
- It is clearly stated where any of the presented work or results were submitted as an independent article to a scientific journal.
- This thesis has not been submitted for any other degree or qualitfication at any other university or institution.
- I have acknowledged all main sources of help.

Kevin Patrick Treder, 07.10.2022

Abstract

Heterogeneous catalyst materials play a key role in modern society, as many processes in the chemical and energy industry rely on them. Optimising their performance is directly connected to a large potential of reductions in energy consumption, and thus to a more sustainable future. A fundamental part in the optimisation process is represented by materials characterisation. This is often done using *in situ* (Scanning) Transmission Electron Microscopy ((S)TEM) imaging in order to obtain a full understanding of the catalyst performance in different environments and temperatures at high resolution. However, the analysis of corresponding dynamical datasets is often time-consuming and requires manual intervention alongside tailored post-processing routines. At the same time, the emergence of direct electron detectors allowing for the acquisition of datasets at kiloHertz frame rates, as well as novel imaging techniques raised data generation rates significantly and created a need for new, reliable and automated data processing techniques.

This work introduces *nNPipe* as Deep Learning based method for the automated analysis of morphologically diverse heterogeneous catalyst systems. The method is based on two Covolutional Neural Networks (CNNs) that were exclusively trained on computationally generated HRTEM image simulations and allow for rapid and precise analysis of raw 2048 x 2048 experimental HRTEM images. The performance of *nNPipe* is demonstrated in a realistic automated imaging scenario where statistically significant material properties are inferred accurately. Moreover, time-efficient and reproducible retraining methods based small experimental datasets are described for both, further performance improvements and adaption to new imaging scenarios. In this context, a potentially new pathway for the generation of suitable training datasets obtained by thousands of mostly non-expert annotations is highlighted. Finally, the analytical capacities of the *nNPipe* method are showcased on time-resolved datasets in two advanced applications scenarios: i) Live image analysis during sample acquisition and ii) Analysis of the particle coalescence of an *in situ* heated Pd/C catalyst.

Acknowledgements

I remember quite well what I have written in my application to the University of Oxford when I was asked something along ‘Why do you pursure to enrol in this programme at this particular university?’ My answer was that I believed that I would have a unique opportunity to gain knowledge and skills in a research field I truly wish to expertise in, but also that the University of Oxford creates an extraordinary environment to widen horizons, shape personal character, and to be sustainably inspired. **I now recognise that it was far more than that.**

During my journey for this degree, I was incredibly lucky to have four outstanding supervisors who supported me at every single stage and who I am deeply grateful for.

I would like to thank Dr. Judy Kim for an exceptional supervision and who made me feel welcome in the research group and in Oxford from day one. Not only was she a fantastic teacher in theoretical and experimental electron microscopy, but was also always available when I had questions or concerns, ensured I had equipment and environment I could thrive in, and especially had plenty wise suggestions and ideas without which this work would not have been possible.

I would like to thank Prof. Dr. Angus Kirkland for excellent supervision and collaboration on the many aspects presented in this thesis. In particular, I would like to thank Prof. Dr. Angus Kirkland for numerous fruitful discussions I genuinely enjoyed, for his sharp sight in complex issues, for suggesting the most efficient, whilst effective ways and ideas to tackle various challenges, for fastest proof-reading I have ever witnessed, and naturally for providing funding, equipment and an environment for this work.

On the ‘industrial’ side of my supervisory board I would like to thank Dr. Manfred Schuster for his marvellous supervision, especially in *in situ* electron microscopy. A foundation pillar of this work, built on countless hours spent achieving the right experimental *in situ* conditions, would not have been possible without his relentless contribution. Furthermore, I would like to thank him for many great teachings in *in situ* electron microscopy, the discussions on obtained results and more general aspects of life, and for taking time to deal with challenges and to think through next steps.

Lastly, I would like to thank Dr. Doğan Özkaya being an inspiring and fantastic supervisor, who for one part brought in many constructive ideas that were key to this thesis, and for another always highlighted the interconnections of this work with materials science and industry in general, which enabled a wider understanding of its impact. Furthermore, I would like to thank Dr. Doğan Özkaya for his supervision, organisation, and collaboration during a highly educational and insightful internship at Johnson Matthey plc that resulted in a new experimental approach.

In this context, I would also like to thank the ESPRC and Johnson Matthey plc for their financial support of this work.

Apart from my supervisors, I would like to mention two people who I had the honour to work with:

I am grateful for an outstanding collaboration with Dr. Chen Huang during the time spent together in the research group and for being a great mentor of coding and Deep Learning. During discussions, Dr. Chen Huang always provided additional and critical points of view which led to substantial improvements of approaches and results, encouraged me to strive for perfection, and generally dedicacted a lot of his time to work on joint projects. Without this collaboration, the degree of success this work presents would not have been possible.

Furthermore, I would like to thank Dr. Aakash Varambhia for the tremendous work accomplished together during my internship at Johnson Matthey plc. Thanks to Dr. Aakash Varambhia, I had the chance to learn about the technical background of web-based networks and related data processing, the importance to make software research tools truly applicable, and on how to use various coding resources effectively. Most importantly, a new experimental approach was the outcome of a fruitful collaboration.

This work would not have been possible without further collaborations and support:

I would like to thank Cameron Bell and Dr. Thomas Slater for a highly fruitful collaboration and exchange on Machine Learning and Deep Learning approaches for the analysis of (S)TEM images of nanoparticles which resulted in two independent manuscripts submitted for publication.

On the operational side, I would like to thank the team at ePSIC, Diamond Light Source, namely Dr. Thomas Slater, Dr. Mohsen Danaie, Dr. Christopher Allen and Matt Spink who supported my work with dedicated guidance and support during experimental sessions.

Moreover, I would like to express my gratitude for a highly disruptive and great collaboration with Dr. Michele Darrow, Dr. Olly King, Dr. Avery Pennington and Dr. Mark Basham, with whom I had the chance to start the joint Zooniverse project 'Science Scribbler: Key2Cat'.

Special thanks go furthermore to all (former) members of the Angus Kirkland research group, especially to Morgan Jackson, Dr. Emanuela Liberti, Dr. Ali Mostaed, Dr. Dimitrios Bellos, Dr. Ofentse Makgae and Arthur Moya who I really enjoyed working with.

Finally, I would like to thank the many people that supported me outside of work:

I would like to express my deep gratitude, admiration and love to my partner Ariana Dima, who supported me thorugh every moment during the last four years despite the long distance between the United Kingdom and Germany. It was her who always found the right words when research became particularly challenging, when I was in doubt, and who highlighted countless bright moments in life when outside of work.

My most profound thanks are dedicated to my parents, Elisabeth and Andreas Treder, who supported me throughout my whole academic career and life, who assured I would always have what I needed, and were there for me at any time. Amongst the most important things, it is their values that I have the privilege to live by and which I believe are a fundamental reason of why I could pursue this degree.

Furthermore, I would like to thank Eva Papanikita and Ioannis Dimas for their countinuing support and for being an ongoing source of inspiration and motivation during the last four years.

My dearest and special thanks go to the wonderful people I had the chance to live with in 13 Alexandra Road: Alessandro Lodi, Michał Wójcik, Mrinalini Penumaka, Vinita Govindarajan and Cecilia Ferrari. Their friendship, support, and countless memories in- and outside the pandemic have a special place in my heart and made my time in Oxford exceptional.

Table of Contents

1 Introduction 1
2 Literature Review 5
2.1 Time-resolved Imaging 5
2.1.1 A brief historical Overview of *in situ* (S)TEM 8
2.1.2 Nanomechanical Studies 10
2.1.3 Microelectromechanical Systems 11
2.1.4 Environmental TEM (ETEM) 18
2.1.5 Stroboscopic Microscopy (UTEM) 19
2.1.6 Dynamic TEM 21
2.1.7 Direct Electron Detectors 25
2.1.8 Challenges of time-resolved Imaging 31
2.2 Applications of Deep Learning in Electron Microscopy 33
2.2.1 Principles 33
2.2.2 Applications in Physical Sciences 37
2.2.3 Applications in Life Sciences 48
2.3 Materials for Heterogeneous Catalysis 51
2.3.1 Properties of Nanoparticle Systems 51
2.3.2 Gold (Au) Nanoparticles 52
2.3.3 Palladium (Pd) Nanoparticles 53
2.3.4 Deactivation Mechanisms 55
2.4 Chapter Summary 57
3 Theoretical Considerations 59
3.1 Electron Emission 59
3.1.1 Continuous Electron Emission 59
3.1.2 Pulsed Electron Emission 63
3.2 HRTEM Imaging 68
3.2.1 Image Formation 68
3.2.2 Wave Aberration Function 71
3.3 HRTEM Image Simulation 73
3.3.1 Approaches for Image Simulation 73
3.3.2 The Multislice Method 74
3.4 Deep Learning 77
3.4.1 Single Layer Perceptron 77
3.4.2 Multilayer Neural Network 78
3.4.3 Optimisation 79

3.4.4 Gradient Descent Strategies 83
3.4.5 Convolutional Neural Networks 86
3.5 Chapter Summary 89
4 Methods 91
4.1 HRTEM Imaging 91
4.1.1 Au/Ge and Pd/C 91
4.1.2 Pd/C *in situ* Heating Experiments 93
4.2 Computational Methods 95
4.2.1 Sample Model Generation 95
4.2.2 Multislice Simulations and Training Data 98
4.2.3 Network Architectures, Training and Inference 100
4.2.4 Ground Truth Datasets and Inference Performance 104
4.3 Chapter Summary 105
5 nNPipe: A deep neural network pipeline for statistically significant metrics of heterogeneous catalyst systems 107
5.1 The nNPipe Method 107
5.2 Results 110
5.2.1 HRTEM Image Inference 110
5.2.2 Statistically Significant Metrics 116
5.2.3 Deployment in Automated Imaging 123
5.3 Discussion 125
5.4 Chapter Summary 128
6 Retraining and Adaption Strategies 129
6.1 Neural Network Retraining 129
6.1.1 Principles 129
6.1.2 Retraining Methods 131
6.1.3 Results 134
6.1.4 Discussion 148
6.2 Democratised Preparation of Training Data 151
6.2.1 Study Preparation 152
6.2.2 Preliminary Results 159
6.2.3 Discussion 163
6.3 Chapter Summary 164
7 Advanced Applications 165
7.1 Live Image Analysis 165
7.1.1 Setup Structure and Functionality 165
7.1.2 Experimental Setup Test 168
7.1.3 Discussion 170

7.2 Analysis of time-resolved *in situ* Datasets using *nNPipe* 171
7.2.1 Methods 171
7.2.2 Results - *in situ* Heating 175
7.2.3 Results - Analysis of time-resolved *in situ* Datasets 183
7.2.4 Discussion 186
7.3 Chapter Summary 190
8 Conclusion and Outlook 191
9 References 195
10 Appendix 227
10.1 Additions to Theoretical Considerations 227
10.1.1 Pulsed Electron Emission 227
10.2 Sample Model Generation 229
10.3 Multislice Simulations and Training Data 246
10.4 Network Architectures, Training and Inference 257
10.5 nNPipe: HRTEM Image Inference 269
10.6 Retraining *nNPipe* 281
10.7 Democratised Preparation of Training Data 293
10.8 Live Image Analysis 302

1 Introduction

The pursuit of a globally sustainable future is closely linked to the development of effective and efficient heterogeneous catalysts. They are of essential importance to the chemical industry [1] and to energy; approximately 25 % of industrial energy use is linked to the production of chemicals alone. Simultaneously, the majority of energy intensive processes are dependent on heterogeneous catalysts, underlining their key role and potential for the reduction of energy consumption [2].

Catalysts are materials '[..] that increase the rate of a reaction without modifying the overall standard Gibbs energy change in the reaction'[3], and even small quantities can convert considerable amounts of reactants ideally under mild conditions [4]. Generally, a distinction between homogeneous, heterogeneous and biocatalysts can made. Homogeneous catalysts – often organometallic complexes – are present in the same phase as the reactant, whilst heterogeneous catalysts exist in a different phase to the reactants. For the latter, the catalyst is mostly solid, while the reactants are either in a gaseous or liquid phase. Finally, biocatalysts utilise enzymes to increase the rate of specific chemical reactions [5].

For optimisation and design to be successful, catalysts must be analysed at several scales across space and time [4]. Trial-and-error approaches were applied in the past, but major advances in our understanding of surface science and reactions, as well as theoretical and experimental methodologies have enabled the targeted conception and production of catalytic processes [2] . In this regard, Electron Microscopy (EM) plays a major role in materials characterisation. In particular, (Scanning) Transmission Electron Microscopy ((S)TEM) is commonly used to analyse key properties and processes of heterogeneous catalysts in various environmental conditions close to *in situ* or *in operando* for both the chemical and physical spectrum [6]. Most importantly, (S)TEM instruments give access to an unmatched spatial resolution down to the sub-100 pm regime, as shown by data that clearly resolves the Ga-Ga dumbbell structure in GaN [7]. However, the operation and maintenance of (S)TEM instruments requires a high degree of competence and is not straightforward, despite significant technical advancements. Expert knowledge is furthermore key for the interpretation and processing of

images and spectra recorded, often involving manual or (semi)–automated post-processing which depend on the sample considered. Both the acquisition of optimal experimental (S)TEM data and corresponding analysis are time-consuming and can be thus considered a bottleneck in the loop between creation of new, improved catalyst materials and their performance-related characterisation. At the same time, the emergence of direct electron detectors [8] and new imaging techniques have led to an unprecedented rate of data generation [9], making accurate analysis even more time consuming. Hence, the need for new, precise and automated approaches is clearly recognised.

As is often the case in scientific research, it was advances from an unrelated discipline that inspired an entirely new analytical approach in EM to address this challenge. Deep Learning (DL) [10], a representation learning method that is part of Machine Learning (ML), has been increasingly employed for EM research in the recent past [11] (see Section 2.2) and has opened up a plethora of possibilities for fast, automated and intelligent data analysis. Nonetheless, the development and application of effective DL-based methods has its own obstacles and must be therefore conducted in an appropriately-conceived, scrupulous manner.

This work describes a novel DL-based approach for the automated analysis of morphologically diverse heterogeneous catalyst materials, named '*nNPipe'*. It allows for time-efficient and accurate analysis of raw 2048 x 2048 pixel HRTEM images by semantically segmenting nanoparticles and gives concomitant access to key material properties with statistical significance. *nNPipe* was developed with a particular emphasis on applicability to real-world, industrial heterogeneous catalysts imaged under various conditions and was successfully used for the analysis of both Au/Ge and Pd/C systems.

In the following chapters, all aspects relevant to a complete understanding of the *nNPipe* method and its potential to low contrast and large datasets generated by dynamic imaging are described. Chapter 2 summarises developments in time-resolved imaging, highlights recent applications of DL in EM in both physical and life sciences and introduces nanoparticle systems. Theoretical foundations to this work are introduced in Chapter 3, while Chapter 4 discusses experimental and computational methods. Chapters 5, 6 and 7 form the core of this work, in which the *nNPipe* method is explained in detail and where the experiments performed are discussed. First, *nNPipe* is applied in a realistic, automated imaging scenario

for the rapid analysis of material properties at statistical significance. Secondly, ‘Retraining and Adaption Strategies’ highlight how the analytical capabilities of the *nNPipe* method can be further enhanced or transferred to new, unseen catalyst systems, and the introduction of a new strategy to source training data brings Chapter 6 to a close. Third, ‘Advanced Applications’ explores the application of *nNPipe* for time-resolved imaging in two scenarios: Section 7.1 introduces a live imaging setup which generates first-order, quantitative information during sample observation or acquisition in real time. Section 7.2 describes an *nNPipe* driven analysis of time-resolved datasets acquired during *in situ* heating of a Pd/C catalyst, whereby the growth of nanoparticles was studied as a function of temperature and time at four independent sample positions. Automated analysis of an *in situ* movie acquired at 100 frames/s at 400 °C allowed for the precise estimation of the mean nanoparticle area growth rate. Finally, the experimental results and insights are summarised along with an outlook into the future in given in Chapter 8.

2 Literature Review

This chapter puts the work described subsequently into context with the existing literature and discusses image data generation and processing. Sections '2.1 Time-resolved Imaging' and '2.2 Applications of Deep Learning in Electron Microscopy' are based on the Literature Review submitted for the Transfer of Status examination at the Department of Materials, University of Oxford (2019) and K. P. Treder, C. Huang, J. S. Kim *et al.*, *Microscopy* (2022), **71**, i100-i115 respectively.

2.1 Time-resolved Imaging

Ever since the first Transmission Electron Microscope (TEM) instrument was constructed by Max Knoll and Ernst Ruska in 1931 [12,13], the pursuit to achieve higher spatial resolution remained a driving factor for instrument development [14] (Figure 2.1). The last century [13] saw numerous scientific contributions ultimately establishing the (S)TEM as a powerful instrument for modern materials characterisation with access to a broad signal-space in both, the spatial and chemical domain [14,21]. Throughout these developments, the aspect of time-resolution was arguably less frequently

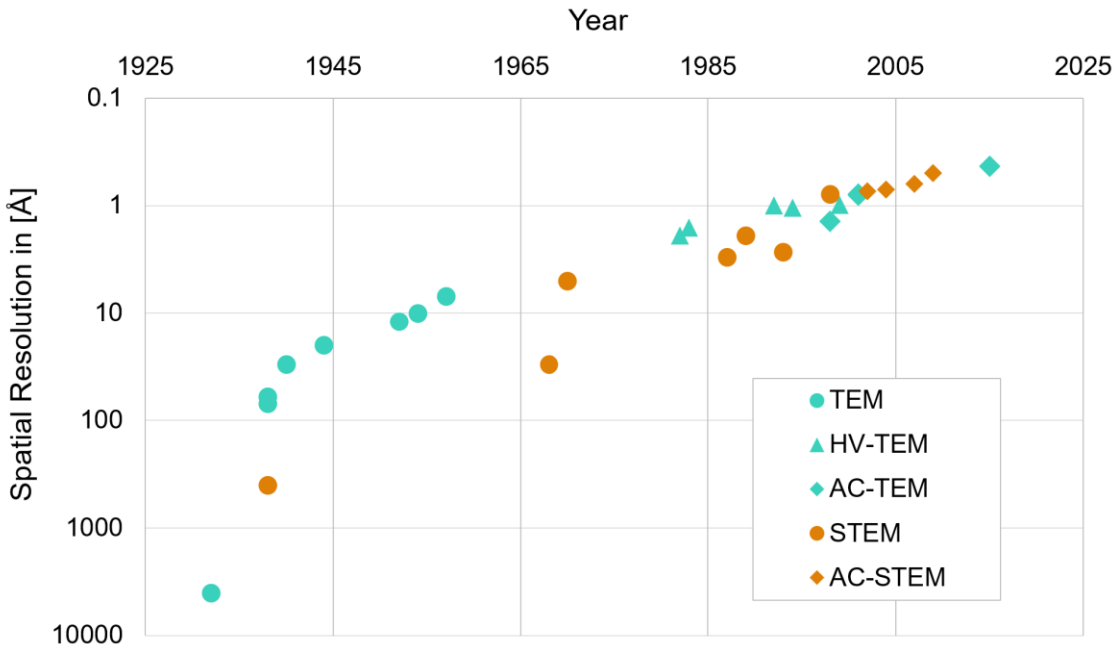


Figure 2.1: Improvement of spatial resolution in electron microscopy over time in years. Graph inspired by Rose [15] and Oxley *et al.* [16] , while values and respective references were obtained from Knoll and Ruska [12], Haguenau *et al.* [13] , von Ardenne [17,18] , Smith *et al.* [19,20] and Oxley *et al.* [16].

explored. Nonetheless, Hirsch *et al.* [22] did not only prove the theoretical concept of dislocations experimentally using TEM imaging [23–25], but also observed and described their *motion* on (111) slip planes that further elucidated the role of dislocations in plastic deformation and its connection to strain- and work-hardening [21,26]. Their work can therefore be considered a milestone of EM that strengthened the importance of *direct observations of dynamical processes* without which a fundamental understanding of materials and their properties remains unattainable. In the course of time, a whole discipline dedicated to the acquisition and analysis of dynamic, time-resolved events on the nanometre scale was established: *In situ* (S)TEM.

The Oxford English Dictionary definition of *in situ* means ‘in its original place’ or ‘in position’ [27]. On the other hand, an alternative translation ‘at the scene of the event’ seems more applicable to discussions herein, and possibly even for many natural science disciplines in general. It can be interpreted that experimental boundary conditions - ‘the scene’ - need to be set up such that a natural phenomenon under consideration - ‘the event’ - is likely to occur. The alert reader will recognise that successful measurements must be therefore subject to the question whether the phenomenon *can* be observed in the first place. In other words: Can we establish experimental conditions such that a specific natural phenomenon will occur and if so, are we able to perform measurements using EM? More recently, purpose-built sample holders, modifications to the (S)TEM instrument, direct electron detectors, *etc*., have played an important role in this context. It is these technical advances that have allowed observations of materials exposed to liquid or gaseous environments, rapid phase transformations, and beam sensitive materials in higher temporal resolution [28–30]. However, this does not mean these technical advances do not establish their own trade-offs [31]. Figure 2.2 shows exemplary non-equilibrium phenomena and the respective spatial and temporal resolutions required for measurement to provide an overview.

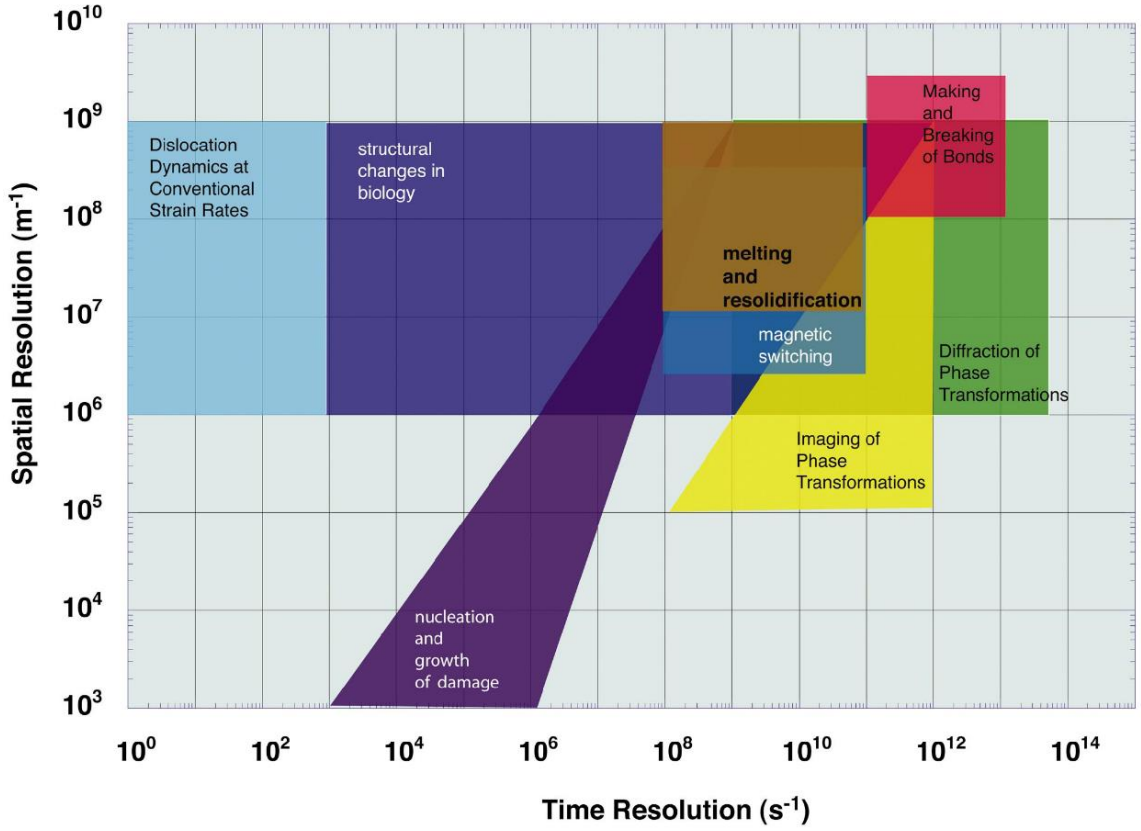


Figure 2.2: Selected non-equilibrium phenomena classified by their spatial and temporal resolution. The time-resolution is defined as the time required in single-shot imaging (see section 2.1.6) to visualise a considered irreversible phenomenon at a given spatial resolution. The spatial resolution can be defined with respect to imaging techniques, where the lower boundary of bounding boxes refer to the smallest resolvable feature and the upper boundary by the typical field of view. Graph as seen in King *et al.* [32].

Herein, the current state of time-resolved imaging in EM is discussed. First, a brief overview of the historical development of *in situ* imaging is given. Subsequently, the various imaging techniques and approaches to record time-resolved data are highlighted individually, followed by a discussion of the role of direct electron detectors and their related data generation. Insights shall be described on how dynamics and processes can be revealed using modern EM approaches; why time-resolved imaging has gained increasing importance; how different techniques access different spatial and temporal resolution, and most importantly, why novel, automated analysis approaches using Machine (ML) and Deep Learning (DL) will play a key role in this field in the future.

2.1.1 A brief historical Overview of *in situ* (S)TEM

The 1960s marked the beginnings of *in situ* EM where studies aimed to solve problems such as fatigue in metals [33]. The potential of *in situ* EM was recognised in these early stages, when expectations for direct measurements of crystal defects and dynamics of materials and processes were raised [34]. However, microstructural changes in thin-foil specimens differed from those encountered in bulk counterparts due to free surfaces acting as potential nucleation sites, providing fast diffusion paths or interacting with the microscope environment. Challenges were also encountered with the reliability and construction of specimen stages that would fit the constrained objective lens volume of a 100 kV instrument, eventually resulting in declining interest. [34]

In situ studies regained attention when high-voltage electron microscopes (HVEM) were introduced in the 1970s [35,36]. Significantly higher acceleration voltages (up to 3 MV) enabled higher image resolution by shortening electron wavelengths, but also introduced a three-fold increase in the working volume in the pole-piece gap and a four- to six-fold increase in specimen thickness. Hence, *in situ* experiments became more practical [34,37], but occurrence of dynamic processes still solely relied on the interaction between the electron beam and the sample [38–41]. It took several more years of technical development until dedicated sample holders would provide adequate stability to observe thermally activated processes in controlled manner. In 1985, Gibson *et al.* reported the observation of silicon surface reconstructions with samples annealed to over 1000 °C [42]. Sinclair and Parker [43] observed the solid-phase epitaxial regrowth (SPER) of silicon between 500 °C and 800 °C and suggested a heterogeneous nucleation mechanism being responsible for crystallisation rather than an atom-by-atom process. Following studies revealed the process to be thermally activated with $E_a = 2.7$ eV and to be driven by a single intrinsic mechanism, which was believed to be interfacial bond breaking or rearrangement [44,45]. To date, the specific atomistic mechanism responsible for SPER has not been fully identified [46]. The following years were characterised by continuing efforts in improving image resolution [14] (Figure 2.1), where a peak was reached with the introduction of aberration correctors [15,47–50]. This resolution extension also enabled atomic resolution imaging of lighter elements at lower acceleration voltages compared to HVEM. Further advances in technology

and techniques such as incoherent (S)TEM imaging [51–53], EDX and EELS [14,54,55], environmental columns [56,57], as well as microelectromechanical systems (MEMS) [28,58,59] led to a paradigm shift standardising *in situ* acquisition and image analysis over previously established post mortem approaches [29]. Figure 2.3 underlines this progress by showing the number of publications over time as modified from Sinclair *et al.* [41].

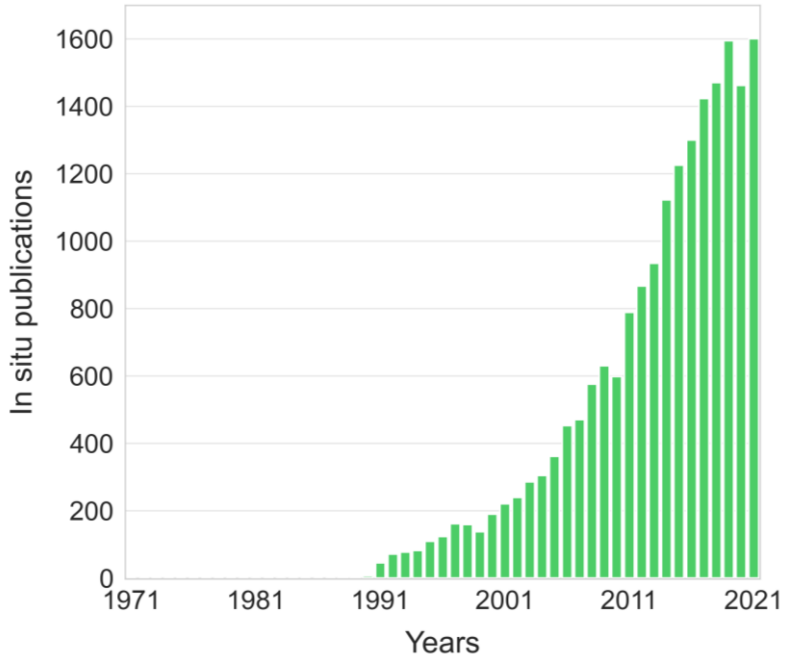


Figure 2.3: Number of publications identified with the keywords 'in situ' and 'TEM' for each year in the timespan between 1971-2021 on 'Web of Science' [60]. Only articles and review articles are considered in this figure. Note that entries between 1971 and 1991 are non-zero. Comparison from Sinclair *et al.* [41].

2.1.2 Nanomechanical Studies

The standardisation of *in situ* image acquisition also played an important role in nanomechanics. This (S)TEM field is not only of central importance for understanding size effects in materials, dislocations dynamics and their interaction [61], but also links to the atomic-scale where predictions of mechanical behaviour can be obtained by modelling suitably [62]. Initially, devices required for nanomechanical *in situ* measurements were only available through custom and dedicated manufacturing [63], but were progressively turned into modules with broad toolkits returning values for loads and displacement [61], leading to their commercialisation at the beginning of this century (Figure 2.4).

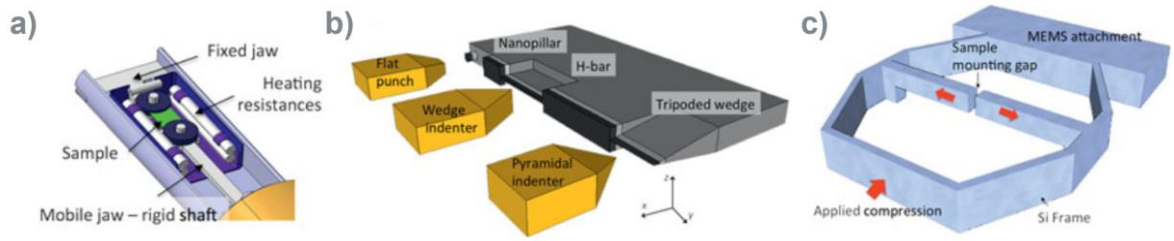


Figure 2.4: Examples of in situ (S)TEM methods for nanomechanics, modified from Yu *et al.* [61]. **a)** Tip of a classical straining holder with a high-temperature option. **b)** Schematic of different straining configurations used in nanoindentation. **c)** Theta (θ) geometry for a push-to-pull MEMS device.

A recent study is given by Li and Minor [64], who described dislocation traces from frame-by-frame analysis of TEM images that were acquired during an *in situ* tensile test on a defect-scarce Nickel single crystal. The crystallography of the nucleated dislocations was consistent with Molecular Dynamics (MD) simulations, resulting in the determination of an activation energy of $Q = 0.37$ eV and an activation volume of $\Omega = 30b^3$, with $b = a/6[11\bar{2}] = 0.1439$ nm for single dislocation nucleation events. Another example is given by Wang *et al.* [65], who developed an *in situ* welding method, with which tungsten nanowires were fabricated inside a (S)TEM instrument. Nanowires can be fabricated without pre-existing defects, enhancing the prospect for data at atomic resolution to be coupled to atomistic simulations. The authors showed that deformation twinning is the dominant deformation mode at low temperature and low strain rates when loading along $\langle 100\rangle, \langle 110\rangle$ and $\langle 111\rangle$ directions. The interested reader is also referred to relevant review articles where *in situ* (S)TEM: [61–63,66–74].

2.1.3 Microelectromechanical Systems

The first [75] MEMS device that featured both electrical circuits and amorphous silicon nitride windows (a-SiN_x) was introduced Zhang *et al.* [76]. The authors conducted calorific measurements to investigate the melting point depression of indium thin film, whereby a link between these measurements and particle size distributions was established by *ex situ* TEM imaging of nanoparticles deposited on the a-SiN_x windows of individual samples after heating. Although Heide proposed the concept of using electron transparent windows as early as 1962 [77], a standardised MEMS device with a-SiN_x for *in situ* experimentation was only introduced 43 years later [58] (Figure 2.5). Compared to the work of Zhang *et al*. [76], the sample holder had external electrical contacts making efficient operation inside the instrument viable [75]. Currently, a-SiN_x windows are predominantly used for quantitative *in situ* experiments, but it should be noted that alternatives such as graphene have been proposed for *liquid cell* applications [78]. The following sections discuss the principles, contemporary research and limitations of *in situ* (S)TEM. Particular focus is given to *heating*, *gas cell* and *liquid cell* systems, as they are most closely related to the work presented and are routinely applied in the analysis of heterogeneous catalyst materials.

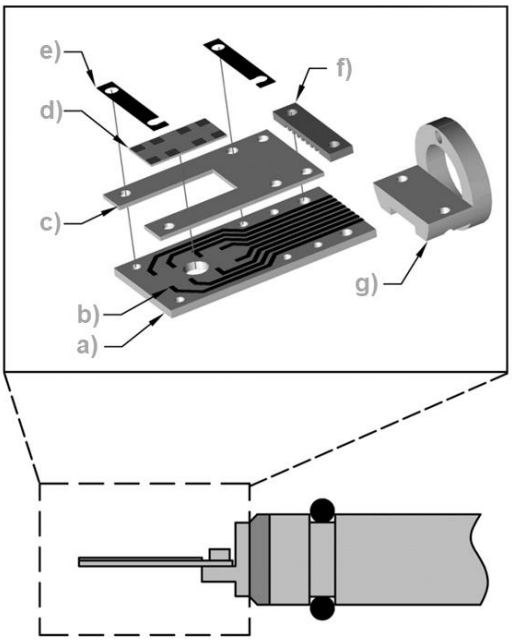


Figure 2.5: Schematic illustration of a typical MEMS sample holder. The bottom section shows a side view of the assembled holder. The top section shows the individual components: **a)** Baseplate, **b)** Electrical contacts, **c)** Alignment frame, **d)** MEMS device, **e)** spring clips to hold the device in place, **f)** Clamp to align and hold the contacts, and **g)** connector to the holder body. Scheme and description from [58].

2.1.3.1 Heating Chips

Compared to alternative technologies [42,43,79–82], MEMS based heating holders [58,76] have key advantages [83]: Temperature differences only affect the small working volume around the sample, ensuring precise and rapid control including low sample drift even at high magnifications. Typically, heating rates and working temperatures are limited at *ca.* $\sim 10^5$ K/s and 1300 °C, but kept below 10 K/s to minimise drift (> 1 nm/s) for atomic resolution imaging [84]. Heating impact is often analysed by

temperature-strain simulations in finite-element analysis software such as COMSOL [85] during the development stage.

Different arrangements of SiN_x windows on the MEMS chip have been proposed [83,86–93] for optimal working conditions and were eventually also considered as lower windows for *gas cells* (see *2.1.3.2*). Specific geometries became commercially available as exemplified by DENS Solutions [94] and Protochips Inc. [95] (Figure 2.6). Simultaneously, these MEMS chips accelerated the widespread application and impact of *in situ* heating experiments at atomic resolution. One well-known example is the study by Warner *et al.* [98] who probed the influence of temperature effects on graphene edge terminations. Production of zigzag edges was known to occur by anisotropic etching supported by metal nanoparticles and hydrogen at elevated temperatures, but characterisation of the mechanism was lacking. The authors used HRTEM and STEM imaging between room temperature and 800 °C on deposited graphene. A drastic change from zigzag dominated edge structures towards armchair dominated configurations was shown to occur between 500 °C and 600 °C due to a combination of reduced chemical etching and increased flipping rates of carbon atoms. A subsequent study [99], revealed the nanocrystalline structure of graphene oxide previously hidden by amorphous contamination.

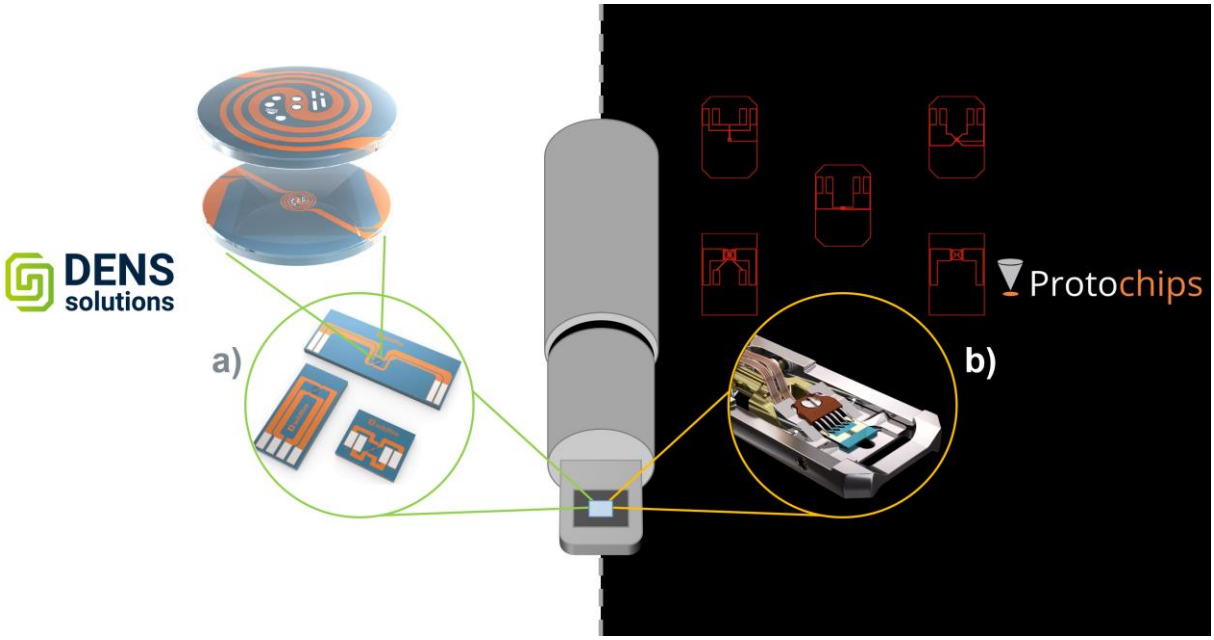


Figure 2.6: Commercially available MEMS based heating sample holders for in situ (S)TEM. The **left** side depicts the wildfire system [96] from DENS Solutions [94]. The **right** hand side shows the Fusion Select system [97] from Protochips [95]. **a)** Various wildfire chip architectures for different (S)TEM holders. The central part of the chip exhibits the same geometry and is enlarged above. A detailed report is given by Omme *et al.* [93]. **b)** Mounted Fusion Select heating chip. Alternative chip geometries suitable for various (S)TEM holders are shown as the red schematic diagrams above.

Heating chips are of particular relevance to the field of heterogeneous catalysis, where measurements of nanoparticle size distributions are a important in determining catalytic activity. Nanoparticle size distributions have been observed to shift towards larger particle sizes over time while also losing free active surfaces through mechanisms such as particle coalescence and Ostwald ripening [100–102] at high temperatures. Chapter 7 of this work discusses this particular aspect with the application of DL approaches for the analysis of an *in situ* heated palladium nanoparticle catalyst. Others [101] have described *in situ* HRTEM and STEM imaging on commercially available and laboratory prepared Ag and Pt nanoparticles at temperatures ranging between 25 °C and 500 °C. Sintering was observed in two different modes: (i) Particle migration along surface areas of the support with subsequent coalescence and (ii) an Ostwald ripening-like process where atoms and atom clusters dissolve onto the substrate followed by transport to another nanoparticle. Though on-particle surface diffusion coefficients were measured between $4.2 \cdot 10^{-16} - 1.1 \cdot 10^{-20}\ \mathrm{cm^2/s}$ for sintering of $12\ \mathrm{nm} - 40\ \mathrm{nm}$ Ag nanoparticles, these values were lower than those extrapolated from higher temperatures and as reported in thin film measurements. This was attributed to the presence of carbon on the surface of the Ag nanoparticles constraining the sintering process. Yao *et al.* [102] also verified the influence of surface compositions on particle migration. Low composition Pt alloyed nanoparticles (*i.e.* $PtFe_3$, $PtCo_3$, $PtNi_3$) were investigated by combining *in situ* heating TEM and STEM with EELS. A higher transition metal content in the surface of the nanoparticles was linked to the formation of oxides between 200 °C and 400 °C, which were responsible for enhanced diffusion during particle coalescence at higher temperatures (500 °C - 600 °C) due to the formation of oxygen vacancies originating from metal oxide reduction by the carbon support. In contrast, Pt-rich surface compositions led to lower particle mobility and atomic diffusion.

2.1.3.2 Gas Cell Chips

Gas cell chips can be thought of as an extension to heating chips: The central part of the gas cell is the lower chip (Figure 2.7a), which only differs by the inclusion of gas inlets and outlets (Figure 2.6). Assembly is completed by an additional rubber-ring and top chip that features another SiN_x window to ensure transparency to the electron beam while encapsulating the sample in gaseous environment. Figure 2.7 depicts the individual parts of a DENS climate nanoreactor [103] . Note the slightly different SiN_x window arrangement compared to that in Figure 2.6. Following Heides [77] proposal for an enclosed cell, Creemer *et al.* implemented the first modern gas cell nanoreactor [59,75,104] and demonstrated its functionality at ambient pressure and up to 500 °C. Their invention was subsequently extended for use with liquid environments (see *2.1.3.4*) with remarkable impact, especially in catalysis [75]. Dai *et al.* [105] recorded *in situ* STEM data from Pt_3Co nanoparticles to study the formation of the Pt-rich shell in core-shell nanoparticles responsible for enhanced oxygen reduction reaction (ORR) activity. Earlier works had already described the formation of such particles, but property tuning followed a trial-and-error approach [105] which underlines the importance of gas cell imaging in developing a fundamental understanding. An atomically resolved disorder-to-order transition occurring at 720 °C at 1 atm oxygen pressure was identified, together with Pt layer-by-layer growth on a {100} surface of the nanoparticles (Figure 2.8). Other studies also combined multiple characterisation methods to expose mechanisms behind reduced catalytic activity in high detail [106,107]. Beck *et al.* [107] correlated *in situ* TEM, ambient pressure XPS, *in situ* XRD and DFT calculations to re-evaluate the strong metal-support interaction (SMSI), which was first observed in oxide supported metal-titania catalysts, and describes the loss of capacity to chemisorb molecules and intermediates [108].

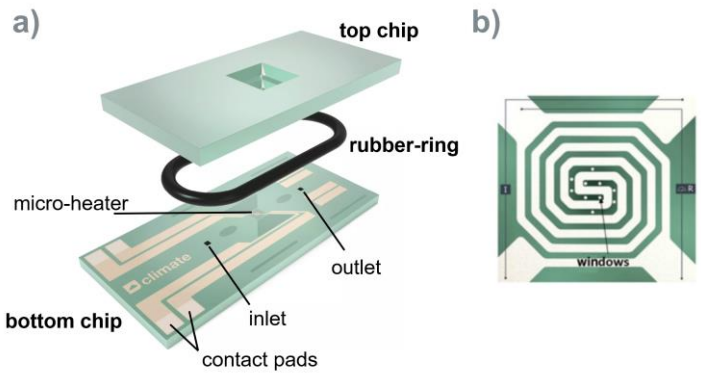


Figure 2.7: a) Gas cell geometry including bottom chip, rubber-ring and top chip of the DENS climate gas cell holder. Schematic as shown in [103], while the respective description was inferred from [92]. **b)** Top-view of the micro-heater and silicon nitride windows for deposition of sample material [92].

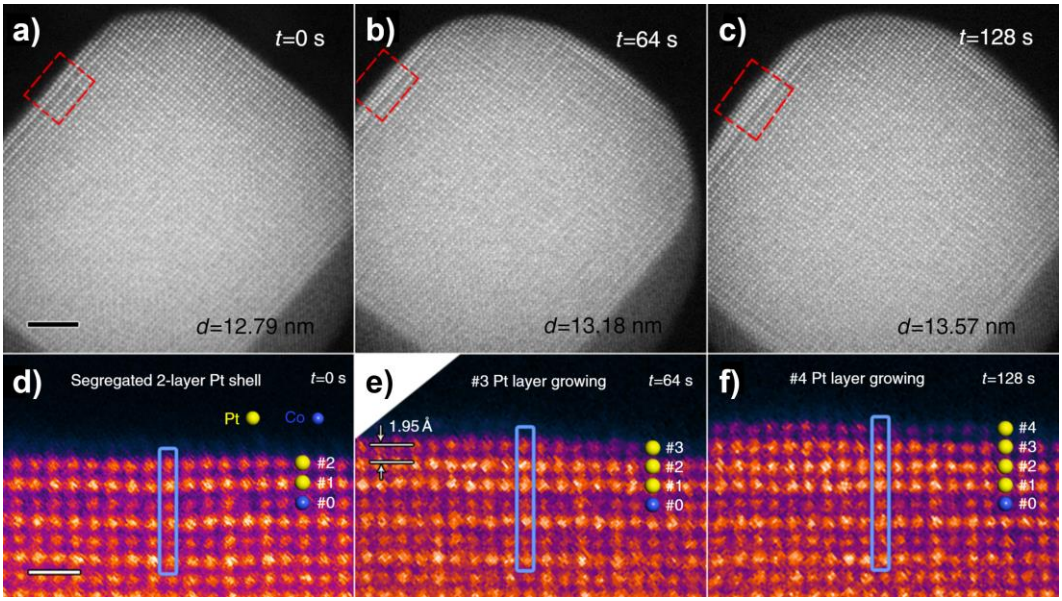


Figure 2.8: Layer-by-layer growth of a {100} Pt shell during oxygen annealing. **a) – c):** Sequential HAADF images taken at 0, 64, and 128 s during oxygen annealing at 300 °C, respectively. Scale bar, 2 nm. **d) – f):** Corresponding enlarged false-colour panels of a {100} surface (indicated by red boxes) in **a) – c)**, respectively. Yellow and blue spheres represent Pt and Co atoms, respectively. Scale bar, 5 Å. Apart from labelling **a) - f)** Figure and description as seen in Dai *et al.* [105].

Using time-resolved imaging (6.4 fps) and their combined characterisation approach, Beck *et al.* [107] observed the encapsulation of multiple Pt nanoparticles by reduced titanium oxide (TiO_x) and a complex formation mechanism revealed: During initial exposure to H_2 gas at 1 bar and 600 °C, Pt nanoparticles were gradually covered with amorphous coating containing Ti. Consecutive exposure to 1 bar of O_2 at 600 °C was expected to remove the coating, but increased its volume instead. As confirmed by additional EELS measurements, it exhibited disordered and crystalline TiO_2, alongside amorphous TiO_2 particles atop the Pt nanoparticle. Restoring of the H_2 atmosphere induced dynamic changes in the overlayer, including migration of amorphous TiO_2 particles to the Pt nanoparticle edge followed by their shrinking, along with deformation of the platinum lattice and Pt particle edge, overall thickness reduction and Ti enrichment of the coating. This process was once again reversed by restoring the O_2 atmosphere, whereby the appearance of altering structural defects and the reformation of a crystalline TiO_2 layer were observed [107]. Eventually, the obtained results disproved an earlier proposed overlayer removal process during oxidation. Further examples [109–111] showcasing the broad range of possibilities for this technique and encompassing review articles [112–116] provide further guidance.

2.1.3.3 Liquid cells

SiN_x windows do not only offer the advantages of being easily fabricated and electron transparent, but also exhibit high mechanic stability [117] allowing for the introduction of liquids into a similar reactor geometry as used for *gas cell* experiments [59]. A schematic illustration of the *liquid cell* architecture under working conditions is shown in Figure 2.9.

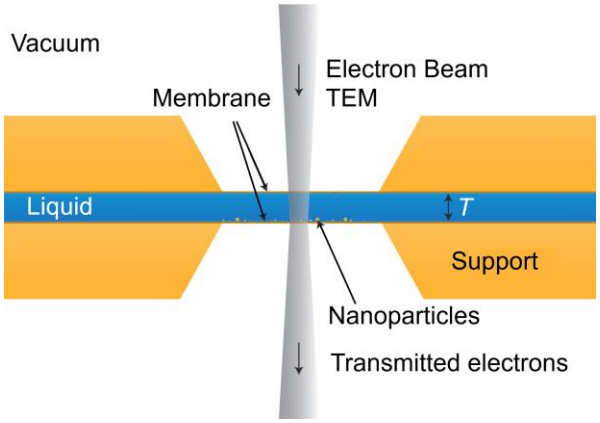


Figure 2.9: Schematic illustration of TEM imaging of nanoparticles fully enclosed in a liquid cell. Illustration and description modified as of [117].

The integration of electrical contacts in a liquid environment was an especially effective response to the need for an analysis technique in electrochemistry that offered sufficient spatial and temporal resolution. A first working setup was described by Williamson *et al.* [118] who captured the deposition of Cu clusters and their growth, providing new insights into the initial stages of clustering. Currently, *liquid cell* reactors are being used to study battery materials [119,120], nanoparticles [121–123] and biological materials [124–126]. Detailed reviews on the application of *liquid cells* for *in situ* (S)TEM experimentation are provided by de Jonge *et al.* [117,127], Ross [28], Wu *et al.* [128], Hodnik *et al.* [129] , Liao *et al.* [130] , Pu *et al.* [131] and Sun *et al.* [132].

2.1.3.4 Limitations

Although MEMS based sample holders have enabled countless, new experiments to be carried out inside (S)TEM instruments, it is equally important to point out challenges and limitations.

i) The use of SiN_x windows introduces additional electron scattering and thus a lowered spatial resolution and signal-to-noise ratio (SNR) at the detector stage, although due to their amorphous nature, no additional diffraction contrast can be expected [59]. Optimum working conditions can be therefore defined in an equilibrium between the highest attainable spatial resolution (thin SiN_x windows) and highest mechanical stability to avoid bulging of the chip in order to preserve the experimental conditions (thick SiN_x windows).

Additionally, species necessary for the creation of the experimental conditions (*i.e.* gases or liquids) cause further electron scattering. For *liquid cells*, graphene can be used as an alternative encapsulation material as its π-conjugated carbon network forms a repulsive field preventing penetration of atoms and molecules [78], and has been demonstrated to achieve both good spatial and temporal resolution [127]. It should be noted however, that for these cells the external stimuli, such as heating, cannot be applied [78].

ii) Using MEMS based chips to deposit samples can create new sources of chemical interaction. Although chemically inactive in most cases [133], SiN_x windows have been observed to crystallise above 800 °C under oxygen-containing atmospheres or to corrode in atmospheres consisting of water vapour and oxygen or mixtures of O_2 and H_2 [75]. Catalytic properties can also be affected by inhomogeneous chip surfaces and compositions, potentially leading to local fluctuations in activity [75,134].

Another important aspect are the interactions of high energy electrons with encapsulated gases or liquids. For *gas cells*, those include heating effects, radiolysis, knock-on damage and contamination [115,116] which may alter outcomes. In the case of *liquid cells*, the two most significant interactions are electrical charge building up on the SiN_x windows and inducing unwanted particle movement due to repulsion, as well as radiolysis which can for instance initiate the formation of bubbles due hydrolysis of water molecules [131].

iii) For *in situ* TEM, the temporal resolution with which dynamical events are observed is governed by conventional electron sources as well as the achievable SNR. It is furthermore limited by the maximum detector readout speed, while for *in situ* STEM scanning speed also has to be taken into account. Since the number of transmitted electrons is reduced due to additional scattering (see i)), detector performances of paramount importance. This aspect will be revisited in 2.1.7, where the current status of direct electron detectors is discussed.

Despite these constraints it has been possible to image beam sensitive materials exposed to a gaseous environment at atomic resolution as recently demonstrated by Xiong *et al.* [135].

2.1.4 Environmental TEM (ETEM)

There is also the concept of regulating the gas pressure inside the (S)TEM column as proposed by Marton in 1935 [114,136] . It introduced small apertures for the objective lens pole pieces that would limit gas leakage into the column of the instrument [114]. The approach eventually advanced into differently pumped environmental TEMs (ETEMs) for which Boyes and Gai played a key role in achieving atomic resolution imaging [56,57]. Figure 2.10 shows the schematic setup of a typical ETEM, which includes a series of differentially pumped zones that create a pressure gradient between the high pressure sample region in the objective lens, and the high vacuum in the rest of the instrument. The immediate advantage is that regular sample holders can be used for experiments, potentially enabling ETEM tomography [137] or mechanical testing. Furthermore, no physical objects obstruct the path of incident electrons, which is particularly advantageous for high spatial resolution and spectroscopy [138]. Conversely, only gas pressures to up to *ca.* 2000 Pa [138] can be realised, weakening links between experimental observations and real-world working conditions. Lower differential pumping apertures also limit applicable collection angles for (S)TEM imaging [114]. Lastly, the time-resolution limit is similarly given by the technical properties of electron detectors (see above). Due to improved spatial resolution and contrast, *in situ* ETEM imaging is however likely to reach higher SNRs and hence better interpretability of acquired data. At this point, it is worth asking whether there was a dedicated effort to improve the time-resolution of (S)TEM instruments, especially given decades of developments in achieving higher spatial resolutions and enabling *in situ* experimentation; in particular whether the readout speed of electron detectors always dictates the highest achievable time-resolution.

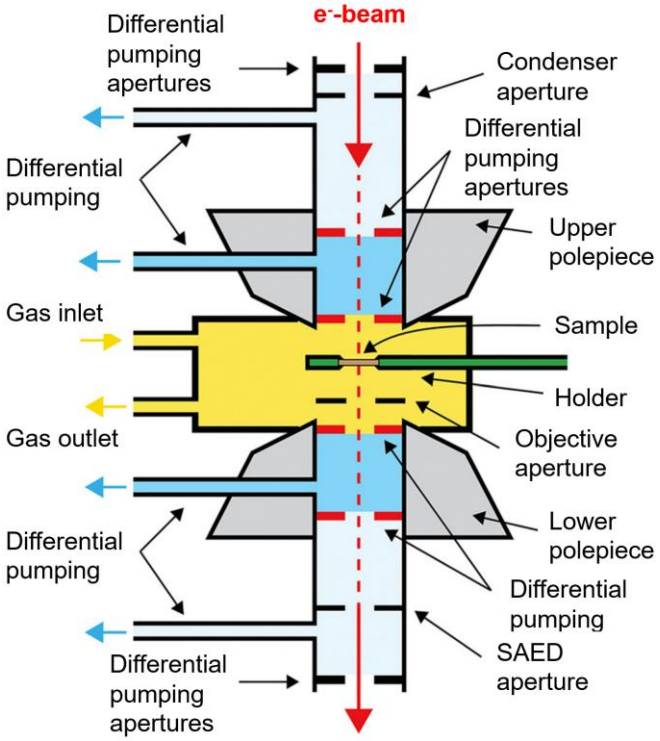


Figure 2.10: Schematic of differentially pumped environmental TEM (ETEM). Figure adapted from Gai *et al.* [114].

2.1.5 Stroboscopic Microscopy (UTEM)

First dedicated efforts to improve the time-resolution of a (S)TEM instrument can be traced back to the 1960s [139], when the idea of periodically illuminating a specimen while simultaneously pumping the specimen with an excitation pulse emerged. In early experiments, Bostanjoglo and Rosin [140] were able to reveal anharmonic wall motion with a 'static friction', alongside harmonic domain rotation in NiFe by combining Lorentz microscopy and stroboscopic illumination with pulse widths of 50 ns. The 1980s saw the integration of pulsed lasers into the TEM column [139] resulting in two main modes: Single-shot [141] and stroboscopic [142] imaging. The latter uses pump laser pulses on a specimen to trigger a specific sample state. Subsequently, pulses containing one to several electrons probe the specimen at pre-defined, time-resolved points repeatedly until sufficient signal is accumulated. Modern implementations – usually termed ultrafast TEM (UTEM) – reach time-resolutions down to the femtosecond regime with sub-nm spatial resolution [139]. This is realised a single laser: Pulses of 100 fs or less are separated by a beam splitter and directed towards the specimen and a photocathode with relative arrival times that can be adjusted by controlling respective path lengths [139]. A notable extension of this experimental approach is ultrafast electron diffraction (UED) [142].

However, there is fundamental limitation: Since signals are obtained in a cumulative manner, image acquisition *via* UTEM is only applicable to reversible phenomena. Nevertheless, this technique has provided insights with unmatched spatial and temporal resolution as shown in numerous works by Zewail *et al.* [142–145]. For instance, Zewail *et al.* [145] combined (S)TEM imaging with femtosecond laser pulses and EELS into ultrafast spectrum imaging (USI) and analysed the spatiotemporal dielectric response of plasmonic Ag nanoparticles after excitation with visible light. Feist *et al.* [146] used a comparable approach to demonstrate the coherent quantum state manipulation of free-electron populations. Ultrashort electron pulses with a spot diameter of 15 nm were focused close to an optically excited conical gold tip, where the near-field mediated optical interaction with free electrons. This interaction generated multiple spectral sidebands, whereby each sideband corresponded to the emission or absorption of an integer number of photons (Figure 2.11).

An almost perfect correspondence to the behaviour of a multilevel model Hamiltonian was observed, which may be an important feature for employing free electrons in quantum information technology [146]. More recent developments demonstrated the generation of attosecond pulse trains [147], as well as the inclusion of cold field-emitter-guns (FEGs) for higher brightness [148]. Reviews showcasing more applications of UTEM, related methods and their principles are given in various articles of Zewail *et al.* [142,149–151] as well as Campbell *et al.* [139].

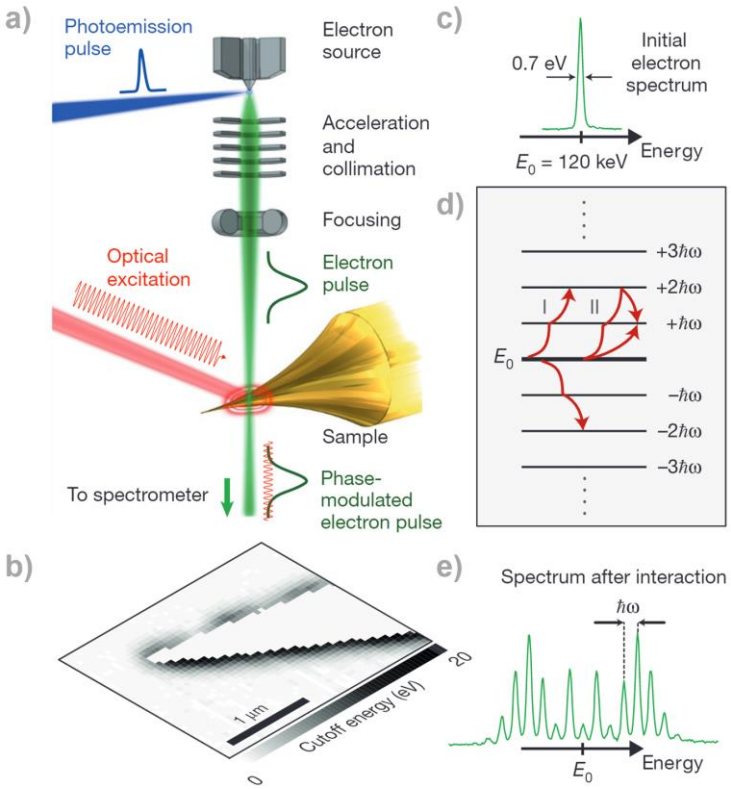


Figure 2.11: Schematic and principles of coherent inelastic electron scattering by optical near-fields**. a)** Experimental setup. **b)** Raster-scanned image of the energy cut-off in the inelastic electron scattering spectra, representing the local transition amplitude. **c)** Incident kinetic energy spectrum (full-width at half-maximum, 0.7 eV) centred at E_0 = 120 keV. **d)** Energy level diagram of ladder states with spacing $\hbar\omega$ coupled to the initial state at E_0. Arrows indicate sequential multistate population transfer (type I) and interfering quantum paths (type II) leading to multilevel Rabi oscillations. **e)** Example of kinetic energy spectrum after the near-field interaction, exhibiting a spectral comb with multiple sidebands separated by the photon energy and modulated in occupation. Figure and description from [146].

2.1.6 Dynamic TEM

The other time-resolved method – single-shot imaging – was pioneered by Bostanjoglo *et al.* at the Technische Universität (TU) Berlin [139,152–154], Germany. This approach uses laser excitation to extract single electron bunches of high brightness and current from an electron gun. In doing so, enough electrons are extracted to acquire images of transient, non-equilibrium phenomena such as phase transformations instantaneously with high time-resolution. Bostanjoglo *et al.* [152] used a laser pulses to locally heat a tungsten harpin emitter for electron emission [26], and obtained electron pulses of 50 ns FHWM and currents of up to 1.6 mA in a first setup. Later, the SNR was increased by switching from thermoemission to photoemission and an additional sample laser to trigger processes and reactions was introduced. Rapidly switching electrostatic deflection plates allowed the capture of up to three frames to track dynamics, with electron pulse lengths down to 7 ns [26,153,154]. Amongst other results, the research group acquired phase transformations of various metal thin films with ns time-resolution [155–158]. Assuming a maximum acquisition speed of $\sim 1\,\mathrm{ms}$ for conventional HRTEM, this operational mode achieves time-resolutions which are six orders of magnitude higher and it is generally referred to as Dynamic TEM (DTEM).

However, this technique also faces fundamental limitations. The generation of electron pulses of high current and brightness requires the instantaneous ejection of $\sim 10^9$ electrons per pulse, which induce space charge effects [139] degrading the spatial resolution. In previous examples, a maximum spatial resolution of $\sim 200\,\mathrm{nm}$ was achieved, but the authors [154] also pointed out that the additional loss of spatial resolution was due to a non-optimised detection system and electron shot noise. A more thorough comparison of conventional and pulsed electron sources is discussed in Chapter 3 of this work.

The group at Lawrence Livermore National Laboratory (LLNL) built a more modern DTEM instrument from 2003 onwards [26]. Specifically, a JEOL JEM-2000FX instrument was used as a base and was substantially more advanced than the Siemens Elmiskop 1A at the TU Berlin, not only promising to achieve much higher image quality due to lower chromatic aberration, but also more stability. An additional condenser lens, the ‘C0 lens’ [26] optimised the ratio between emitted electrons and those collected in the imaging plane. Early results showed a maximum achievable spatial resolution of

< 20 nm with a time-resolution of 10 ns, which is about an order of magnitude higher than reported previously. Both instruments are schematically compared in Figure 2.12.

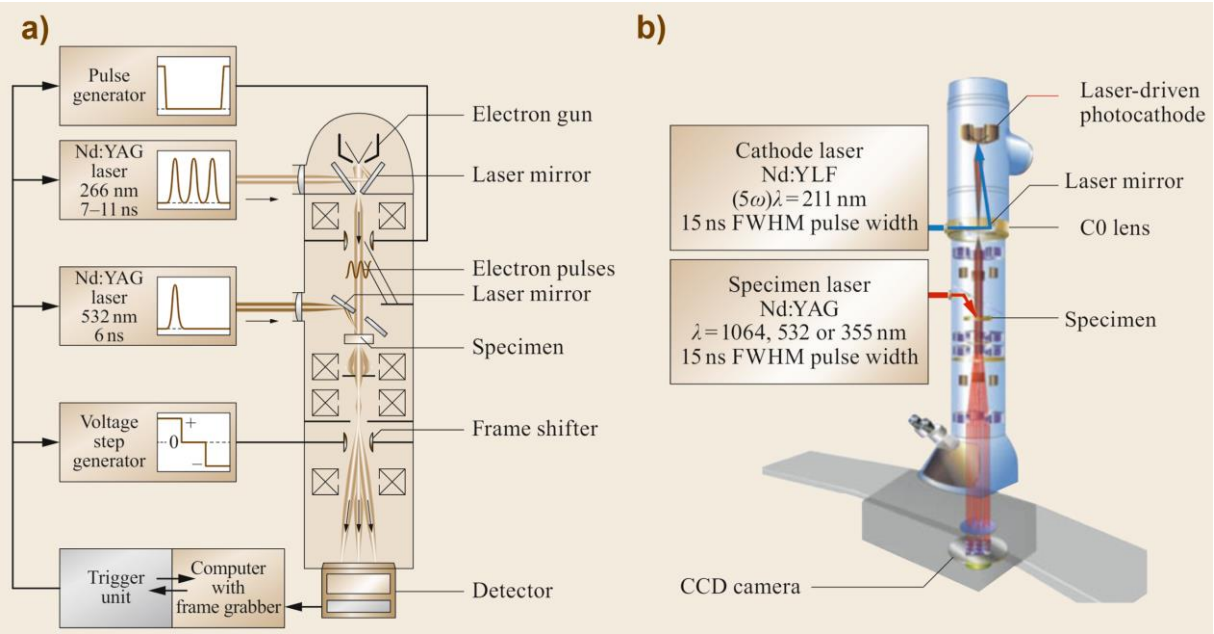


Figure 2.12: Schematic comparison of DTEM instruments proposed by **a)** TU Berlin and **b)** LLNL. Taken from Campbell *et al.* [139].

Subsequently, an arbitrary waveform generator (AWG) cathode laser system was added in conjunction with a high-speed electrostatic deflector array [159,160] . This allowed pre-definition of pulse durations and delays where each pulse would capture an image of the sample at a given time. Electrostatic deflectors directed each pulse to a sub-quadrant of a high-resolution Charged-Coupled Device (CCD) camera, effectively generating a multi-frame movie with a time-resolution purely defined by pulse lengths. Although other three-frame realisations existed (see above), most experiments would only be feasible with single pump-probe approaches on multiple positions of a sample. Hence, the chronological history of irreversible events such as phase transformations were only linked to an 'averaged' estimate. The new capability of acquiring multi-frame movies on pristine sample positions in the DTEM led to a new name: movie-mode DTEM (MMDTEM). Its working principle is shown in Figure 2.13 [139,159,160].

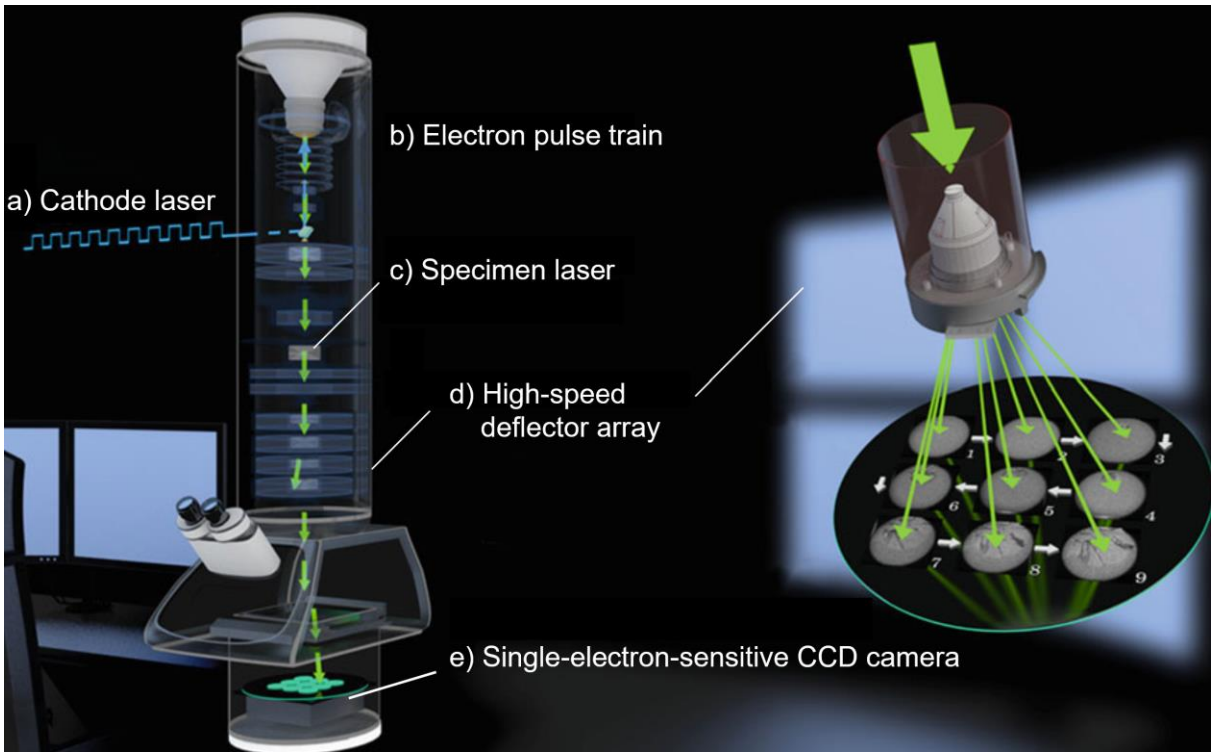


Figure 2.13: Schematic of MMDTEM **a)** arbitrary waveform generation (AWG) cathode laser producing a train of UV pulses to generate **b)** Electron pulses with the same temporal pattern. **c)** The specimen laser initiates the process to be studied, and **d)** A high-speed deflector array (shown in detail on the right-hand side of the image) rapidly switches each pulse to a different region of **e)** a single-electron-sensitive CCD camera. Figure with modified fonts and modified description taken from Campbell *et al.* [139].

New insights into various non-equilibrium phenomena were obtained with this instrument, as shown in by Kim *et al.* [161], who observed the phase transformation reaction front of $Al/Ni_{0.91}V_{0.09}$ reactive multilayer foils (RMLF). Measurements with 15 ns FHWM pulses revealed a transient cellular phase exceeding the original bilayer periodicity of the RMLF and were consequently linked to binary alloy solidification during undercooling, and to a solid solution eutectoid transformation. Corresponding diffraction data allowed a structural determination of individual observed phases, and confirmed that the cellular phase was formed from the two phases of ordered B2 NiAl and liquid. A similar approach was used by Grapes *et al.* [162], who combined DTEM with nanocalorimetry to investigate the phase transformation of Al/Ni bilayers when heated at rates of 830 K/s and ultimately confirmed that the phase transformation sequence observed at slower heating rates in DSC, namely $Al + Ni \rightarrow Al_3Ni + Ni \rightarrow Al_3Ni_2 + Ni \rightarrow AlNi$, occurs in the same manner at high heating rates. Santala *et al.* [163] employed MMDTEM to visualise the crystallisation dynamics of an amorphous GeTe film with nanosecond time-resolution (Figure 2.14). Numerous other application examples are also given by Campbell *et al.* [139].

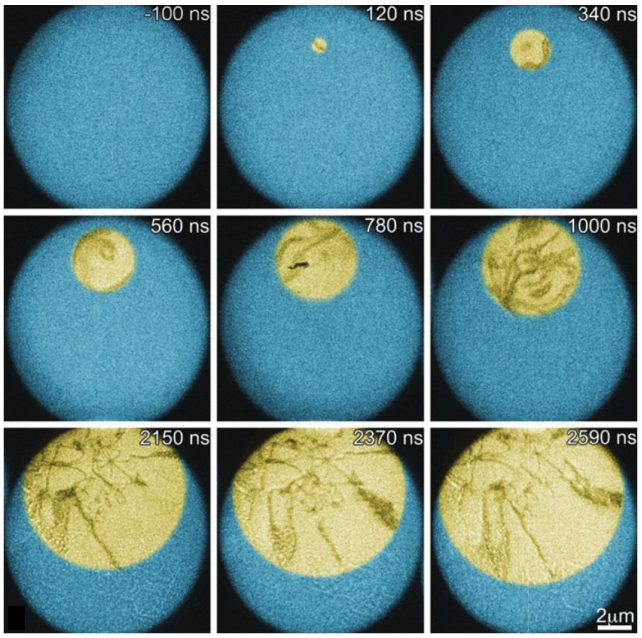


Figure 2.14: Growth of crystalline regions (yellow) into amorphous GeTe (blue) (false colour) in two nine-image series of 17.5 ns electron pulses after 4.7 µJ laser shots. The time signature in each frame is relative to the time of the peak specimen laser intensity with an uncertainty of ±3 ns. Image and description as seen in Santala *et al.* [163].

In DTEM standard CCD detectors can be used as the temporal resolution is entirely defined by the applied pulse length and movies are acquired by deflecting individual pulses to sub quadrants of the detector (MMDTEM) [139]. Meanwhile, time-resolution limitations in continuous illumination (S)TEM imaging are governed by the acquisition capabilities of the detector. While CCD detectors use indirect coupling *via* suitable scintillators for signal generation, Complementary Metal Oxide Semiconductor (CMOS) detectors are radiation-hardened and use electrons for signal generation directly. Direct operation allowed for overcoming the inherent limitations of indirect CCD detection with modern CMOS detectors achieving significantly higher imaging performance and running at higher acquisition speeds. In order to obtain a full comparison of the major time-resolved approaches in TEM, the following section highlights the main aspects of direct electron detectors.

2.1.7 Direct Electron Detectors

Electron detectors are semiconductor devices that convert primary electrons involved in (S)TEM image formation into a digital signal [8]. Since aberration correction has advanced the spatial resolution of (S)TEM instruments significantly in the course of time (see Section 2.1.1), digital detectors are now recognised as a limiting factor in experiments [164]. Initially, first-generation CCD were used alongside photographic film since their resolution and sensitivity was inferior, despite allowing for instant digital image acquisition [8]. It was not until the late 1990s when CCD detectors were technically advanced to feature large dynamic ranges and higher sensitivity than many photographic emulsions, and were thus preferred. Their working principle is described as follows: First, primary electrons illuminate a YAG single crystal or phosphor powder scintillator that is optically coupled to the CCD chip *via* fibre or lens optics, generating a signal [165] (see Figure 2.15). Once a signal is received, it is read-out off-chip. However, this *indirect* signal acquisition imposes inherent limitations to the detector such as relatively poor resolution, limited field of view and low efficiency due to electron and photon scattering in the scintillator, and optical blurring in the coupling chain (see Figure 2.15) [164,166,167].

Recent advances in CMOS devices raise a considerable advantage over CCD detectors as they are radiation-hardened, and are therefore suitable for *direct* signal generation, *i.e.* without a scintillator: Primary electrons create electron-hole pairs inside the chip which are *directly* converted into a signal [8]. There are two main architectures: Monolithic Active Pixel Sensors (MAPS) and Hybrid Pixel Sensors (HPS) (Figure 2.15). While MAPS devices are backthinned and have readout electronics integrated inside the sensitive layer, HPS devices have their readout-circuits bump-bonded to the pixel readout structure wafer [8]. Both architectures feature active (smart) pixel technologies enabling several readout modes. This is in contrast to CCD devices that can only allow integrating detection (*i.e.* accumulation of signal for a set time and linearly converting stored charge into an intensity signal). Most importantly, active pixels allow for counting detection where each incident electron is detected *individually*, isolated and localised. This equals a direct conversion *individual* single pixel counts [8] and consequently allows for virtually noiseless electron detection. Overall, the key features of direct

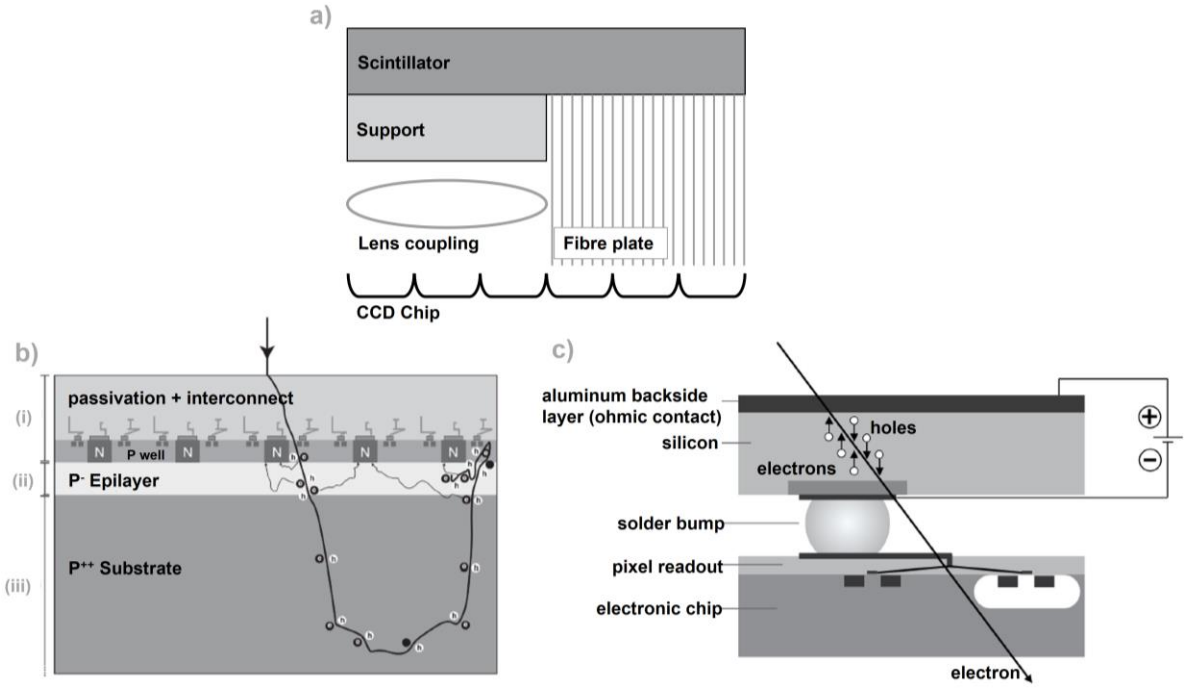


Figure 2.15: Schematics of different detector architectures comparing *indirect* (**a**) and *direct* (**b** and **c**) electron detection. **a)** Two variants of a CCD detector, depicted by two different halves. Incoming electrons are scattered in the scintillator and emit photons. Either a lensoptical (left half) or a fiberoptical (right half) coupling system subsequently transfers generated photons to the CCD chip, where electron-hole pairs are generated and a signal is read out. Schematic adapted from [165]. **b)** MAPS detector. (i) P-well passivation layer with (5 µm) with readout electronics. (ii) P^- Epilayer for appropriate signal generation. (iii) Heavily doped substrate. The track of a single, high-energy electron is illustrated with a potential track in a non back thinned detector [172]. **c)** Single exemplary pixel of the Medipix2 detector. Holes are being drawn to the readout electronics through the solder bump, effectively achieving full depletion of the detector. Additionally, the path of an incoming electron passing through the sensor layer (top) and the readout chip is drawn to illustrate the functionality. Schematics of **b)** and **c)** were redrawn from Farqui and McMullan [168].

electron detectors are increased sensitivity and improved resolution close to the Nyquist frequency. While the former allows for image acquisition at lower electron doses, the latter enables the resolving of smaller features at lower optical magnifications for a given pixel size and hence a greater field of view [8].

2.1.7.1 Characterisation Metrics

Quantitative characterisation of detectors requires the determination of measurement performance as a function of spatial frequency ω. Note that ω can be described by components of a discrete Fourier transform as electron detectors are usually designed as discreet (pixelated) 2D arrays. The detector performance is measured relative to the Nyquist frequency given as $\frac{1}{2}\Delta d$ (where Δd is the pixel spacing) and plotted as a modulation transfer function (MTF) [165,168,169] :

$$\mathrm{MTF}(\omega) = \frac{\mathrm{Modulation}(\omega)_{\mathrm{out}}}{\mathrm{Modulation}(\omega)_{\mathrm{in}}} = \left(\frac{I(\omega)_{\max} - I(\omega)_{\min}}{I(\omega)_{\max} + I(\omega)_{\min}}\right)_{\mathrm{out}} / \left(\frac{I(\omega)_{\max} - I(\omega)_{\min}}{I(\omega)_{\max} + I(\omega)_{\min}}\right)_{\mathrm{in}} = \sqrt{\frac{P(\omega)_{\mathrm{out}}}{P(\omega)_{\mathrm{in}}}} \tag{2.1}$$

where $I_{\min}$ and $I_{\max}$ refer to the minimum and maximum measured intensities of a deterministic test pattern with frequency ω respectively, P denotes the power and subscripts 'in' and 'out' refer to the image before and after detection [169]. The MTF can also be described as the Fourier Transform of the point-spread function (PSF) as:

$$\mathrm{MTF}(\omega) = FT(\mathrm{PSF(r)}) \tag{2.2}$$

where r refers to a real space coordinate. The PSF is essentially the blurring kernel of the detector [169], represented as:

$$Image_{out} = \mathrm{PSF}(r) \otimes Image_{in} \tag{2.3}$$

It is always favourable to achieve high values of the MTF over the broadest range of ω. However, given that electron detectors consist of finite pixels, the MTF decays towards the Nyquist-limit [169] and as such provides a measure of the detector *resolution* as a function of spatial frequency.

The second important characterisation measure is the Noise Power Spectrum (NPS) which describes the spatial frequency dependence of the noise as its Wiener spectrum [8]:

$$\mathrm{W}(\omega) = \frac{1}{N}\left\langle \left| \int I(x,y) \exp\left(-2\pi i(xu + yv)\right) dxdy \right|^2 \right\rangle \tag{2.4}$$

where N is the number of pixels in an image and I refers to an image of pure noise acquired with uniform illumination. One can think of this Wiener spectrum as a discrete Fourier transform of an image consisting only of Poisson noise showing a series of uncorrelated events, where each event generates a single count. In this scenario, the NPS would be independent of the spatial frequency and could be described as n, the mean dose at all spatial frequencies. Note that a large peak at $\omega = 0$ needs to be considered due to edge effects introduced by a Fourier transform of a finite detector and that there are different normalisation approaches that have been proposed [8]. The normalised NPS (NNPS) is obtained by:

$$\mathrm{NNPS}(\omega) = \frac{\mathrm{NPS}(\omega)}{\mathrm{NPS}(0)} \tag{2.5}$$

and is often assumed to be constant for a detector. This is a reasonable assumption for high and medium electron doses, but needs to be carefully reconsidered in scenarios where detector readout noise contributes significantly to an image [8].

The last quantitative measure of detector performances introduced here is the Detection Quantum Efficiency (DQE), which relates the input to output signals [169,170]:

$$\mathrm{DQE}(\omega) = \frac{\mathrm{SNR}(\omega)_{\mathrm{out}}}{\mathrm{SNR}(\omega)_{\mathrm{in}}} \tag{2.6}$$

Whereby SNR denotes the signal-to-noise ratio. Equation (2.4) essentially provides a measure of electron detection *efficiency* again as a function of spatial frequency, which is especially important for low-dose applications. Note that the DQE has a quadratic relationship to the MTF [169]:

$$\mathrm{DQE}(\omega) = \frac{n_{\mathrm{out}}^2 \cdot \mathrm{MTF}^2(\omega)}{n_{\mathrm{in}} \cdot \mathrm{NPS}(\omega)} \tag{2.7}$$

In this case, n_{out} is the average signal of the output image (without a test object), while n_{in} refers to the input electron dose. Setting $n_{\mathrm{out}} = n_{\mathrm{in}} = 1$ and assuming a constant $\mathrm{NPS}(\omega)$, we obtain the DQE of an ideal detector:

$$\mathrm{DQE}(\omega) = sinc^2(\pi\omega/2) \tag{2.8}$$

2.1.7.2 Monolithic Active Pixel Sensors (MAPS)

A quantitative comparison of different MAPS detectors has been provided by McMullan *et al.* [171], who considered the commercially available back-thinned detectors and applied the knife-edge method to determine MTFs and DQEs. Their results are shown in Figure 2.16 and show that all of these MAPS sensors have a higher DQE than photographic film. Note that all of the considered detectors can be run in counting mode, which is of particular advantage for time-resolved applications as single frames can be extracted without prior image integration. In the discussed study, counting mode was however only used for the K2 Summit [171]. This is also observed in Figure 2.16, where the K2 Summit detector

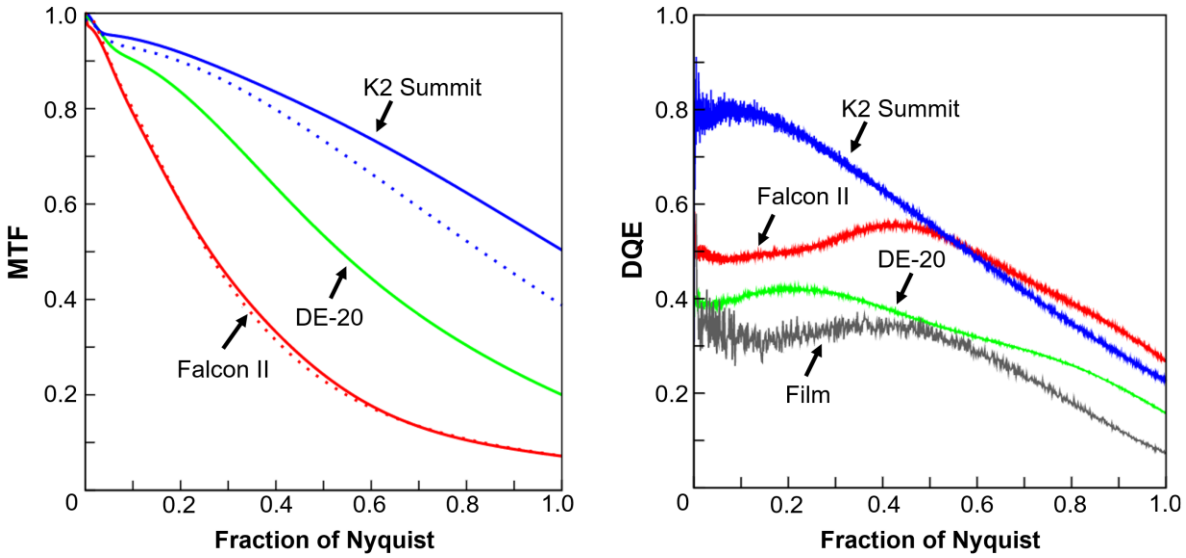


Figure 2.16: Measured MTFs (left) and DQEs as function of Fraction of Nyquist. **Left:** Solid lines represent MTFs in super-resolution mode, while dotted lines correspond to normal resolution mode in the case of the K2 Summit and to noise power spectra method in case of the Falcon II. **Right:** DQEs of the MAPS detectors. Additionally, the DQE of photographic film was added for comparison. Graphs with modified fonts taken from [171].

achieves particularly high DQEs at low fractions of Nyquist. Another study [172] compared the performance of photographic film, a CCD detector as well as HPS and MAPS detectors. MAPS detectors showed most promising results for applications with acceleration voltages > 300 keV [168,171], while the same held true for the Medipix2 HPS detector for electron energies < 120 keV. For the latter at higher acceleration voltages, the lateral spread of incident electrons degrades the spatial resolution due to the detector thickness [168]. The newest generation of MAPS detectors such as the Direct Electron Celeritas enables a time-resolution of up to 1,960 fps (~ 0.5 ms/frame) at 1024 x 1024 pixel or 26,400 frames (~ 0.038 ms/frame) at 256 x 256 pixel spatial resolution [173]. Based on the discussions provided by McMullan *et al.* [171,172], backthinned ($20 - 50$ µm) MAPS detectors have superior DQEs and MTFs compared to CCD detectors while offering significantly higher frame rates and should thus be the recording device of choice when conducting time-resolved *in situ* experiments at higher acceleration voltages.

2.1.7.3 Hybrid-Pixel Sensors (HPS)

At lower acceleration voltages, increased electron scattering in thin MAPS detectors would lead to signal detection across multiple pixels and consequently degrade performance. In that respect, HPS detectors can have an advantage over MAPS detectors [174], as their pixel pitch is larger (55 µm vs. 6.4 µm for Medipix3 [174] and DE-20 [171] detector respectively). The Medipix HPS detector was

derived from a series of detectors originally developed for particle physics at CERN [168,175]. In principle, every single pixel of the chip works as an independent detector [175]: Incident electrons create electron-hole pairs in the top sensor layer on which a bias is applied. Subsequently, charges drift to a collection electrode and reach the amplifier of the readout-chip by passing through a solder bump (Figure 2.15).

The latest generation of the Merlin Medipix3 detector [176] is based on 55 x 55 μm pixels with a 500 (or 375) μm thick sensor layer that can be arranged in 256 x 256 pixel or 512 x 512 pixel arrays, and runs at 1,800 fps (~ 0.55 ms/frame) in 12-bit mode. It provides a single-pixel mode (SPM), as well as a charge summing mode (CSM), where the latter is used to mitigate the effects of charge sharing that can either occur when incident electrons disperse laterally across multiple pixels or when secondary electron/hole charge clouds are broadened by diffusion. In simple terms, this is accomplished by the digital circuitry within the pixels that compare generated charges within multiple pixels affected by a single event and only register the one with highest charge [174]. SPM enables predefinition of a threshold value that has to be exceeded for a count to be registered in a pixel. Mir *et al.* [174] determined the MTF and DQE of the Medipix3 detector at acceleration voltages of 60 kV and 80 kV and clearly showed a straightforward improvement of both when CSM was applied. However, it did not yield ideal identification of single electron events and recovery of events leading to charge distribution. In accordance with other studies [168,175], the authors concluded that the CSM of Medpix3 is best applied for low-dose imaging.

2.1.8 Challenges of time-resolved Imaging

In this section, an overview of various time-resolved methods is provided, highlighting the state-of-the art techniques that give access to various spatial and temporal resolutions in EM. These techniques can be regarded as complementary to each other, but individual technical limitations hinder the overlaps necessary for a complete understanding of processes and dynamics occurring in a given sample using only one technique. This is illustrated in Figure 2.17, which maps the minimum and maximum accessible spatial and temporal resolution of the techniques discussed.

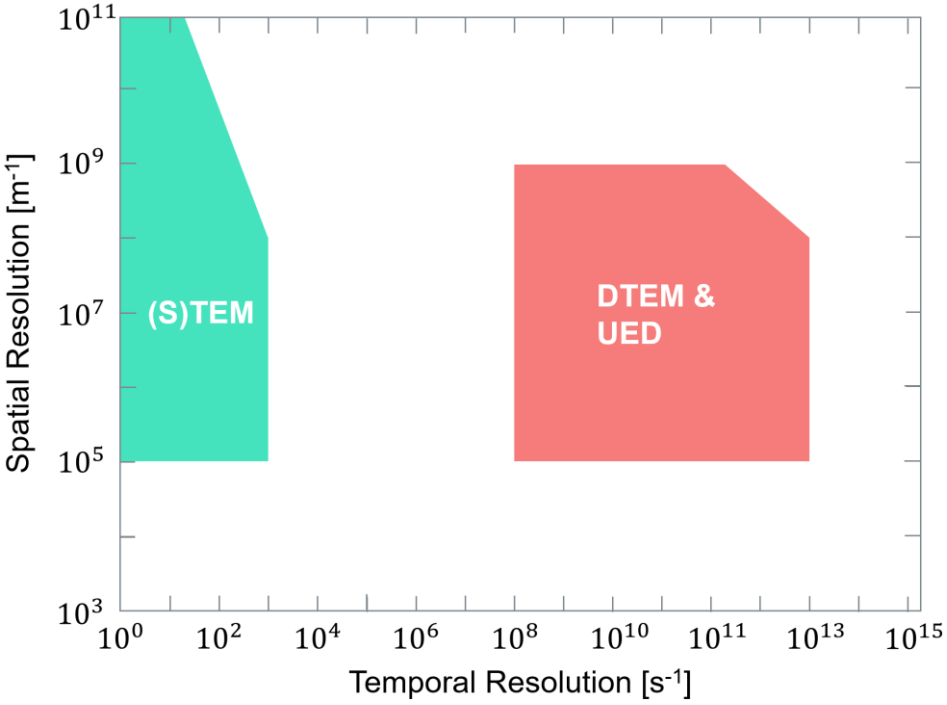


Figure 2.17: Minimum and maximum accessible spatial and temporal resolution for selected techniques.

In practice, this is encountered during *in situ* HRTEM imaging, where continuous electron beams are used for operation. Although direct electron detectors enable sub-ms image acquisition (see 2.1.7), faster acquisitions imply less signal per frame for a finite maximum current. Introduction of gases or liquid and SiN_x windows impose additional scattering, reducing the usable signal (see 2.1.3). DTEM on the other hand works against limited signal by extracting high current electron pulses, but inherits lower spatial resolution due to beam degradation (see 2.1.6). Repeated pump-probing of the sample offers a balance between the two, but is only applicable to reversible phenomena (see 2.1.5). Potential hardware solutions such as using MMDTEM-like deflector systems in TEM imaging are promising pathways to bridge current gaps [139].

In addition to a requirement for higher spatial and temporal resolution, it is arguably even more important to reconsider what is being captured by these technically advanced instruments: data. Spurgeon *et al.* [9] correlated the various electron detectors and their respective data produced over time (Figure 2.18). Developments show a steep increase in data produced per given time interval. In fact, the range in which it is feasible to treat data manually was surpassed in about 2010, accelerating research towards more automated and augmented analysis approaches. Recently, significant efforts were made in employing automated analysis methods originating from the field of Artificial Intelligence (AI). ML, and in particular DL methods have shown to be extremely efficient in interpreting microscopy data and feature detection [9]. As this work not only demonstrates the application of DL networks on highly heterogeneous catalyst systems (Chapter 5), but also to time-resolved data (Chapter 7), the current status of DL in EM is reviewed in the following for proper contextualisation

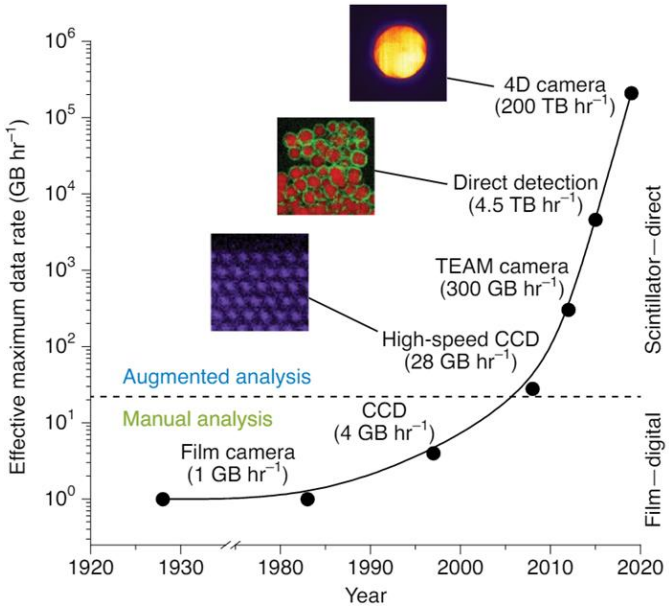


Figure 2.18: Effective maximum data production rates per hour plotted against the year. Newer detector technologies have enabled a steep increase in the data produced per given time interval [9].

2.2 Applications of Deep Learning in Electron Microscopy

Deep Learning (DL) was originally developed within the field of Machine Learning (ML) and had led to significant progress in the solution of a variety of computer vision problems [177]. Driven by the impact of direct electron detectors generating an increasing amount of data, new imaging modes and measurements on complex specimens [9,178,179], recent reports discussed the use of DL approaches for improved automated data analysis in EM. The following sections present applications of these in the fields of *physical* and *life* sciences, while to the latter are only discussed briefly. A more encompassing review is given in Treder *et al.* [11] which also serves as basis of this part of this chapter. It must also be noted, that longer established ML approaches such as Principal Component Analysis (PCA), sparse coding, k-means clustering, random forest and others have also broad application to EM data, but are only covered in a comparative or supplementary context. The interested reader is referred to other reviews and articles discussing other microscopy techniques and AI/ML resources and approaches *via* the following references: [9,180–184].

2.2.1 Principles

Hierarchical architectures [185], commonly known as neural networks, are able to learn high-level representations and form the core of DL approaches. Raw data is used as input and passed through non-linear functional modules transforming them into increasingly abstract representations. The network then learns the complex functions required for specific tasks such as classification or object detection in an automated fashion [10]. The paradigm of DL is manifested in *how* those representations are generated; these are *entirely* learned from data using a general-purpose learning procedure [10].

The widespread success of DL started in 2012 when Krizhevsky *et al.* [186] submitted a convolutional neural network (CNN) for the ImageNet Large Scale Visual Recognition Challenge and achieved ground-breaking results. Their approach was quickly adopted by the computer vision community and found application in numerous areas such as consumer electronics, pedestrian detection, autonomous driving and computer-aided medical diagnosis [187,188]. Nowadays, deep neural networks are an inherent part of modern society.

CNNs can be assigned to solve different tasks and hence need to produce an appropriate output. A clear distinction is made to understand different applications properly. Consider a set of mutually exclusive classes. These written as $\boldsymbol{c} = [c_1, c_2, c_3, \dots, c_n]$, with $n \in \mathbb{N}$ denoting the number of classes considered. *Classification* is defined as the function f_{class} that assigns one or multiple instances of the real-valued vector $\boldsymbol{c}$ to a real-valued input image I_{in}:

$$f_{class}: I_{in}^{j \times k \times l} \rightarrow \boldsymbol{c}^{n \times 1} \tag{2.9}$$

with $j, k, l \in \mathbb{N}$ being the image dimensions, *i.e.* the image size in pixels and number of colour channels. *Classification* forms the foundation for more complex tasks such as *object detection* f_{od} and *semantic segmentation* f_{seg} [189]. In case of the former, the goal is not only to identify the instance with class c_n in a given image, but also to determine its bounding boxes [187]:

$$f_{od}: I_{in}^{j \times k \times l} \rightarrow D^{m \times 5} \tag{2.10}$$

where m refers to the number of detected instances. A given entry i in the matrix $D^{m \times 5}$ equals to the vector $\boldsymbol{d}_i = [c_n, x, y, w, h]$, with x, y being a pair of image coordinates and w, h the width of a bounding box, respectively. In *semantic segmentation*, every single pixel of a given image is classified. This segments the image into *meaningful* – *semantic* – regions [188]:

$$f_{seg}: I_{in}^{j \times k \times l} \rightarrow I_{out}^{j \times k \times l} \tag{2.11}$$

where the function f_{seg} projects an input image I_{in} into a segmented output image I_{out}. Other important computer vision applications in EM include image *inpainting* and *denoising*. *Inpainting* describes the interpolation of defective image parts without noticeable changes with respect to the remainder. *Denoising* on the other hand aims to remove additive noise from an image that is inevitably generated during the acquisition, compression and transmission of images [190,191].

This can be expressed based on Eq. 2.9 as:

$$f_{restore}: I_{in}^{j \times k \times l} \rightarrow I_{out}^{j \times k \times l} \tag{2.12}$$

in this case, the restoring function $f_{restore}$ projects a defective or noisy image I_{in} to the restored output image I_{out}. Figure 2.19 illustrates several exemplary micrographs and data that are typically recorded in EM.

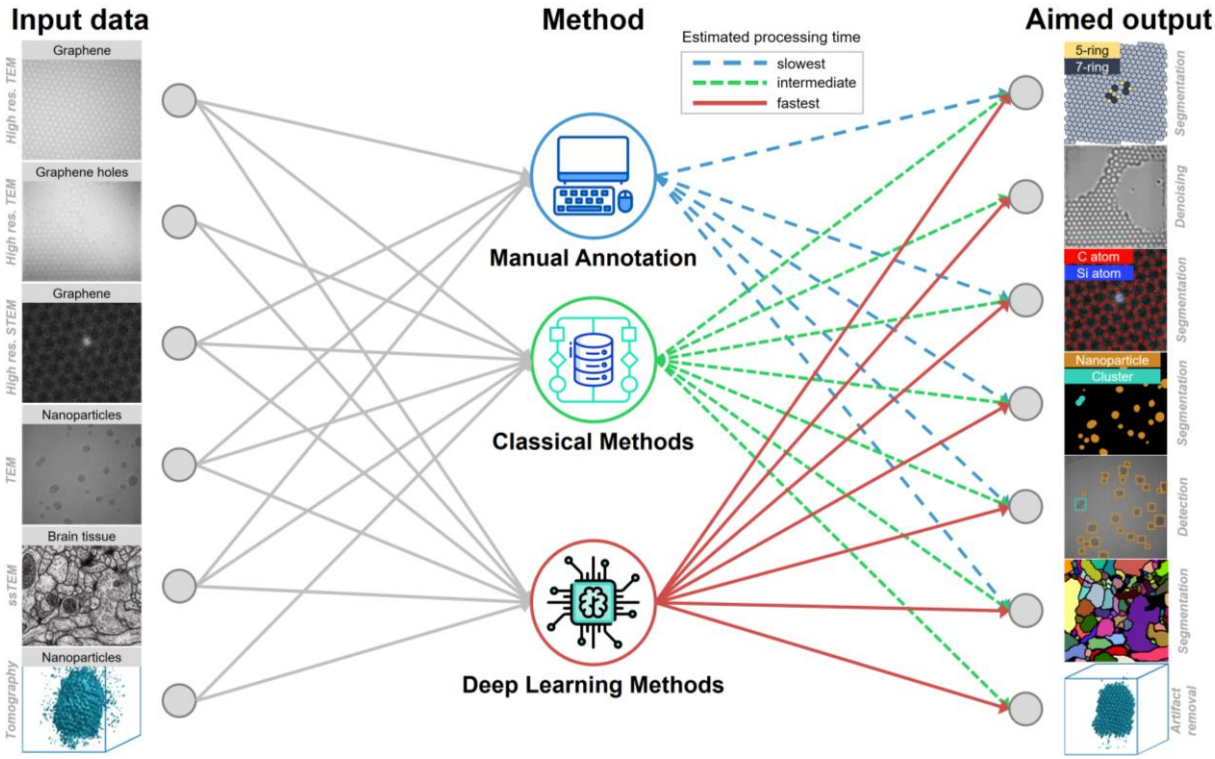


Figure 2.19: Exemplary EM data processed through different routes for an aimed output. Aimed outputs can be generated through different pathways, but the estimated processing time and performance of DL-based approaches outperforms other methods [11]. In the current context, classical methods refer to (semi-) automated algorithms and ML methods which differ depending on the given analysis task. The output of DL-based methods is mainly dependent on the training images used. Images of 'High res. STEM Graphene, 'ssTEM Brain Tissue' and 'Tomography Nanoparticles' with respective outputs obtained from [194,262,337], respectively. Figure and caption modified from [11].

For *inpainting* and *denoising* CNNs, Generative Adversarial Networks (GANs) are established as state-of-the art methods [190,192]. GAN architectures set a generative model G against a discriminative model D, where both are typically neural networks by themselves [193]. During training, the discriminator D learns to determine whether a specific sample was drawn from original input data or generated by the generative model G, while G tries to generate outputs mimicking the original samples such that D eventually makes mistakes. Consequently, the training procedure follows a mini-max optimisation problem where the saddle point is reached at a minimum for G and a maximum for D. When that point is attained, G is considered to have captured the real distribution of true examples [193]. While the above definitions shall serve as a guideline to understand DL applications in EM, a more in-depth evaluation of the foundations, training procedures and common architectures is given

in Chapter 3. The following highlights applications of DL-based approaches used in EM in more detail. Some of the CNNs described are encountered more often than others which is mainly dependent on the required outputs and the analysis task under consideration. They can however be intrinsically linked as many segmentation networks are encoder-decoder versions of their corresponding classification counterparts. For example, SegNet is an encoder-decoder version of VGG and the same holds true for ResUNet/ResNet and Fully Convolutional DenseNet/DenseNet. Table 2.1 [11] gives a comprehensive overview of most used architectures in EM and lists them according to their application areas. Note that not all of the examples provided in Table 2. [11] are discussed individually in the following and that a more comprehensive overview is given in Treder *et al.* [11], which is the basis of this section.

2.2.2 Applications in Physical Sciences

2.2.2.1 Atomically resolved STEM

The earliest examples of modern DL-based approaches in EM have been reported for semantic segmentation of atomically resolved STEM images [194–197] that combined fully convolutional neural networks (FCNs) [198–200] with sophisticated post-processing routines. Ziatdinov *et al.* [194] used training data entirely generated from simulations to train a CNN for automated defect segmentation and identification in graphene and $Mo_{1-x}W_xSe_2$ monolayers and demonstrated that training bases with limited *a priori* knowledge can be applied to identify a wider range of defect types in experimental images. In subsequent work, the dynamics and phase evolution in Mo-doped WS_2 during electron irradiation, as well as vacancy and Si impurity identification in graphene were evaluated with a two-step DL approach [195,196]. Identified defect structures were either categorised manually or by using an unsupervised Gaussian Mixture Model (GMM). Briefly, the latter can be described as a weighted sum that assumes all underlying data points are drawn from a mixture of finite Gaussian distributions with unknown parameters [201,202]. This enabled consecutive modelling of defect structures, the use of density function theory (DFT) and a comparison with experimental Scanning Tunneling Microscopy (STM) images of the same sample possible.

Table 2.1: Underlying neural network architectures with respective applications in EM. Superscripts I-IV refer to classification (I), object detection (II), semantic segmentation (III) and image denoising or inpainting (IV) where applicable.

Underlying Network	Application in EM
Multi-layer perceptron (MLP) [203]	**Materials analysis** Carbon nanotubes (CNTs): quality assessment[I] [204] and image segmentation[III] [205] Nanoparticles: size analysis in agglomerates[I] [206] Semiconductors: composition and thickness analysis[I] [207] **Phase retrieval** CBED analysis [208–210] Image wave reconstruction in holography[I] [211,212] **Techniques** EELS low-loss region analysis[I] [213] **Connectomics** Brain tissue segmentation[III] [214–216] Cryo-EM Particle picking for single-particle analysis (SPA)[II] [217,218]

Autoencoder (AE) [219]	**Material analysis** Transformations in graphene [220] **Cryo-EM** Particle picking for SPA[II] [221] Denoising of cryo-EM images[IV] [222]
LeNet [223]	**Material analysis** CNT chiral index determination[I] [224] Semiconductors: phase boundary analysis [225] **Cryo-EM** Particle picking for SPA[II] [226]
AlexNet [186]	**Material analysis** Metal alloys: Microstructure recognition[I] [227] **Diffraction** Position averaged convergent beam electron diffraction (PACBED) analysis[I] [228] **Cryo-EM** Particle picking for SPA[II, III] [229–231] **General applications** Micrograph labelling[I] [232]
VGG [233]	**Material analysis** Metal alloys: microstructure classification and segmentation[I, III] [234,235] Semicondutors: defect recognition in III–V/Si crystalline materials[I, III] [236] Classification of carbon nanomaterials[I] [237] **Cryo-EM** Pruning of false positives in SPA[I] [238]
DenseNet [239]	**Material analysis** Metal alloys: defect analysis[III] [240] **Techniques** Super-resolution reconstruction of TEM images [241] **Virus analysis** Virus image classification[I] [242]
ResNet [243]	**Material analysis** Nanoparticles: detection in SEM images[II] [244] **Cryo-EM** Part of cryo-EM analysis pipeline[II, IV] [245]
InceptionNet [246]	**Cryo-EM** Particle picking for SPA[III] [247]

	Cryo-ET Subdivision of tomograms into structurally homogeneous subsets[I] [248] **General applications** Categorization of EM images [249]
YOLO [250]	**Cryo-EM** Particle picking for SPA[II] [251,252] Ice-thickness determination[I, II] [253] Part of cryo-EM analysis pipeline[I, II] [254]
FCN [198]	**Cryo-EM** Particle picking for SPA [255]
SegNet [199]	**Material analysis** 2D materials: segmentation of graphene and MoWSe2 STEM images[III] [194] , phase evolution and dynamics in Mo-doped WS2[III] [195] , generation of graphene defect libraries[III] [196] **Phase retrieval** Prediction of structure and phase from diffraction patterns [256]
U-Net [200]	**Material analysis** 2D materials: analysis of beam-induced reactions of Si atoms on graphene[III, IV] [197], atomic structure analysis in graphene and nanoparticles[III] [257] Nanoparticles: segmentation for in situ experiments[III] [258,259], segmentation and defect identification[III] [260], column height determination in metallic nanoparticles[III] [261] **Tomography** Atomic-resolution tomography [262] Reconstruction of noisy 3D EDX tomograms [263] **Cryo-EM** Data pruning approaches for SPA [264] Image denoising[IV] [245,265–267] Resolution estimation and validation of cryoEM maps[I] [268] Atom structure determination from cryo-EM maps [269] **Virus analysis** Optimising network architectures for virus analysis[III] [270] **General applications** ADF-STEM image training library and image analysis app [271] Image-to-image inverse problems [272]
ResUNet [273]	**Material analysis** 2D materials: Analysis of WeSe2-2xTe2x point defects[III] [274] **Connectomics** Segmentation of brain tissue images[III] [275]

	Tomography Information recovery and artefact removal[IV] [276,277]
GAN [192]	**Techniques** Denoising of STEM images[IV] [278] STEM micrograph inpainting[IV] [279] SEM image-resolution enhancement [280]
Other	**Material analysis** Nanoparticles: analysis of nanoparticle diffusion behaviour in liquid-cell TEMI [281], detection and segmentation of nanoparticles[II, III] [282] Metal alloys: defect analysis[I, II] [283] **Diffraction** Diffraction pattern classification[I] [284] **Phase retrieval** Solving the ptychographic inversion problem [285] **Techniques** Analysis of manganese EELS spectra[I] [286] Denoising of EM micrographs[IV] [287] **Cryo-EM** Particle picking for SPA[II] [288] Structure modelling[I, II, III] [289–291] **Cryo-ET** Subcellular annotation and feature extraction[III] [292] Open-source platform for data analysis [293] **Connectomics** Brain tissue segmentation[III] [294–296] **Virus analysis** Segmentation of feline calicivirus images[III] [297], classification and detection of herpesvirus[I, II] [298,299], detection of SARS, MERS and HIV viruses[II] [300], image classification [301] **General applications** Entry-level platform for DL tasks[II, III, IV] [302]

2.2.2.2 *Atomically resolved TEM*

An early application to HRTEM data was demonstrated by Kirschner and Hillebrand [207] who trained a neural network to determine the composition and thickness of a semiconductor in 2000. Modern examples include works from Madsen *et al.* [257], who deployed a deep neural network to segment atom positions or columns in HRTEM images of graphene and gold nanoparticles respectively. This method has been followed up by Ragone *et al.* [261], who extended the same network architecture towards atom counting in nanoparticles. Performance improvements were achieved by using realistic Wulff constructed [303] (*i.e.* the minimum energy shape of a nanoparticle, determined by its facet-dependent free surface energy) nanoparticles in the training stage, and a regression-based output for column height determination (Figure 2.20). Finally, Groschner *et al.* [260] modified a U-Net for the segmentation of Au and CdSe nanoparticles in HRTEM images which were subsequently categorised with a random forest classifier (*i.e.* an ensemble of tree-structured classifiers with independent, identically distributed random vectors and where each classifier votes for a class, given an input x) [304] for stacking faults.

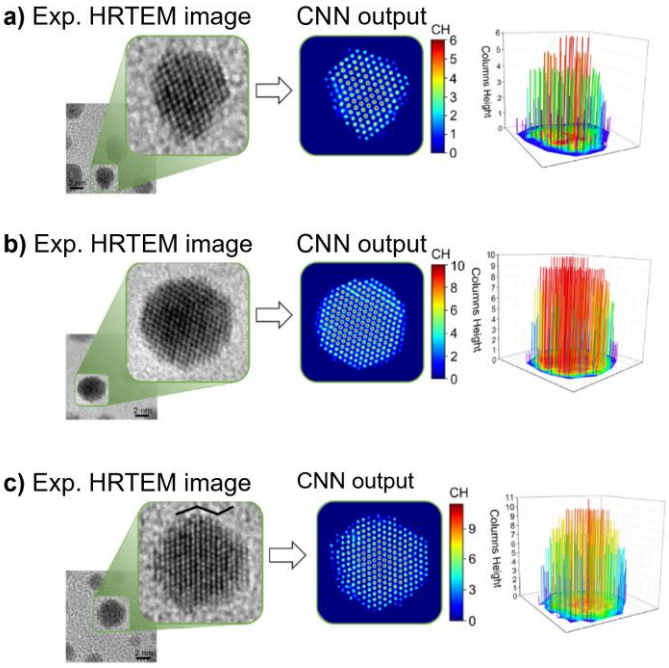


Figure 2.20: Column height predictions of the CNN used by Ragone *et al.* [261] for nanoparticles of the size **a)** 2.8 nm, **b)** 3.9 nm and **c)** 4 nm.

2.2.2.3 *Other (S)TEM Applications*

Several reported works have used DL-based approaches on lower resolution images of nanoparticles [258,259,281,282,305] and agglomerates [206], chiral indices in nanotubes [224], dislocations in metals and alloys [240,283], as well as phase boundaries [225]. Jamali *et al.* [281], Frei *et al.* [206] and Förster *et al.* [224] report examples of alternate pathways where *classification* rather than *segmentation* approaches were used. Using this approach new insights of nanoparticle diffusion in *in situ* liquid-cell

TEM [281] and advances towards rapid determination of CNT chiral indices [224] with improved statistical significance were obtained.

2.2.2.4 Scanning Electron Microscopy (SEM)

Image formation principles are fundamentally different for Scanning Electron Microscopy (SEM) as compared to (S)TEM, but it is equally viable to train neural networks for analysis purposes as demonstrated for carbon nanotubes (CNTs). Al-Khedher *et al.* [204] used an MLP to characterise the curvature and alignment of CNTs in SEM images. A similar MLP was used by Trujillo *et al.* [205], but used for *semantic segmentation* of SEM/TEM data instead. Modarres *et al.* [249] opted for a more advanced architecture alongside a transfer learning routine on a diverse set of SEM data for an accurate *classification* of nanostructures. Transfer learning was further used to train a VGG-16 network for carbon nanomaterial classification [237].

In a commercial context, SEM is often the characterisation method of choice to characterise metal alloys by Secondary Electrons (SE), Backscattered Electrons (BSE) or Electron Backscatter Diffraction (EBSD) and carried out by human experts. However, Azimi *et al.* [235] showed that a segmentation analysis of steel microstructures could be carried without sacrificing accuracy by combining a VGG-16 network with an FCN. DeCost *et al.* [234] combined the features found in the intermediate layers of a VGG network with unsupervised learning techniques to obtain a classification routine for ultrahigh carbon steel with superior performance in comparison with non-DL based methods. Lastly, it was shown that it is possible to use synthetic datasets for training [236,244] a RetinaNet [306] with a pre-trained ResNet-50 backbone for *object detection* of nanoparticles in SEM images.

2.2.2.5 Spectroscopy

One of the key capabilities of electron microscopes is the measurement of chemical properties with energy-dispersive X-ray analysis [263] (EDX) or electron energy-loss spectroscopy (EELS). However, both have specific limitations. The EEL spectrometer calibration can change over time due to zero-loss peak (ZLP) energy drift, precluding comparison of absolute peak positions [286]. Chatzidakis and Botton [286] demonstrated that it is possible to overcome this limitation by implementing DL-based

calibration invariant EELS spectrum analysis. Compared to other approaches that harness publicly available data for training, the authors exploited in-house stored EELS spectra of Mn^{2+}, Mn^{3+} and Mn^{4+} acquired under different experimental conditions. Results showed that the neural network was able to identify spectral peaks merely by their shape whilst position invariance was maintained. The method was further demonstrated on digitised spectra taken from other publications and still generated the correct output.

Roest *et al.* [213] focused on the ultra-low loss region of EELS spectra which contained overlapped contributions from both the ZLP and the low-loss inelastic interactions. In particular, the authors aimed to subtract the ZLP from the spectrum by precise parametrisation. A Monte Carlo replica method served as the foundation for the construction of a probability distribution in the experimental data space, which was subsequently propagated by a neural network to a probability density space of ZLP models. A Neural Network Parton Distribution Function approach [308], originally developed to study quark and gluon substructures in the proton [309] was subsequently used. This procedure was able to disentangle the ZLP contribution from low-loss inelastic scattering and was applied to WS_2 nanoflowers characterised by a 2H/3R polytypic crystalline structure where a bandgap distribution of $E_{BG} \cong 1.6^{+0.3}_{-0.2}$ eV with an indirect bandgap preference was verified consistent with previous studies.

2.2.2.6 Tomography

An important contribution to tomography was provided by Ding and co-workers [276], who proposed a joint two-step model to inpaint missing-wedge information in sinograms to reduce residual artefacts in reconstructed tomograms based on GANs [192]. Importantly, the authors demonstrated superior performance when compared to established methods such as weighted back projection (WBP), simultaneous algebraic reconstruction technique (SART) or total variation minimisation (TVM), hence underlining the potential of improving analysis approaches using DL. The follow-up study [277] reported the successful removal of missing wedge artefacts from an atomic-resolution tomogram, improving the resolution from ~ 1.42 Å to 0.71 Å. In this case however, a single GAN was used for artefact removal, whereby a U-Net++ [310] served as the generator G. Those results are remarkable in that training was carried out using very generic open training datasets including Massachusetts General

Hospital (MGH), the National Biomedical Imaging Archive (NBIA), the ImageNet library, as well as simulated images mimicking cross-sectional images of faceted and rounded nanocrystals [277]. Lee *et al.* [262] trained a 3D U-Net [311] on image simulations for tomogram artefact removal and achieved atomic structure measurements with a precision of 15 pm. The largest improvements in the tomograms were achieved at the surface of a 4 nm Pt nanoparticle, which was attributed to the correct identification of atoms that were otherwise obscured due to missing wedge artefacts. accurate reconstruction of the nanoparticle enabled a detailed strain analysis which confirmed theoretical predictions of compressive stresses along ⟨100⟩ and tensile stresses along ⟨111⟩ directions (Figure 2.21).

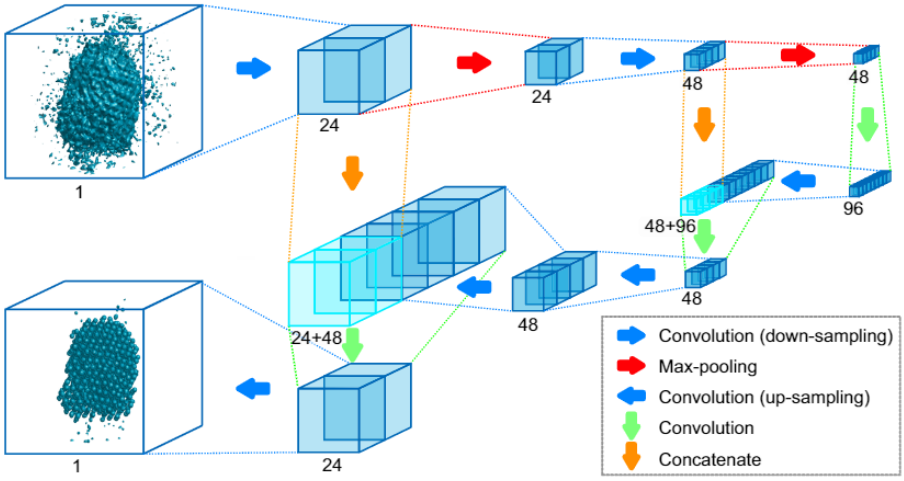


Figure 2.21: Architecture of the deep learning augmentation used by Lee *et al.* [262].The deep learning augmentation follows a 3D U-Net architecture. The boxes represent feature maps while the number of channels is shown below each box.

2.2.2.7 Image Denoising and Inpainting

Analogous to *semantic segmentation*, it is possible to use pairs of noisy (training) and clean (target) images to train a neural network for *denoising* [191]. A fundamentally different strategy was presented by Lehtinen *et al.* [313], who showed that clean images are not necessarily required: Using the L_2- loss for neural network training has the trivial, but crucial advantage in that the calculated point estimate will remain unchanged even if the training targets are replaced with random numbers. However, this only holds true as long as their expected value matches the expected value of the replaced training targets. This has the consequence that both the training *and* target images can be drawn from corrupted images – *i.e.* a network can learn how to remove photon noise given only pairs of noisy images. Since large datasets required for training neural networks are usually lacking such unsupervised approaches [314–318] are of particular interest.

Wang *et al.* [278] have implemented a GAN [192]; Noise2Atom [278] which applies a discriminator *D* to differentiate between images composed of pure Gaussian peaks and images translated from noisy experimental images into denoised ones. Performance was benchmarked against established denoising methods including their own work based on Multiresolution Deep Convolutional Neural Network (MCNN) [272]. Compared to Noise2Atom, MCNN was trained on simulated images and the authors emphasise that it is difficult to include all potential noise levels encountered in STEM imaging and hence to train the network for all potential scenarios.

Ede and Beanland have published work on both image denoising [287] and inpainting [279], with training conducted by corrupting images from their published database [319] with Poisson noise or binary masks resembling partial scans respectively. An architecture similar to DeepLab3 [320,321] was used and outperformed eight other denoising methods [287]. Considering their inpainting work, one particular aspect is notable in the context of EM. The low SNR limitation encountered for beam-sensitive materials can be interpreted inversely: Sparse sampling reduces the electron dose at the sample significantly and missing information can be inferred by algorithms that aim for minimal information loss such as natural neighbour interpolation or exemplar-based inpainting [322]. This is similar to an approach that has been already demonstrated using compressive sensing [323,324], but which requires

a fast deflection system to be applied for STEM imaging. In contrast, Ede and Beanland [279] only assumed patterns that are likely to be realised with standard deflection systems, and secondly used STEM images with different characteristics for training to ensure generalisation. Results from a multiscale GAN confirmed significant dose reductions with a factor 17.9 with a 3.8 % test error and 87.0 with a test error of 6.2 % respectively. The limitation of this approach are reconstructions of regions with unpredictable features not covered by the electron beam, which makes application for dynamics difficult. Examples are shown in Figure 2.22.

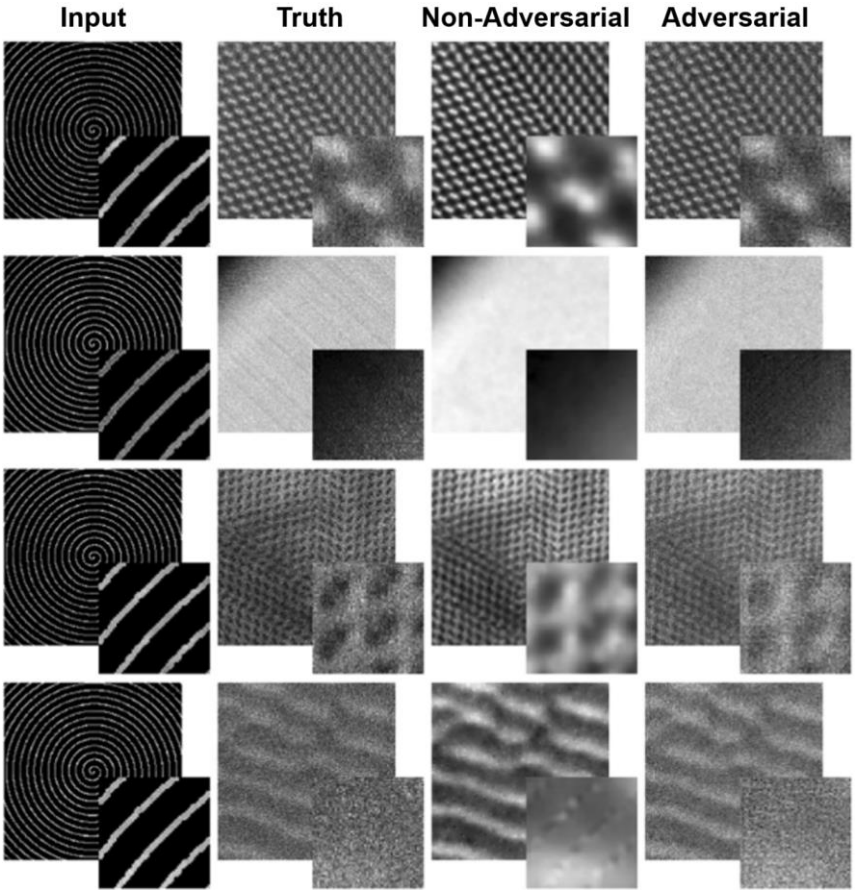


Figure 2.22: Blurred spiral scan inputs (1/20 pixel coverage) as well as adversarial and non-adversarial image completions for a 512 x 512 test set as described in Ede and Beanland [279]. While 'Non-Adversarial' refers to outputs generated from the generator network trained using an auxiliary trainer for mean squared error minimisation exclusively, 'Adversarial' refers to outputs obtained from the generator network after being fine-tuned as part of a GAN (see 2.2.1) to produce more realistic images. Further details are given in Ede and Beanland [279].

2.2.2.8 General EM applications

Besides dedicated solutions, an increased interest in applications with more general purposes is notable in the field. This includes metadata annotation, for which Weber *et al.* [232] used a transfer learning approach on an AlexNet-based network to transfer manually labelled images to the entire data store of the National Center of Electron Microscopy (NCEM). Results showed a classification accuracy of *ca.* 80 % and point to a potential strategy to increase efficiency and simplicity in data searching in big data projects.

As already noted, it is difficult to generate training data for computer vision tasks in EM independently, suggesting high interest in publicly available resources for the successful application of DL methods. Ede posted used datasets [319] alongside network architectures and interactive visualisations for public use. The research group of Xin recently published a desktop app with a graphical user interface for ADF-STEM image analysis [271] that included a training library consisting of a diverse set of simulated images, and the U-Net based FCN AtomSegNet that could be trained for various tasks such as super-resolution and denoising.

An online solution has been provided by von Chamier and co-workers [302], effectively making DL approaches accessible to the wider microscopy community. ZeroCostDL4Mic is an entry level DL deployment platform based on Google Colaboratory (Colab) allowing users to perform computer vision tasks such as *object detection*, *semantic segmentation* or *image restoration* without in-depth knowledge. As noted by the authors, this environment could be useful for prototyping or deployment of small to medium-sized projects using state-of-the-art networks. Their work only discussed biomedical images, but the extension to the physical sciences is obvious.

Lastly, even areas such as instrument alignment have been subject to DL-based approaches. Rotunno *et al.* [325] reported a TEM orbital angular momentum sorter using a DL-based misalignment measurement.

2.2.3 Applications in Life Sciences

The ‘resolution revolution’ [326] was mainly driven by the introduction of direct-electron detectors (see 2.1.7) and marked the beginning of a new era of structural biology. Single-particle cryogenic EM (SPA cryo-EM) has emerged as a preferred method for structural determinations [327,328], but corresponding specimen preparation and data analysis remain complex and time-consuming. Unlike conventional (S)TEM imaging, cryo samples of bio molecules are usually dispersed along random orientations in vitreous ice for SPA. Generally, high-resolution 3D reconstructions are obtained by averaging classes, defined by their projection angles [329]. Thousands of individual particles are commonly required to achieve a sufficient SNR and to cover all necessary projection angles, ultimately leading to a successful reconstruction. The initial step – *i.e.* the selection of suitable molecules – is usually called ‘particle picking’ and often performed in a manual or semi-automated fashion [328,330]. However, it is also one of the first tasks for which DL-based approaches in cryo-EM are applied [217,218,331]. Nowadays, DL-based applications are an area of intense activity in the field of cryo-EM, offering multiple solutions for particle picking [221,226,229,247,251,252,266,288], alongside some critical steps [330] in the 3D reconstruction [238,254,264,268,332,333], for solving heterogeneous cryo-EM structures [334], for denoising of cryo-EM images [221,222,245,265,266], for cryo-electron tomography [248,292,293] and even for frameworks aiming for automation spanning from raw data input to the final reconstruction of the molecule under consideration in short time frames [245,254].

Once a reconstruction is obtained, the final step requires the correct assignment of an atomic structure which is of fundamental importance to understand underlying mechanisms, molecular dynamics, structure comparison and associated applications such as drug design. The procedure and context of this step is reviewed by Malhorta *et al.* [335] and Kim *et al.* [336]. Reconsidering DL methods, 3D data instead of images have to be considered for structure modelling. The aim is to predict the back-bone structure with Cα atoms playing a central role. This is achieved in several studies [269,289–291] reporting results that outperformed state-of-the-art prediction methods. The above applications are summarised in the overview presented in Figure 2.23. A related area of research to both, cryo-EM, and physical sciences is the field of virus recognition and analysis where DL approaches have equally been

applied [242,270,297–301]. Lastly, DL-based approaches have had crucial impact in neuroscience, where it is of critical importance to visualise and to understand internal neuronal connections. A new branch of biotechnology – 'connectomics' – has emerged that combines advanced sample preparation, serial block-face SEM (SBEM), big data, automated image segmentation and 3D reconstruction. The state of the art of this technology is the sectioning of brain tissue volumes into thousands of thin slices using SBEM imaging.

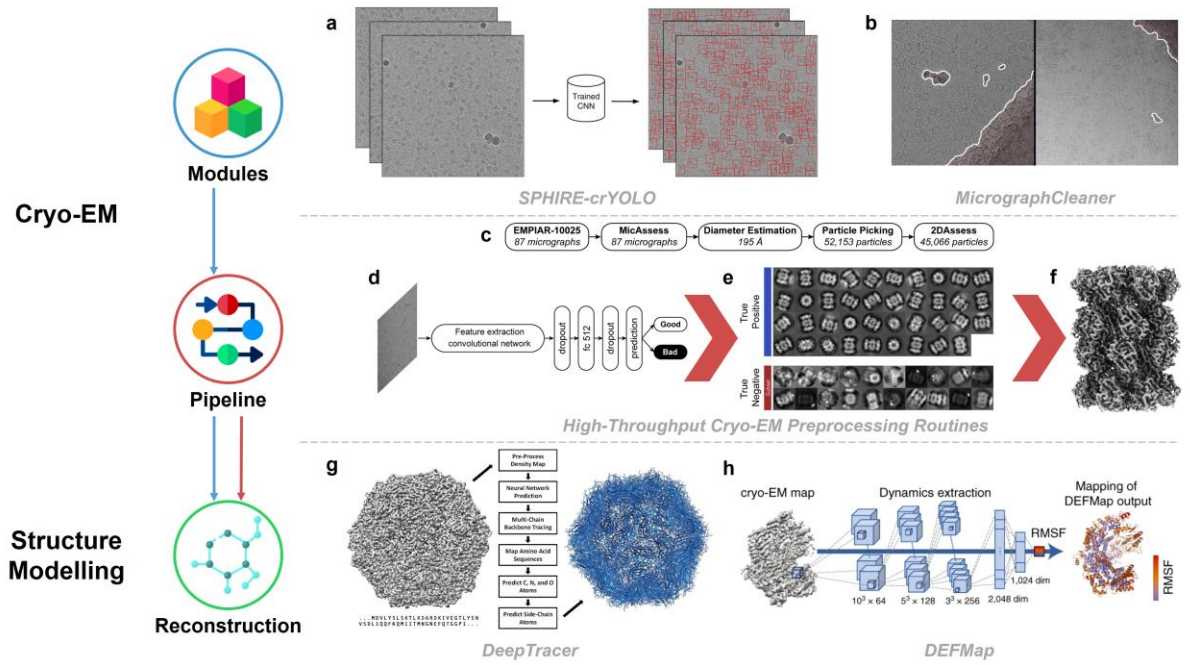


Figure 2.23: Selected examples of deep neural networks applied in the fields of cryo-EM and structural modelling. Modules describe neural networks that were trained for a specific task, *e.g.* **a)** SPHIRE-crYOLO (modified from Fig. 1 of [251]), **b)** MicrographCleaner (modified from Fig. 1 of [264]) for the reduction of false positives. Pipeline refers to applications that employ multiple steps with neural networks for automated multi-stage processing of datasets. **c)** Workflow of the T20S proteasome (EMPIAR-10025) of the pipeline introduced by Li *et al.* [254]. The particle picking step in this pipeline is conducted by the 'general' crYOLO network. **d)** Architecture of MicAssess, a neural network for assessing motion corrected cryo-EM micrographs for their quality. **e)** 2D Assess, a network used for 2D classification into 'good' and 'bad' obtained averages.**f)** 3D electron density volume output obtained by the pipeline. Reconstruction refers to applications that use neural networks to reconstruct structures of dynamics associated with atomic fluctuations from cryo-EM electron density maps. **g)** Model determination pipeline of DeepTracer (modified from Fig. 1 of [269]). **h)** Workflow of the prediction stage in DEFMap [291]. Figure and description from Treder *et al.* [11].

It is at the *semantic segmentation* stage where neural networks can be readily applied and were introduced as early as 2007 [214,215], but it took another five years until DL approaches were recognised as state-of-the-art method [337]. Although the respective approaches were yet not fully automated successful, reconstructions of the mouse retina [216] and a fly brain [338] triggered high interest along with advancements in ML methods [275,294–296] and innovative data annotation through cloud-based environments and crowdsourcing [216,295,337]. At this point, the interested

reader is directed to an article by Ishii *et al.* [339], who recently summarised CNN-based segmentation methods in connectomics, as well as to a more thorough discussion of applications in *life* sciences by Treder *et al.* [11].

Before providing the theoretical concepts required for the full understanding of this work (Chapter 3), the relevance of nanoparticle systems, their various applications, contemporary research and deactivation mechanisms are highlighted briefly. This is of particular relevance for the interpretation of results presented in Chapter 5 and Chapter 7 and thus emphasis on gold (Au) and palladium (Pd) nanoparticles is given.

2.3 Materials for Heterogeneous Catalysis

2.3.1 Properties of Nanoparticle Systems

Numerous industrial sectors are highly dependent on heterogeneous catalyst systems, which reinforces the need for their optimisation, especially with respect to a more environmentally friendly future. In heterogeneous catalysis, reactants are usually in gaseous or liquid phase, while the catalyst generally is mostly solid [5] and systems such as metal nanoparticles supported on carbon or oxides are often applied [340]. Other important (nano) material systems commonly used include zeolites, carbon nanotubes (CNTs), silicoaluminophosphates, *etc.* [341–345], and are equally of high relevance to catalytic applications and research. While this section is restricted to supported metal nanoparticles, references to comprehensive reviews discussing other heterogeneous catalyst systems in more detail are provided in addition: [4,341–347].

Nanoparticles are defined as 'particle of any shape with dimensions in the $1 \cdot 10^{-9}$ and $1 \cdot 10^{-7}$ m range' [348], whereby this definition includes tubes and fibres. The 100 nm (or $1 \cdot 10^{-7}$ m) limit is imposed as novel properties differing from their bulk counterparts emerge at this length scale [348]. These size-induced effects include the quantum size effect [349], the reduction of the melting temperature [350], and an increasing contribution of surface atom properties with decreasing nanoparticle size due to an increasing surface-to-volume ratio. Particularly the latter is crucial: In simplified terms, heterogeneous catalysis proceeds by first adsorbing the reactants on the catalyst surface, followed by a chemical reaction of the absorbed reactants, and finally concluded by a desorption of the products [351]. A high surface-to-volume ratio hence gives rise to a large number of active sites for catalytic reactions to occur [352].

Catalytic performance is not only determined by its activity after all. Instead, it is given by taking additional metrics including catalyst stability, selectivity and regenerability (deactivation) into account, which themselves depend on interacting factors such as the synthesis method, porosity, metal-support interaction, crystal structure, *etc.* [5]. Ongoing efforts in catalyst optimisation created numerous preparation routes which allow for the control of size and shape of dispersed nanoparticles [2], and hence also of their performance. Categorisations may follow distinction into bottom-up and top-down

[353], or chemical and physical preparation routes [354,355], whereby examples for the chemical preparation route include co-precipitation, electrochemical reduction, micro emulsions or chemical vapour deposition [354,355]. Approaches such as microwave irradiation, plasma reduction, or pulsed laser ablation are used in physical preparation [354,355]. A more thorough introduction of common nanoparticle powder preparation techniques would go beyond the scope of this section and is therefore provided as references [353–355].

2.3.2 Gold (Au) Nanoparticles

Gold nanoparticles have been used for thousands of years [356], contrasting to other nanoparticle systems. In modern research, Au nanoparticles were initially considered to be chemically inert and hence to exhibit poor catalytic properties, but it was realised that dispersion of Au nanoparticles of size $< 10\,\mathrm{nm}$ on a suitable support creates a highly active catalyst applicable to a wide range of hydrogenation and oxidation reactions, as for instance the oxidation of alcohols and alkenes or the hydrochlorination of ethyne [354]. Moreover, Au nanoparticles are amongst the most stable and exhibit excellent biocompatibility and low toxicity, making them suitable for bioimaging, medical diagnoses and even for treatment of cancer [357]. Liang *et al.* [358] studied TiO_2 supported Au nanoparticles for alkynes using morpholine as a co-catalyst in basic conditions and demonstrated recyclability of the catalyst by simple filtration[360]. Sharma *et al.* [359] reviewed the oxidation of alcohols using gold nanoparticles and thoroughly discussed various support materials, alongside oxidation mechanisms, the effect of the particle size, *etc.* He *et al.* [360] used atomic resolution ETEM (see Section 2.1.4) to investigate size dependent structure dynamics of ceria (CeO_2) supported Au nanoparticles in CO oxidation at room temperature. Amongst other insights, the authors report an interplay between Au nanoparticles and adsorbed CO molecules, whereby results suggested the generation of dynamic ‘single atoms’ on the surface in sufficiently small clusters [360]. Figure 2.24 shows exemplary structural changes of ceria supported Au nanoparticles as observed by the authors.

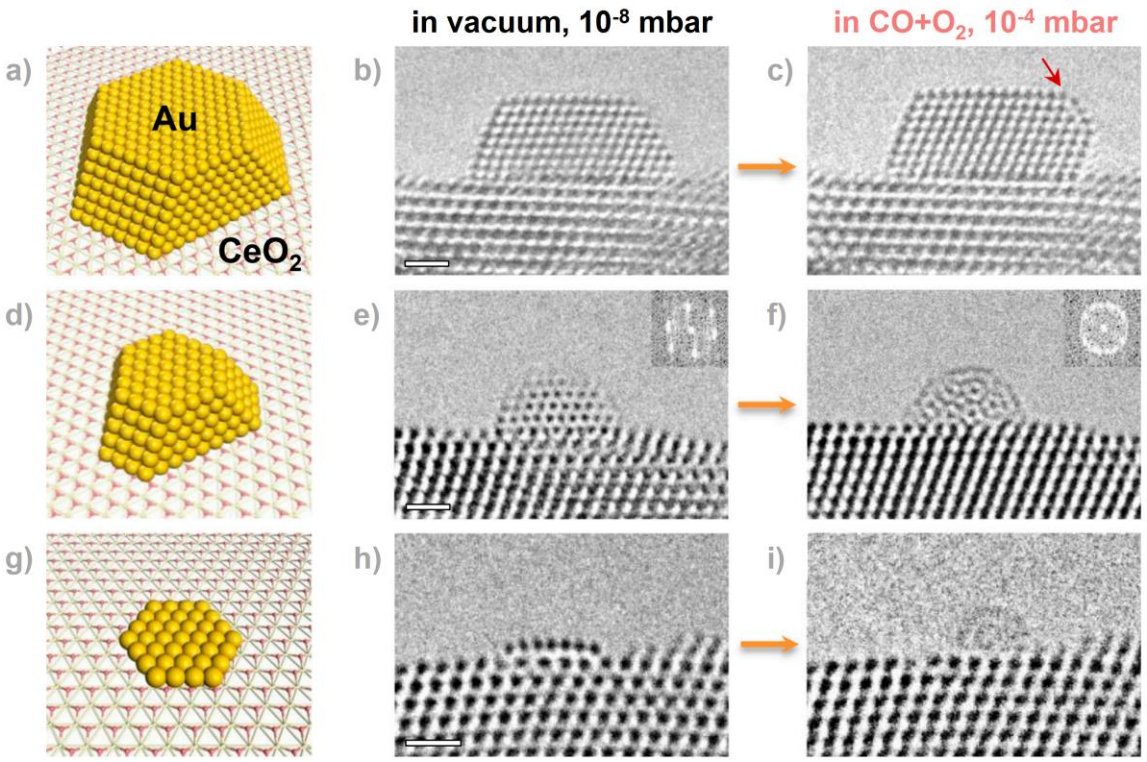


Figure 2.24: Dynamic structural changes of various Au nanostructures supported on CeO2(111) upon exposing to the reactant gases. **a**-**c** Model (**a**) and HRTEM images (**b** and **c**) of a 4 nm Au nanoparticle with *ca.* 1,000 atoms; the arrow in **c** indicates the Au(100) surface reconstruction. (**d**-**f**) The 2 nm Au nanoparticle with *ca.* 100-200 atoms; the Insets show corresponding fast Fourier transforms of the AuNP. (**g**-**i**) The *ca.* 2 nm Au single layer with ~ 40 atoms. (Scale bars: **b**, **e** and **h**: 1 nm; also apply to **c**,**f** and **i** respectively. Figure and description adapted from [360].

The Au/Ge system introduced in Chapter 4 is within core interest of this work. It shall be noted, that the sample considered is rather used as a calibration standard in (S)TEM imaging instead as a heterogeneous catalyst itself, but can be understood as a blueprint to other Au nanoparticle heterogeneous catalyst systems, for which the proposed method of this work should be applicable as well. Moreover, other DL-based approaches [258] conducted on similar specimen enable a more direct comparability, which is the reason why this sample was ultimately chosen.

2.3.3 Palladium (Pd) Nanoparticles

Palladium is among the most varied used metals in heterogeneous catalysis, especially for C-C coupling reactions (Heck, Suzuki, Sonogashira) [354,361]. Other exemplary reactions include the synthesis of vitamin intermediates or oxidation of alcohols, bezenes and alkenes[354]. More recent, exemplary studies on Pd nanoparticle systems are given by Rivera-Cárcamo *et al.* [362], who report a cooperative catalysis operating between carbon supported single atoms of Pd and Pd nanoparticles for alkene hydrogenation and isomeration. Ament *et al.*[363] proposed the intercalation of positively charged Pd

nanoparticles of 3.8 nm between negatively charged nanosheets of L-titanate as an alternative catalyst system architecture, which was shown to be highly active on oxidation of CO. Cao and co-workers [364] reviewed recent advances in metal-support interaction of Pd-based catalysts for automobile emission control. Pd is widely used as catalyst in three-way catalytic converters where the following core reactions take place[364]:

Oxidation reactions $$C_mH_n + (m + n/4)O_2 \rightarrow mCO_2 + n/2\ H_2O \quad (2.13)$$

$$2CO + O_2 \rightarrow 2CO_2 \quad (2.14)$$

$$2H_2 + O_2 \rightarrow 2H_2O \quad (2.15)$$

NO reduction reactions $$C_mH_n + NO \rightarrow N_2(N_2O, NH_3) + H_2O + CO_2 \quad (2.16)$$

$$CO + NO \rightarrow CO_2 + N_2(N_2O) \quad (2.17)$$

$$H_2 + NO \rightarrow N_2(N_2O, NH_3) + H_2O \quad (2.18)$$

Hydrogen production $$CO + H_2O \rightarrow CO_2 + H_2 \quad (2.19)$$

$$C_mH_n + 2mH_2O \rightarrow mCO_2 + (2m + n/2)H_2 \quad (2.20)$$

The works introduced for the Au and Pd systems include only several examples and yet clearly indicate the importance of ongoing research for several fields. To not only understand the catalyst systems in various conditions and at different time resolutions, but also to streamline the optimisation process further, more reliant and faster analytical tools are indispensable. While other characterisation techniques, such as UV/Raman/IR/X-ray spectroscopy, magnetic resonance methods including NMR and MRI, and other microscopy techniques can be used for *in situ* characterisation of catalyst materials [365] , *in situ* (S)TEM gives *simultaneous* access to both physical and chemical domain at high temporal resolution (consider examples in Section 2.1.3) [115], especially in HRTEM. Conclusively, the development of advanced analysis methods to characterise heterogeneous catalyst materials on the nanometre scale can be stated to be of particular importance for this imaging technique and is therefore pursued in the form of a DL-based approach in this work. This is also within the interest for a deeper understanding of deactivation mechanisms.

2.3.4 Deactivation Mechanisms

During operation, nanoparticle catalyst systems often suffer from deactivation originating from metal leaching and nanoparticle sintering at high temperatures, leading to loss of crucially important active surfaces. Once sintered, regeneration of catalyst nanoparticles requires costly separation from the support and subsequent reloading. Preventing the sintering process altogether is therefore a subject of widespread interest in catalysis research [366]. Two main mechanisms have been proposed in literature [100–102,366] for the sintering mechanism in nanoparticle systems:

(i) Ostwald ripening [367,368]. From a more general perspective, first-order phase transformations result in a two-phase mixture whereby the second phase ends up being dispersed in the first, introducing a large interfacial surface area. The system subsequently aims to reduce its total energy by increasing the size scale of the second phase and thus decreasing the total interfacial surface area which is called Ostwald ripening [367,368]. In nanoparticle systems, Ostwald ripening consists of individual metal atoms leaving one nanoparticle, diffusion of the metal atom on the support, and subsequent adhesion to a different metal nanoparticle. Since larger nanoparticles are energetically more favourable for the system due to lower surface energy, this process results in the growth of larger nanoparticles on the expense of small ones [100,340]. A more encompassing treatment on Ostwald ripening in general is given in reviews by Voorhees [367] and van Westen and Groot [368], while Hansen *et al.* [100], Campbell [340], and Wang *et al.* [366] focus of nanoparticle systems.

(ii) Particle coalescence. In this case, entire metal nanoparticles can diffuse across the support in Brownian motion. If two (or more) nanoparticles come intro close proximity, coalescence occurs [100,340].

Both mechanisms have been studied using *in situ* (S)TEM imaging with examples given in Section 2.1.3.1 and 2.1.3.2. Note that Hansen *et al.* [100] propose a three-stage sintering model for nanoparticle systems in which either of the mechanisms can be more apparent.

Lastly, deactivation can also occur due sublimation of nanoparticles due to the melting point depression in nanoparticles (see 2.3.1) [350]. This was studied by Asoro *et al.* [369], who conducted *in situ* HRTEM studies on Ag nanoparticles with diameters between 5 and 50 nm. Consistent with the Kelvin equation, a decrease of sublimation temperature with decreasing nanoparticle size was observed.

2.4 Chapter Summary

In this literature review, the two overarching fields of time-resolved imaging and applications of DL in EM were discussed first, followed by an introduction of materials for heterogeneous catalysis. For time-resolved imaging, important aspects of the history of EM have been summarised together with relevant imaging techniques. For both, key works have been highlighted and discussed. Lastly, direct-electron detectors alongside their quantitative characterisation metrics have been explained and the emerging challenge of data processing in time-resolved imaging has been emphasised.

Methods arising from DL can provide potential solutions to these challenges. In order to evaluate the applicability and performance thereof, various reports of DL-methods in the field of *physical* and *life* sciences of EM were described. It can now be stated that DL methods are a highly impactful, data-driven and general approach that can be tailored for many problems in EM and it can be expected that the number of published reports will increase in the future.

Finally, the importance of heterogeneous catalyst systems and their applications were highlighted with particular emphasis on metal nanoparticle systems. As time-resolved (S)TEM imaging provides one of the most important analytical tools to this research field, the development of more reliant and faster analytical tools is of core importance, for which DL-based methods can be considered a key enabler.

3 Theoretical Considerations

This chapter outlines the essential theoretical concepts relevant to the work presented, with more specific considerations within other chapters. Since this work mainly presents a new *application* of DL for the interpretation of HRTEM image analysis, the following background is well-known and publicly available. More detailed discussions are given in the references used herein but can also be found in many other textbooks and in the primary literature.

3.1 Electron Emission

In this section one essential part of all electron microscopes is described briefly: electron sources. In conjunction with Section 2.1, different types of electron sources accessing distinctive time-resolution domains are considered: i) continuous electron sources [370–372] and ii) pulsed electron sources [139,148,373,374].

3.1.1 Continuous Electron Emission

The electrons used for image formation originate from the electron source. It emits electrons into the vacuum of the electron microscope and is required to produce beams of high brightness, spatial and temporal coherence [370], which can be realised by two types of electron emission, namely *thermionic* and *field emission* [372].

3.1.1.1 Thermionic Emission

Consider electrons in a conducting material. The work function ϕ_w defines the energy barrier that needs to be overcome for electron emission, whereas the potential energy $V(z)$ of an electron at a distance z normal from the material surface can be calculated by using a mirror charge with opposite sign behind the surface.

$$V(z) = \phi_w - \frac{e^2}{16\pi\epsilon_0}\frac{1}{z} - e|\boldsymbol{E}|z \tag{3.1}$$

where ϵ_0 is the vacuum permittivity, $V = -e|\boldsymbol{E}|z$ refers to the superposed potential energy of the mirror charge in an applied electric field $\boldsymbol{E}$ and e is the electron charge. For thermionic emission to occur, electron sources are be heated leading to a broadening of the Fermi distribution $f(E)$ at the Fermi level E_F with emitted electrons gaining sufficient energy to overcome ϕ_w at the tail of $f(E)$. The emitted current density j_c, a key characterisation metric, can be described by Richardson's law [370] as:

$$j_c = AT_c^2 \exp(-\frac{\phi_w}{kT_c}) \tag{3.2}$$

in which $k = 1.38 \cdot 10^{-23}$ J/K is Boltzmann's constant, T_c the cathode material temperature and $A \cong 12 \cdot 10^5$ A/K^2 m^2 is a material dependent constant. While the typically used tungsten hairpin would be heated by the flow of current through the material itself, LaB_6 sources rely on thermal contact to a low-workfunction emitter material such as tungsten as supporting material.

3.1.1.2 Schottky Emission

Increasing the strength of the electric field $\boldsymbol{E}$ at the tip of the electron source results in an overlap of potential energies (see Eq. 3.1), causing an effective decrease of the work function $\Delta\phi_w$, also known as the Schottky effect [370,372].

The effective work function $\phi_{w'}$ is obtained as:

$$\phi_{w'} = \phi_w - \Delta\phi_w = \phi_w - e\sqrt{\frac{e|\boldsymbol{E}|}{4\pi\epsilon_0}} \tag{3.3}$$

where field strengths on the order of 10^8 V/m are typical and ϵ_0 refers to the dielectric constant. Note that this decrease is different from field-emission (see below) as electrons must still overcome the work function $\phi_{w'}$.

3.1.1.3 Field emission

High electric fields $\boldsymbol{E}$ not only reduce the work function ϕ_w, but also decrease the potential barrier width b at the matter-vacuum boundary. Assuming a typical tip radius of $r \leq 0.1$ µm and a field strength $|\boldsymbol{E}| \geq 10^9$ V/m, b becomes less than 10 nm, which is small enough for electrons to be emitted *via*

quantum-mechanical tunnelling [370]. In this scenario, the source does not need to be heated if ultrahigh vacuum is applied [371]. The corresponding current density is estimated by the Fowler-Nordheim formula [370]:

$$j = \frac{k_1 |\boldsymbol{E}|^2}{\phi_w} \exp\left(-\frac{k_2 \phi_w^{3/2}}{|\boldsymbol{E}|} \right) \tag{3.4}$$

with the constants k_1 and k_2 being only weakly depended on $|\boldsymbol{E}|$ and ϕ_w. Table 3.1 compares core properties of commonly used electron sources [372], while Figure 3.1 shows plots of the potential barriers for different types of electron emission [370].

Table 3.1: Direct characteristics comparison of commonly used electron sources (not effective for electron gun).

Characteristic	Thermionic W hairpin	Thermionic LaB_6	Schottky FEG (ZrO on W)	Cold-FEG (W single crystal)
ϕ (eV)	4.5	2.7	2.8	4.5
J_c (A/m^2)	~ 10^4	~ 10^6	~ 10^7	~ 10^9
α (rad)	~ 10^{-2}	~ 10^{-2}	~ 10^{-3}	~ 10^{-4}
d (µm)[a]	~ 50	~ 10	~ 0.02-0.03	~ 0.01
Brightness ($A/(m^2sr)$)[b]	~ 10^9	~ 10^{10}	~ 10^{11}-10^{12}	~ 10^{12}-10^{13}
Temperature (K)	2700	1800	1800	300
Energy spread (eV)	1-3[c]	0.6-2[c]	0.6-1.5[c]	0.3
Vacuum (Pa)	~ 10^{-2}	~ 10^{-4}	~ 10^{-7}	~ 10^{-8}

[a] Specific values dependent on tip configuration.
[b] Values given for 100 keV electrons and scale with accelerating voltage. Reduced brightness given by B/V, with V referring to the accelerating voltage in V.
[c] Energy spread dependent on operation conditions, such as Wehnelt bias, extraction voltage of the Schottky FEG or the filament heating current.

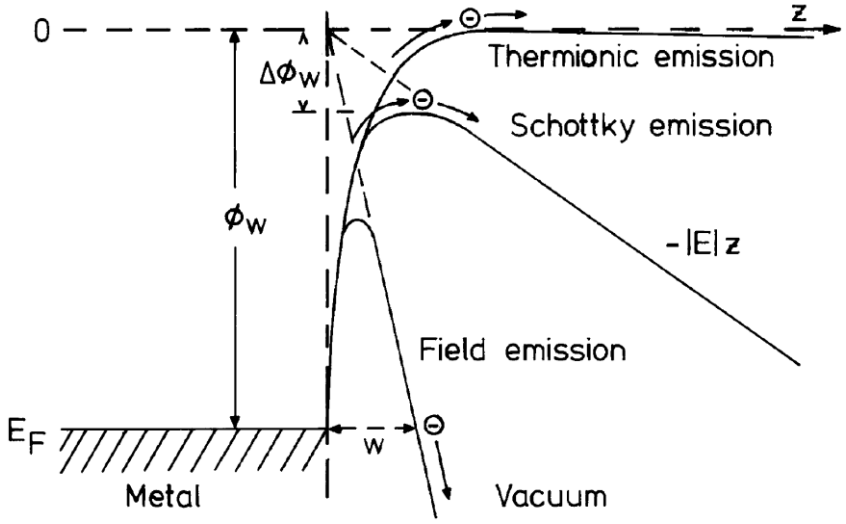


Figure 3.1: Electron potential energy $V(z)$ at the metal-vacuum boundary for different types of electron emission. For $E > E_F$, electrons have to overcome ϕ_w for thermionic emission or $\Delta\phi_w$ for Schottky emission. In the case of field emission, electrons can tunnel through a barrier of width w. Graph taken from [370].

With the core characteristics and emission types having been introduced for various electron sources, it is now of relevance to discuss how these affect electron beam and image properties. This influence can be determined by assessing the performance of the electron source, quantified through metrics such as *brightness, coherence* and *stability* [371].

3.1.1.4 Brightness

The brightness of an electron source is defined as the current density per unit area per unit time. Due to their different sizes and geometries, electrons exit the source surface non-uniformly, which can be described by an angular distribution. Considering an electron source diameter d_0, a cathode emission current i_e and a divergence semi-angle α_0 (note that d_0 and α_0 are defined at the first cross-over of the electrons, *after* emission into the vacuum), the brightness can be written as [372]:

$$\beta = \frac{i_e}{\pi\left(\frac{d_0}{2}\right)^2 \pi{\alpha_0}^2} = \frac{4i_e}{(\pi d_0 \alpha_0)^2} \tag{3.5}$$

Note that β is measured in $\mathrm{A/m^2\ sr^1}$ and increases linearly with accelerating voltage.

3.1.1.5 Spatial Coherence

This metric is directly related to the effective source size and defined as d_c for coherent illumination:

$$\Delta x \gg d_c = \frac{\lambda}{2\alpha} \tag{3.6}$$

with λ being the electron wavelength and α the angle subtended by the source at the specimen and Δx refers to the distance between two wave coherence maxima [370,371]. For a fixed wavelength and with no apertures, spatial coherence is thus achieved when using field emission sources due to their small size. It can be further increased using small acceleration voltages (large λ) and small apertures. For the same source type small sources are more spatially coherent than large ones and give rise to a better phase contrast [371].

3.1.1.6 Temporal Coherence and Energy Spread

Temporal coherence determines the effects of energy spread (and hence wavelength variation) in the electrons emitted from the source on the coherence of the beam [371]. In this context, a coherence length λ_c can be defined as:

$$\lambda_c = \frac{v\mathrm{h}}{\Delta E} \tag{3.7}$$

with h being the Planck's constant, v the electron velocity and ΔE as the energy spread of the beam. Equation 3.7 illustrates the importance of a small ΔE (Table 3.1).

3.1.2 Pulsed Electron Emission

Pulsed electron emission is fundamentally different from continuous emission. To build a pulsed TEM, a commercially available column often serves as a base-platform for one or multiple lasers triggering a pulse of electrons in both UTEM and DTEM (see 2.1.5 and 2.1.6 respectively) [139]. As both the spatial and temporal domain accessed by those two techniques are determined by their operational modes, the following describes pulsed electron emission and gives a *qualitative* comparison to continuous electron emission. However, a direct *quantitative* comparison is difficult due to differences in the inherent physics of electron emission and is hence not attempted.

3.1.2.1 Electron Pulses in UTEM

Lasers based on Ti-doped sapphire lasing at 800 nm and generating femtosecond pulses (pulse durations ~ 100 fs) with repetition rates in the MHz regime are typically used in UTEM. Only one laser is employed and a beam splitter separates the original pulse into two pathways directed towards the electron gun and specimen respectively. The former reaches a photocathode after being converted to the third harmonic (267 nm), where photoelectrons are generated. Approximately one electron is emitted into the optical column of the instrument per pulse [139]. From this model, a limited comparison can be considered:

i) *Brightness*: It is possible to compare the brightness of a pulsed source with a continuous one by considering the normalised, time-averaged brightness B_n and scaling it with the

effective duty cycle of the pulsed source, and a typical FHWM electron pulse duration, resulting in the normalised peak brightness B_{np} [373]. Feist *et al.* [373], show that values comparable to a continuous Schottky emitter can be obtained for laser-induced field emission when electron pulses free of space charge effects are considered. From an image acquisition point of view, brightness as defined *3.1.1.4* is important for high information content in an acquired image. In contrast, image acquisition in UTEM is often based on cumulating single-electron pulses [139].

ii) *Spatial coherence*: While the spatial coherence for continuous electron sources is defined by the distance d_c (compare *3.1.1.5*) and as such is a property intrinsically related to the source geometry, it is dominated by the photon distribution that follows laser irradiation in the case of photocathodes. An illustration for the spatial coherence of a laser driven Schottky field emitter is given in Figure 3e in the study provided by Feist *et al.* [373]. High spatial coherence has also been reported by Houdellier *et al.* [148], who used a cold FEG photocathode where the illuminating laser is larger than the photocathode tip radius and hence the latter determines the spatial coherence.

iii) *Temporal coherence*: Feist *et al.* [373] measured a zero-loss peak of 0.6 eV (FHWM) on $1T - TaS_2$ and PCMO using a Schottky-type field emitter while Houdellier *et al.* [148] determined an energy resolution as low as 1 eV for an ultrafast operated cold-FEG. Hence the temporal coherence is determined by the source material and lies in a similar regime as for continuous electron emission (compare Table 3.1) for single-electron sources.

3.1.2.2 Electron pulses in DTEM

In recent developments of DTEM at LLNL, a Ta disk of 800 µm diameter welded on a conventional wire hairpin emitter was used (Figure 3.2). The fifth harmonic of a Nd:YLF laser (211 nm) was used to achieve high photoemission efficiency [139] with pulse lengths between 15 ns and > 1 µs. Compared to single-electron emission in UTEM, $\sim 10^9 - 10^{10}$ electrons are required in DTEM to generate an image or diffraction pattern that is not limited by noise. This results in electron pulses of high brightness and currents in the low mA range, orders of magnitude higher than for continuous electron emission. However, the emission of high current pulses leads to subsequent degradation of the spatial resolution due to space charge effects which is one fundamental limitation of DTEM at intermediate voltages (see section 2.1.6) [139]. Other core properties of this mode can be summarised as follows:

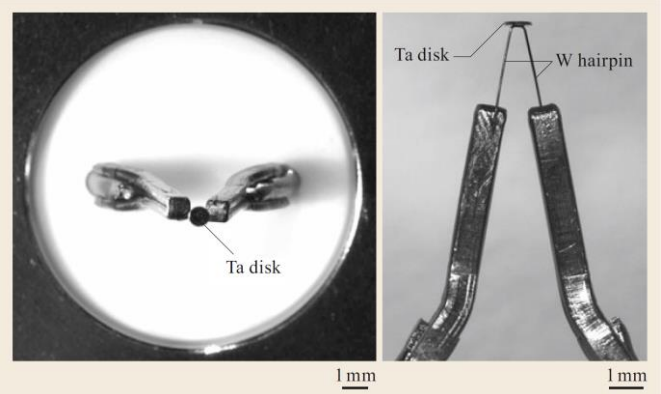


Figure 3.2: Photographs of 800 µm Ta disk welded on a wire hairpin emitter, used for single-shot TEM at LLNL. Figure as seen in [139].

i) *Brightness*: Overall, DTEM is a signal-limited technique and the brightness of the electron source (or pulses respectively) is thus of particular significance. Equation 3.5 can be rewritten as:

$$\beta_{source} = \frac{Ne}{(\pi r^2)(\pi \alpha^2)\Delta t} \tag{3.8}$$

with N as the number of electrons, r the transverse pulse radius, α the local convergence semi-angle and Δt the pulse duration. For a specific (intermediate) accelerating voltage, the brightness would ideally be constant with increasing extraction voltage, but is degraded due to space-charge effects and loss of coherence in practice. Note that beam degrading space charge effects affecting losses in the temporal resolution (Δt in (Eq. 3.8)) can be mitigated by increasing the acceleration voltage [139]. An alternative, instrument-based metric is the coherent fluence N_c [139]:

$$N_c = \frac{N\lambda^2}{r\alpha^2} = \frac{\pi^2 \beta_{source} \Delta t}{e} \tag{3.9}$$

which is the number of electrons per transverse coherence area per pulse. Similar to UTEM sources, it is in principle possible to compare the brightness of continuous electron sources with those used in DTEM for a set time-interval (*i.e.* the pulse duration). However, since the most recent source used for DTEM (Figure 3.2) is not employed in (S)TEM and both operational modes have entirely different image formation mechanisms, this comparison is not attempted here. A comparison of the coherent fluence N_c is however given in Figure 8.7 adapted from Campbell *et al.* [139].

ii) *Spatial coherence*: The resolution in (S)TEM imaging is achieved through optimisation of brightness, energy spread, lens aberration and sample quality. These are all factors that are compromised in DTEM instruments [139]. Increasing the brightness at the sample usually comes at the cost of lower spatial coherence [374] and it disadvantageous to discard spatially incoherent electrons using apertures [160]. Hence, it can be stated that the spatial coherence must be inferior which is indirectly supported by the previously mentioned Figure 8.7 [139], as the coherent fluence N_c is related to the lateral coherence $r_c = \lambda/\alpha = 2d_c$ (Eq. 3.6). Treatment of the two characteristic variables for electron bunches, namely transverse and longitudinal coherence, can be found in Appendix 10.1,

iii) *Temporal coherence*: Due to electron-electron interactions in high density pulses, the energy spread ΔE is expected to be large in DTEM, resulting in low temporal coherence [139]. An energy spread of 8.7 eV was measured for the DTEM prototype at TU Berlin [139] while an $\Delta E = 3$ eV was assumed in the discussions of the coherent fluence previously mentioned. More recently, Kuwahara *et al*. [375] reported an energy spread between $0.4\ \text{eV} - 20\ \text{eV}$ for picosecond pulses of different charges in a spin-polarised TEM running at 30 kV. Notwithstanding those developments, the temporal coherence of DTEM sources is likely to be inferior as compared to modern continuous electron sources used currently (Table 3.1)

In summary, this section has highlighted the various types of electron emission and outlined property metrics. Only a *qualitative* direct comparison between continuous and pulsed electron sources was attempted since operational modes and the inherent physics of electron emission differ substantially.

3.2 HRTEM Imaging

Electrons that are emitted into the vacuum of the instrument column are accelerated and passed through multiple condenser lenses, aberration correctors, an objective lens as well as projector lenses until they eventually reach the detector stage and an image is recorded. All of these stages are relevant to image formation in (S)TEM and the interested reader is referred to the textbooks mentioned at the beginning of this chapter for more detail. Here, only the essential theory for HRTEM image formation is described based on the chapter 'Atomic Resolution Transmission Electron Microscopy' by Kirkland, Chang and Hutchison [376].

3.2.1 Image Formation

The formation of an HRTEM image can be summarised in three steps. First, the electron beam interacts with the sample causing electron scattering in the sample plane. Second, a diffraction pattern at the back focal plane of the objective lens is formed. Third, an image is formed in the image plane. All of the above planes are connected *via* Fourier transforms and a theoretical framework connecting these individual parts is required, which is discussed based on the simplest scenario below. Assuming a wave description for the electron beam, consider a thin specimen, whose only effect is to alter the phase of an electron wave while leaving the amplitude unchanged. Furthermore, absorption is neglected. This is the *phase object approximation* (POA), which includes multiple scattering, but ignores Fresnel diffraction within the specimen. Under this approximation the specimen exit-wave complex amplitude is then expressed as:

$$\psi_e(x,y) = \exp(-\mathrm{i}\sigma\phi_p(x,y)) \tag{3.10}$$

with σ being an interaction constant given by:

$$\sigma = 2\pi me\frac{\lambda_r}{\mathrm{h}^2} \tag{3.11}$$

in which m and λ_r are the mass and relativistically corrected electron wavelength. The second relevant variable is the 2D projection of the specimen potential given by:

$$\phi_p = \int_{-t/2}^{t/2} \phi(x,y,z)dz \tag{3.12}$$

For extremely thin specimens consisting of low atomic number elements, the mean inner potential is small and hence phase changes are also small, allowing for 3.10 to be expanded as:

$$\psi_e(x,y) \approx 1 - \mathrm{i}\sigma\phi_p(x,y)) \tag{3.13}$$

which is known as the *weak phase object approximation* (WPOA). Note that within the WPOA, kinematic scattering is assumed which equates to a significantly higher intensity of the central unscattered waves as compared to diffracted waves. Fourier transforming equation 3.13, the complex amplitude of the wave in the back focal plane of the objective lens is obtained as:

$$\psi_d(u,v) \approx \delta(u,v) - \mathrm{i}\sigma\mathcal{F}\{\phi_p(x,y)\} \tag{3.14}$$

Subsequently, ψ_d needs to be further modified to consider the phase shifts introduced by the objective lens as well as an objective aperture. The former can be included using the wave aberration function $W(u,v)$ (Section 3.2.2) while the latter is described with a simple top-hat function:

$$\begin{aligned} P(k_x,k_y) &= 1 \quad |\boldsymbol{k}| \le r \\ P(k_x,k_y) &= 0 \quad |\boldsymbol{k}| > r \end{aligned} \tag{3.15}$$

in which $\boldsymbol{k}$ refers to the reciprocal wave scattering vector and r to the aperture radius. This leads to an expression for the WPOA with lens and aberration effects included as:

$$\psi'_d(k_x,k_y) \approx \delta(k_x,k_y) - \mathrm{i}\sigma\mathcal{F}\{\phi_p(x,y)\} \times P(k_x,k_y)\exp\left(iW(k_x,k_y)\right) \tag{3.16}$$

The image amplitude in real space is obtained by applying the inverse Fourier transform:

$$\psi_i(x,y) \approx 1 - \mathrm{i}\sigma(\phi_p(-x,y)) \times \mathcal{F}^{-1}\left\{P(k_x,k_y)\exp\left(iW(k_x,k_y)\right)\right\} \tag{3.17}$$

Cancelling the cosine terms, the recorded image intensity to first order is expressed as:

$$I(x,y) = \psi_i(x,y)\psi_i^*(x,y) \approx 1 + 2\sigma\phi_p(-x,-y)^* \times \mathcal{F}^{-1}\left\{\sin\left(W(k_x,k_y)\right)P(k_x,k_y)\right\} \tag{3.18}$$

This expression shows that the image contrast is proportional to the projection of the specimen convolved with an impulse response function of the microscope. The impulse response function can be more accurately described by including the partial spatial and temporal coherence (see Section 3.1) of the electron source which dampen the transfer of higher frequencies, and the electron detector MTF (see Section 2.1.7). The basic WPOA can also be extended in various ways to consider different models for image formation [376]. One extension is the *projected charge density* (PCD) approximation. This includes the effects of multiple scattering while retaining a projection approximation. Starting from Eq. 3.10, the amplitude of the exit-wave in the back-focal plane is given as:

$$\begin{aligned}\psi_d(k_x, k_y) &= \mathcal{F}\{\exp(-\mathrm{i}\sigma\phi_p(x,y))\} \times \exp(-\mathrm{i}\pi\Delta C_1\lambda(u^2+v^2)) \\ &\approx \phi_p(k_x, k_y)(1 + i\pi\Delta C_1\lambda|\boldsymbol{k}|^2)\end{aligned} \tag{3.19}$$

which is valid for small defocus values ΔC_1 provided $\phi_p(k_x, k_y)$ can be treated as a Fourier transform of $\exp(-\mathrm{i}\sigma\phi_p(x,y))$. This approximation is valid if all aberration coefficients are small, i.e for aberration corrected HRTEM imaging. Through similar treatments as for the WPOA and further mathematical reformulations, the intensity in the image plane is:

$$I(x,y) = 1 + \left(\frac{\Delta C_1\lambda\sigma}{2\pi\epsilon_0\epsilon}\right)\rho_p(x,y) \tag{3.20}$$

where $\rho_p(x,y)$ is the projected total charge density including the nuclear contribution.

3.2.2 Wave Aberration Function

Consider an ideal lens. A point object at position (x, y) in the object plane creates a spherical wavefront in the diffraction plane, which subsequently contracts to a conjugate point in the image plane. However, electromagnetic lenses cannot be aberration free, and consequently the image of the point object is blurred. A wave aberration function $W(u, v)$ can be defined as the distance between ideal and actual wavefronts in the diffraction plane as function of the point object position in reciprocal coordinates (u, v). This is also illustrated in Figure 3.3.

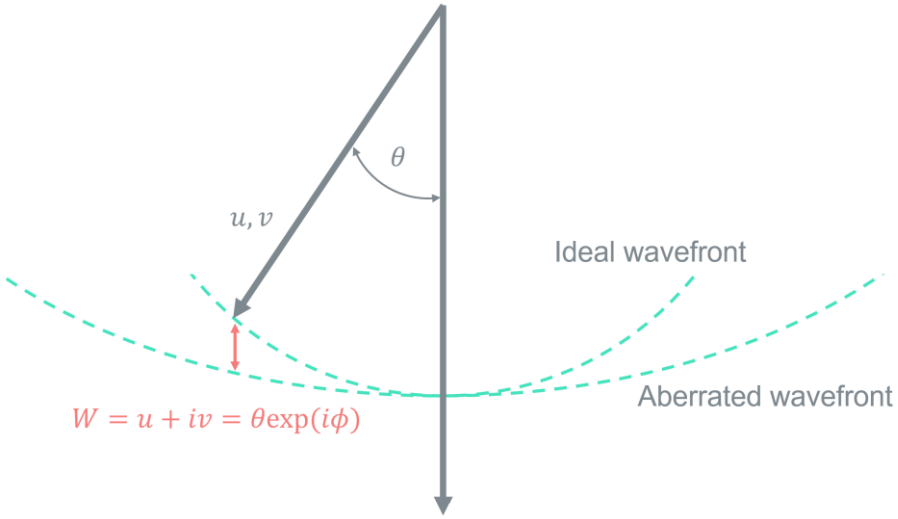


Figure 3.3: Schematic adapted from Kirkland *et al.* [376] which describes the origin of the wave aberration function $W(u, v)$ that describes the deviation of the ideal wavefront and the aberrated wavefront.

Substituting the complex position variable $\omega = u + iv$ and only considering axial aberrations which are a function of ω only, $W(\omega)$ can be Taylor expanded to third order terms as:

$$\begin{aligned} W(\omega) = \mathrm{Re}\Big\{\frac{2\pi}{\lambda}\Big(&A_0\lambda\omega^* + \frac{1}{2}A_0\lambda^2\omega^{*2} + \frac{1}{2}C_1\lambda^2\omega^*\omega \\ &+ \frac{1}{3}A_2\lambda^3\omega^{*3} + \frac{1}{3}B_2\lambda^3\omega^{*2}\omega \\ &+ \frac{1}{4}A_3\lambda^4\omega^{*4} + \frac{1}{4}S_3\lambda^4\omega^{*3}\omega \\ &+ \frac{1}{4}C_3\lambda^4\omega^{*2}\omega^{*2} + \cdots\Big)\Big\} \end{aligned} \tag{3.21}$$

Converting to polar notation *via* $\sin(\omega) = \boldsymbol{k}\exp(i\phi)$, $A_n = |A_n|\exp(i\alpha_n)$ and $B_n = |B_n|\exp(i\beta_n)$ yields and expression where radial and azimuthal dependencies are more apparent.

$$\begin{aligned} W(\boldsymbol{k},\phi) = \frac{2\pi}{\lambda}\Big(& |A_0|\lambda k\cos(\phi - \alpha_0) \\ & + \frac{1}{2}|A_1|\lambda^2 k^2\cos 2(\phi - \alpha_1) + \frac{1}{2}C_1\lambda^2 k^2 \\ & + \frac{1}{3}|A_2|\lambda^3 k^3\cos 3(\phi - \alpha_2) \\ & + \frac{1}{3}|B_2|\lambda^3 k^3\cos(\phi - \beta_2) \\ & + \frac{1}{3}|A_3|\lambda^4 k^4\cos 4(\phi - \alpha_3) \\ & + \frac{1}{3}|S_3|\lambda^4 k^4\cos 2(\phi - \sigma_3) + \frac{1}{4}C_3\lambda^4 k^4 + \cdots \Big) \end{aligned} \tag{3.22}$$

Table 3.2 lists all axial aberration coefficients to the third order using the notation introduced by Typke and Dierksen [376,377].

Table 3.2: HRTEM aberration coefficients to third order [376].

Aberration	Order in ω	Azimuthal symmetry	Name and description
A_0	1	1	Image shift
A_1	2	2	Two-fold astigmatism
A_2	3	3	Three-fold astigmatism
A_3	4	4	Four-fold astigmatism
B_2	3	3	Axial coma
S_3	4	4	Axial star
C_1	2	∞	Defocus
C_3	4	∞	Spherical aberration

3.3 HRTEM Image Simulation

3.3.1 Approaches for Image Simulation

The acquisition of atomic resolution TEM images is not the only requirement that needs to be fulfilled to determine quantitative information from a given sample. Calculations of corresponding image simulations, that replicate the structure under consideration and take into account imaging conditions are required for comparison [376]. Two main methods have been proposed for this purpose: The Bloch wave method, introduced by Bethe with many implementations described in the literature [378] and the multislice method, introduced by Cowley and Moodie [379]. The former uses the Bloch theorem in solid-state physics in which the product of a plane wave and a function with the same periodicity such as a crystal are used to solve the Schrödinger equation in a periodic crystal potential. The periodicity function is subsequently expanded in its Fourier components, effectively reducing the Schrödinger equation to a matrix equation for the Fourier coefficients [376]. While this approach is relatively accurate for simple crystals, it is impractical for complex crystals with large unit cells since a large number N of Bloch waves is required for the calculation with the computation time for the matrix solution increasing as N^3.

The multislice method is currently the most commonly used, especially in HRTEM. In contrast to the Bloch wave method, efficient Fast Fourier Transform (FFT) algorithms and more readily available computing resources have answered initial accuracy constraints caused by limited sampling and a limited number of slices or diffracted beams. Since the multislice method plays a central role in this work, it is treated in more detail as outlined by Kirkland *et al.* below [376]. More extensive descriptions are also available [378].

3.3.2 The Multislice Method

The core paradigm of this approach is the virtual division of a specimen into multiple, thin slices orthogonal to the incident electron wave, enabling the effects of the specimen potential (*i.e.* transmission in real space) and Fresnel diffraction (*i.e.* propagation in Fourier space) to be evaluated separately. The specimen potential is modelled using electron scattering factors for which different approaches can be selected [380]. The slices have to be thin enough to be treated as WPO whilst holding on to periodic boundary conditions in the direction orthogonal to the incident electron wave.

The Schrödinger equation for the wavefunction ψ_f of an electron in the electrostatic potential $V(x,y,z)$ of a specimen is given as:

$$\left[-\frac{\hbar^2}{2m}\nabla^2 - eV(x,y,z)\right]\psi_f(x,y,z) = E\psi_f(x,y,z) \tag{3.23}$$

where m refers to the relativistically corrected mass of an electron and $\hbar$ is Planck's constant h divided by 2π. It can be assumed, that for high-energy electrons, motion predominantly occurs in the z-direction. Hence, $\psi_f(x,y,z)$ can be treated as a product of the solution of the free Schrödinger equation ($V \equiv 0$) propagating in z and a wavefunction describing the effects of the specimen:

$$\psi(x,y,z) = \exp(2\pi kz)\psi(x,y,z) \tag{3.24}$$

with k being the inverse wavelength of an electron. Substituting equation 3.24 into $E = h^2k^2/2m$ gives

$$-\frac{\hbar^2}{2m}\left[\nabla_{xy}^2 + \frac{\partial^2}{\partial z^2} + 4\pi k\frac{\partial^2}{\partial z} + \frac{2meV(x,y,z)}{\hbar^2}\right] \times \psi(x,y,z) = 0 \tag{3.25}$$

Restricting this further to elastic scattering processes, the components k_x and k_y are proportional to the scattering angle while the change Δk_z is proportional its square. Assuming small scattering angles, the relation $\left[\frac{\partial^2\psi}{\partial z^2}\right] \ll \left|\nabla_{xy}^2\psi\right|$ holds and can be neglected in the paraxial approximation. This leads to a first-order differential equation in z which can be expressed as:

$$\frac{\partial\psi(x,y,z)}{\partial z}[\mathrm{A} + \mathrm{B}]\psi(x,y,z) \tag{3.26}$$

in which $A = \frac{i\lambda}{4\pi}\nabla_{xy}^2$, $B = i\sigma V(x,y,z)$ and σ defined as in equation 3.11. Solving equation 3.26 yields

$$\psi(x,y,z+\Delta z)=\exp\left\{\int_{z}^{z+\Delta z}[A(z')+B(z')]dz'\right\}\psi(x,y,z) \tag{3.27}$$

which for small Δz, can be reduced to:

$$\psi(x,y,z+\Delta z)=\exp\left[\frac{i\lambda}{4\pi}\Delta z\nabla_{xy}^{2}+i\sigma V_{\Delta z}(x,y,z)\right]\psi(x,y,z) \tag{3.28}$$

whereby

$$V_{\Delta z}(x,y,z)=\int_{z}^{z+\Delta z}V(x,y,z')dz' \tag{3.29}$$

refers to the projected specimen potential between z and $z+\Delta z$. Equation 3.28 cannot be written as a product of two exponential functions as commutative property does not valid in this case. However, the following approximation is valid as both, $\frac{i\lambda}{4\pi}\Delta z\nabla_{xy}^{2}$ and $i\sigma V_{\Delta z}(x,y,z)$ and are small:

$$\begin{aligned}\exp(\varepsilon A+\varepsilon B)&=\exp(\varepsilon A)\cdot\exp(\varepsilon B)+\frac{\varepsilon^{3}}{2}[B,A]+O(\varepsilon^{3})\\&=\exp(\varepsilon A)\cdot\exp(\varepsilon B)+O(\varepsilon^{3})\end{aligned} \tag{3.30}$$

Which can be applied to equation 3.28 to give:

$$\psi(x,y,z+\Delta z)=\exp\left(\frac{i\lambda}{4\pi}\Delta z\nabla_{xy}^{2}\right)t(x,y,z)\psi(x,y,z) \tag{3.31}$$

Note that the term $\exp(i\sigma V_{\Delta z}(x,y,z)$ has been rewritten as $t(x,y,z)$ and refers to the transmission function for the specimen slice between z and $z+\Delta z$.

The other operator can be rewritten as:

$$\exp\left(\frac{i\lambda}{4\pi}\Delta z\nabla_{xy}^{2}\right)=\exp\left(i\alpha\frac{\partial^{2}}{\partial x^{2}}\right)\exp\left(i\alpha\frac{\partial^{2}}{\partial y^{2}}\right) \tag{3.32}$$

where $\alpha=\frac{\lambda\Delta z}{4\pi}$.

Subsequently, the Fourier transform for the x-component is considered:

$$\begin{aligned}
&\mathcal{F}\left\{\exp\left(i\alpha\frac{\partial^2}{\partial x^2}\right)f(x)\right\} \\
&\quad = \int \exp(-2\pi kx)\exp\left(i\alpha\frac{\partial^2}{\partial x^2}\right)f(x)dx \\
&\quad = \sum\frac{(i\alpha)^n}{n!}\int \exp(-2\pi kx)f^{(2n)}(x)dx \\
&\quad = \sum\frac{(i\alpha)^n}{n!}(-2\pi k)^{2n}\int \exp(-2\pi kx)\,f(x)dx \\
&\quad = \exp(-4i\pi^2k^2\alpha)\mathcal{F}\{f(x)\}
\end{aligned} \tag{3.33}$$

Doing the same for the y-component yields:

$$\begin{aligned}
\mathcal{F}\left\{\exp\left(\frac{i\lambda}{4\pi}\Delta z\nabla_{xy}^2\right)\right\} &= \mathcal{F}\left\{\exp\left(i\alpha\frac{\partial^2}{\partial x^2}\right)\exp\left(i\alpha\frac{\partial^2}{\partial y^2}\right)\right\} \\
&= \exp(-i\pi\lambda(k_x^2+k_y^2))\mathcal{F}\{f(x,y)\}
\end{aligned} \tag{3.34}$$

which gives a final equation for a single multislice step as:

$$\psi(x,y,z+\Delta z) = \mathcal{F}^{-1}\{P(k_x,k_y)\}\mathcal{F}\{t(x,y,z)\psi(x,y,z)\} \tag{3.35}$$

with the propagation function $P(k_x,k_y)$ in Fourier space given by

$$P(k_x,k_y) = \exp\left(-i\pi\lambda\Delta z(k_x^2+k_y^2)\right) \tag{3.36}$$

This calculation is performed for individual slices until the full range z of the sample considered has been included. The above algorithmic implementations work very efficiently using FFTs, but the approximations introduced can impose limitations under certain conditions. As the number of beams accounted for in the simulation is limited by the discrete sampling of the applied unit cell, it is only reliable as long as scattering into beams that are not included in the simulation is negligible. This can be addressed by including a large number of beams, which is practicable with modern computation. Lastly, the error due to the paraxial approximation can be minimised in the majority of cases by using slices that are sufficiently thin [376].

3.4 Deep Learning

Deep Learning (DL) approaches, especially as applied in computer vision, form an integral part of this work. Compared to other scientific disciplines, such as computer science, modern DL approaches were applied relatively recently in EM (see 2.2). Hence, the essential theoretical concepts relevant to this work are introduced in the following sections based on the books by Goodfellow, Bengio and Courville [381], Géron [382], as well as Aggarwal [383]. It should be noted that a number of concepts relevant for the full functioning of the DL methods applied in this work are not discussed in detail, and the reader is referred to resources given above.

3.4.1 Single Layer Perceptron

The earliest algorithms recognised in this field today aimed to replicate learning as it occurs in biology or even in the brain [381], leading to the name 'artificial neural networks' (ANNs) . The simplest neural network is a single layer perceptron shown in Figure 3.4. From this illustration, a prediction situation of the form $(\bar{X}, y)$ is considered, where $\bar{X} = [x_1, \dots, x_d]$ with d feature variables and $y \epsilon \{-1,1\}$ represents the observed value of the binary class variable. $\bar{X}$ is transmitted through the input layer consisting of d nodes with weights $\bar{W} = [w_1, \dots, w_d]$ to the output node,

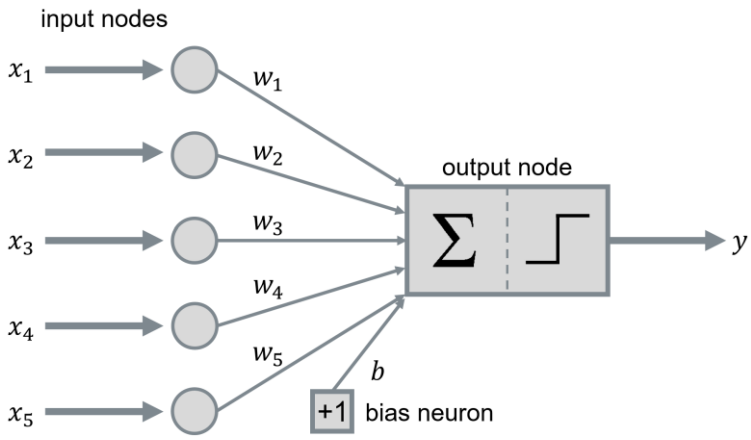


Figure 3.4: Schematic architecture of a single layer perceptron as in [383].

where $\bar{W} \cdot \bar{X} = \sum_{i=1}^{d} w_i x_i$ is computed and a class prediction $\hat{y}$ is obtained as:

$$\hat{y} = \text{sign}(\bar{W} \cdot \bar{X} + b) = \text{sign}\left(\sum_{i=1}^{d} w_i x_i + b\right) \tag{3.37}$$

where the sign function maps a real value to -1 or +1 [383]. Note the invariant part of the prediction b, also refered to as *bias*, is also included in equation 3.37 for completeness. It can be regarded as a 'neuron' that contributes constant values to the output node and is not influenced by the input $\bar{X}$. The key goal of neural networks is to minimise the prediction error, which can be formulated in a least-squares form as:

$$Minimise_{\bar{W}} L = \sum_{(\bar{X},y)\,\in\,\mathcal{D}} (y - \hat{y})^2 = \sum_{(\bar{X},y)\,\in\,\mathcal{D}} \left(y - \text{sign}\left(\sum_{i=1}^{d} w_i x_i + b \right) \right)^2 \qquad (3.38)$$

in which $\mathcal{D}$ refers to the dataset containing all feature-label pairs used for training (see below). This and similar types of minimisation objective functions are usually referred to as *loss functions*.

3.4.2 Multilayer Neural Network

Multilayer neural networks, or multilayer perceptrons (MLPs), consist of one input layer, one or multiple computational layers that are referred to as *hidden layers*, and an output layer. A schematic overview, showing the most common forms of representations is given in Figure 3.5.

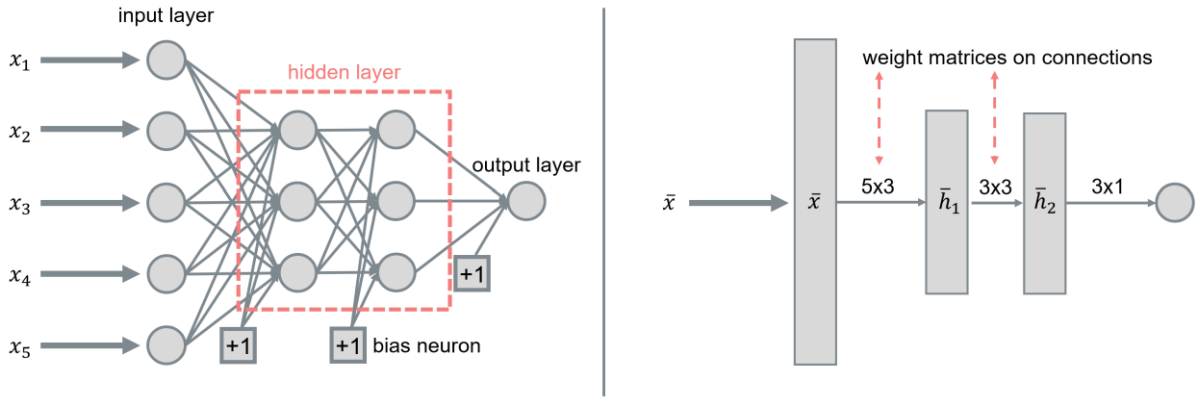


Figure 3.5: Different representations of the same basic MLP architecture containing two hidden layers and a single output layer. **Left:** Each unit contains single scalar variables, which are often represented by a single vector unit. **Right:** Vector units are represented as rectangles and have connection matrices between them. Figure adapted from [383].

Neural networks containing deep stacks of hidden layers are called deep neural network (DNN) and the research field studying architectures (models) containing deep stacks of computations is called Deep Learning [382]. The details of the distinction between MLP and DNN by simply comparing the number of hidden layers is too simplistic [383], but thorough explanations are outside the scope of this chapter.

3.4.3 Optimisation

3.4.3.1 Backpropagation

Having set a neural network, it would be ideal that layer weights $\bar{W}$ are selected such that a correct output is obtained for different inputs. However, researchers were not initially able to tune – or *train* – MLPs automatically for this goal. Rumelhart, Hinton and Williams [384] found a ground-breaking method still used today: *backpropagation.* Briefly, the backpropagation algorithms computes gradients of all model parameters (weights, biases) with respect to the network's error function in one forward and one backward pass. A *gradient descent* step can then be performed towards a lower error. The process repeated until convergence is achieved [382].

In neural networks, compositions of functions are computed for individual nodes. It is thus possible to view a pass through two nodes as the composition function $f(g(\cdot))$, where the first node exhibits $g(\cdot)$ and the latter $f(\cdot)$. Assuming a sigmoid function for both composition functions and weights w, this results in

$$f\big(g(w)\big) = \frac{1}{1 + \exp\left(-\frac{1}{1 + \exp(-w)}\right)} \tag{3.39}$$

which would lead to demanding computation effort when the derivative is determined. Thus, *backpropagration* leverages the *chain rule of differential calculus* and in its simplest form can be written as:

$$\frac{\partial f(g(w))}{\partial w} = \frac{\partial f(g(w))}{\partial g(w)} \frac{\partial g(w)}{\partial w} \tag{3.40}$$

which is also referred to as the *univariant chain rule*. The key principle is that a composition of functions are applied on weights w to subsequently generate an output, where the gradient of the output can be written as product of local gradients along 'a path' (consider the left side of Figure 3.5). Computation is therefore simplified since each local gradient is only dependent on local input and output. As an example, the function $f(y) = \cos(y)$ and $g(w) = w^2$ are considered in Figure 3.6.

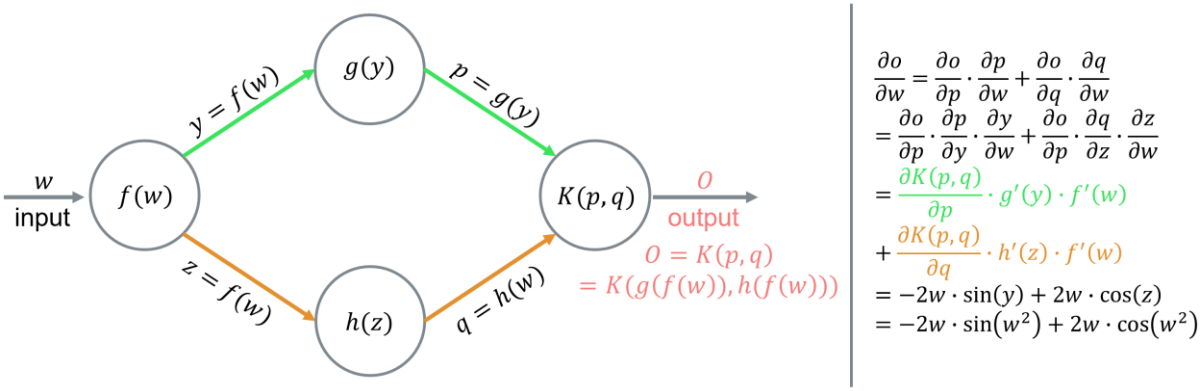


Figure 3.6: Example of an applied chain rule. Products of node-specific partial derivatives of different paths from weights w to output o are aggregated, yielding the derivative of output O with respect to w. Figure adapted from [383].

In reality, neural networks are not represented as paths as hidden layers receive multiple inputs. Equation 3.40 can be generalised into the *multivariable chain rule*, for which the function $f(g_1(w), \dots, g_k(w))$ receiving k inputs generating $g_1(w), \dots, g_k(w))$ must be considered in the following form [383]:

$$\frac{\partial f(g_1(w), \dots, g_k(w)))}{\partial w} = \sum_{i=1}^{k} \frac{\partial f(g_1(w), \dots, g_k(w)))}{\partial g_i(w)} \cdot \frac{\partial g_i(w)}{\partial w} \tag{3.41}$$

In summary, the backpropagation algorithm consists of the following steps [382]:

i) The training dataset is split into so-called *mini-batches* which are all passed through the network for training. A pass of all mini-batches during training is called an *epoch*.

ii) *Mini-batches* are passed through the network to generate an output. During this *forward pass* the output of all neurons/layers are computed and preserved.

iii) Subsequently, the network output error is calculated using a *loss function* (see above).

iv) The contribution of all neurons to the *loss function* is computed by propagating the error gradient backwards through the whole neural network.

v) A Gradient Descent step updates the weights with the error gradients computed.

3.4.3.2 Activation Functions

The treatment above underlines the importance of non-linear output functions. Contrasting the sign-function introduced in Section 3.4.1, the gradient of the logistic (sigmoid) function, *i.e.* $\sigma(z) = 1/(1 + \exp(-z))$ is non-zero for nearly all values. Furthermore, non-linear functions preserve the model complexity: If linear activation functions would be used, multiple weight matrices could be mathematically summarised into a single one and would conclusively act as a single layer perceptron. Deep stacks of layers (and corresponding weight matrices) are important to DNNs, as a sufficient number of layers can in principle approximate any desired function [382].

In addition to the sigmoid function, popular functions are

$$\tanh(z) = 2\sigma(2z) - 1 \tag{3.42}$$

and

$$\mathrm{ReLU}(z) = \max(0, z) \tag{3.43}$$

where in both cases, z refers to input received from a previous layer. The hyperbolic tangent function is an S-shaped function with output ranging from -1 to 1 and generates outputs centred around 0, speeding up convergence. The rectified linear unit function is not differentiable at $z = 0$, but has the advantage of being computed rapidly and has been shown to work well in practice [382]. However, due to its former property, gradient values of 0 can arise resulting in 'dead neurons'. An expedient is given by using the 'LeakyReLU' function:

$$\mathrm{LeakyReLU}(z) = \max(\alpha z, z) \tag{3.44}$$

in which α is a tuneable parameter, *i.e. hyperparameter*, set to 0.01 in many practical implementations.

3.4.3.3 Unstable Gradients

In practice, this type of optimisation poses challenges. Gradients tend to shrink with propagation to lower layers, leaving them nearly unchanged and thus hampering convergence. Similarly, the opposite can occur in which case gradients grow rapidly until the network diverges. The former is called the *vanishing gradients problem* while the latter is known as the *exploding gradients problem*. They were firstly identified by Glorot and Bengio [382,385] who flagged specific aspects of the training procedure used at that time as being responsible. As a result, the authors proposed a new method of weight

initialisation with which the variance of the outputs of each layer would be equal to the variance of the respective input, leading to an equal 'signal flow' in forward feeds and backpropagation, named the *Xavier initialisation*:

$$\sigma = \frac{1}{fan_{avg}} \tag{3.45}$$

which is the variance applied when a normal distribution is assumed. Alternatively, a uniform distribution of weight values can be assumed between $-r$ and r with

$$r = \sqrt{\frac{3}{fan_{avg}}} \tag{3.46}$$

Note that both of the above assume a sigmoid function and with $fan_{avg} = (fan_{in} + fan_{out})/2$ referring to the connection weights. A further reduction of unstable gradients can also be achieved by using variants of the ReLU function (compare with equation 3.44) [382].

Although the above measures significantly reduce the probability of unstable gradients at the beginning of the training, they do not guarantee their reappearance in later stages. This can be addressed by *batch normalisation*, that zero-centres and normalises each input and subsequently scales and shifts the result with two parameter vectors per layer. To perform the former two, an estimation of the standard deviation and mean are required. These are determined on the currently used mini-batch and can be described by the following four steps [382].

i) $$\mu_B = \frac{1}{m_B}\sum_{i=1}^{m_B} x^{(i)} \tag{3.47}$$

ii) $$\sigma_B^2 = \frac{1}{m_B}\sum_{i=1}^{m_B}\left(x^{(i)} - \mu_B\right)^2 \tag{3.48}$$

iii) $$\hat{x}^{(i)} = \frac{x^i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{3.49}$$

iv) $$z^{(i)} = \gamma \otimes \hat{x}^{(i)} + \beta \tag{3.50}$$

Where μ_B refers to the vector of input means computed over the mini-batch B containing $m_B$isntances, σ_B refers to the vector of standard deviations, equally determined over the mini-batch B, $\hat{x}^{(i)}$ denotes the vector of normalised and zero-centred inputs for instance i, ε represents a *smoothing term* (a small number) , γ refers to the output scale parameter vector for the layer, $\otimes$ is an element-wise multiplication in this context (instead of an convolution), β is the output offset parameter vector for the layer (each input is offset by the corresponding shift parameter), and $z^{(i)}$ is the output.

3.4.4 Gradient Descent Strategies

The sections above were explained under the assumption of a *steepest-descent method* for parameter learning, where the gradient of the applied *loss function* is directly used for parameter updates. The gradient descent can however be unstable and not always lead to the best direction of improvement. Hence, alternative strategies are commonly applied and selected examples are briefly described below [383].

3.4.4.1 Momentum-Based Learning

Momentum-based strategies consider the possibility of 'zigzagging', *i.e.* conduction of contradictory gradient descent steps that cancel each other out partially and consequently slow down the learning process of the network. Moving along an 'averaged' direction from the last few steps can counteract this behaviour. In a conventional scenario. updates based on a *loss function L* are written as:

$$\bar{V} \leftarrow -\alpha \frac{\partial L}{\partial \bar{W}};\ \bar{W} \leftarrow \bar{W} + \bar{V} \tag{3.51}$$

where $\bar{V}$ represents the gradient descent vector and α refers to the learning rate. Using a momentum-based descent, Eq. 3.51 is modified to:

$$\bar{V} \leftarrow \beta \bar{V} - \alpha \frac{\partial L}{\partial \bar{W}};\ \bar{W} \leftarrow \bar{W} + \bar{V} \tag{3.52}$$

where $\beta \in (0,1)$ is the *momentum parameter* or *friction parameter* [383].

An alternative strategy to ensure consistency in gradient descent steps is taken up by parameter-specific learning rates. While some parameters might oscillate strongly, other parameters with small partial derivatives tend to be more stable.

3.4.4.2 AdaGrad

The AdaGrad algorithm tracks the aggregated squared magnitude of the partial derivative of each parameter during training, whereby the square-root of this value is proportional to the root-mean-square slope of the parameter considered. The algorithm performs updates of the aggregate value A_i of the ith parameter according to:

$$A_i \leftarrow A_i + \left(\frac{\partial L}{\partial w_i}\right)^2 \quad \forall i \tag{3.53}$$

while the ith parameter w_i is updated *via*:

$$w_i \leftarrow w_i - \frac{\alpha}{\sqrt{A_i}}\left(\frac{\partial L}{\partial w_i}\right) \quad \forall i \tag{3.54}$$

Introducing the term $\sqrt{A_i}$ in the denominator can be interpreted as a 'signal-to-noise' normalisation as A_i determines the historical magnitude of the gradient. Overall, the algorithm encourages faster relative moments in directions with smaller gradient values, but with consistent sign [383].

3.4.4.3 RMSProp

Compared to *AdaGrad*, *RMSProp* uses exponential averaging instead of the squared gradients to compute A_i. This prevents premature slowdown in the gradient progress. The idea is to use a decay factor $\rho \in (0,1)$ while weighting the squared partial derivatives within the last t updates by ρ^t. Updates in *RMSProp* are computed as:

$$A_i \leftarrow \rho A_i + (1-\rho)\left(\frac{\partial L}{\partial w_i}\right)^2 \quad \forall i \tag{3.55}$$

Again, the square-root of this value is used to normalise the gradient. Following this step, the update used for the global learning rate α is similar to the one described in equation 3.54 [383].

3.4.4.4 Adam

This algorithm unifies many of the aspects described above and combines them into an efficient way to apply gradient descent steps. It uses 'signal-to-noise' normalisation, while exponentially smoothing the first-order gradient to incorporate momentum. Analogous to *RMSProp*, A_i refers to the exponentially averaged value of the ith parameter w_i. The update is carried out as described in Eq. 3.55. However, an

exponentially smoothed value of the gradient is also retained, referred to as F_i. In this case, the smoothing is computed with a decay parameter ρ_f.

$$F_i \leftarrow \rho_f F_i + (1 - \rho_f)\left(\frac{\partial L}{\partial w_i}\right) \quad \forall i \tag{3.56}$$

Which can be interpreted as a variation of the pure *momentum-based method* discussed in Section 3.4.4.1. Lastly, the update in the ith iteration is computed with learning rate α_t by:

$$w_i \leftarrow w_i - \frac{\alpha_t}{\sqrt{A_i}} F_i \quad \forall i \tag{3.57}$$

In summary, there are two key differences to the algorithms discussed above and in particular to *RMSProp*. First, the gradient is substituted with an exponentially smoothed value resembling momentum. Secondly, the learning rate α_t is now dependent on the iteration index t:

$$\alpha_t = \alpha\left(\frac{\sqrt{1 - \rho^t}}{1 - \rho_f^t}\right) \tag{3.58}$$

The most important details on neural networks along with optimisation (training) strategies have now been introduced. Finally, Convolutional Neural Networks are discussed as this type of DNN are most relevant to the application presented in this work.

3.4.5 Convolutional Neural Networks

3.4.5.1 Using Convolutions

Convolutional Neural Networks (CNNs) are a special type of neural network that processes data with predetermined grid-like topology, such as images that can be considered as 2D grids of pixels. As the name suggests, this type of network employs mathematical convolutions in at least one of the layers [381] and this function can therefore be considered as the most important building block. The convolution operation can be described by considering a single output $x(t)$ describing the position of an object at time t in a tracking scenario. Furthermore, the tracking is assumed to be noisy. To obtain a less noisy estimate of the object position, several measurements $x(t)$ are performed, where more recent measurements are more important. This can be taken into account by a weighting function $u(a)$ in which a refers to the measurement age. Applying the weighted average at every moment, a new function s is obtained that describes a smoothed estimate of the objects position as:

$$s = \int x(a)u(t-a)da = (x * u)(t) \tag{3.59}$$

For CNNs, x is usually referred to as *input*, u as the *kernel* and s sometimes as the *feature map* [381]. To apply the mathematical convolution to grid-like topologies with discrete values a discrete convolution must be considered:

$$s(t) = (x * u)(t) = \sum_{a=-\infty}^{\infty} x(a)u(t-a) \tag{3.60}$$

Furthermore, it is important to consider that the input is usually a multidimensional array of data, while the kernel is equally a multidimensional array of parameters, both of which are tensors. It is also assumed that the functions are all zero, apart from the finite cases for which values are stored, allowing the infinite summation to be written as a finite summation over all array elements. Lastly, convolutions are used on more than one axis at the time:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-m) \tag{3.61}$$

where K is a two-dimensional kernel and I is a two-dimensional input, *i.e.* an image [381]. In practice, this is realised by neurons that are only connected to certain *receptive fields* of corresponding previous layers. For example, the first layer only sees part of the image, while each neuron in the second layer is connected to neurons located in a corresponding small rectangle of the first layer (Figure 3.7). In this way, individual neurons can 'focus' on small low-level features in the first hidden layer and subsequently merge them into more abstract, higher-level features in the second *etc.* [382]. More specifically, this also refers to *sparse interactions*. As the kernel is smaller than the input, fewer parameters need to be stored for each convolution and effectively reduces memory requirements and statistical efficiency. CNNs are initialised by approaches such as the *Xavier initialisation* (see 3.4.3.3) and thus, all convolutional layers and applied kernels are initially randomised. During training, the weights of neurons can be represented as small images that are of the size of receptive fields – or *filters* (see Figure 3.7). Each of these filters will react to different image features during training and evolve automatically according to the task assigned [382]. Conventionally, multiple filters are applied that generate different feature maps after each convolution (Figure 3.8).

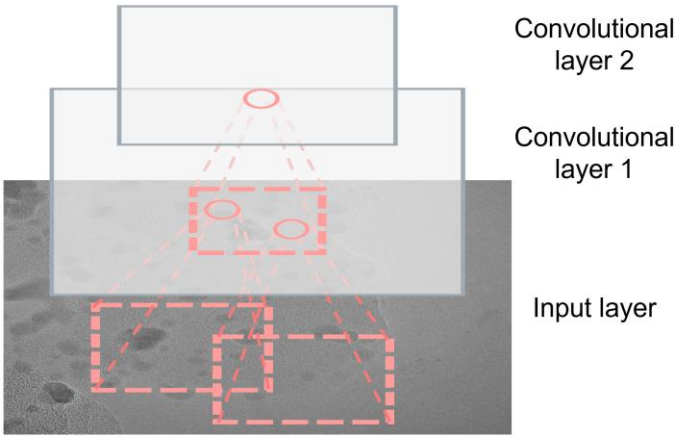


Figure 3.7: Layers of a CNN with receptive fields, demonstrating how neurons are connected. Figure adapted from [382] and modified.

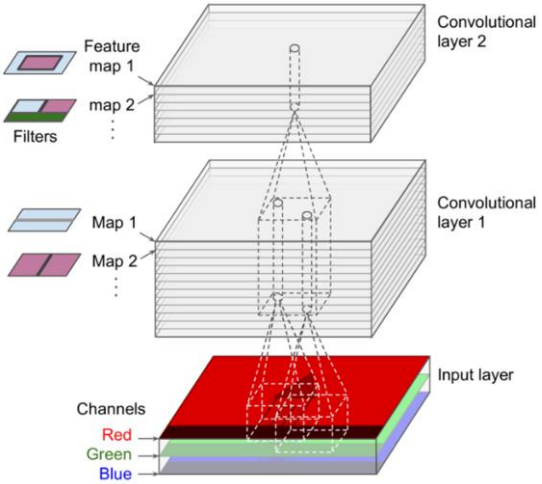


Figure 3.8: CNN with multiple feature maps and three-channel input image. Figure adapted from [382].

3.4.5.2 Pooling

Convolutional layers in a CNN function in three stages. First, several convolutions are performed in parallel and a set of linear activations are computed. Next, the linear activations are passed through a non-linear activation function. Finally, a pooling function is used to substitute the output at a given location with summary statistics of nearby outputs. A prominent example is the *max pooling* function, which generates the maximum output within a rectangular neighbourhood. Overall, this step increases *invariance* to small translations, which is especially useful for cases where it is more important that specific features are represented in the input rather than their precise positions [381].

3.4.5.3 Architectures

In very simple terms, common CNN architectures are built up by repeating the sequence of several convolutional layers, a variant of the ReLU activation function and a pooling layer, which eventually processes an input image into increasingly smaller feature maps. In the end (or top respectively), a regular feedforward neural network composed of fully connected layers is added, from which a prediction is obtained (Figure 3.9).

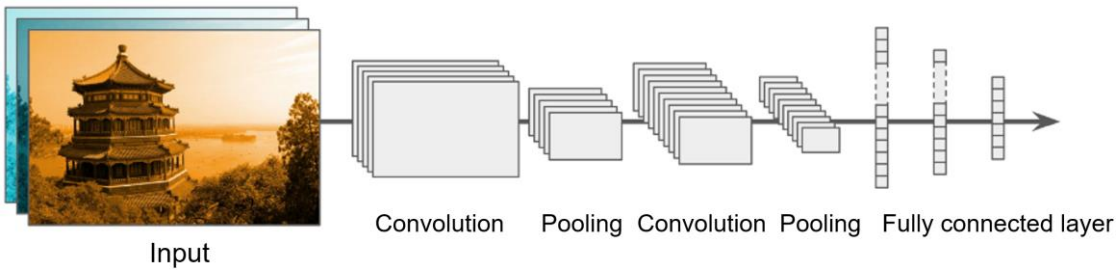


Figure 3.9: Simplified, common CNN architecture. Figure adapted from [382].

There are several well-known architectures, often used as examples and more thoroughly explained in textbooks [382,383] and in the numerous references provided in Section 2.2 that highlight DL applications in EM. Overall, the influence of different operations within different layers of DNNs are researched in the field of Deep Learning and new architectures are frequently released to solve dedicated tasks, and in some cases, different networks would compete against each other [186].

3.5 Chapter Summary

This chapter has highlighted core theoretical considerations of important aspects relevant to this work, including different modes of electron emission, HRTEM image formation and simulation along with the DL fundamentals. These are used in subsequent experiments and simulations.

4 Methods

In this chapter, the key experimental, computational methods and approaches relevant to this work are described. Further details can be found in subsequent chapters. Note that many aspects of the methods outlined herein form part of Treder, Huang, Bell *et al.*, *npj Compu. Mat.*, second revision, 2022.

4.1 HRTEM Imaging

This section outlines techniques and parameters used to acquire experimental HRTEM datasets.

4.1.1 Au/Ge and Pd/C

4.1.1.1 Sample preparation

A standard calibration sample of gold nanoparticles supported on ultrathin (*ca.* 4 nm) amorphous germanium (Au/Ge) was prepared by JEOL Ltd. and provided on a lacy-carbon copper TEM grid. A single sample was used to acquire all Au/Ge related images in this work. Samples of palladium nanoparticles (1 wt. %) on active carbon were synthesised from alginate, fired at 450 °C, and provided as powder by Johnson Matthey plc. Exact details of the powder preparation procedure are protected by the intellectual property of Johnson Matthey plc. TEM grids were prepared using a dry deposition method (Figure 4.1): A small amount of Pd/C powder was put on the tip of a thin spatula and placed on a clean optical glass slide. The powder was then carefully crushed in small circular motion without additional force using a second optical glass slide. Finally, a standard TEM copper grid was sandwiched

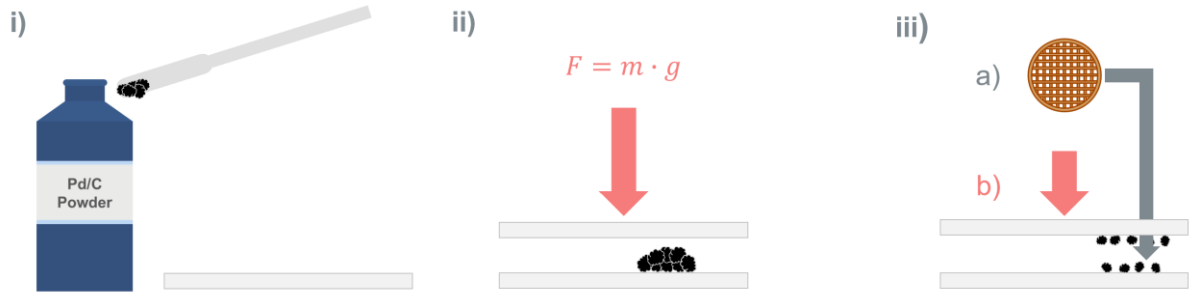


Figure 4.1: Schematic illustration of the dry sample preparation method. **i)** Pd/C powder taken up from a container using the tip of a thin spatula and placed on a glass slide. **ii)** Powder sandwiched between two glass slides and crushed with no additional force and careful circular motion. **iii)** A standard TEM copper grid is sandwiched between exposed glass slides.

between the two glass slide sides that were exposed to the Pd/C powder. These as prepared sample grids were used for HRTEM imaging.

4.1.1.2 Experimental Image Acquisition

HRTEM imaging was conducted using a double-corrected JEOL ARM300CF at the electron Physical Science Imaging Centre (ePSIC), Diamond Light Source, Harwell Research Campus, UK. The instrument was operated at 300 kV, TEM-S mode, spot size 1 and a 150 µm CL-1 aperture at parallel beam illumination with aberrations [377] corrected to 28 mrad as shown in Table 4.1.

Table 4.1: Typical aberration values used for HRTEM imaging of the Au/Ge and Pd/C standard datasets, Typke notation [377,386].

Aberration	B_2 [nm]	A_2 [nm]	C_3 [µm]	S_3 [µm]
Typical value	< 55.3	< 56.6	(< 8.1)	< 3.73

Images were acquired on a Gatan OneView Model 1095 4096 x 4096 CMOS detector at magnifications between 300,000 – 400,000X and binned by 2 x 2 pixels resulting in a final image size of 2048 x 2048 pixels and a sampling size of $0.0824 - 0.0618$ nm/pixel respectively. To demonstrate the capabilities of the *nNPipe* approach in this work (see Chapter 5), a variety of scenarios were taken into account, *i.e.* in- and out-of-focus images, varying particle loads, favourable and unfavourable image contrast, *etc.* This was aimed to mimic a real experimental imaging scenario that could be encountered during an automated imaging procedure. In total, 93 and 96 images were considered for the Au/Ge and Pd/C dataset respectively. Figure 4.1 shows two typical HRTEM images of the samples.

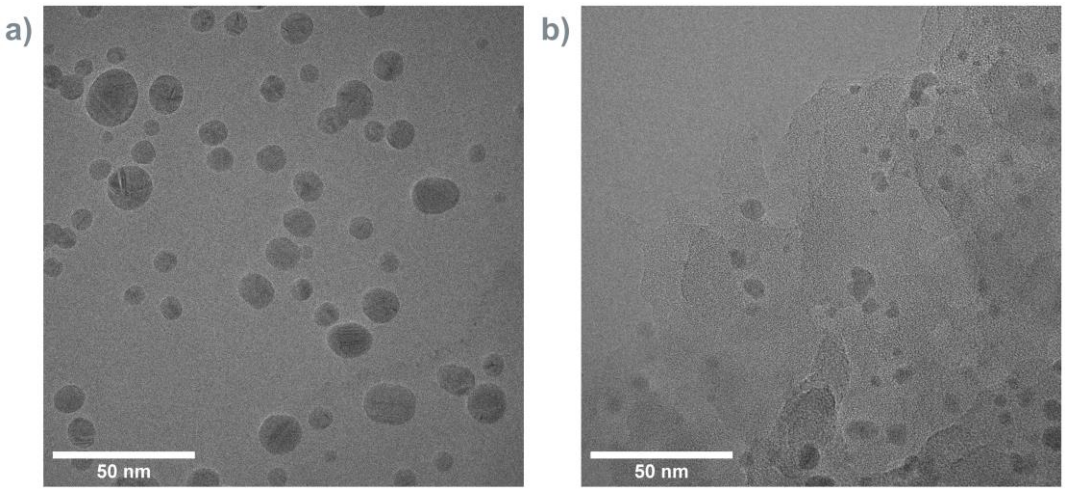


Figure 4.2: Typical HRTEM images acquired for two sample systems. **a)** Au/Ge system, total dataset size: 93 images. **b)** Pd/C system, total dataset size: 96 images.

4.1.2 Pd/C *in situ* Heating Experiments

4.1.2.1 Sample Preparation

In situ heating experiments were carried out using a pristine DENS Solutions Wildfire chip [96] compatible with JEOL sample holders. A new, dry sample deposition method was used for the dataset in this work. Previous experiments showed that the sample appearance of the Pd/C powder deviated from the one used for HRTEM imaging on standard lacy carbon copper grids when a wet preparation route using $CHCl_3$ was used. The dry deposition method comprised of the following steps:

i) A thin oar-shaped spatula was used to put a small amount of Pd/C powder on a clean optical glass slide.

ii) The powder was mechanically crushed by placing another glass slide on top of the powder with no additional force and careful circular motion.

iii) A pristine DENS Solutions wildfire chip [96] was taken up with a coated tweezer and carefully placed on the exposed surfaces of the exposed sides of the glass slides with SiN_x windows facing down. Subsequently, the chip was taken up again and carefully placed on another area of the exposed side of the glass slides. This was repeated several times.

iv) The chip was turned around and checked under the light microscope. If multiple SiN_x windows were covered with small flakes of the Pd/C powder, the deposition was considered complete. Otherwise, step iii) and iv) were repeated.

v) The prepared chip was mounted onto the TEM holder and checked for functionality.

A schematic illustration of the described method is additionally given in Figure 4.3 below.

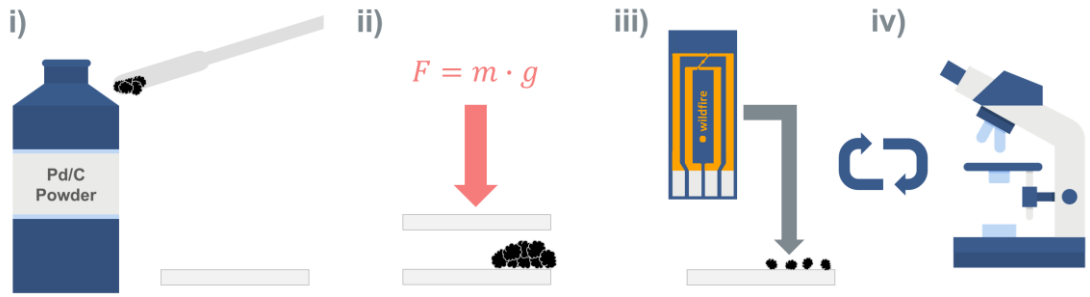


Figure 4.3: Schematic illustration of the dry sample deposition method for heating chips with numbers matching the steps in text. **i)** Pd/C powder is taken up using a thin spatula and placed on a glass slide. **ii)** Pd/C powder is sandwiched between two glass slides and mechanically crushed without additional force and careful circular motion. **iii)** Heating chip is repeatedly placed on crushed powder with SiN_x windows facing down. **iv)** Chip is checked under light microscope.

4.1.2.2 In Situ Imaging

In situ heating experiments were conducted on a probe-corrected JEOL ARM200F at ePSIC in HRTEM mode. The acceleration voltage was set to 200 kV in TEM-S mode at spot size 3, CL-1 aperture of 100 µm inserted and probe-corrector turned off. These settings were chosen as they formed a compromise between image contrast and beam-sample interaction over time: A lower spot size or bigger CL-1 aperture would have resulted in an improved SNR, at the cost of sample movement and sample vibration. At lower spot sizes and without CL used, the carbon and underlying SiN_x windows were furthermore found to disintegrate after several minutes of electron beam exposure. The settings above did not alter the sample visibly over the course of hours and even at elevated temperatures. Images and movies were acquired on a Gatan OneView Model 1095 4096 x 4096 pixel CMOS detector at varying magnifications. The images were automatically binned by 2 x 2 pixels during acquisition which led to a final image size of 2048 x 2048 pixels. Movies were acquired at 100 frames/s with additional details given in Chapter 7. Figure 4.4 shows two typical images acquired from a dry-deposited, *in situ* heated Pd/C sample.

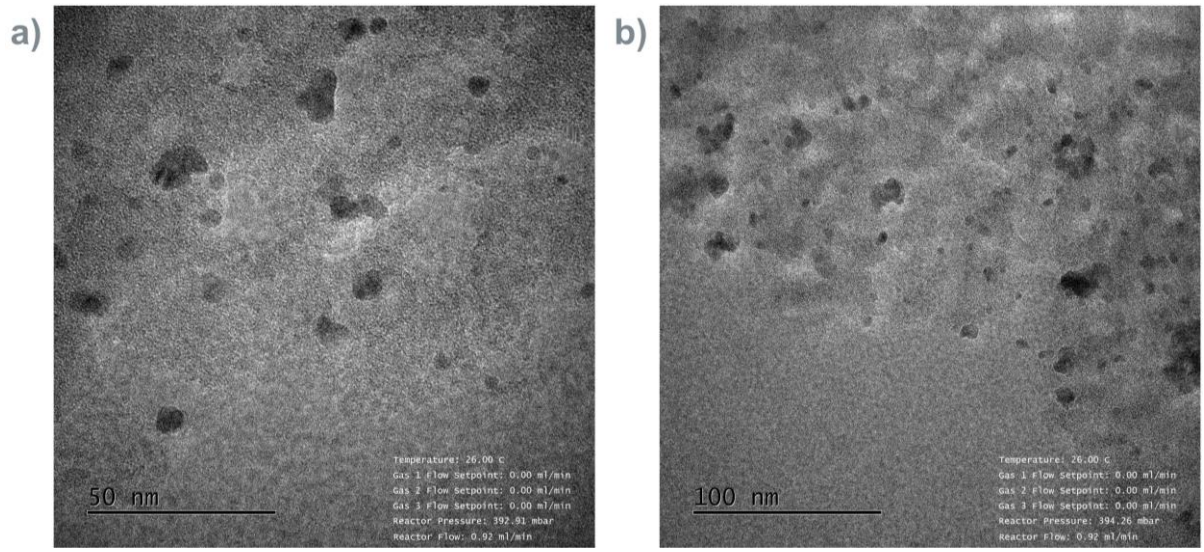


Figure 4.4: *In situ* TEM images of the Pd/C sample deposited on a heating chip. **a)** Image acquired at room temperature at 300,000X. **b)** Image of the same Pd/C sample at room temperature and 150,000X. Note that this image was acquired on a different sample position than **a)**.

4.2 Computational Methods

In this section computational methods used in the presented work are described with relevant code detailed in the Appendix.

4.2.1 Sample Model Generation

This section describes the application of DL-based computer vision methods for experimental HRTEM imaging. In order to obtain a high-performing neural network pipeline (see Chapter 5), individual network architectures need to be trained for generalisation on data not included during training, *i.e.* 'unseen' data. This can be achieved by using large training datasets [383]. Such datasets are usually rare for EM applications and are based on experimental images, which introduce an unintended, but inherent bias applied by experts manually annotating the ground truth labels. In this context, ground truth labels refer to the *true* target or measurement (here: binary map) that a neural network aims to approximate with minimal error and which are used during training or inference.

Herein, an alternative approach is pursued. Training data is obtained by constructing computational models mimicking various Au/Ge and Pd/C samples. These computational models were used as input to an HRTEM multislice algorithm to generate a large dataset, whereby the dataset size can be tuned by including different sample model geometries, aberrations and image augmentations. A further advantage from the ground truth annotation is that since computational models and multislice simulations are linked to each other the coordinates of nanoparticles and aggregates of nanoparticles (henceforth referred to as 'clusters') are known, and thus can be assigned as with absolute certainty. This effectively eliminates unwanted bias introduced by manual annotation in training. The only residual bias in computational/simulation based training dataset is given by how well the simulations approximate experimental images.

Sample models mimicking Au/Ge and Pd/C catalyst systems consisting of $\{x, y, z\}-$atom coordinate lists were hard coded in customised Mathematica 12.1 notebooks. Characteristic properties such as nanoparticle size and aspect ratio distributions were obtained by manually measuring *ca.* 200 nanoparticles per system with an elliptic approximation. The appearance of the amorphous support

material was modelled using a random distribution of points in 3D space with an average interatomic distance derived from material densities available in literature [387,388]. Morphologies of sample systems were approximated by repeated qualitative comparison of multislice simulations and experimental images. A flat sample support was assumed for the Au/Ge system, in contrast to the Pd/C system which exhibited a diverse sample support morphology. To obtain an adequate training data set resembling the carbon support material, 2D polyconic map projections of country outlines were processed using a random set of erosion blur and rotations. Several country polygons – henceforth denoted as 'countrygons' – were combined to form a unique shape and linearly extended into the third (z-coordinate) direction to form a 3D countrygon layer into which carbon coordinates were assigned. Up to five layers were stacked and slightly shifted in $\{x, y\}$-directions to create areas of thicker and thinner support material.

Nanoparticles were accounted for by including a weighted, random number of commonly observed shapes for respective catalyst systems, *i.e.* icosahedra, cuboctahedra, decahedra and ellipsoids into the model, whereby ellipsoids were given the highest probability as they were the most encountered in acquired HRTEM images. In addition, two to five nanoparticles were joined to account for clusters present in the experimental images with up to two clusters included in each computationally generated model. For nanoparticles and clusters, atomic coordinates were assigned following a perfect cubic F lattice symmetry and lattice parameters of $a_{Au} = 0.40782$ nm and $a_{Pd} = 0.38907$ nm respectively. The as-generated sample models were not subject to energy-state relaxation *via* Molecular Dynamics (MD) or Density Function Theory (DFT). Figure 4.5 illustrates two typical hull models of both catalyst systems while Figure 4.6 shows the stepwise construction process of an individual Pd/C model.

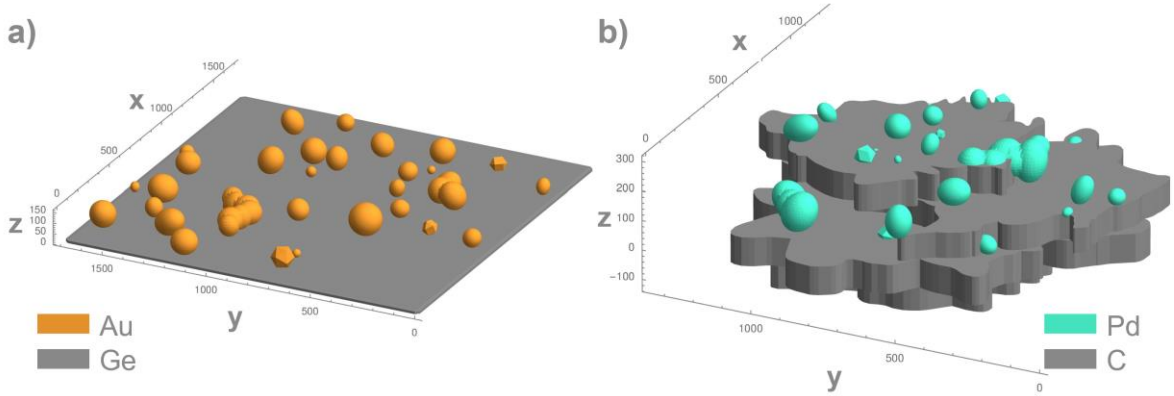


Figure 4.5: Examples of computationally generated models of respective catalyst systems. **a)** Au/Ge. **b)** Pd/C.

Table 4.2 summarises the sample properties assigned in the generation process. Respective Mathematica 12.1 codes with thorough code explanations are available in the Appendix Section 10.2.

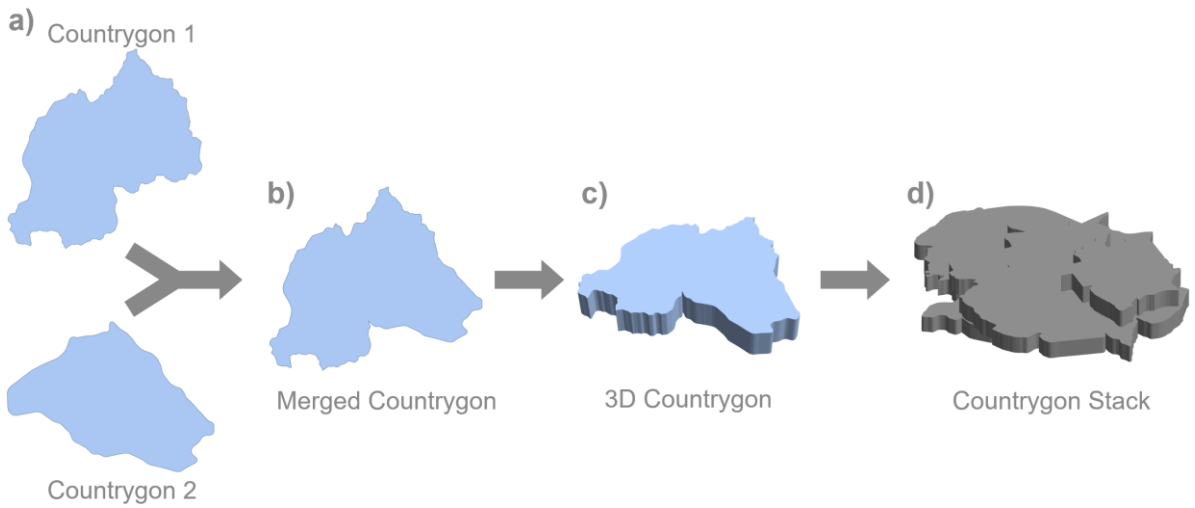


Figure 4.6: Stepwise construction of the carbon support in computational Pd/C models. Labels included in the steps **a)-d)** serve as step explanation supporting the main text (see above).

Table 4.2: Assigned properties for computational generation of considered sample systems.

Sample system	Au/Ge	Pd/C
Layer properties	$\rho = \left(3.96 \pm 0.08 \frac{\mathrm{g}}{\mathrm{cm}^3}\right)$[387] w = [100; 170] nm d = [1.0; 2.2] nm	$\rho = 1.7 \frac{\mathrm{g}}{\mathrm{cm}^3}$ [388] w = [100; 130] nm d = [4.0; 12.0] nm
Model properties	Number of layers = 1 Number of nanoparticles = [10; 25] Number of clusters = [1; 3]	Number of layers = [1; 5] Number of nanoparticles = [10; 25] Number of clusters = 2
Total model number	3,500	3,500

4.2.2 Multislice Simulations and Training Data

The sample models were used an input for the HRTEM multislice implementation MULTEM [389,390]. Although the low spatial frequency features (*i.e.* the shape of nanoparticles) aimed to be analysed with the DL-based method do not strictly require phase contrast simulations, their use raises two crucial advantages: (i) Experimental images (Section 4.1.1 and 4.1.2) clearly indicated atomically resolved features inside the metal nanoparticles. Hence, using adequate simulations provides additional image features for improved training and inference performance of neural networks. (ii) If other users decided to use this method at higher magnifications in the future, fast preparation of new training data using the exact same preparation steps is ensured.

Once the models were passed into MULTEM, each of the 3,500 models (see Table 4.2) was augmented five times with a randomised selection of aberrations that matched typical experimental conditions, as well as a randomised defocus spread modelling partial temporal coherence to introduce blurring into the image simulations. Furthermore, image simulation histograms were modified using a selection of experimental images and noise functions including varying SNR, Gaussian- and Poisson noise and beam-shape dependent variations in brightness were applied. In a final step, the sampling of the image simulations was adjusted to 0.0824 nm/pixel to match images in the experimental dataset (see 4.1.1). As computational models varied in size (Table 4.2), this last step led to varying size of image simulations. Applied aberrations and criteria for image simulations are summarised in Table 4.3.

Table 4.3: Applied aberrations, GLD criterion and total training dataset sizes for HRTEM multislice simulations.

Applied aberration values	$C_0 = [-5;\ -50]$ nm $C_{3,0} = [-0.05;\ 0.5]$ mm $A_1 = [0;\ 25]$ nm $A_{1,\theta} = [0;\ 360]$ ° $B_2 = [0;\ 50]$ nm $B_{2,\theta} = [0;\ 360]$ ° $A_2 = [0;\ 120]$ nm $A_{2,\theta} = [0;\ 360]$ ° $\sigma = [9; 16]$ nm	$C_0 = [-30;\ -160]$ nm $C_{3,0} = [-0.05;\ 0.5]$ mm $A_1 = [0;\ 25]$ nm $A_{1,\theta} = [0;\ 360]$ ° $B_2 = [0;\ 50]$ nm $B_{2,\theta} = [0;\ 360]$ ° $A_2 = [0;\ 120]$ nm $A_{2,\theta} = [0;\ 360]$ ° $\sigma = [9; 16]$ nm
GLD criterion	0.025	0.025
Images for Object Detection	140,000	140,000
Images for Semantic Segmentation	480,730	618,090

When comparing the image simulations to experimental images, it was noticed that some nanoparticles exhibited a significantly lower contrast with respect to the sample support than observed in the former. The first *nNPipe* training attempts (Section 4.2.3 and Chapter 5) showed that such instances would diminish the quality of the training data as the computer vision task became unclear and parts of the support material were falsely inferred as 'nanoparticle'. Potential mitigation of this effect could have been achieved through MD or DFT calculation-based energetic relaxation, but was not pursued due to the significant computational resources required. Instead, a minimum Graylevel-Difference (GLD) criterion measuring the contrast difference between individual nanoparticles and the support material was introduced. Individual nanoparticles with a GLD lower than a set empirical value would be removed from the original sample model coordinate list (Table 4.3). Pruned models were re-simulated with aberration and noise values matching the previous generation and formed the final training dataset. A comparison of a sample model before and after pruning is shown in Figure 10.1 (Appendix), while a comparison of the final models obtained with experimental images is shown in Figure 4.7. It is noted that the pruning step did lead to 'empty' sample models in some cases. All MATLAB code required to reproduce the image simulations presented is available in the Appendix (Section 10.3).

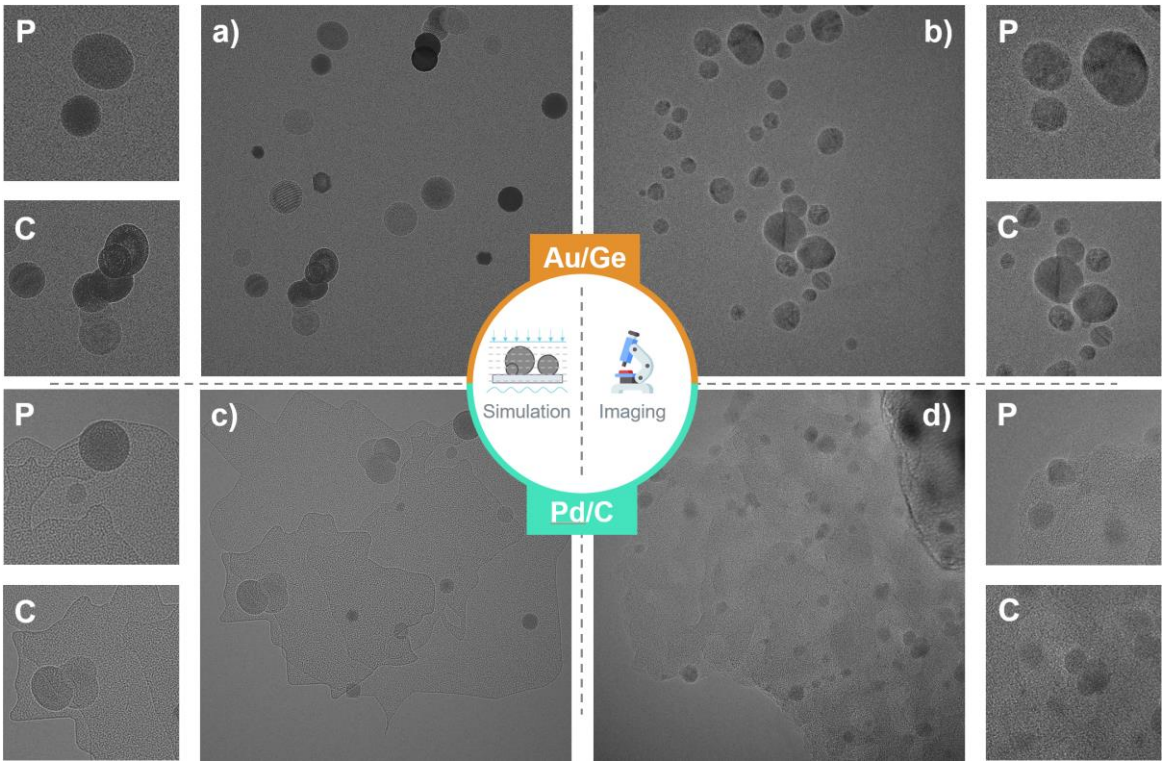


Figure 4.7: Comparison of typical HRTEM multislice simulations and experimental images. **a)** Au/Ge system image simulation. **b)** Experimental HRTEM image of Au/Ge. **c)** Pd/C image simulation. **d)** Experimental HRTEM image of Pd/C system. Additional examples of 'P' and 'C' image sections indicate typical examples of individual nanoparticles and

4.2.3 Network Architectures, Training and Inference

4.2.3.1 Architectures

The YOLOv5 [391] object detection CNN used in this work is based on its first implementation 'YOLO' introduced in 2016 by Redmon *et al.* [250]. In contrast to previously published work, this algorithm reformulates object detection as single regression problem. This imposes an S x S grid on each input image to predict classes and B bounding boxes with respective confidence scores if the centre of an object falls into a given grid cell. This can be expressed as $\Pr(Object) \cdot IOU_{pred}^{truth}$, whereby IOU refers to the Intersection Over Union (of the bounding boxes) and a confidence score of $\Pr(Object) = 0$ should be predicted if no object falls into a grid cell. Finally, C conditional class probabilities $\Pr(Class_i|Object)$ are predicted for each grid cell. Single passes are used for training and prediction phases exclusively, which gives the YOLO algorithm a significant advantage in processing speed [250]. An updated version of the YOLO algorithm, such as YOLOv3 [392] shows performance improvements by introduction of using predefined anchor boxes instead of grids which can be selected using k-means clustering amongst others [392,393]. The newest implementation at the time this research was conducted was YOLOv5 [391] which was chosen for its high inference speed, high inference performance and simple deployment on customised training data. The architecture was downloaded from https://github.com/ultralytics/yolov5 and the largest version of the network, namely YOLOv5x was used for all experiments and applications presented in this work. This version resizes all input images into a 640 x 640 pixels for both training and inference.

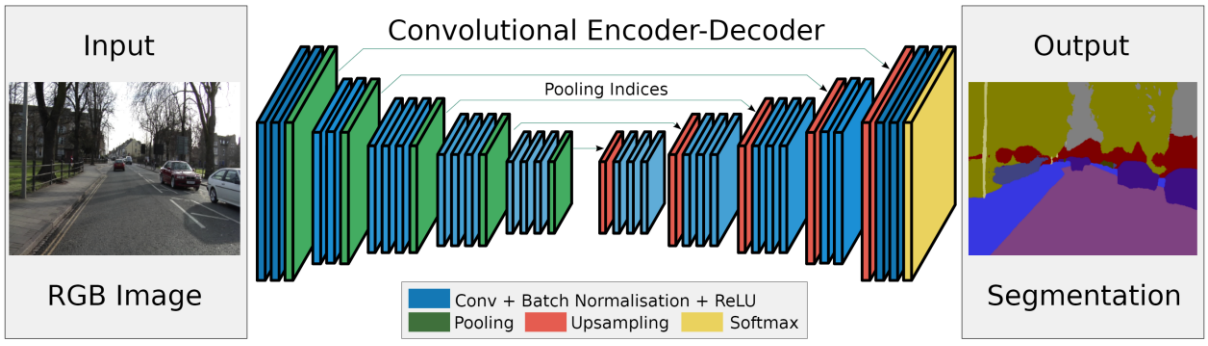


Figure 4.8: Schematic of the SegNet architecture from the original work of Nerd Badrinarayanan *et al.* [394]. The CNN was originally developed for RGB segmentation of image data from everyday scenes, but was tailored for the segmentation of nanoparticles by replacing the Softmax layer with a Logistic Sigmoid layer at the top of the network.

For semantic segmentation, the CNN architecture 'SegNet' developed by Badrinarayanan *et al.* [394] was chosen for its high performance and easy implementation. The network architecture (Figure 4.8) was rebuilt in a Mathematica 12.1 notebook (Appendix Section 10.4) while the final softmax layer was replaced by a logistic sigmoid layer adapted for a binary classification.

4.2.3.2 Training

The training data was split into $90/10$ % training and validation sets which were randomly chosen from the image simulations generated from each network (Table 4.3). Both networks were trained for individual material systems separately and *ab initio* on a NVIDA GeForce 2080 RTX Ti GPU with ADAM optimisation (see Section 3.4.4.4) and with a learning rate of 0.001 applied. During training, various performance metrics were automatically measured and used to determine the best state (*i.e.* weights) of the network. These included:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.1}$$

where TP refers to true positive instances and FP refer to false positives,

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.2}$$

in which FN denotes false negative instances. As an example, consider five instances in a given image. High Precision can be achieved by simply predicting a single instance correctly ($TP = 1, FP = 0$), while the other four would be omitted. Conversely, a high recall is achieved even if more than five instances are inferred by the network ($TP = 1$, $FN = 0$), as false positives are not taken into account. On their own, neither Precision nor Recall reflect the performance of a network, which is why both metrics are often considered together [382] as there is a trade-off between the two (*i.e.* the higher the Recall the lower the Precision). A useful metric to assess the performance of object detection networks is the mean Average Precison (mAP), which is defined as the mean precision of a set of equally spaced Recall thresholds r such as $[0, 0.1, 0.2, \dots, 0.9, 1]$ [395]:

$$\text{mAP} = \frac{1}{11}\sum_{r} Precision(r) \quad ; r \,\epsilon\, [0, 0.1, \ldots, 0.9, 1] \tag{4.3}$$

Using the YOLOv5x network, training was stopped after 100 epochs while weights with the best mAP@0.5 on the validation data were automatically saved for image inference. Training histories automatically generated by the YOLOv5x implementation for both material systems are provided in Figure 4.9 and 4.10. Applied hyperparameters, as well as code for full training dataset preparation are supplied in Appendix Section 10.4.

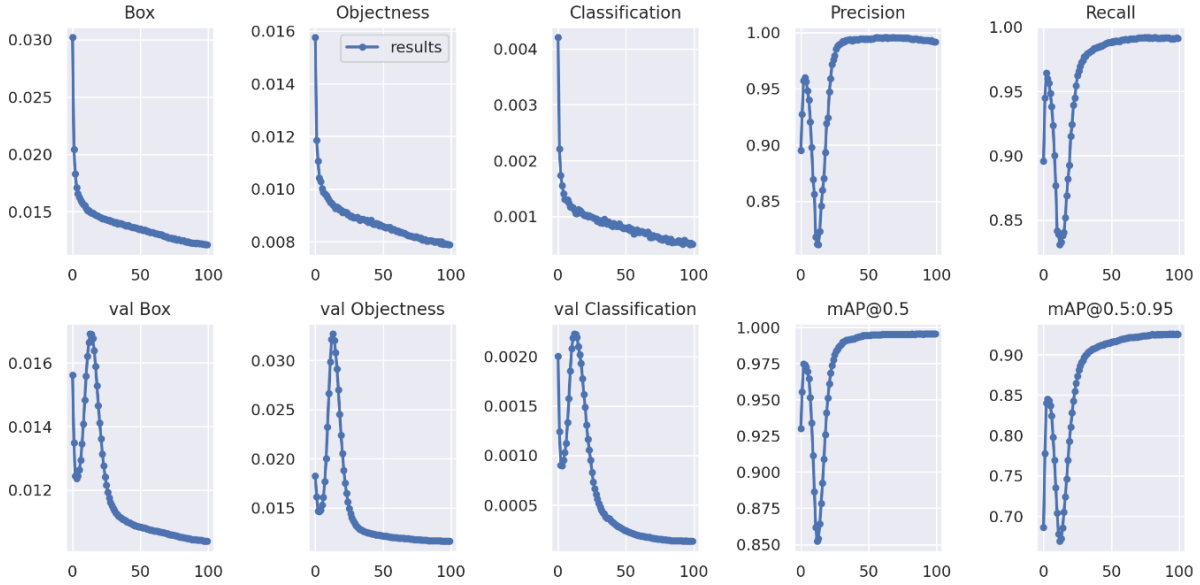


Figure 4.9: Au/Ge YOLOv5x training history plotted for various performance metrics. Weights with best mAP@0.5 using validation datasets were automatically saved for inference. All horizontal axes in the graphs refer to Epochs.

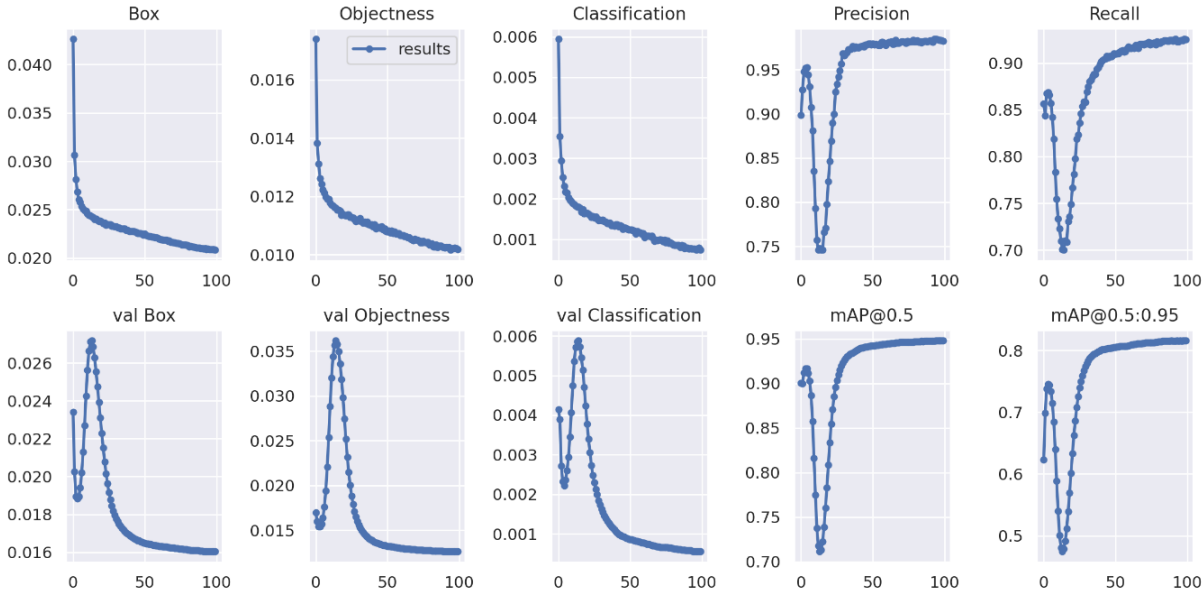


Figure 4.10: Pd/C YOLOv5x training history plotted for various performance metrics. Weights with best mAP@0.5 for validation datasets were automatically saved for inference. All horizontal axes in the graphs refer to Epochs.

The local minima observed for the metrics ‘Precision’, ‘Recall’, ‘mAP@0.5’ and ‘mAP@0.5:0.95’ in Figure 4.9 and Figure 4.10 may be interpreted as a short initial training period in which the network parameters were optimised unfavourably. This may have been caused by factors such as the values of the network weights generated during initialisation, applied hyperparameters such as learning rate, the training algorithm, or by the training data itself. Since the final weights used for inference (see Chapter 5.2.1) were taken based on highest achieved performance (determined by highest ‘mAP@0.5’) and a grid search potentially eliminating factors responsible for the observed minima would have been time-intensive, further investigation was not pursued.

For SegNet, training images of 256 x 256 pixels were extracted from previously generated image simulations using MATLAB scripts provided in Appendix (Section 10.4). The training was initially set to 100 epoch for each SegNet network, but stopped early once the validation loss did not improve any further or overfitting (Figure 4.11a) was recognised by increasing validation error. In the case of the Au/Ge system, this was achieved after 82 epochs with a loss of $2.77 \cdot 10^{-3}$ and $3.27 \cdot 10^{-3}$ for training and validation respectively, while for the Pd/C system the final losses were $1.73 \cdot 10^{-3}$ and $2.32 \cdot 10^{-3}$ for training and validation after 71 epochs. Figure 4.11 shows the training history of SegNet for both material systems.

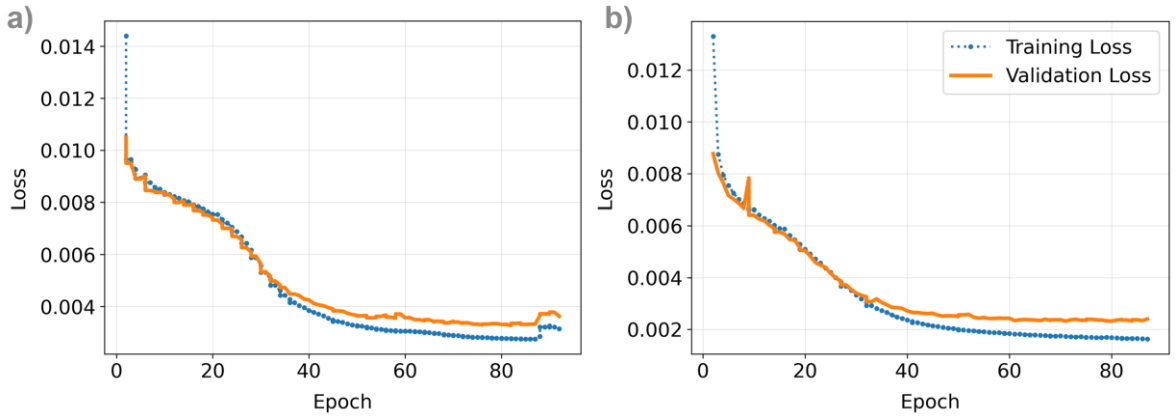


Figure 4.11: SegNet training history with training and validation loss shown as dependency of the training epoch for **a)** Au/Ge system and **b)** Pd/C system. The legend shown in **b)** holds true for both graphs.

4.2.4 Ground Truth Datasets and Inference Performance

4.2.4.1 Ground Truth Data

Ground truth datasets based on the HRTEM images acquired for both the Au/Ge and Pd/C systems were prepared by two domain experts (Cameron G Bell and Kevin P Treder) using manual segmentation in FIJI for performance measurements. Both experimental ground truth datasets contained a low number of cases where a subjective judgement for a segmentation was unavoidable and examples are provided in Figure 4.12 below. This dataset formed a basis for further alternation depending on the experiment analysed and corresponding remarks described in chapters 5-7.

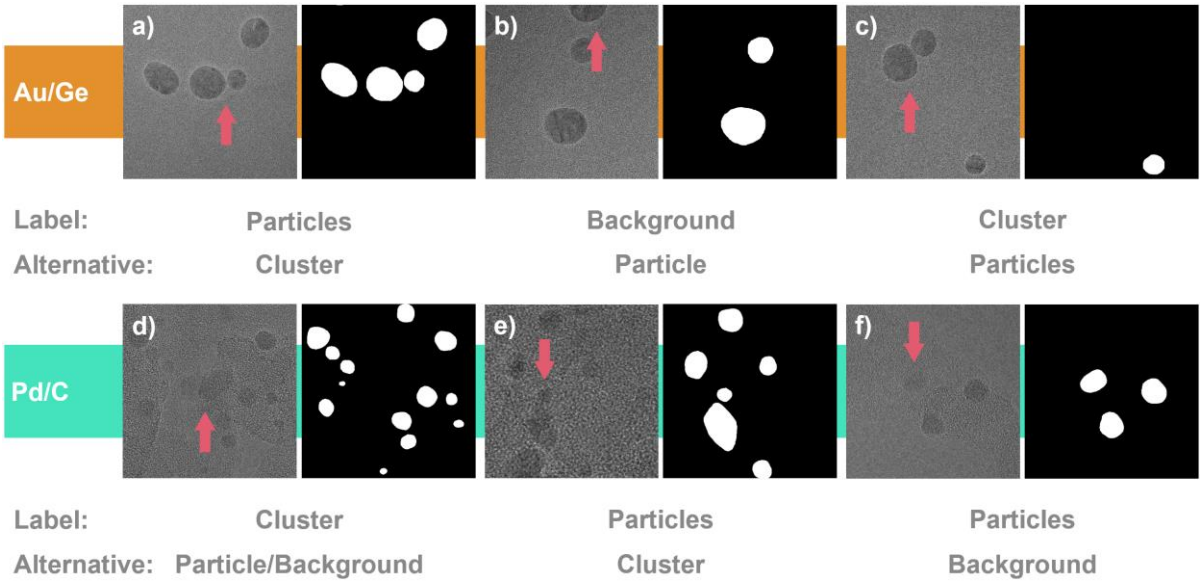


Figure 4.12: Selected examples of manual segmentation for experimental ground truth images where a subjective judgement on the assigned label was unavoidable. **a) – c)** Examples of the Au/Ge system and **d) – f)** Examples of the Pd/C system. 'Label' refers to the assigned label (as shown in the binary map), while 'Alternative' represents potential alternative labels that could have been assigned as well. The assigned and potential alternative labels are added below each example, while red arrows point at the respective instance in question. Note that decisions of this nature were more common in the Pd/C dataset.

4.2.4.2 Inference Performance

Unless otherwise stated, inference performances was measured by overlaying 2048 x 2048 experimental ground truth with inferred binary maps produced by *nNPipe* and calculating the pixel-based F1-Score:

$$\mathrm{F1} = 2\frac{Precision \cdot Recall}{Precision + Recall} = \frac{TP}{TP + \frac{FP + FN}{2}} \tag{4.4}$$

which reflects a harmonic mean of the Precision and Recall. Hence, the F1-Score is only high if both, Precision *and* Recall are high [382].

4.3 Chapter Summary

This chapter has summarised the key experimental and computational methods used. For the former, a detailed overview on sample preparation and HRTEM imaging has been given, while sample model and training dataset generation, as well as used CNN architectures and training procedures were described for the latter with codes added in the appendix.

5 nNPipe: A deep neural network pipeline for statistically significant metrics of heterogeneous catalyst systems

In this chapter, the development of a DL-based method for automated analysis of statistically significant metrics in heterogeneous catalyst systems is described. This method forms the core of the work presented and establishes a basis for further deployment strategies and applications described in subsequent chapters. Note that the approach discussed here, alongside parts of Chapter 4, are currently under review in Treder, Huang, Bell *et al.*, *npj Compu. Mat.*, second revision, 2022.

5.1 The nNPipe Method

The emergence of automated imaging routines and direct electron detectors (Section 2.1.7) set a new standard for data collection by enabling the acquisition of large data stacks over a wide range of sample positions at high temporal resolution. These advances have triggered immediate consequences for data analysis: Robust, high-throughput methods that require minimal manual intervention have become essential for accurate data analysis within reasonable timeframes. Traditional and pre-determined (semi-) automated approaches [396–402] fulfil those criteria only in a very limited fashion and thus, an increasing number of studies using the application of ML and DL approaches have been described (Section 2.2).

In the field of heterogeneous catalysis, it is important to understand the performance [403,404] of a catalytic system in various environments [405–409]. This is governed by material characteristics and dynamics such as nanoparticle size and morphology distributions or nanoparticle-support interactions, and commonly determined using (S)TEM. Analysis of HAADF images at statistically significant levels is often time-consuming and involves manual and instrument dependent post-processing of acquired images.

The application of DL-based approaches is well-suited to this challenge and many researchers have demonstrated successful application of object detection and semantic segmentation for low [282,410–

412] and high resolution images [257,260], liquid-cell *in situ* experiments [259,281] or compared the performance of different U-Net architectures [200,258,305]. However, in light of automated and fast data acquisition, several important aspects remain unaddressed. For instance, studies [258,282,305] are often based on topologically uniform samples in which nanoparticles are clearly discernible from the background. Furthermore, ground truths for training datasets are obtained by inherently biased manual labelling of experimental images, and workable array-sizes of individual images are linked to available computing resources which limits inclusive and wide-spread deployment.

This chapter introduces *nNPipe*, a DL-based method for an automated analysis of morphologically diverse heterogeneous catalyst materials. *nNPipe* is an abbreviation for nanoparticle ('NP') neural network ('nN') pipeline ('Pipe') and uses two standalone CNNs working in serial fashion to semantically segment raw 2048 x 2048 experimental HRTEM images of various sample morphologies and produces an output of the same size. CNNs, namely YOLOv5x [391] and SegNet [199], were trained on an aberrated and augmented training dataset of TEM image simulations derived from computationally generated models exclusively (Chapter 4). The workflow of *nNPipe* is fully described in Figure 5.1 below.

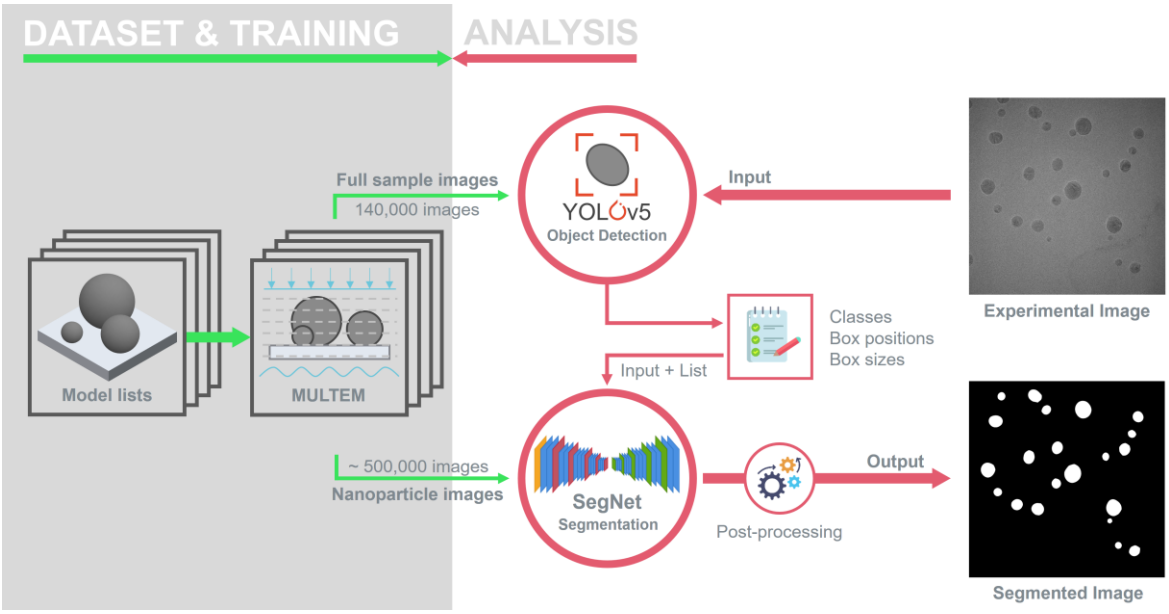


Figure 5.1: *nNPipe* training and inference workflow. Paths highlighted by green arrows indicate computational model generation, image simulation and individual training of the YOLOv5x [391] and SegNet [199] CNNs (see Chapter 4). Red arrows highlight the path of image inference: A raw experimental image (here an HRTEM image of Au/Ge) is passed to the YOLO object detection which predicts bounding boxes using 'nanoparticle' and 'cluster' as classes. This output is used to create 256 x 256 cropped sections of 'nanoparticle' instances in the raw experimental images. Cropped images are passed to the SegNet module where binary masks of individual nanoparticles are generated. Finally, bounding boxes and binary masks are treated through system dependent post-processing routines to obtain a 2048 x 2048 binary map.

Although approaches such as pure semantic segmentation or instance segmentation [413] offer potential alternatives for the processing of this data, a two-step pipeline was chosen for the following reasons:

i) *Precision*

Industrially relevant catalyst materials such as the Pd/C system described above contain nanoparticles that are challenging to discern from the support material. *nNPipe* processes experimental images as a whole and extracts relevant nanoparticle instances from a general context, which are subsequently semantically segmented at full image resolution.

ii) *Speed*

YOLO [250] and its more recent implementations such as YOLOv3 [392] and YOLOv5 [391] are distinguished by very high inference speed and performance (see Chapter 4). A targeted semantic segmentation of 256 x 256 image sections furthermore ensures low GPU load and fast processing.

iii) *Modularity*

The CNNs used in this work were trained on separate training datasets and are connected by simple processing functions. The networks can be consequently regarded as modules that can be deployed as standalone solutions depending on the desired output. In addition, the modules can be further developed and used with different image sizes (see Chapter 6).

iv) *Feasibility*

nNPipe is based on established and publicly available architectures with minimal modification. Instead of using a customised architecture for optimal performance, the performance of *nNPipe* is mainly governed by the training dataset, which allows for fast and user-friendly deployment.

nNPipe has been tested on HRTEM datasets containing different scenarios and sample morphologies that might be encountered in automated imaging (Chapter 4). Ramifications for materials characterisation are examined based on a comparison of binary maps and statistically significant metrics between the *nNPipe* output and a manually determined ground truths.

5.2 Results

5.2.1 HRTEM Image Inference

For inference, *nNPipe* was first used to process raw 2048 x 2048 experimental HRTEM images by the YOLOv5x object detection module that output images containing bounding boxes of predicted classes, alongside corresponding lists of vectors $\boldsymbol{d_i} = [c, x, y, w, h]$ for which the confidence threshold was set to 0.5. The variables were given in YOLO-format, *i.e.* $\in [0, 1]$, while c refers to the classes 'nanoparticle' or 'cluster'. The confidence threshold furthermore ensured filtering of instances which might be incorrectly recognised as relevant. In this work, the focus is set on a precise analysis of nanoparticles exclusively. Although both modules of *nNPipe* were trained on examples of 'nanoparticles' and 'clusters', a differentiation between the nanoparticles constituting a cluster in projection is challenging and requires tailored data analysis approaches as discussed by Frei *et al.* [414]. Instances inferred as 'cluster' or cut by image edges were consequently filtered after object detection or in post-processing steps (Figure 5.1) and manually labelled ground truths were modified accordingly (Chapter 4). In a second step, centred and cropped images of 256 x 256 pixels were extracted from raw experimental images using the bounding box information predicted by the object detection module and passed to the SegNet module for semantic segmentation. Lastly, segmented nanoparticle instances were subject to sample dependent post-processing routines to reconstruct a 2048 x 2048 binary mask. Meyer water-shedding was included for the Pd/C dataset to enhance nanoparticle separation from the surrounding support material with varying contrast. Full inference timings were independently measured for CNN modules and material systems and are summarised in Table 5.1. It is noted that the stated timings include the image writing process and differences between material systems are attributed to a longer post-processing for the Pd/C system. Relevant code for the generation of 2048 x 2048 binary maps, as well as a comparison of the different post-processing routines is provided in Appendix 10.5.

Table 5.1: Inference times for CNN modules used in *nNPipe* for the two considered material systems.

	Au/Ge	Pd/C
Experimental images	93	96
YOLOv5x inference time	16.72 s (0.03 s/image)	19.42 s (0.03 s/image)
SegNet inference + post-processing time	108.80 s (1.17 s/image)	280.92 s (2.93 s/image)
Total analysis time	125.25 s (1.35 s/image)	300.34 s (3.13 s/image)

Once datasets were fully processed by the *nNPipe* method, the inferred output was overlaid with a manually labelled ground truth and compared for TP, FP and FN pixels from which F1-Scores were calculated. Figure 5.2 shows typical cases of the output generated by the YOLO object detection module and colour-coded binary maps for performance evaluation for the Au/Ge system. Additional, close-ups of binary masks are given in the Appendix, Figure 10.2.

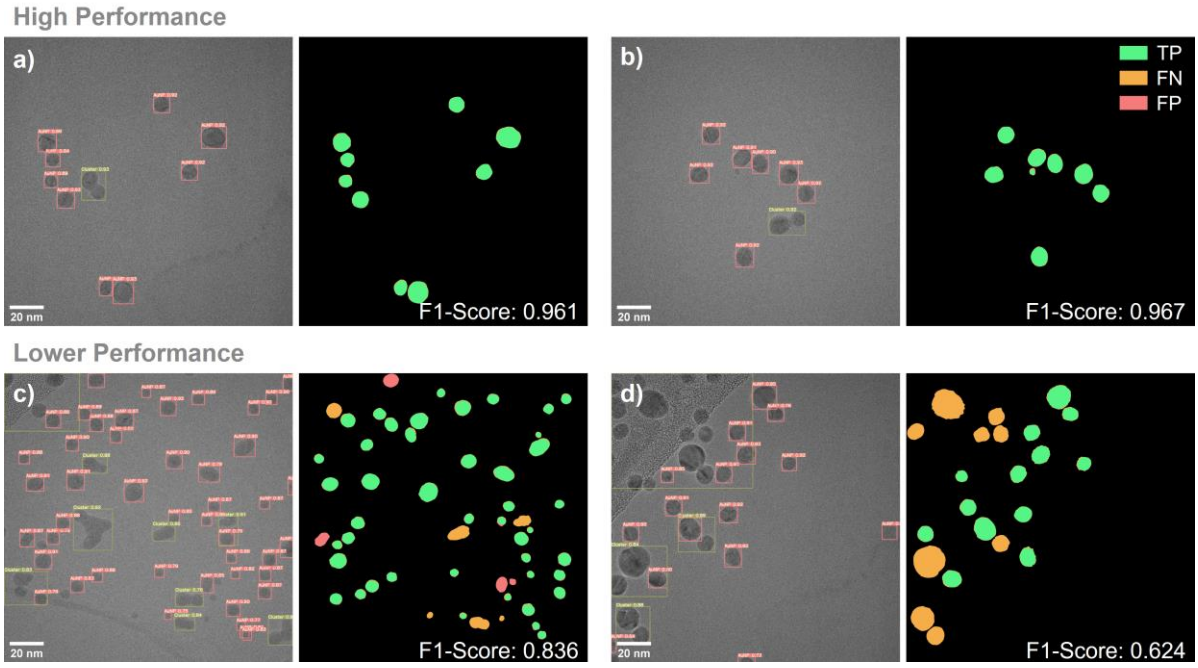


Figure 5.2: Selected examples of intermediate object detection output and colour coded binary masks for the Au/Ge system. **a)** and **b)** show cases of high inference performance, while **c)** and **d)** show lower performance. The legend provided on the top right of **b)** holds true for all segmentation masks. In the object detection output, nanoparticle instances are labelled as 'AuNP' (red boxes) while clusters are labelled as 'Cluster' (yellow boxes).

Examining high F1-Scores (0.961 and 0.967) and mostly TP contributions in Figure 5.2a and b, it is clear that the *nNPipe* method is able to deliver nearly identical segmentation masks compared the ground truth. Closer inspection (Figure 10.2, Appendix) reveals FN pixel contributions that can be associated with a border mismatch between the inferred output and ground truth labels. The total error contribution due to this mismatch is revisited in Section 5.2.2. 'Statistically Significant Metrics'. Figure 5.2c and d show examples of lower segmentation performance where the former nanoparticles cut by the image border were not correctly filtered in the post-processing stage or a 'cluster' instance was classified as 'AuNP' and hence led to FP contributions. Conversely, nanoparticles labelled as 'cluster' contribute to FN pixels in both cases. Unknown image features, such as the lacy carbon on the top left corner of Figure 5.2d are furthermore identified as a very large cluster and hinder the correct

segmentation of individual nanoparticles in that area. The observed inference of the object detection network can be understood by comparable greyscale values of the lacy carbon with respect to clusters and a significantly larger size of the instance as compared to single nanoparticles. Overall, the SegNet module of *nNPipe* achieves a high performance as identified instances are precisely separated from the background which also includes shapes differing from those provided in the training dataset, underlining generalisation capabilities.

By analogy with the Au/Ge system, the final segmentation performance was assessed for the Pd/C catalyst. A differing performance was expected since the Pd/C catalyst is more diverse morphologically and in dynamic range resulting in a more varied experimental dataset. Nevertheless, the *nNPipe* method yielded a high segmentation performance shown in Figure 5.3 below.

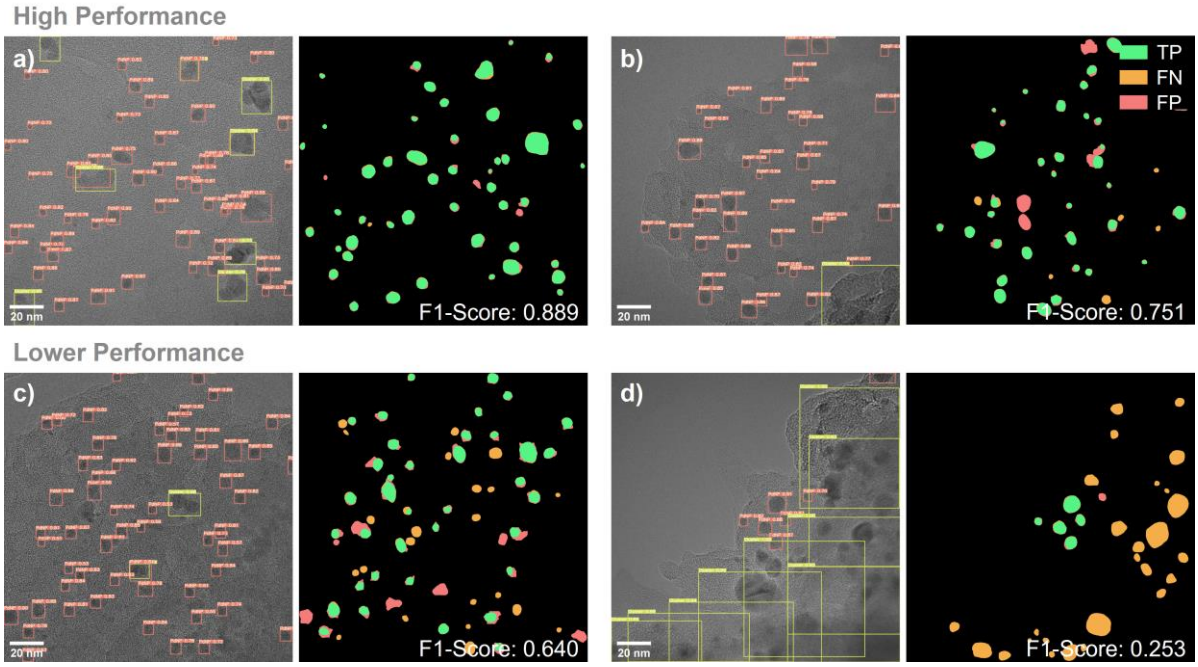


Figure 5.3: Selected examples of intermediate object detection output and colour coded binary masks for the Pd/C system. **a)** and **b)** show cases of high inference performance, while **c)** and **d)** show lower performance. The legend provided on the top right of **b)** holds true for all segmentation masks. In the object detection output, nanoparticle instances are labelled as 'PdNP' (red boxes) while clusters are labelled as 'Cluster' (yellow boxes).

The inference output shown in Figure 5.3a is very close to the experimental ground truth. All nanoparticle and cluster instances were picked up correctly by the YOLOv5x object detection module and accurately segmented subsequently (F1-Score 0.889), although similar border mismatches as in the case of Figure 5.2a and b can be observed (Figure 10.3, Appendix). Their respective contribution to the total error is revisited in Section 5.2.2. Several darker features of the amorphous carbon support material

were furthermore identified as nanoparticles, effectively contributing FP pixels to the segmented mask. Compared to the other examples provided in Figure 5.3, a more uniform background is present in this case which gives rise to improved contrast and hence segmentation performance. Despite stronger morphology variations in examples 5.3b-d, *nNPipe* is still able to achieve high F1-Scores as shown in Figure 5.3b (F1-Score 0.751). Almost all nanoparticles are detected and segmented correctly, whereby FP contributions arise from a cluster labelled as two nanoparticles in the object detection stage as well as some poorly resolved nanoparticle borders (Figure 10.3, Appendix). This example highlights the generalisation capabilities of the SegNet module once again, as the network is able to adapt to cases not included in the training data.

Lower inference performance is shown in Figure 5.3c (F1-Score 0.640) where higher rates of FN and FP can be attributed to a thicker support material, causing a lower contrast between nanoparticles instances and the background. This is partially reflected in the performance of the YOLOv5x object detection module, which misses several nanoparticles that result in FN contributions in the segmented mask. FN and FP contributions are furthermore linked to larger border mismatches (Figure 10.3, Appendix), while pure FP pixel instances arise due to misidentification of darker areas in the support material as 'nanoparticles'. A potential strategy to increase inference capabilities in the low contrast border area, and thus decrease the FN and FP contributions due to border mismatch, might be given by increasing the image noise contributions, such as Gaussian- or Poisson noise (see Section 4.2.2) in parts of the training dataset.

It is important to assess cases in which *nNPipe* does not perform well, to understand associated consequences for applications such as automated imaging. Hence, Figure 5.3d shows a low performance example for completeness with a sample position with thin, clear carbon surrounded by a thick area of support material. Nanoparticles located on the thin carbon area are segmented at high precision with mostly TP contributions visible (compare Figure 10.3, Appendix). This is in contrast to all nanoparticles located in the thick area of the support material that were misidentified as 'cluster' in the object detection stage and resulted in numerous FN pixel instances at segmentation. As also observed in the Au/Ge system (Figure 5.2d), unknown large instances with greyscale values comparable to individual

nanoparticles or clusters tend to be identified as 'clusters'. The experimental datasets were additionally analysed with established automated segmentation methods to establish a broader context for the analysis performance of *nNPipe*. Specifically, a routine of global and local Otsu thresholding (*i.e.* unsupervised threshold selection for image binarisation based on maximum seperability of different classes in gray levels)[415] featuring a rolling ball filter value of 600, Gaussian kernel size of 7, a local filter size of 50 pixel and a minimum nanoparticle size of 50 pixels was applied to both datasets after initial contrast inversion and without additional water shedding using the ParticleSpy [416] python package, as described in Bell, Treder, Kim *et al.* [417]. Generated outputs were subject to similar performance measurements as described above and Figure 5.4 compares historgram Kernel Density Estimation (KDE) plots of F1-Scores obtained by all methods. KDE plots were obtained using Scott's rule [418] that uses Gaussian kernels for automatic histogram bin width estimation *via:*

$$h = \sigma n^{-1/(d+4)} \tag{5.1}$$

where h equals the bin width, σ refers to the sample standard deviation, n denotes the sample size and d the dimensions. Note that Eq. 5.1 was simplified to match the used implementation in SciPy [419].

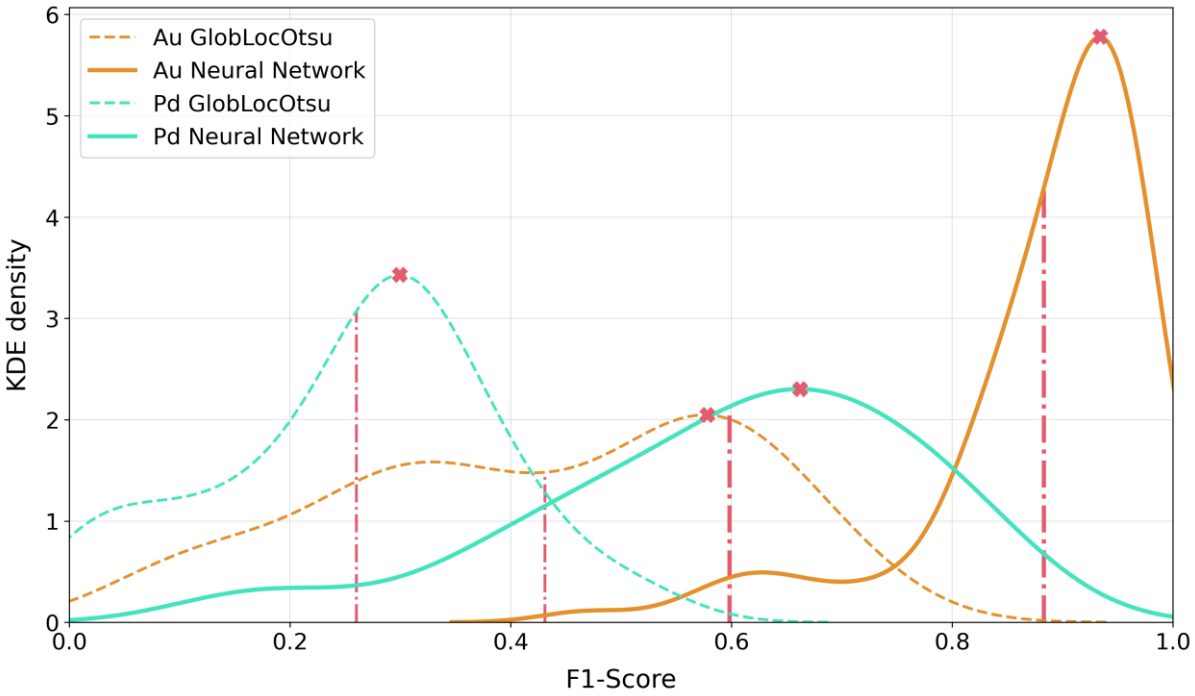


Figure 5.4: Comparison of KDE density plotted against F1-Scores for all segmentation methods considered. Plots were generated by using a Gaussian kernel and Scott's rule [418]. F1-Score means and maximum values are highlighted with dash-dotted red lines and red crosses respectively. **Total F1-Scores:** Au GlobLocOtsu = 0.431, Au Neural Network = 0.883, Pd GlobLocOtsu = 0.260, Pd Neural Network = 0.598. **KDE maxima:** Au GlobLocOtsu = 0.578, Au Neural Network = 0.934, Pd GlobLocOtsu = 0.300, Pd Neural Network = 0.663. For the materials systems considered, semantic segmentation using neural networks (*nNPipe*) yields higher inference performance.

Beginning with the performances for the Au/Ge system and segmentation methods taken into account (Figure 5.4), both the F1-Score and KDE maximum achieved with the *nNPipe* method are significantly higher than for the combined global and local Otsu thresholding. The KDE plot furthermore appears sharper for the *nNPipe* inference (denoted as 'Au Neural Network' in Figure 5.4), suggesting a high F1-Score probability around the KDE maximum of 0.934. Considering the Pd/C system, the F1-Score mean and F1-Score at maximum KDE yield higher values for *nNPipe* (denoted as 'Pd Neural Network' in Figure 5.4) than for combined global and local Otsu thresholding. However, the KDE plot follows a broader distribution and yields lower F1-Score values as compared to the Au/Ge system, which is attributed to a more varied analysis performance. Reciprocally, the *nNPipe* Pd/C distribution underlines the morphological diversity of the system (compare with Figure 5.2 and 5.3) and more varied imaging scenarios included in the dataset, and conclusively refers to a likelihood of inference errors. It can be clearly stated that for both, the Au/Ge and Pd/C datasets, higher performance in semantic segmentation is achieved by the *nNPipe* method and therefore justifies its use over more established approaches.

5.2.2 Statistically Significant Metrics

In order to understand the effects of the applied method to material characterisation in an automated imaging scenario, key material properties important to potential catalytic performance were measured on inferred output and ground truth labels. The following presents measurements of the nanoparticle area which were determined *via* the 'particle_analysis' function of the ParticleSpy [416] package. This function also allows consideration of other nanoparticle properties such as major and minor axis length, circularity or eccentricity to address the full spectrum of features relevant to the performance of catalyst materials. Figure 5.5 shows histograms of inferred and ground truth distributions with corresponding KDE-plots, as well as F1-Scores above set thresholds with calculated mean areas and count instances for the full Au/Ge dataset.

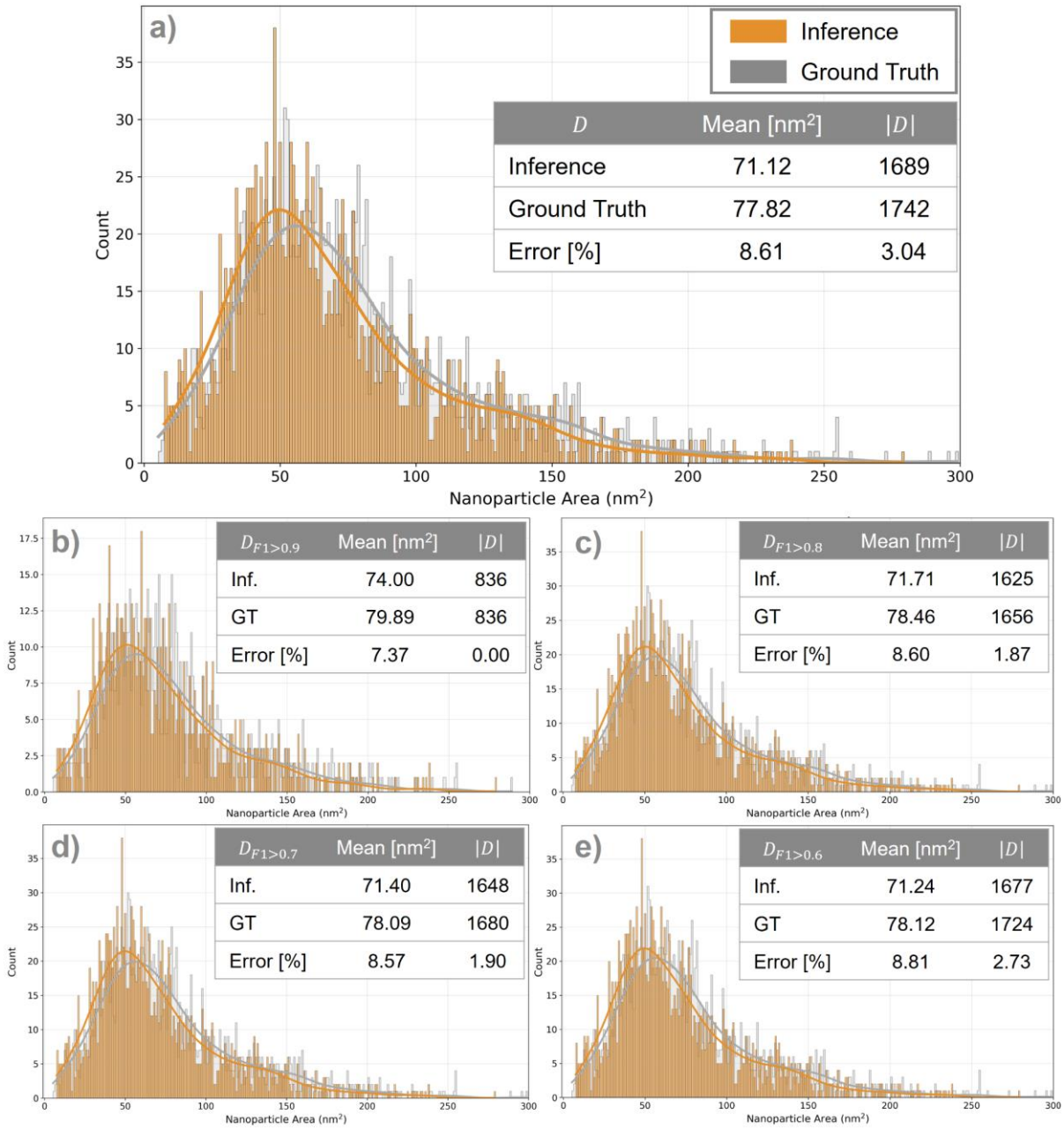


Figure 5.5: Nanoparticle area histograms (1 nm²/bin) with corresponding KDE-plots for inferred *nNPipe* output and ground truth labels of the Au/Ge dataset. All KDE-plots were generated using a Gaussial kernel and Scott's rule [418]. Tables summarising the calculated mean nanoparticle area (nm²) and nanoparticle counts ($|D|$) in F1-Score thresholded datasets are also shown inset. **a)** Full Au/Ge dataset. **b)** Images with F1-Score > 0.9. **c)** Images with F1-Score > 0.8. **d)** Images with F1-Score > 0.7 and **e)** Images with F1-Score > 0.6. Note that the smallest GT nanoparticle was measured at 5.29 nm², while the smallest inferred nanoparticle was 7.51 nm², which may hint at the smallest nanoparticle size measureable at given settings.

Generally, all errors for calculated nanoparticle areas and nanoparticle counts are low in all cases shown in Figure 5.5. The smallest deviation from the ground truth is observed for the dataset containing images with F1-Score > 0.9, shown in Figure 5.5b. While the nanoparticle count does not differ from the ground truth, the nanoparticle area differs by 7.37 % although this sub-dataset only takes images into account where *nNPipe* performed the best. Progressive inclusion of sub-datasets with lower F1-Scores (Figure 5.5c - e) results in an increase of the error to 8.81 % in the nanoparticle area and

2.73 % in the nanoparticle counts. For contextualisation, Figure 10.4 (Appendix) shows selected examples of HRTEM images that were inferred in these F1-Score intervals.

A close similarity of inferred and ground truth KDE-plots is evident for all cases shown in Figure 5.5. In addition, Table 5.2 summarises the most common nanoparticle areas (x_{ymax}) and full widths at half maximum (FWHM) for Figure 5.5b – e.

Table 5.2: Comparison of selected KDE-plot properties between ground truth (GT) and inferred datasets for the Au/Ge system.

F1-Score	1-0.9		1-0.8		1-0.7		1-0.6	
Property	x_{ymax}	FWHM	x_{ymax}	FWHM	x_{ymax}	FWHM	x_{ymax}	FWHM
Inference	51.16	10.15	49.79	21.15	49.79	21.45	49.79	21.89
GT	56.62	9.50	56.91	19.73	56.91	19.94	56.91	20.48

From this evaluation, it is clear that the FWHM between inferred and ground truth KDE-plots remain similar in all sub-datasets considered, while a nearly constant difference of *ca.* 7 nm^2 is associated with calculated x_{ymax} values. Revisiting the analysis conducted for Figure 5.2, the immediate question on whether this error mostly originates from nanoparticle border mismatches or pure FN and FP instances arises. Consequently, five pairs of inferred and ground truth images were revisited for F1-Score intervals of $[0.9, 1], [0.8, 0.9], [0.7, 0.8], [0.6, 0.7]$, whereby only instances consisting of mostly TP pixel contributions were considered to isolate the nanoparticle border mismatch error and to track the error evolution with decreasing segmentation performance. Table 5.3 summarises calculated results of the error estimation while Figure 5.6 displays typical image differences between inferred output and ground truth labels.

Table 5.3: Calculated error estimations for pure nanoparticle border mismatch between inferred output and ground truth labels for the Au/Ge dataset, based on determined nanoparticle areas in nm^2.

F1-Score	1-0.9	0.9-0.8	0.8-0.7	0.7-06
Inference	76.53	68.65	62.77	63.99
Ground Truth	84.35	76.34	68.35	70.18
Error [%]	9.27	10.07	8.16	8.82

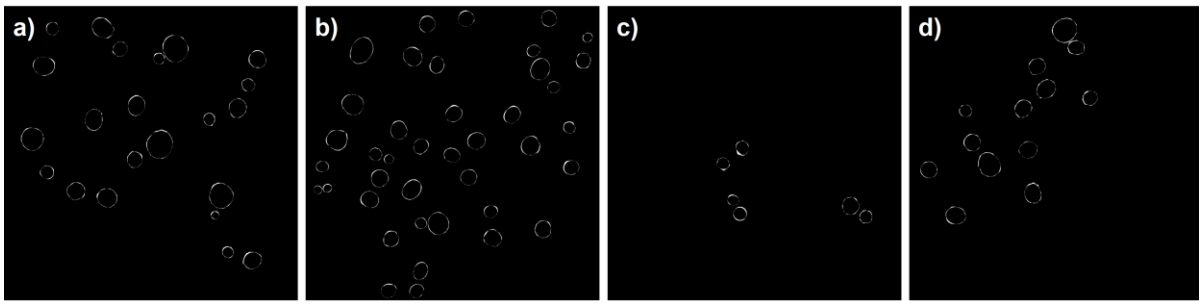


Figure 5.6: Selected examples of images generated for the border mismatch comparison across different F1-Score intervals. Images were obtained by determining the image difference between inferred output and ground truth labels for instances mostly consisting of TP pixel contributions exclusively. **a)** [0.9, 1]. **b)** [0.8, 0.9]. **c)** [0.7, 0.8]. **d)** [0.6, 0.7].

As summarised in Table 5.3, the border mismatch error remains constant in an interval between 8.16 % and 10.07 % for the sub-datasets considered. Additionally taking Figure 5.6 into account, a characteristic shape of the border mismatch is revealed where the inner edge appears considerably smoother than the outer edge. This observation is consistent with the analysis carried out for Figure 5.2, where small contributions for FN and FP pixels were found at the border of nanoparticle instances with high TP ratios. Based on this data, it can be stated that *nNPipe* slightly underestimates nanoparticle borders. This can be interpreted as a bias of the SegNet module originating from the training on HRTEM image simulations in which nanoparticles exhibited predominantly smooth surfaces, due to approximation of computational models with respect to the real catalyst sample (Chapter 4). Smooth surfaces are subsequently more likely to be imposed on identified nanoparticle instances during inference. Within the outlined error intervals, the error origin in the inferred data must predominantly originate from a borer mismatch between inferred output and ground truth labels and result into small histograms shifts towards smaller nanoparticle areas in conclusion. Similar magnitudes of mismatch errors can be equally expected for manual ground truth labels prepared by other human experts.

Analogous to the analysis conducted for the Au/Ge system, the measurement performance of *nNPipe* on the Pd/C system is shown in histograms and KDE-plots, as well as tables summarising mean nanoparticle areas and nanoparticle counts for various F1-Score intervals in Figure 5.7. As more images were inferred at lower F1-Scores (Figure 5.4), the deviation from the ground truth for calculated errors and the histogram is higher for the full dataset in the Pd/C system as compared to the Au/Ge system (Figure 5.7a). This is especially apparent when comparing Figure 5.7a and e, where the latter only

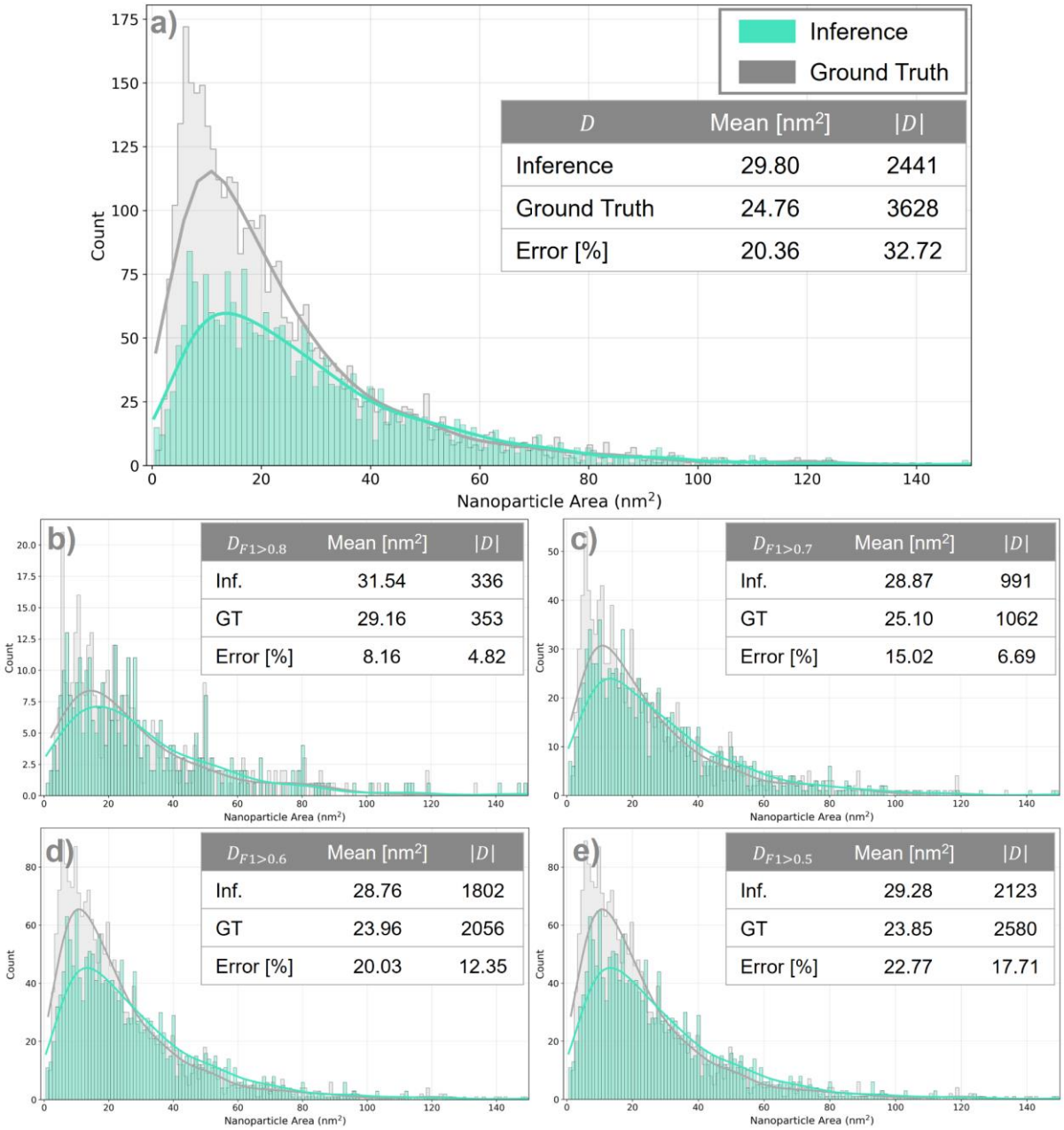


Figure 5.7: Nanoparticle area histograms (1 nm²/bin) with corresponding KDE-plots for inferred *nNPipe* output and ground truth labels of the Pd/C dataset. All KDE-plots were generated using a Gaussial kernel and Scott's rule [418]. Tables summarising the calculated mean nanoparticle area (nm²) and nanoparticle counts ($|D|$) in F1-Score thresholded datasets are added for completeness. **a)** Full Pd/C dataset. **b)** Images with F1-Score > 0.8. **c)** Images with F1-Score > 0.7. **d)** Images with F1-Score > 0.6 and **e)** Images with F1-Score > 0.5. Note that the smallest GT nanoparticle was measured at 0.726 nm², while the smallest inferred nanoparticle was 0.374 nm². This suggests that *nNPipe* is in principle not limited in segmenting smallest nanoparticles, but that FP contributions may also arise at this length scale, similar to others.

considers images of an F1-Score > 0.5. The error for the mean nanoparticle area remains comparable, while the error for the count is significantly higher for the full dataset. This can be explained by considering FN instances missed by wrong classification in the object detection stage, as highlighted for Figure 5.3d. The most prominent difference in KDE-plots can be seen for nanoparticle areas around 15 nm², which is consistent with local lower contrasts affecting smaller nanoparticles and resulting in

misclassified or undetected instances from poorer performance in the segmentation stage. On the other hand, errors remain significantly lower for the datasets with F1-Scores > 0.8 and > 0.7, especially for the nanoparticle count. Figure 10.5 (Appendix) displays several examples of images inferred in the various F1-Score intervals. Similarly, high inference performance is also reflected in the shape and peak positions of KDE-plots summarised in Table 5.4 below.

Table 5.4: Comparison of selected KDE-plot properties between ground truth (GT) and inferred datasets for the Pd/C system.

F1-Score	0.9-0.8		0.9-0.7		0.9-0.6		0.9-0.5	
Property	x_{ymax}	FWHM	x_{ymax}	FWHM	x_{ymax}	FWHM	x_{ymax}	FWHM
Inference	17.06	38.71	13.23	34.80	13.23	32.69	13.23	32.69
GT	14.36	32.76	10.50	27.67	10.45	24.08	10.45	24.08

Calculated values of most common nanoparticle areas x_{ymax} and KDE-plot FHWM settle at 13.23 and 34.80 for the inferred, and 10.45 and 24.08 for the ground truth data respectively once sub-datasets with F1-Score > 0.6 are included. Although these differences arise due to an overestimation from the *nNPipe* method, a high performance can nonetheless be attributed, particularly when respective shapes of both KDE-plots are taken into account (Figure 5.7).

A deeper understanding of the error origin was attained by separating the border mismatch error from entire FN and FP instances as described for the Au/Ge system above. Table 5.5 shows calculated nanoparticle border mismatch errors, while Figure 5.8 shows selected examples of image differences in the F1-Score intervals $[0.8, 0.9], [0.7, 0.8], [0.6, 0.7]$ and $[0.5, 0.6]$ respectively.

Table 5.5: Pure nanoparticle border mismatch error estimations based on determined nanoparticle areas in nm^2 for inferred output and ground truth labels.

F1-Score	0.9-0.8	0.8-0.7	0.7-0.6	0.6-0.5
Inference	30.37	22.87	28.31	27.30
Ground Truth	29.16	23.70	25.18	23.58
Error [%]	4.14	3.50	12.43	15.77

Compared to the border mismatch edges shown in Figure 5.6, the ones in Figure 5.8 have a more varied appearance. Smooth shapes indicating network bias due to approximations in the training dataset can be recognised for both inner and outer edges of the border mismatches and some cases exhibit thicker

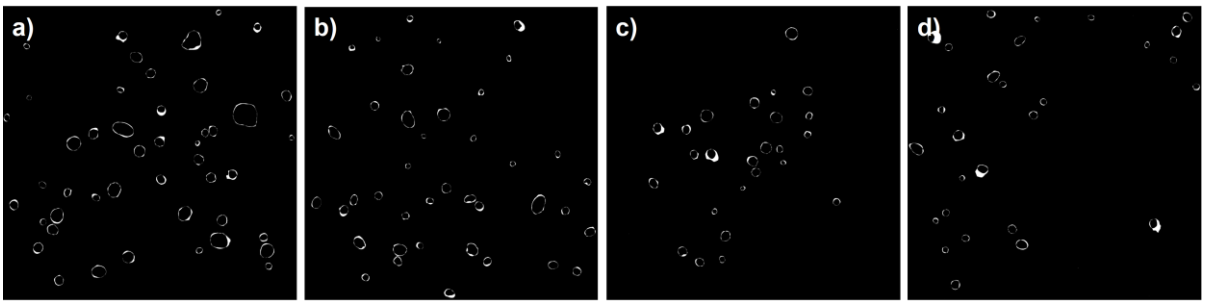


Figure 5.8: Selected examples of images generated for the border mismatch comparison across different F1-Score intervals in the Pd/C dataset. Images were obtained by determining the image difference between inferred output and ground truth labels for instances mostly consisting of TP pixel contributions exclusively. **a)** [0.8, 0.9]. **b)** [0.7, 0.8]. **c)** [0.6, 0.7]. **d)** [0.5, 0.6].

bulges (Figure 5.8). Their origin can be explained by contrast variations between individual nanoparticles and the supporting amorphous carbon that were not included in the training of the SegNet module of *nNPipe*. At the same time, a low average error of 3.82 % is obtained when considering all datasets with F1-Scores > 0.7 (Table 5.5). This indicates that a significantly larger proportion of nanoparticle area error (Figure 5.7) is due to pure FN and FP instances contributing to the statistics. Additionally, border mismatch errors rise to 12.43 % and 15.77 % for sub-datasets with F1-Scores > 0.6 and > 0.5 respectively.

5.2.3 Deployment in Automated Imaging

As it has been shown in Section 5.2.2, it is possible to obtain statistically significant metrics of thousands of nanoparticles at high inference performance and speed using *nNPipe*. However, it is emphasised that the results obtained were compared against manual ground truth labels to measure performance for different sub-datasets.

In the context of an automated imaging scenario, it is necessary to consider various sample morphologies (Chapter 4), and also account for the fact that is impossible to conduct F1-Score-based image selection, as a ground truth is usually not available. A reasonable estimate can however be given by comparing the number of instances detected during YOLOv5x object detection and SegNet semantic segmentation stages. If the Instance Difference (ID) is below an empirically set threshold, a higher probability for mostly *TP* nanoparticle instances should follow. It is introduced as:

$$\mathrm{ID}_i = \left| o_i^{\mathrm{YOLOv5x}} - o_i^{\mathrm{SegNet}} \right| \tag{5.2}$$

where o_i is number of output instances of the YOLOv5x or SegNet module, respectively. Here, only the class i 'nanoparticles' is considered, consistent with previous discussion. In Figure 5.9 and Figure 5.10, datapoints representing individual HRTEM image inference F1-Scores are plotted against the respective calculated ID, both obtained from the corresponding *nNPipe* output.

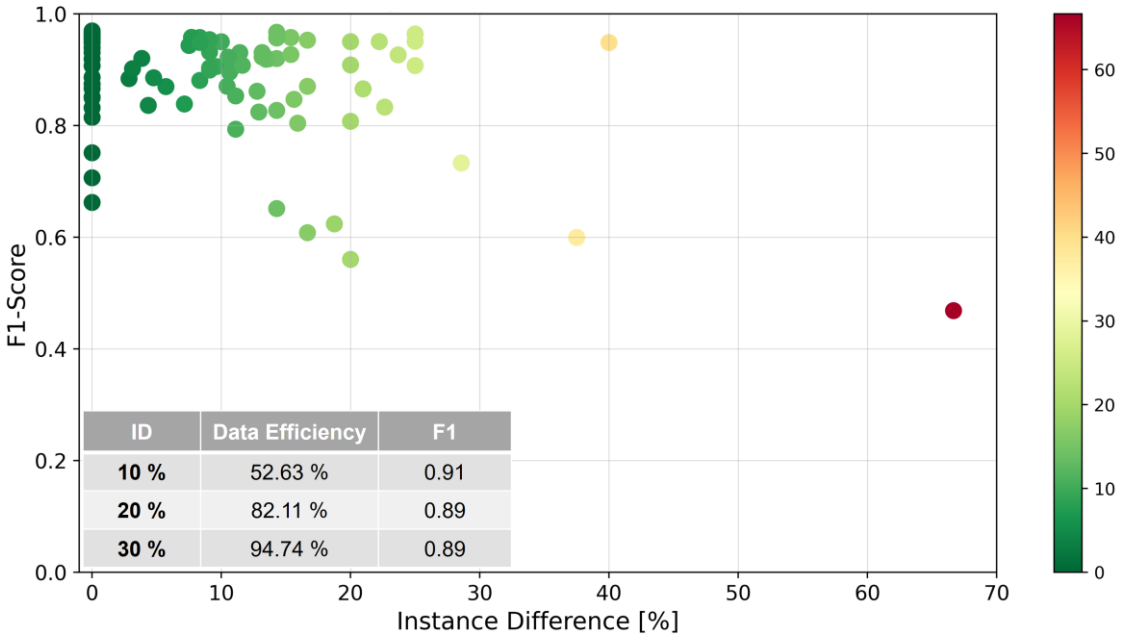


Figure 5.9: Point-plot of F1-Scores obtained for the *nNPipe* inference of individual HRTEM images in the Au/Ge dataset plotted against the Instance Difference (ID) given in %. The table inset indicates data efficiencies and average F1-Scores for different thresholds of the ID applied. The added colour scale was added to support visual distinction.

The analysis in Section 5.2.2 already showed particularly high F1-Scores achieved by the *nNPipe* method for the Au/Ge dataset. Using the ID is therefore not a strict requirement, but a set threshold of ID = 10 % leads to an improvement of the average F1-Score to 0.91. On the other hand, only 52.63 % of the total experimental dataset would be considered in this scenario, while the F1-Score would only improve by 0.03.

This is in contrast to the Pd/C system for which the ID was originally implemented and where a broader F1-Score distribution exists across the HRTEM dataset. Setting the ID threshold to 7.5 %, the overall F1-Score increases to 0.70, which is over 0.1 higher than for the full dataset and thus a more significant improvement than seen for the Au/Ge system. This boundary condition would be applicable to 40.82 % of the total images acquired, but would meanwhile ensure a more close-to-human performance measurement in automated imaging (Section 5.2.2)

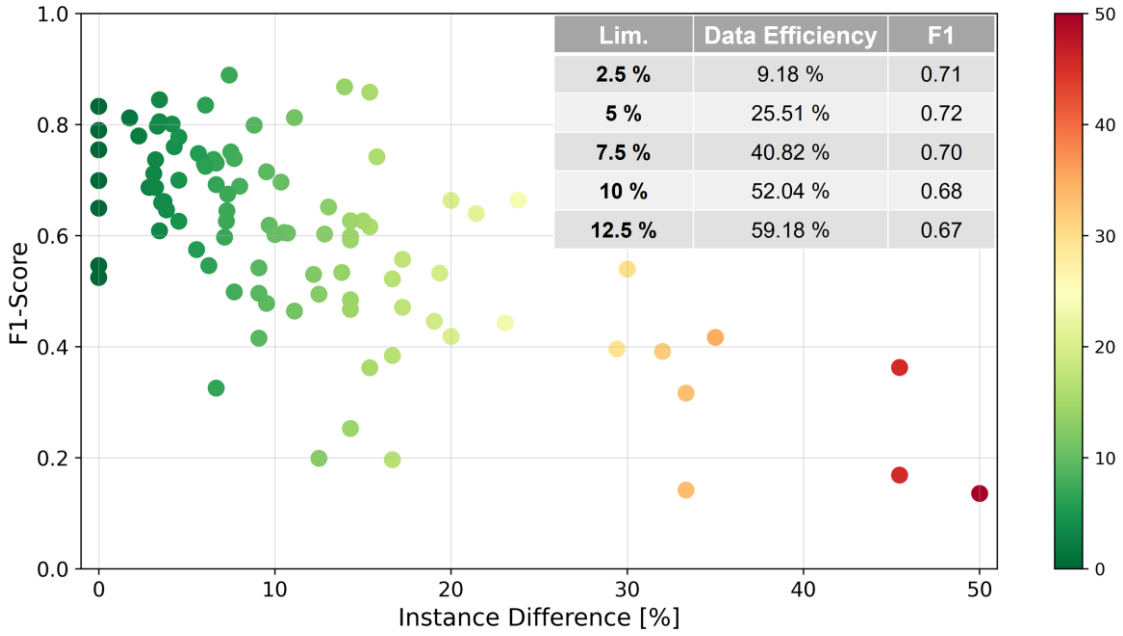


Figure 5.9: Point-plot of F1-Scores obtained for the *nNPipe* inference of individual HRTEM images in the Pd/C dataset plotted against the Instance Difference (ID) given in %. The table inset indicates data efficiencies and average F1-Scores for different thresholds of the ID applied. The added colour scale was added to support visual distinction.

5.3 Discussion

Image inference performance as shown in Figures 5.2. and 5.3 reveal the possibility to train different neural network architectures for the analysis of experimental HRTEM images using computational models and multislice simulations [389,390] exclusively. Most significantly, this strategy is effective for morphologically diverse heterogeneous catalyst systems such as Pd/C, although the sample complexity was simplified during modelling (Chapter 4). Nevertheless, application of the *nNPipe* method yielded F1-Scores higher than 0.85 and illustrates the potential of DL-based approaches of generalisation. Importantly learned tasks can be applied to images differing from the ones provided in training with high performance.

Based on the results presented, the following three advantages of this training strategy can be summarised:

i) It is possible to tailor sample models for individual purposes

ii) Exact image-label pairs can be extracted from image simulations

iii) There are virtually unlimited training images and models that can be generated.

In contrast, it must be noted that steps including the sample model generation, multislice simulation and augmentation, as well as training of the individual network require a considerable amount of time before the method can be employed for the interpretation of experimental HRTEM images. An image analysis based on other ML methods or shallow networks [258,417] that have been shown to perform well on image datasets comparable to the images of the Au/Ge system might thus be advantageous for smaller datasets.

Certain limitations of the *nNPipe* method have been highlighted in the context of the various results discussed in 5.2.2. For instance, very thick support material included buried nanoparticles and nanoparticles with unclear borders. These posed a challenge for accurate detection by the YOLOv5x object detection module and resulted in higher *FN* particle rates during semantic segmentation. As adequate scenarios were not included in the in the image simulations and training datasets the modules of *nNPipe* could not address these instances for experimental images appropriately during inference.

Similarly, this holds true for the nanoparticle shapes that exhibited a larger variety in appearance in the experimental images compared to the training dataset. Although the SegNet module showed clear generalisation capabilities towards unexpected input, the resulting border mismatch between the ground truth and *nNPipe* output can be understood as bias in the neural network (compare 5.2.2). A potential expedient to this is the application of retraining strategies and has been conducted for the method and datasets considered. Corresponding results and discussions can be found in Chapter 6.

A particularly high performance was achieved for the Au/Ge dataset which featured a very high similarity of histograms and KDE-plots (Figure 5.5). Sub-datasets with F1-Scores > 0.9 and > 0.8 showed counting errors below 2 %, while nanoparticle areas differed only by 7.37 % and 8.60 % respectively (Figure 5.5b and c). In conjunction with the analysis outlined in Table 5.3 and Figure 5.6, it is clear that the *nNPipe* method achieves close to human-like performance for datasets with uniform sample morphology and during automated image acquisition.

High inference performance was also demonstrated for HRTEM images of the morphologically diverse Pd/C system. However, for this system inclusion of images with large areas of very thick carbon support material should be avoided. Measurement performance remains close to the one for the Au/Ge system for images with F1-Score > 0.7 (Figure 5.7), while higher deviations from the ground truth become more likely for images with F1-Scores < 0.7. This is also shown in Figure 5.3c and d where respective field of views include areas of thicker carbon support material that washout individual nanoparticle contrast and lead to higher rates of border mismatches and pure FN and FP contributions. Hence, a close-to-human performance can only be expected in cases similar to Figures 5.3a and b. Additional tools such as the ID criterion can improve data selection and thus high performance for automated imaging scenarios.

In summary, the *nNPipe* method demonstrates a successful DL-approach for automated analysis of morphologically diverse heterogeneous catalyst systems. Relatively few computationally generated models were required to obtain large synthetic datasets that approximate experimental HRTEM images to train *nNPipe* towards high analytical performance. Assuming ~ 30 minutes for a full manual segmentation of an experimental image, *nNPipe* provides a significant advance in efficiency as it merely

required 125.25 s (2.08 min) and 300.34 s (5.01 min) for the analysis of a Au/Ge and Pd/C dataset containing 93 and 96 HRTEM images respectively. This equates to a speed increase of 1100 x and 580 x.

5.4 Chapter Summary

In this chapter, the core method of this work has been described. *nNPipe* is a DL-based approach for the automated analysis of morphologically diverse heterogeneous catalyst systems based on two networks. The method's context and functionality has been presented and the measurement performance was assessed using HRTEM images of a test Au/Ge and a real Pd/C catalyst system. Ramifications for the measurement of statistically significant metrics of selected nanoparticle properties were analysed and discussed in the context of an automated imaging scenario. Lastly, the methods functionality was considered from a practical point of view and a deployment strategy to yield high-performance metrics was suggested.

6 Retraining and Adaption Strategies

Herein, retraining and adaption strategies are described based on the *nNPipe* method introduced in Chapter 5. Section 6.1 discusses aspects of retraining and how it can further improve the performance and applicability of *nNPipe*, and Section 6.2 highlights a new training dataset preparation strategy based on public annotation carried out on the Zooniverse platform.

6.1 Neural Network Retraining

nNPipe is a highly effective and precise method for automated analysis of morphologically diverse heterogeneous catalyst systems. However, the necessary sample modelling, dataset generation and network training impose a large overhead. It takes considerable effort and time to obtain adequate networks if *nNPipe* were to be applied on a related analysis problem. Other methods such as shallow networks [258,417] or unsupervised learning offer potential alternatives [276–279].

On the other hand, it is possible to tune neural networks for tasks related to the original training dataset and hence adapt them to a new domain. This process is called retraining for which different strategies can be applied. The popular approaches of *transfer learning* and *fine-tuning* [420] are applied on *nNPipe* and the results with corresponding conclusions for widespread use are discussed subsequently.

6.1.1 Principles

In *transfer learning*, features learned by a neural network on one problem are reused on a new, similar problem. *Transfer learning* can be applied in scenarios where training datasets are too small to conduct *ab initio* training and is realised by disabling parameter updates during retraining for specific layers (‘freezing’) which normally involves all but the top layer or newly added ones. The top layers are subsequently trained using new data and learn to make predictions by using mostly ‘old’ features [420].

Fine-tuning on the other hand consists of retraining the full network at a significantly lower training rate than carried out originally. This allows for incremental adaption of pretrained features in the whole network to a new dataset [420]. Figure 6.1 shows a schematic differentiation.

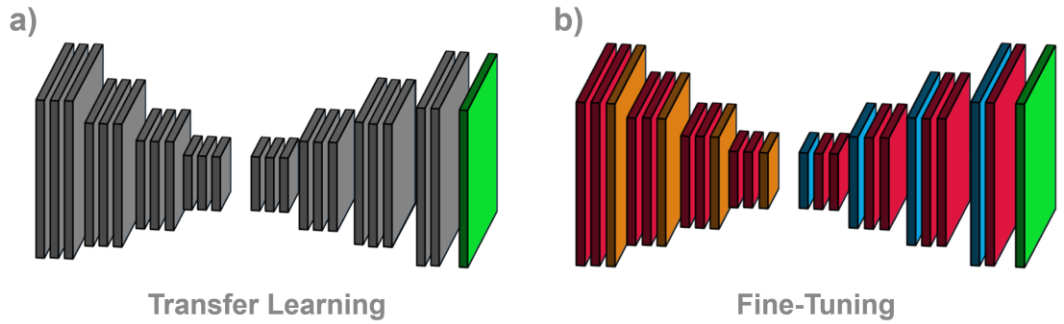


Figure 6.1: Schematic comparison between **a)** *transfer learning* and **b)** *fine-tuning* based on a typical encoder-decoder CNN architecture. Gray layers refer to layers of the network which are 'frozen' during training, *i.e.* where parameter updates are not allowed. Coloured layers refer to network layers which are updated during retraining. Fine-tuning can also be considered as an optional step after transfer learning [420].

Since *nNPipe* was trained on a large training dataset obtained from computational models, retraining the individual models appears to be suitable approach for several reasons:

i) Investigations on performance changes after retraining may have important implications for users willing to leverage pretrained networks, especially since large training datasets are not common for DL-based methods in EM.

ii) Specific imaging scenarios such as small, low contrast nanoparticles or nanoparticles distributed amongst thick layers of support material posed a considerable challenge for the networks trained on the Pd/C dataset. Retraining *nNPipe* may extended analytical capabilities, especially for imaging scenarios not included during training.

iii) Bellos *et al*. [421] demonstrated that relatively small training datasets can be sufficient for effective knowledge transfer. If similar approaches are applicable for the HRTEM images in this work, the use of *nNPipe* and other DL-based methods in EM may become much more practicable as the time required for data preparation and training would be reduced.

For the above reasons, the application of *transfer learning* and *fine-tuning* on *nNPipe* were individually investigated for the Au/Ge and Pd/C system.

6.1.2 Retraining Methods

The retraining methods applied here are selected by considering a realisitic retraining scenario, in which a neural network (pipeline) is retrained for a specific dataset in a time-efficient, yet performance-oriented manner. Building on the results of Bellos *et al.* [421], retraining *via fine-tuning* is directly compared to *transfer learning* instead of being applied as an optional step after *transfer learning* [420], which implies only a single retraining step per network.

6.1.2.1 Retraining Datasets

Manually labelled HRTEM images of Au/Ge and Pd/C nanoparticle systems were considered for retraining *nNPipe*. The open-source annotation software Visual Object Tagging Tool (VOTT) [422] was used to annotate HRTEM images (Section 4.1) with bounding boxes distinguishing individual nanoparticles and clusters according to ground truth images as described in Section 4.2.4.1. The obtained annotation files were used alongside corresponding HRTEM images to train the YOLOv5x module [391] . Note that a more recent version of YOLOv5x was downloaded from github.com for *transfer learning* as it introduced a 'freeze' argument that enabled disabling of parameter updates for selected network layers using a simple syntax. The previously used network architecture (Chapter 5) of YOLOv5x was used for both retraining methods to ensure direct comparability. Training images for the SegNet module [199] were obtained by generating 256 x 256 pixel image sections of individual nanoparticles and empty areas (*i.e.* images of the vacuum and support material) for generalisation. Analogous to Chapter 5, a ground truth consisting of individual nanoparticles not cut by image edges was considered for performance comparison.

Two main variables were considered to trace their respective influence on network performance after retraining. Experimental Subset Size (ESS) and augmentation. Values were selected empirically and set at 10 %, 25 %, 50 % and 100 % for the ESS, and to 1 %, 5 % and 10 % for augmentation, which refers to the relative percentage of the final training dataset size after augmentation and with respect to the original training (Section 5.1). Note that augmentation was conducted offline and that iterations in loops were adapted with respect to each ESS to contain overgeneration of data (see below).

Once respective ESS were randomly selected, their images were initially augmented by a factor of eight using four-fold rotations and horizontal flipping, creating a 'Seed'. Secondly, a nested loop randomly augmenting the data with previously tested values for contrast adjustment, brightness variation, Gaussian blurring and dead pixels, where at least one image per loop did not contain any dead pixels. Lastly, copies of augmented images were cropped to image arrays of $65\,\% - 95\,\%$ of the original 2048 x 2048 pixel size. To match an exact number of images for the respective training dataset sizes, redundant images were randomly selected to be subsequently deleted. The resulting set was split into 90/10 % for training and validation of the YOLOv5x network. This was done three times per experimental subset to achieve a three-fold cross-validation.

Training datasets for the SegNet module of *nNPipe* were generated from non-cropped images of the training dataset of the YOLOv5x network (see above). As images contained both nanoparticles and empty areas, training datasets larger than ultimately required were initially generated. Hence, redundant images were also randomly selected and deleted to match the set final training dataset size. Table 6.1 and 6.2 summarise the total dataset sizes for both material systems, while corresponding MATLAB code for image augmentation is supplied in the Appendix Section 10.6.

Table 6.1: Retraining dataset sizes for individual *nNPipe* modules and for the Au/Ge dataset. The original dataset size for the YOLOv5x module was 140,000, while the SegNet module was originally trained on 480,730 images. The abbreviation 'imgs' refers to images in respective ESS.

YOLOv5x dataset sizes	10 imgs / 10 %	24 imgs / 25 %	47 imgs / 50 %	93 imgs / 100 %
1 % of org. training	1,400	1,400	1,400	1,400
5 % of org. training	7,000	7,000	7,000	7,000
10 % of org. training	14,000	14,000	14,000	14,000
SegNet dataset sizes	10 imgs / 10 %	24 imgs / 25 %	47 imgs / 50 %	93 imgs / 100 %
1 % of org. training	4,808	4,808	4,808	4,808
5 % of org. training	24,037	24,037	24,037	24,037
10 % of org. training	48,073	48,073	48,073	48,073

Table 6.2: Retraining dataset sizes for individual *nNPipe* modules and for the Pd/C dataset. The original dataset size for the YOLOv5x module was 140,000, while the SegNet module was originally trained on 618,090 images. The abbreviation 'imgs' refers to images in respective ESS.

YOLOv5x dataset sizes	10 imgs / 10 %	24 imgs / 25 %	48 imgs / 50 %	96 imgs / 100 %
1 % of org. training	1,400	1,400	1,400	1,400
5 % of org. training	7,000	7,000	7,000	7,000
10 % of org. training	14,000	14,000	14,000	14,000
SegNet dataset sizes	10 imgs / 10 %	24 imgs / 25 %	48 imgs / 50 %	96 imgs / 100 %
1 % of org. training	6,181	6,181	6,181	6,181
5 % of org. training	30,905	30,905	30,905	30,905
10 % of org. training	61,809	61,809	61,809	61,809

6.1.2.2 Retraining Process

For both *transfer learning* and *fine-tuning*, the hyperparameters were left at similar values to the original training as described in Chapter 5.1 with the exception of the two learning rates, namely 10^{-4} and 10^{-5}, that were empirically chosen and tested for both retraining methods for comparison. This was only done for training datasets that were augmented to 10 % which were expected to achieve highest performance due to their training dataset size. The best performing approach was subsequently used for smaller datasets. As mentioned above, the first 24 layers were frozen during training for *transfer learning* while parameter updates were allowed for the whole network in *fine-tuning*. Furthermore, the training time was limited to 50 epochs for the YOLOv5x networks and 80 epochs for the SegNet module respectively, to take practicability and potential time-restrictions into account. The maximum training time for both *nNPipe* modules was *ca.* 33 h (8 h YOLOv5x module, 25 h SegNet module).

6.1.3 Results

6.1.3.1 Retraining on Au/Ge data

For each experimental subset, three different sets of randomly selected images were used for retraining to obtain a three-fold cross-validation and mean Average Precision (mAP) curves were calculated to give more general insights into retraining behaviour. The dataset containing 100 % of experimental images as a subset was taken into account to determine maximum achievable performance. Figure 6.2 shows an epoch-averaged comparison of mAP@0.50 values between *transfer learning* and *fine-tuning* of training datasets augmented to 10 % of the original training dataset size. Line-plots comparing mean values of mAP@0.50 and mAP@0.95 for each epoch individually are shown in Figure 10.6.

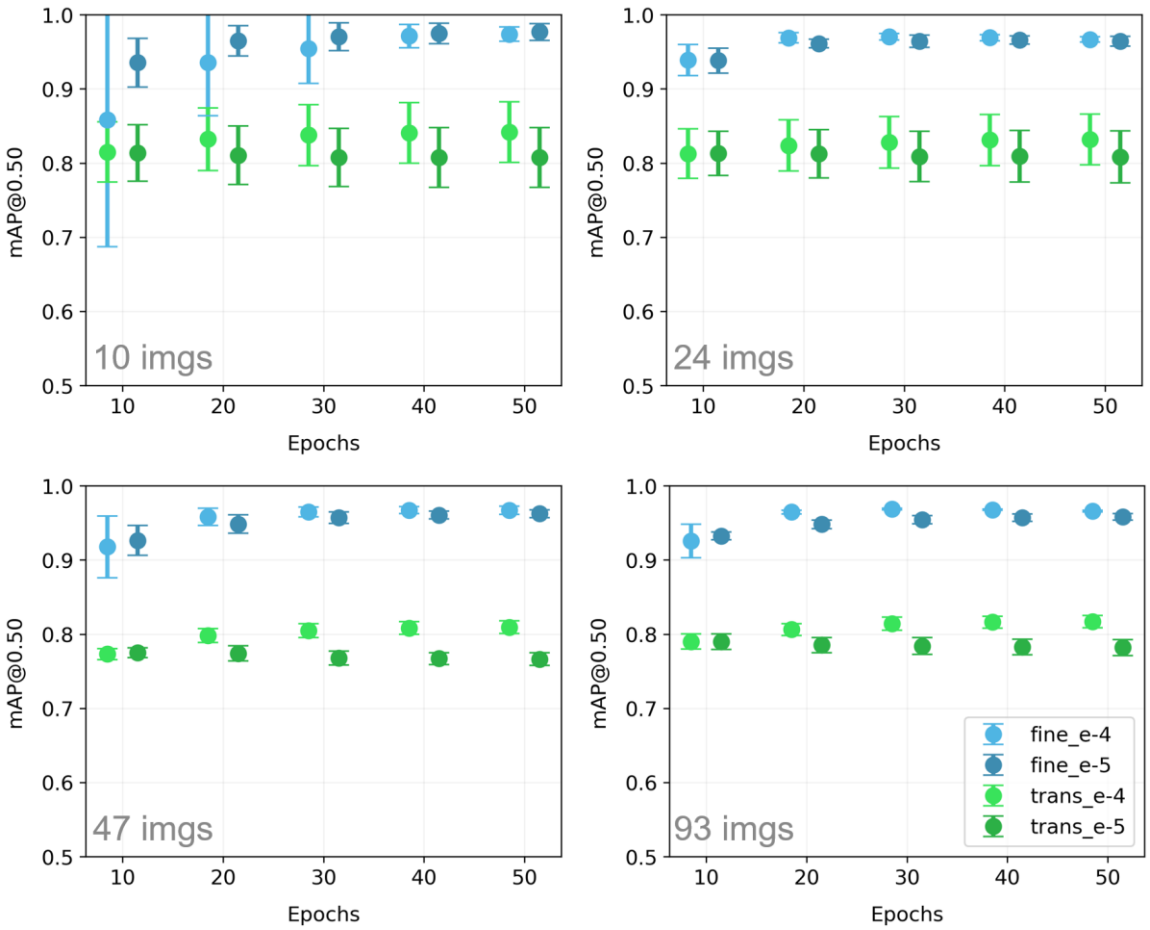


Figure 6.2: Comparison of *transfer learning* and *fine-tuning* applied for the Au/Ge system on the YOLOv5x module at learning rates of 10^{-4} and 10^{-5} respectively. Each point represents the mean mAP@0.50 of three sets of randomly selected instances of given experimental subsets which were averaged over 10 epochs to obtain a meaningful mean standard deviation. The vertical axis refers to mAP@0.50 values and the horizontal axis to epochs. Note that e-4, e-5 scatter points were shifted ±1.5 around their original multiple of 10 epoch number for improved visibility. The legend in the lower right corner is applicable to all plots shown.

The achieved mAP values are above 0.7 for all of the applied retraining approaches, although *fine-tuning* yields the best performance. Considering *fine-tuning* and *transfer learning* separately, learning rates of 10^{-4} appear to achieve slightly higher mAP as compared to learning rates of 10^{-5} with the exception of *fine-tuning* for the experimental subset of 10 images. In this case learning rates of 10^{-5} achieve consistently better mAP up to 30 epochs, after which the difference in learning rates of 10^{-4} is small. Most significantly, mAP values of *ca.* 0.95 are consistently achieved for all experimental subsets after 30 epochs, suggesting that even shorter retraining periods may be applied. However, larger standard deviations for smaller experimental subsets and shorter training times must be taken into account as seen for the experimental subset of 10 images in particular. While larger standard deviations occur in early retraining, those become significantly smaller after 30 epochs. However, lower mean standard deviations are observed for learning rates of 10^{-5} in early training stages in all cases. In addition, larger experimental subset sizes appear to induce lower overall mean standard deviation at earlier stages of retraining with stable mAP values obtained after 20 epochs.

Based on these results, it was decided to continue the retraining of *nNPipe* only by *fine-tuning* as this retraining strategy consistently achieved higher performance for the YOLOv5x module. It is noted that a combined retraining approach, for example *fine-tuning* for YOLOV5x and *transfer learning* for the SegNet module is generally feasible, but was not pursued in order to obtain one distinct retraining approach for the *nNPipe* method with highest performance. Accordingly, the SegNet module was retrained on different experimental subsets augmented to 10 % of the original training dataset size only using *fine-tuning* with results summarised in Figure 6.3. In this case, results are illustrated in a line-plot since the Cross-Entropy loss (Eq 6.1) was only tracked every five epochs and hence a similar figure to Figure 6.2 would be unfeasible. The Cross-Entropy loss is defined as [383]:

$$L = -\sum_{i=1}^{k} y_i \log(o_i) \tag{6.1}$$

where $y_i, \ldots, y_k$ refer to observed outputs for k mutally exclusive classes and o_i refers to the probability generated by the logistic sigmoid layer (see Section 4.2.3).

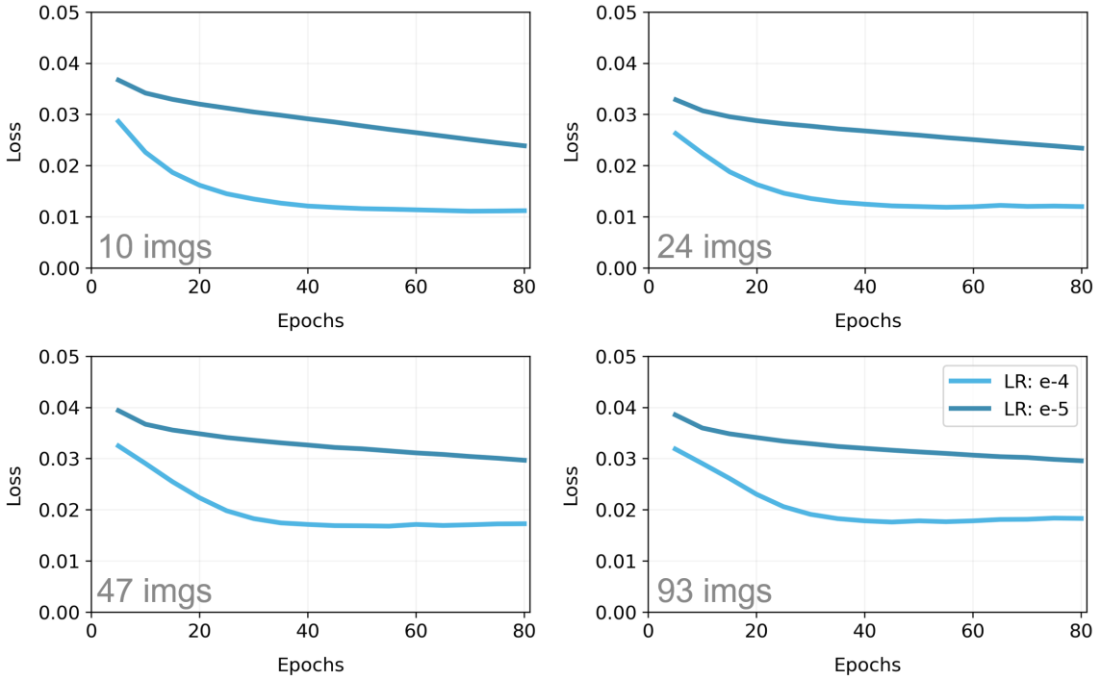


Figure 6.3: Line-plots of Cross-Entropy loss during *fine-tuning* of the SegNet module for the Au/Ge *nNPipe* and measured on validation sets. The vertical axis refers to the Cross-Entropy loss and the horizontal axis to epochs. Each curve was calculated from three sets of randomly selected instances of given experimental subsets which were averaged to result in a mean retraining line-plot.

Compared to the *fine-tuning* as observed in for the YOLOv5x module, a learning rate of 10^{-4} yielded significantly lower Cross-Entropy losses (Eq. 6.1) than for learning rates of 10^{-5}. Additionally, it appears that lower losses are achieved for smaller experimental subsets, which can be understood by a smaller morphological feature variation as compared to larger experimental subsets. A counterintuitive behaviour is also observed for experimental subsets of 47 images and 93 images, where the Cross-Entropy loss curves of 10^{-4} start to increase after approximately 45 epochs, suggesting an onset of overfitting. A potential explanation could be given by reconsidering the training dataset generation method. Larger experimental subsets require less augmentation to yield a specific training dataset size (*i.e.* 10 % of original training data). Hence, those datasets exhibit a relatively high morphological variability of individual instances where less augmented instances of which the former might be learned faster. However, additional in-depth analysis on this aspect would be required as a confirmation. Similarly, to Figure 6.2 and Figure 10.6, smaller experimental subsets exhibit larger mean standard deviation errors per epoch.

Both *nNPipe* modules were consequently *fine-tuned* at learning rates of 10^{-4} for all residual training datasets (see Table 6.1). A direct comparison of the *fine-tuning* process between individual experimental subsets at varying degrees of augmentation, as well as between different experimental subsets for fixed degrees of augmentation are shown in Figures 10.7 – 10.10. Figure 6.4 shows KDE-plots of F1-Scores corresponding to Figure 5.4 for different degrees of augmentation, while similar plots compare F1-Scores with respect to varying experimental subset sizes in Figure 6.5. Table 6.3 and 6.4 list key properties deduced from all KDE-plots shown in Figure 6.4 and 6.5. These results were obtained on the full Au/Ge HRTEM experimental dataset, as otherwise a comparison to the experimental subset of 93 images would be unfeasible. SegNet module weights were selected by considering the first minimal validation loss achieved and *nNPipe* post-processing remained unchaged for direct comparability.

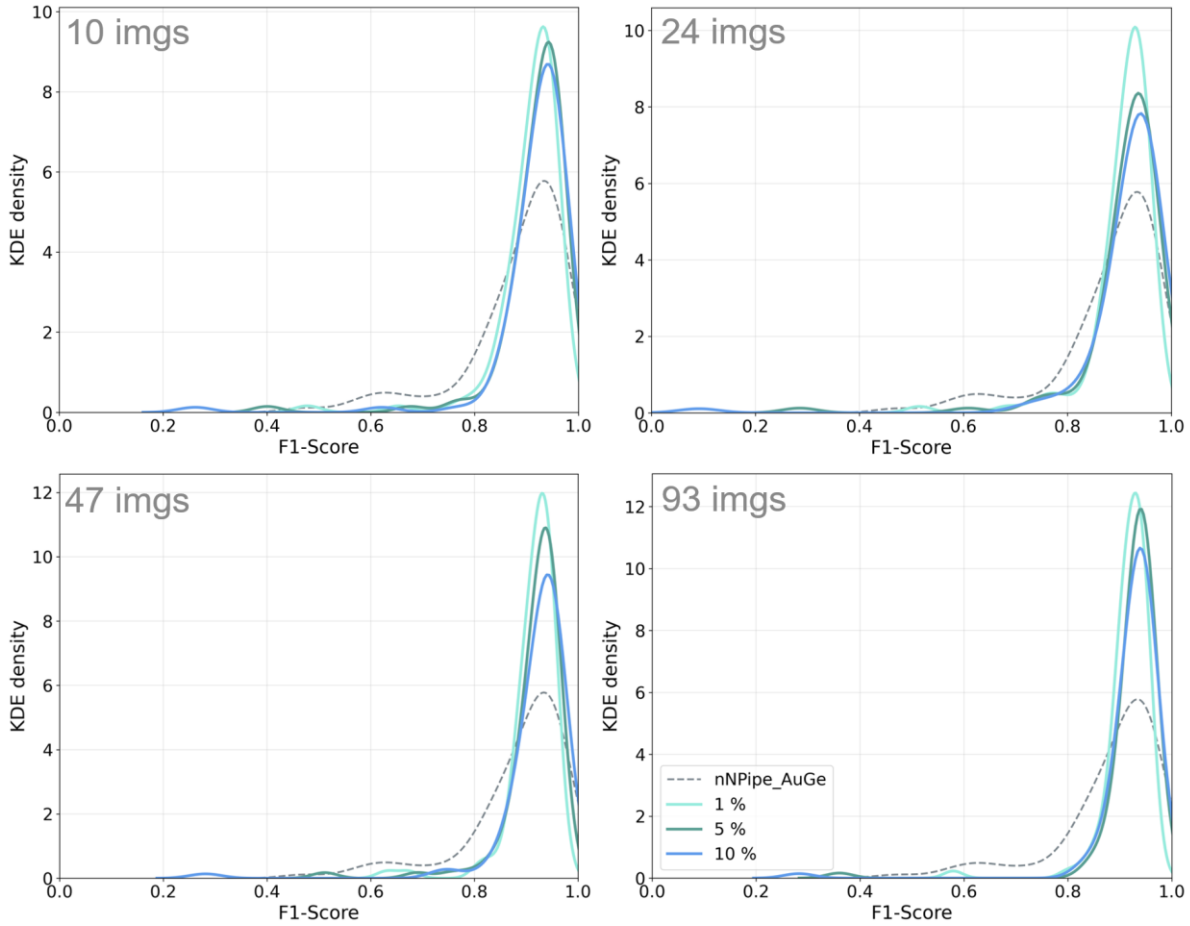


Figure 6.4: KDE-plots of F1-Scores obtained from retrained networks of the *nNPipe* method for different experimental subsets at varying augmentation levels. Plots were generated by using a Gaussian kernel and Scott's rule [418]. The legend added for the experimental subset of 93 images holds true for all graphs shown. Note that the grey dashed line denoted as 'nNPipe_AuGe' refers to the performance of *nNPipe* as originally trained on synthetic data for comparison. The abbreviation 'imgs' refers to 'images'.

Table 6.3: Key quantitative properties of determined KDE-plots for different ESS and degrees of augmentation as shown in Figure 6.4. The abbreviation 'aug.' refers to augmentation.

10 imgs dataset	1 % aug.	5 % aug.	10 % aug.	24 imgs dataset	1 % aug.	5 % aug.	10 % aug.
Mean	0.908	0.918	0.920	Mean	0.907	0.911	0.916
$x_{\max(\mathrm{F1})}$	0.932	0.943	0.939	$x_{\max(\mathrm{F1})}$	0.930	0.936	0.942
FWHM	0.090	0.097	0.105	FWHM	0.087	0.103	0.112
47 imgs dataset	1 % aug.	5 % aug.	10 % aug.	93 imgs dataset	1 % aug.	5 % aug.	10 % aug.
Mean	0.914	0.917	0.921	Mean	0.917	0.928	0.925
$x_{\max(\mathrm{F1})}$	0.930	0.938	0.940	$x_{\max(\mathrm{F1})}$	0.930	0.942	0.939
FWHM	0.077	0.081	0.092	FWHM	0.074	0.077	0.082

The results shown in Figure 6.4 and Table 6.3 show mostly consistent trends for all experimental image subsets. Higher degrees of augmentation yield higher mean F1-Scores and higher F1-Score values at peak KDE ($x_{\max(F1)}$). However, the FWHM increases as well. The former is expected and can be explained by the higher degrees of generalisaiton and precision achieved through training using a larger training dataset. Increasing FWHM can be understood by an increased influence of the retraining dataset on the original parameters of *nNPipe* due to its sheer size. While the mean F1-Score may increase, some of the inference capabilities might be lost resulting in mispredictions in specific cases. An additional indicator is given in all graphs shown in Figure 6.4, where the KDE-plots of 10 % augmentation show distinct, flat tails at low F1-Score values. Augmenting the dataset to 5 % of the original training dataset size might offer an optimum, as F1-Score means and $x_{\max(F1)}$ values are highest for experimental subsets of 10 and 93 images for that degree of augmentation (Table 6.3). Even though these highlighted differences are not extensive (at maximum 0.012 for both F1-Score means and $x_{\max(F1)}$), the datasets augmented to 5 % or 10 % of the original dataset size clearly achive higher performance than datasets augmented to 1 % of the original dataset size in all cases respectively. An important result can be observed when comparing retraining performances of different experimental subsets for set degrees of augmentation as shown in Figure 6.5 and Table 6.4. For training datasets augmented to 10 % of the original training dataset size the the mean F1-Score of the 10 image-subset is only 0.005 lower than for

the one containing 93 images whilst achiving the same value for $x_{\max(F1)}$. Moreover, the 10 image-subset outperforms the 24 image-subset in both F1-Score mean and $x_{\max(F1)}$ achieves very similar mean F1-Score values compared to the 47 image-subset for 5 % and 10 % of augmentation.

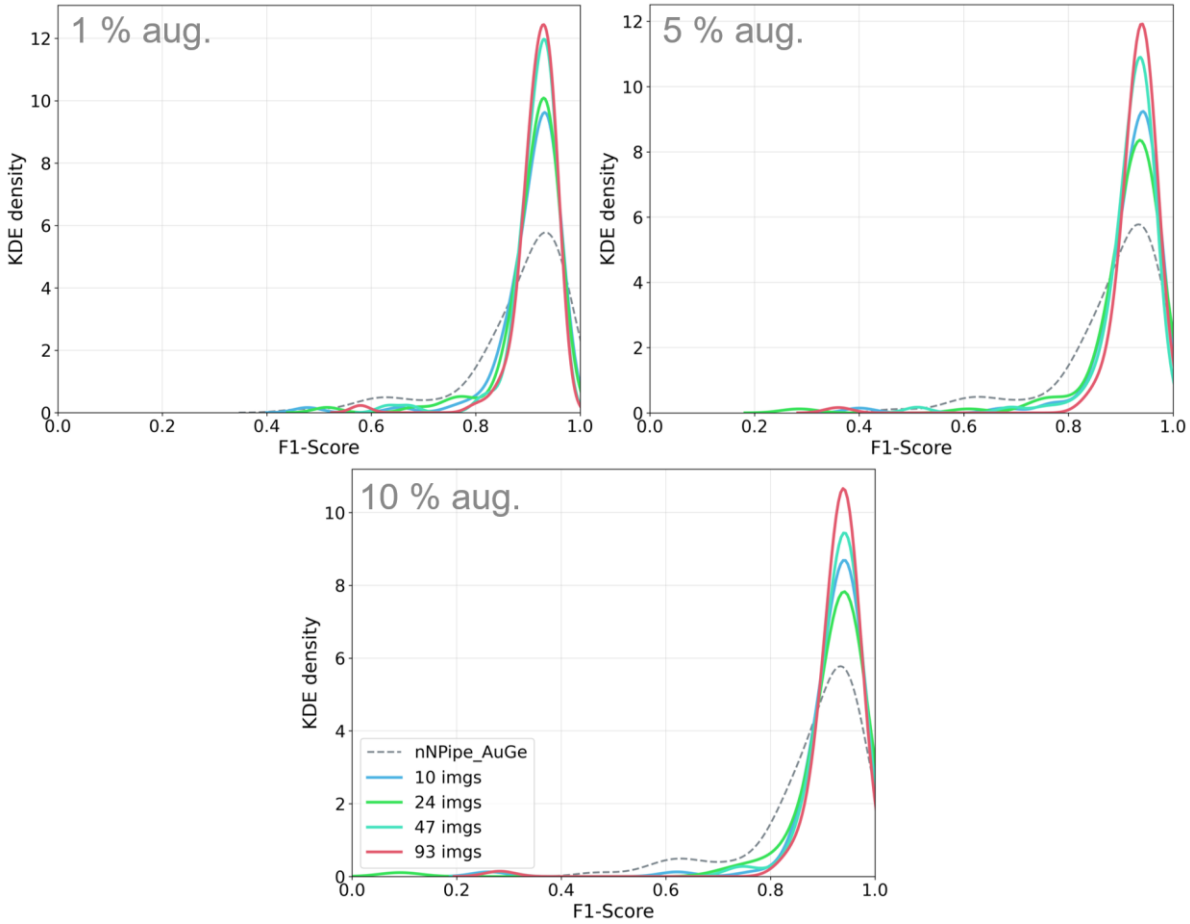


Figure 6.5: KDE-plots of F1-Scores obtained from retrained networks of the *nNPipe* method at different augmentation levels comparing varying experimental subset sizes. Plots were generated by using a Gaussian kernel and Scott's rule [418]. The legend shown for 10 % augmentation (aug.) holds true for all graphs shown, whereby the dashed grey line denoted as 'nNPipe_AuGe' refers to the performance of *nNPipe* as originally trained on synthetic data. The abbreviation 'imgs' refers to 'images'.

Table 6.4: Key quantitative properties of KDE-plots of datasets at different degrees of augmentation and varying ESS as shown in Figure 6.5. The abbreviation 'im.' refers to 'images' of respective ESS.

1 % augmentation					5 % augmentation				
Property	10 im.	24 im.	47 im.	93 im.	Property	10 im.	24 im.	47 im.	93 im.
Mean	0.908	0.907	0.914	0.917	Mean	0.918	0.911	0.917	0.928
$x_{\max(F1)}$	0.932	0.930	0.930	0.930	$x_{\max(F1)}$	0.943	0.936	0.938	0.942
FWHM	0.090	0.087	0.077	0.074	FWHM	0.097	0.103	0.081	0.077

10 % augmentation				
Property	10 im.	24 im.	47 im.	93 im.
Mean	0.920	0.916	0.921	0.925
$x_{\max(F1)}$	0.939	0.942	0.940	0.939
FWHM	0.105	0.112	0.092	0.082

This suggests that only 10 experimental images (*ca.* 10 %) are sufficient to *fine-tune nNPipe* to a significantly higher inference performance which is comparable to retraining scenarios where the full experimental dataset would be used for *fine-tuning*. However, experimental subsets only containing 10 images also introduced the highest mean standard deviations in the first 30 epochs of retraining, as shown in Figures 10.6 – 10.10 (Appendix) and so longer training times are ought to be expected. In practice, the retraining is manifested by an increased analysis performance especially for HRTEM images in which *nNPipe* previously achieved lower F1-Scores. Figure 6.6 shows two examples of experimental HRTEM images and respective performance output alongside F1-Scores for the original Au/Ge system *nNPipe* and after *fine-tuning* with a learning rate of 10^{-4} using an ESS of 10 images augmented to 10 % of the original training dataset.

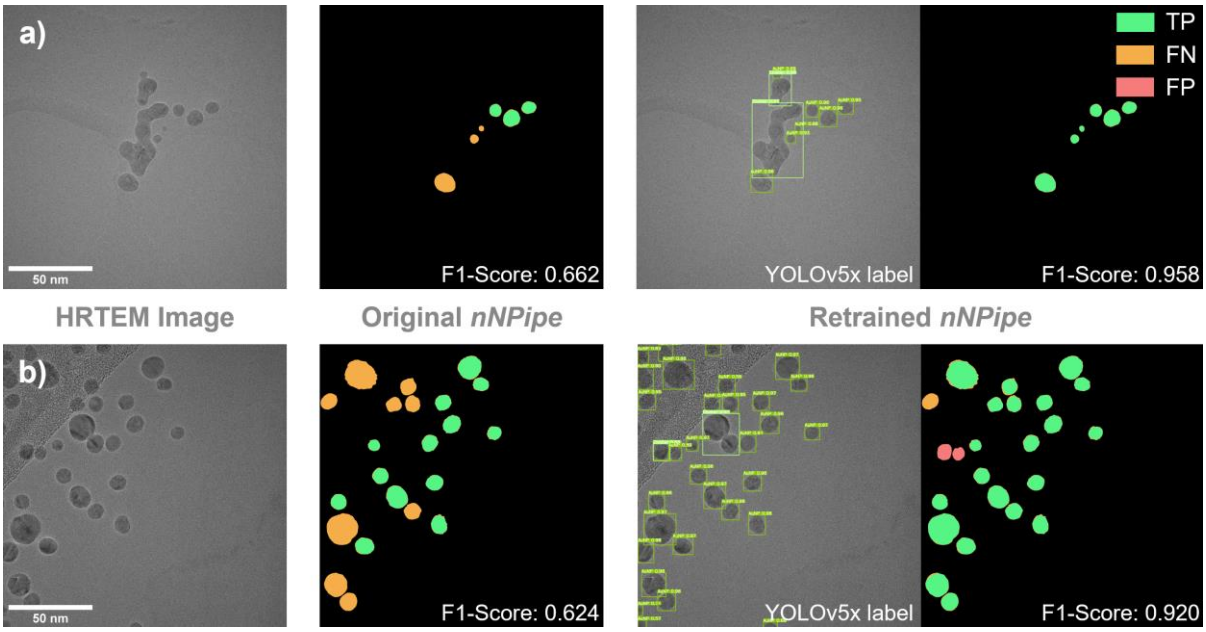


Figure 6.6: Comparison *of nNPipe* inference output in its original state and after retraining for two examples lower than in the majority of cases. **a)** Imaging scenario where three nanoparticles were not segmented correctly (FN pixels) as they were assigned to the cluster in the centre of the HRTEM image. Retraining improved the distinction between individual instances and ensured correct segmentation of individual nanoparticles. **b)** Indivial nanoparticles found on lacy carbon were identified and segmented correctly after training. Previously, the lacy carbon was identified as cluster (compare Figure 5.2). The colour-code shown upper right hold true for all segmentation outputs in this figure. Note that amongst all three ESS of 10 images augmented to 10 %, the best one was selected for this analysis based on the highest mean F1-Scores achieved.

The two examples shown highlight the extension of inference capabilities of both, the YOLOv5x and the SegNet module. While the distinction between individual nanoparticles and clusters works more reliably as seen in Figure 6.5a, unknown instances such as the lacy carbon shown in Figure 6.5b are also now interpreted correctly leading to significantly improved performance.

6.1.3.2 Retraining on Pd/C data

By analogy to retraining conducted on the *nNPipe* method for the Au/Ge system, *transfer learning* was compared to *fine-tuning* for the Pd/C system using three randomly selected sets of images to obtain a three-fold cross-validation. Figure 6.7 shows an epoch-averaged comparison of mean mAP@0.50 values between *transfer learning* and *fine-tuning* for datasets augmented to 10 % of the original training dataset size, while respective line plots are shown in Figure 10.11.

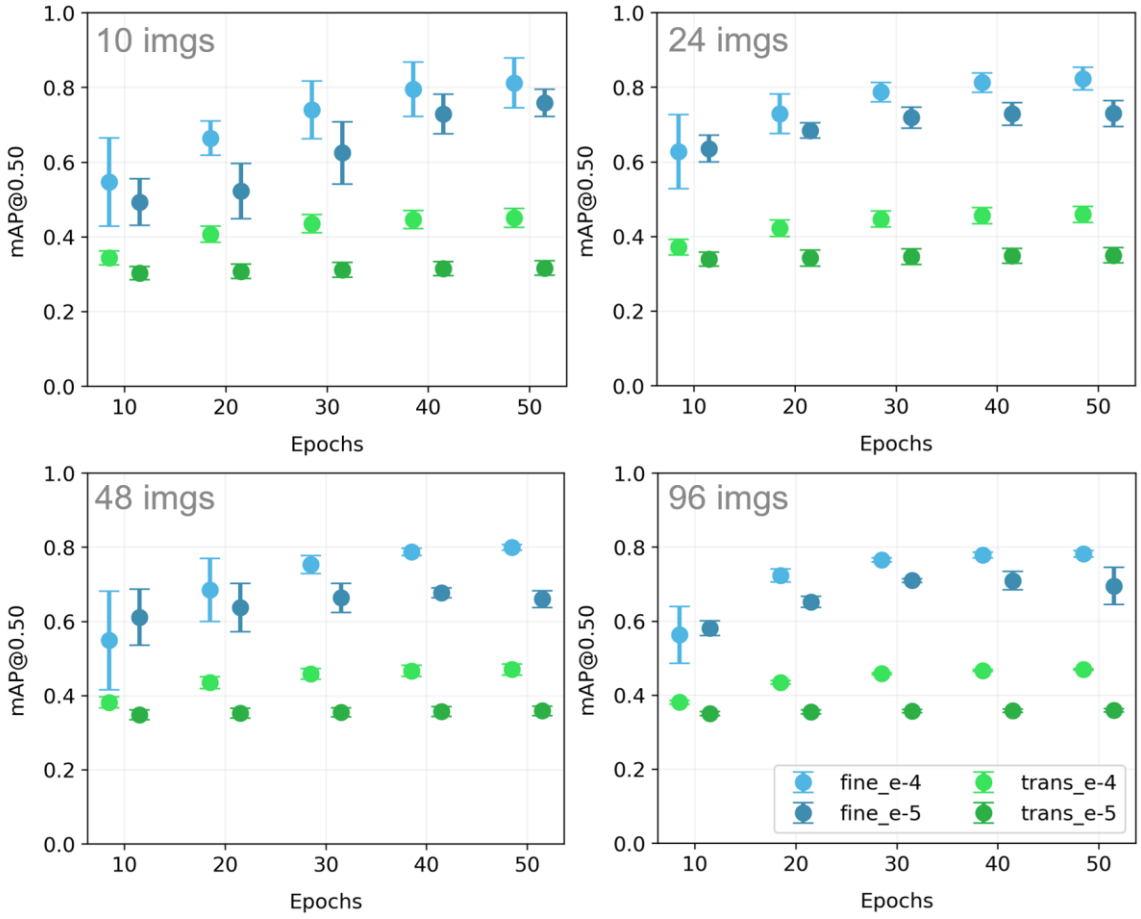


Figure 6.7: Comparison of *transfer learning* and *fine-tuning* for the Pd/C system on the YOLOv5x module at learning rates of 10^{-4} and 10^{-5} respectively. Each point represents the mean mAP@0.50 of three sets of randomly selected instances of given experimental subsets which were averaged over 10 epochs to obtain a meaningful mean standard deviation. The vertical axis refers to mAP@0.50 values and the horizontal axis to epochs. Note that individual scatter points were shifted ±1.5 around their original epoch number for improved visibility. The legend shown for the experimental subset of 96 images holds for all plots shown. The abbreviation 'imgs' refers to 'images'.

Compared to Figure 6.1, the overall mAP@0.50 values achieved during retraining of the Pd/C YOLOv5x module are lower than for the Au/Ge system. This result is expected given the analysis conducted on the *nNPipe* method in Chapter 5, which highlighted a lower performance due to increased morphological diversity for the Pd/C system. Nonetheless these results clearly show an advantage of *fine-tuning* over *transfer learning*, where applied learning rates of 10^{-4} achieve highest mAP@0.50

values and are distinguished by a more pronounced difference with respect to learning rates of 10^{-5} in both cases. Larger ESS equally appear to induce lower mean standard deviations in earlier stages of training, while mAP values increase over the whole retraining period as seen for the 10 images subset in particular. This suggests that longer training periods may be beneficially applied to morphologically complex systems such as Pd/C, which is in contrast to the Au/Ge system.

Observations of retraining the YOLOv5x module confirmed the advantage of *fine-tuning* over *transfer learning* for the considered materials systems and it was thus decided to conduct the next retraining step using *fine-tuning* only. Figure 6.8 highlights Cross-Entropy losses obtained during *fine-tuning* of the SegNet module for different ESS augmented to 10 % of the original training dataset size and at two different learning rates.

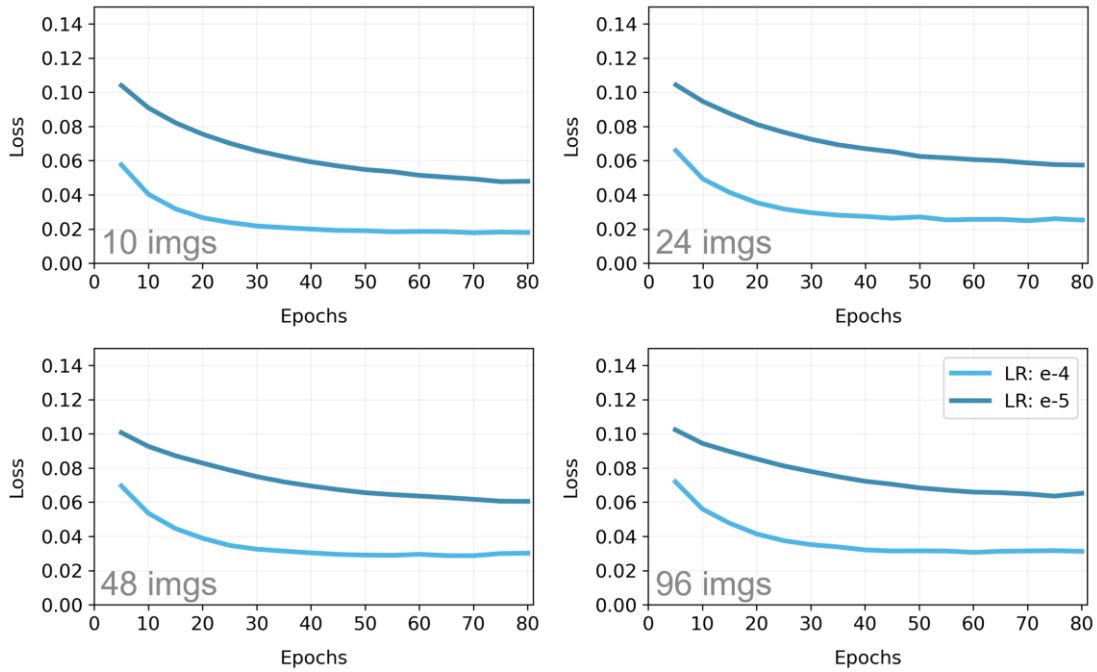


Figure 6.8: Line-plots of Cross-Entropy loss during *fine-tuning* the SegNet module of the Pd/C *nNPipe* measured on validation sets. The vertical axis refers to the Cross-Entropy loss and the horizontal axis to epochs. Each curve was calculated from three sets of randomly selected instances of given experimental subsets which were averaged to result in a mean retraining line-plot. The abbreviation 'imgs' refers to 'images'.

Similar to the. results for the Au/Ge system shown in Figure 6.3, the SegNet module yields significantly lower Cross-Entropy loss if a learning rate of 10^{-4} is applied and the lowest losses are observed for the experimental subsets of 10 images, which again can be understood by comparably smaller morphological feature variation as compared to subsets generated from more experimental images. In contrast, the overall Cross-Entropy losses do not appear to increase proportionally with ESS, nor are

onsets of overfitting observable. This is consistent with the retraining results observed for the YOLOv5x module in Figure 6.7, where performance improvements were noticeable across the entire retraining period and may again highlight that longer training periods are necessary for morphologically diverse material systems. The fact that Cross-Entropy errors are similar for the experimental subsets of 24, 48 and 96 images could indicate that the experimental subset of 24 images already covers a large portion of the new image scenarios that the SegNet module is trained on (*i.e.* particle shapes, contrast, *etc.*), but in contrast to the Au/Ge system the point at which the entire morphological variability is learned is not achieved, ultimately leading to similar *fine-tuning* behaviour and absence of overfitting in all three ESSs.

These results justified the application of *fine-tuning* at learning rates of 10^{-4} for all other training datasets and were comparable to those from the Au/Ge system allowing direct comparison. Direct comparison between *fine-tuning* behaviours of the YOLOv5x and SegNet module for individual ESS at varying degrees of augmentation, and between different ESS for set degrees of augmentation are shown in Figure 10.12 – 10.15. In addition, Figure 6.9 and 6.10 show comparisons of F1-Score KDE-plots determined from the Pd/C HRTEM dataset (see Section 4.1) for either fixed ESS or degrees of augmentation. Table 6.5 and 6.6 summarise quantified key properties of the respective KDE-plots for completeness. By analogy with the Au/Ge system, weights of the SegNet module were selected by considering the first minimal loss calculated on the validation dataset and the system-dependent post-processing routine of *nNPipe* remained unchanged to ensure direct comparison.

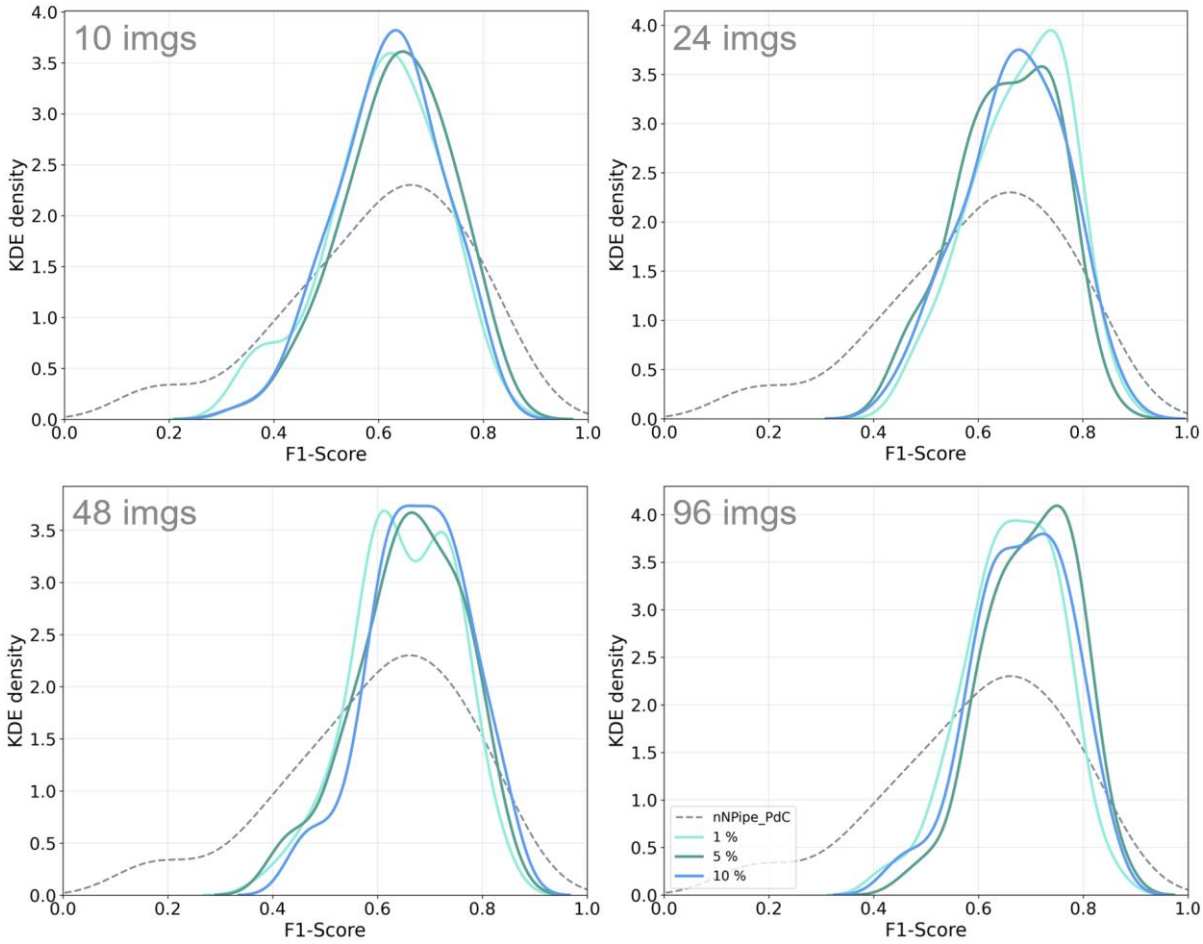


Figure 6.9: KDE-plots of F1-Scores obtained from retrained networks of the *nNPipe* method for different experimental subsets at varying augmentation levels. Plots were generated by using a Gaussian kernel and Scott's rule [418]. The legend added to the experimental subset of 96 images holds true for all graphs shown. Note that the grey dashed line denoted as 'nNPipe_PdC' refers to the performance of *nNPipe* as originally trained on synthetic data for comparison. The abbreviation 'imgs' refers to 'images'.

Table 6.5: Key quantitative properties of KDE-plots for different ESS and degrees of augmentation as shown in Figure 6.9. The abbreviation 'aug.' refers to augmentation.

10 imgs dataset	1 % aug.	5 % aug.	10 % aug.	24 imgs dataset	1 % aug.	5 % aug.	10 % aug.
Mean	0.608	0.637	0.621	Mean	0.680	0.655	0.669
$x_{\max(F1)}$	0.624	0.645	0.634	$x_{\max(F1)}$	0.740	0.723	0.677
FWHM	0.257	0.263	0.256	FWHM	0.246	0.266	0.256
48 imgs dataset	1 % aug.	5 % aug.	10 % aug.	96 imgs dataset	1 % aug.	5 % aug.	10 % aug.
Mean	0.651	0.661	0.679	Mean	0.665	0.701	0.683
$x_{\max(F1)}$	0.614	0.667	0.662	$x_{\max(F1)}$	0.670	0.750	0.722
FWHM	0.257	0.260	0.247	FWHM	0.238	0.241	0.249

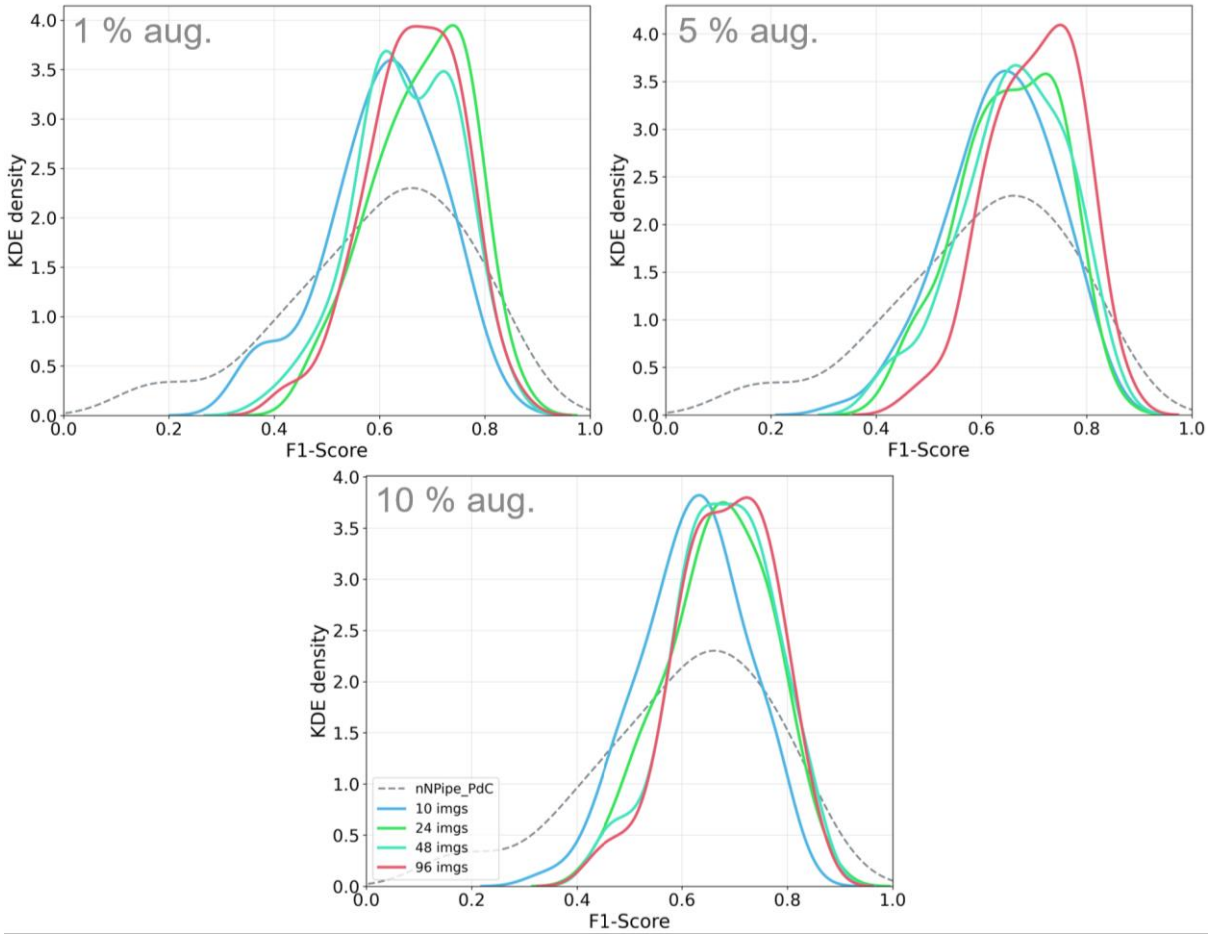


Figure 6.10: KDE-plots of F1-Scores from retrained networks of the *nNPipe* method at different augmentation levels comparing varying experimental subset sizes. Plots were generated by using a Gaussian kernel and Scott's rule [418]. The legend shown for 10 % augmentation (aug.) holds true for all graphs shown, where the dashed grey line denoted as 'nNPipe_PdC' refers to the performance of *nNPipe* as originally trained on synthetic data for comparison.

Table 6.6: Key quantitative properties of KDE-plots of datasets at different degrees of augmentation and varying ESS as shown in Figure 6.10. The abbreviation 'im.' refers to images of respective ESS.

1 % augmentation					5 % augmentation				
Property	10 im.	24 im.	48 im.	96 im.	Property	10 im.	24 im.	48 im.	96 im.
Mean	0.608	0.680	0.651	0.665	Mean	0.637	0.655	0.661	0.701
$x_{\max(F1)}$	0.624	0.740	0.614	0.670	$x_{\max(F1)}$	0.645	0.723	0.667	0.750
FWHM	0.257	0.246	0.257	0.238	FWHM	0.263	0.266	0.260	0.241

10 % augmentation				
Property	10 im.	24 im.	48 im.	96 im.
Mean	0.621	0.669	0.679	0.683
$x_{\max(F1)}$	0.634	0.677	0.662	0.722
FWHM	0.256	0.256	0.247	0.249

Considering the results shown in Figure 6.9 alongside the respective KDE-plot quantifications in Table 6.5, a significantly improved performance of *nNPipe* on the Pd/C HRTEM images is clearly visible for all retraining cases used (Figure 6.9). However, these are are still lower overall compared to any training state of the Au/Ge system while less clear trends unfold (Table 6.5). Larger degrees of augmentation do not necessarily lead to improved mean F1-Scores as can be seen for the ESS of 10 images, 24 images and 96 images (Table 6.5). Except for the ESS of 24 images, training conducted using datasets augmented to 1 % of the original datasets yield lowest performance in individual ESS comparisons. However, an important observation can be seen in ESS augmented to 5 % of the original dataset size, which yield highest values for mean F1-Scores and $x_{\max(\mathrm{F1})}$ in three of four ESS. This may be understood by considering the degree of augmentation with respect to the original *nNPipe* training. Introduction of a high number of instances varying in image contrast, brightness or blurring as compared to the data to be analysed may eventually lead to decreased performance. The 5 % augmentation thus appears to provide a good compromise between maintaining the generalisation capabilities of the original *nNPipe* method and optimisation towards experimental images. Consistently with findings at the training stage of the individual CNN modules of *nNPipe*, the ESS of 10 images achieves lower performance compared to larger ESS. While the highest mean F1-Score achieved is 0.637 for the ESS of 10 images, the lowest amongst all others is 0.651. This suggests that *ca.* 10 % of a new experimental dataset are unlikely to be sufficient for retraining *nNPipe* for a new dataset and a higher number of training instances may be required for more complex catalyst systems. Counterintuitively, the ESS containing 24 images yields high values for mean F1-Scores and $x_{\max(\mathrm{F1})}$ as compared to all ESSs (Table 6.5). As implied during the SegNet retraining already (see above), this ESS may cover most of morphologies and imaging scenarios encountered in the new experimental dataset and forms a compromise between new instances introduced to *nNPipe* and its generalisation capabilities in addition to the one introduced with 5 % augmentation.

This is also seen when Figure 6.10 and Table 6.6 are considered. Although the ESS containing 100 % of experimental images (96 images) achieves the highest performance, the ESS of 24 images yields very similar performance levels, especially for 10 % augmentation (Figure 6.10). In practice, retraining

based on ESS of similar size would however require longer traininig periods (Figure 10.12 and 10.13), which is analogous to the behaviour seen for ESS of 10 images in the Au/Ge system. Lastly, it appears that very low degrees of augmentation (1 % augmentation) lead to strongly varying performances across all ESS which may rationalised by favourable and unfavourable augmentation parameters that were selected randomly. The increased analysis performance was further tested on HRTEM images in which the original Pd/C *nNPipe* attained lower F1-Scores. Figure 6.11 illustrates two selected experimental HRTEM images from the Pd/C system as well as respective outputs generated from the original *nNPipe* method and after *fine-tuning* at a learning rate of 10^{-4} with an ESS of 24 images, augmented to 10 % of the original training dataset.

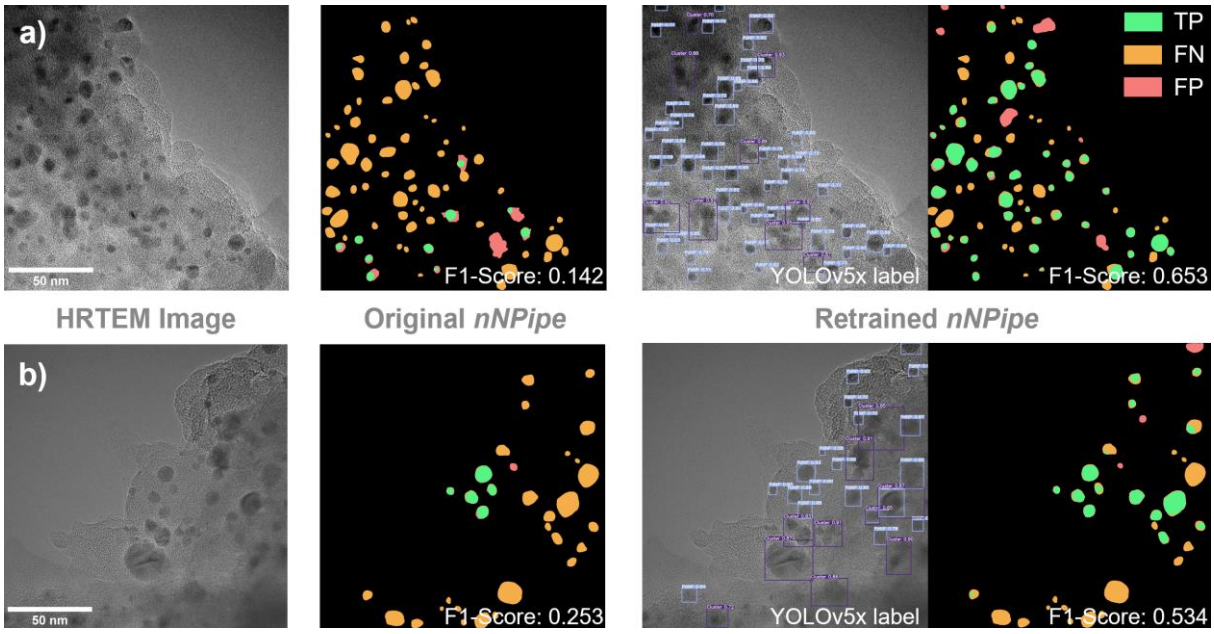


Figure 6.11: Comparison *of the nNPipe* inference output in its original state and after retraining for two examples was lower than in the majority of cases. **a)** Imaging scenario including a thick layer of amorphous carbon support material where the majority of nanoparticles were not identified resulting in a very low inference performance. After retraining, individual instances are identified successfully and a significantly higher number of nanoparticles are correctly segmented. **b)** Nanoparticles originally only identified and segmented in the area of flat amorphous carbon (compare Figure 5.3). Retraining enhances *nNPipes* ability to identify clusters and nanoparticles even in areas of thick support material, leading to a higher number of correct segmentations and a higher F1-Score.

The most notable improvements are observed for lower-performing cases (Figure 6.9 and 6.10), which is why Figure 6.11 highlights these scenarios. Originally, most nanoparticles shown in the HRTEM image of 6.11a were missed in the original *nNPipe*, leading to a very low F1-Score. After retraining, a significantly higher number of instances are correctly identified and the same holds true for the data in Figure 6.11b which was also shown in Figure 5.3. Overall, *nNPipe* appears to gain inference capabilities for areas of thick support material, which are cases previously not included in training.

6.1.4 Discussion

The results shown in Figure 6.4 and 6.5, as well as Figure 6.9 and 6.10 confirmed the conjecture in Section 5.3. The inference capabilities of *nNPipe* can be further improved by retraining CNN modules with experimental images. In particular, imaging scenarios that previously posed a challenge to the method since the original training was conducted on computationally generated samples exclusively can be inferred with high precision after retraining, as demonstrated in Figures 6.6 and 6.11. With regard to statistically significant metrics, it is those images that are required for an even more accurate analysis result. This is underlined in Figures 6.12 and 6.13 that compare nanoparticle area histograms of the original *nNPipe* method and after retraining for the Au/Ge and Pd/C system respectively.

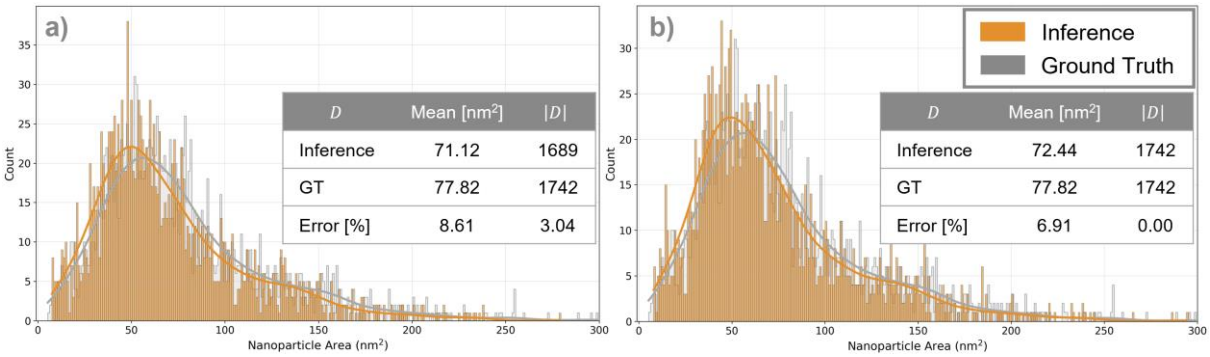


Figure 6.12: Nanoparticle area histograms (1 nm²/bin) for the Au/Ge system with corresponding KDE-plots for **a)** The original *nNPipe* output and **b)** *nNPipe* output after retraining the network with an ESS of 10 images, augmented to 10 % of the original training dataset. The mean F1-Score of the selected retraining set is 0.927. All KDE-plots were generated using a Gaussian kernel and Scott's rule [418]. Tables listing calculated mean nanoparticle areas (nm²) and nanoparticle counts ($|D|$) are added for quantitative comparison. The abbreviation 'GT' refers to ground truth respectively.

In the case of the Au/Ge system, the errors of inferred mean nanoparticle area with respect to the ground truth were reduced from 8.61 % to 6.91 %, while the counting error was reduced to 0.00 %. However, that this does not imply that all nanoparticles have been inferred correctly, but rather that instances mostly consisting of *FN* pixels and *FP* pixels (see Eq. 4.1 and Eq. 4.2) are likely to cancel, as it can be deduced from the results obtained in Figure 6.6. Furthermore, it appears that the *nNPipe* method still exhibits a bias in underestimating the nanoparticle borders and thus infers instances at a smaller area compared to the ground truth (compare Section 5.2.3). However, the overall inference error after retraining is smaller than that originally determined for sub-dataset containing images for F1-Scores > 0.9 only (Figure 5.5b) and thus this result highlights peak inference performance on the full experimental dataset instead of a selected sub-set.

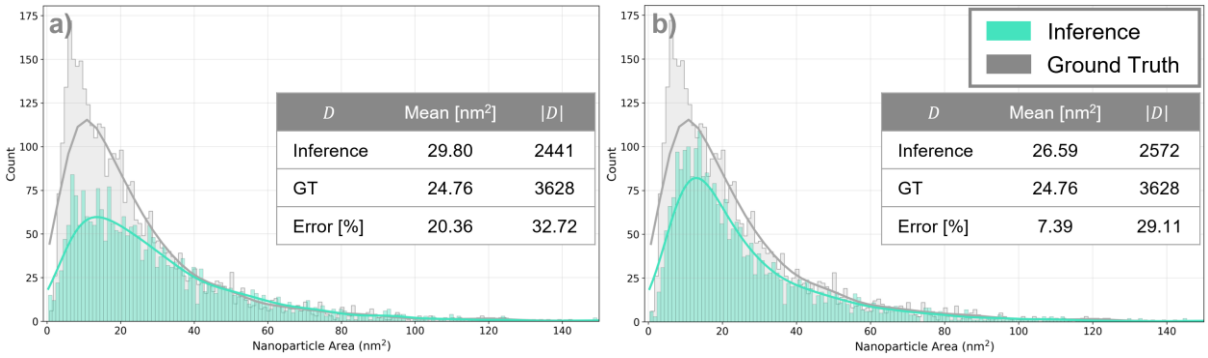


Figure 6.13: Nanoparticle area histograms (1 nm²/bin) from the Pd/C system with corresponding KDE-plots for **a)** Original *nNPipe* output and **b)** *nNPipe* output after retraining the network with an ESS of 24 images, augmented to 10 % of the original training dataset. The mean F1-Score of the selected retraining set is 0.689. All KDE-plots were generated using a Gaussian kernel and Scott's rule [418]. Tables listing calculated mean nanoparticle areas (nm²) and nanoparticle counts ($|D|$) are added for quantitative comparison. The abbreviation 'GT' refers to ground truth.

Considering the Pd/C system in Figure 6.13, a drastic reduction from 20.36 % to 7.39 % is observed for the inferred nanoparticle area, while that for inferred nanoparticle count from 32.72 % to 29.11 % is less significant. The retrained *nNPipe* method renders a KDE-plot shape more similar to the ground truth with a mean nanoparticle area error less than observed for the sub-dataset of images with F1-Scores > 0.9 (Figure 5.7b). As for the Au/Ge system, this indicates a high inference performance for the entire HRTEM Pd/C dataset, but given a higher error for nanoparticle counts, individual users may still choose to apply pruning strategies such as the ID-criterion (Section 5.2.3) to ensure highest possible accuracy.

Most importantly, the retraining strategies applied were selected with emphasis on fast deployment and practicality. For this purpose, a limited number of retraining epochs per module, but also the experimental subset size and augmentation factors leading to specific training dataset sizes have been considered. The key findings can be summarised as follows:

i) For *nNPipe*, retraining *via fine-tuning* yields better results than *transfer learning* whereby learning rates of 10^{-4} have an advantage over learning rates of 10^{-5}. Enabling parameter updates allows the encoding of new, unseen features which could be of particular importance if networks were initially trained on image simulations.

ii) Even the smallest training datasets used for retraining (smallest ESS and augmentation) will lead to a performance improvement of *nNPipe*.

iii) For more uniform heterogeneous systems such as Au/Ge, it is possible to achieve close to the highest inference performance by only using 10 % of experimental images and augmenting them to a final training dataset of 10 % of the original training. Short retraining periods may suffice, but longer retraining is likely to give the best results.

iv) Heterogeneous catalyst systems which are more diverse in morphology and image intensity require larger ESS to yield best performance, which can be linked to a minimum amount of new training scenarios required to be encoded during retraining. Nonetheless, the results here show that an ESS of 25 % can be sufficient to achieve similar results to those obtained using ESS of 100 %. Overall, retraining for this type of heterogeneous catalyst requires longer training periods when compared to more uniform systems.

The above findings have important consequences for a more widespread use of DL-based analysis methods such as *nNPipe*. Depending on the complexity of the material system, future users do not need to generate a large training dataset, but rather annotate a small fraction of (S)TEM images from which a retraining dataset can be generated. It is however important that most new imaging scenarios are covered to ensure best performance. Overall, this reduces the expenditure to obtain an automated analysis system for statistically significant metrics drastically. This is also demonstrated Figure 6.14, highlighting an automated analysis of a Pt/C catalyst system.

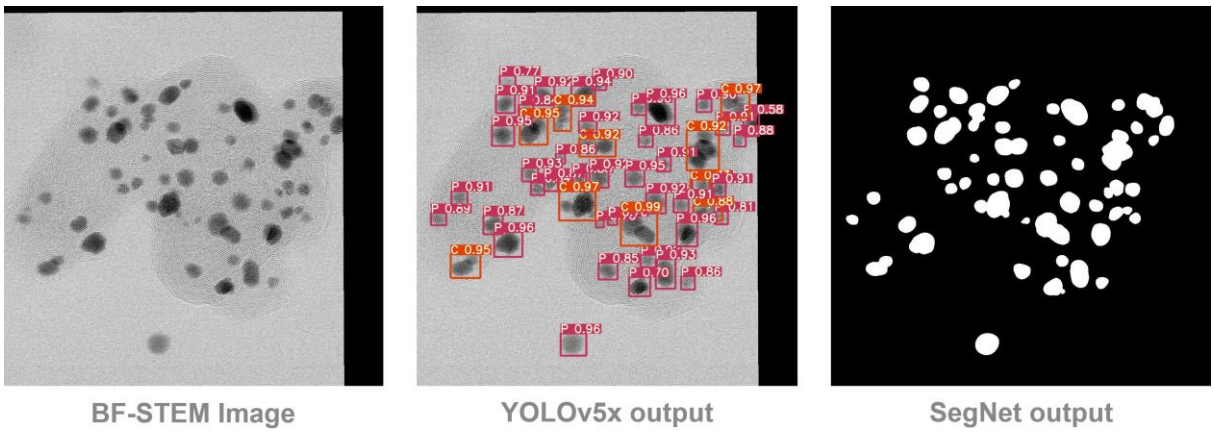


Figure 6.14: Cross-sample retraining example of *nNPipe*. In this case, *nNPipe* was retrained to automatically analyse 1024 x 1024 in situ BF-STEM images of a Pt/C catalyst. Images were acquired on a DENS Solution wildfire heating chip at temperatures ranging from 833 °C to 1100 °C by Dr. Manfred E. Schuster. For automated analysis, *nNPipe* was *fine-tuned* using 10 % of the original experimental dataset containing 144 images in total at a learning rate of 10^{-4}.

6.2 Democratised Preparation of Training Data

The results presented in Section 6.1 unambiguously showed that it is possible to retrain a generalised DL-based method such as *nNPipe* to high inference performance with relatively few experimental images. However, generalised networks may not be available for all materials systems routinely analysed in (S)TEM imaging. This situation forces a decision between two approaches for supervised DL-based analysis methods. Either, (re)training data is generated using computationally driven methods such as image simulations, or by annotating large amount of experimental data. The latter may be time-consuming and introduces and unintended, inherent bias, but may be favoured in practice as knowledge of the former might not always be available.

Hence, important questions on the currently common training dataset paradigm emerge: Is expert knowledge always required to correctly annotate (S)TEM images and is it possible to reduce manual annotation bias by considering multiple assessors for a training dataset? This section aims to establish an initial pathway to answer this question and potentially to a new preparation stategy for DL (re)training datasets in EM.

The following study and results leverage the *zooniverse.org* platform where a large group of mostly non-expert volunteers of different age, gender, ethnicity, occupation and background can annotate HRTEM images of previously introduced heterogeneous catalysts. This project has been conducted in close collaboration with Dr. Michele Darrow, Dr. Olly King, Dr. Avery Pennington and Dr. Mark Basham and was named ‘Science Scribbler: Key2Cat’ (‘key to catalysis’). At the time of writing, 1612 volunteers have contributed to this project in a total of four different annotation workflows described below. The project is still ongoing and thus the results presented here should be considered as preliminary. As will be shown, the insights obtained at this stage are nevertheless promising, which is why they are introduced in a separate section of this thesis and may potentially provide an additional ‘adaption’ method for training dataset preparation.

6.2.1 Study Preparation

This section outlines the framework that was prepared on the zooniverse.org platform for HRTEM images to be annotated at two different stages, reflected in four different workflows. One aim of this study is to compare the annotations obtained to an expert ground truth (Section 4.2.4 and 5.1) quantitatively to obtain insights on training dataset preparation.

6.2.1.1 The Zooniverse Platform

Zooniverse is a web-based research platform (https://www.zooniverse.org) for people-powered science [423], on which over a million of volunteers contribute and assist professional researchers. The platform's key aim is to facilitate research that would otherwise not be viable or practical [423] and covers research projects in the fields of Space, Physics, Climate, Humanities, Nature, Medicine, *etc.* As the core questions formulated in the previous section would require a large number of annotations by non-expert contributors, it was quickly realised that this platform provides a highly suitable space for a useful study, leading to the Science Scribbler: Key2Cat project.

6.2.1.2 Study Datasets

The two previously introduced HRTEM experimental datasets of heterogeneous catalysts (see Section 4.1) were considered for annotation by zooniverse volunteers. However, the preparation of the respective ground truths, both for object detection and binary segmented maps (see Section 4.2.4 and 5.1) is likely to be perceived as complicated, unfeasible and a time-consuming task, especially to volunteers that are introduced to EM or (S)TEM images for the first time.

Therefore, the annotation of individual images was simplified and separated into two different stages to obtain most important information for comparison, namely pairs of (x, y) image coordinates describing centroids of individual nanoparticles or clusters and their corresponding classifications. This was implemented by creating 'Particle Picking' and a 'Particle Cleaning' workflows for both material systems, resulting in four workflows in total.

6.2.1.2.1 Particle Picking Dataset

To ensure high practicality for volunteers, but also multifold inspection of individual catalyst particle instances from which (x, y) image coordinates could be caculated, experimental HRTEM images were pre-processed with methods that were proven to be effective in previous Science Scribbler projects [424–426].

First, histograms of individual HRTEM images were contrast enhanced within the range of values $[mean_{im} - 2.5\sigma_{im}, mean_{im} + 2.5\sigma_{im}]$, whereby $mean_{im}$ refers to the 2D mean and σ_{im} refers to the standard deviation of the image respectively. The value for $2.5\sigma_{im}$ was chosen empirically and specifically, as previous tests suggested that it established an optimum for improved contrast for nanoparticle discernibility without overenhancing contrast of support material or artifacts. Secondly, images were grayscale-padded (Figure 6.15c) to an image size of 2736 x 2736 pixels to enable uniformly cropped images of 684 x 684 pixels with a 50 % overlap (342 x 342 pixels). This resulted in a 7 x 7 cropping matrix which was chosen since cropped sections contained a number of nanoparticle instances that was believed to be suitable for volunteers. Furthermore, this strategy ensured that individual instances would be inspected at least four times since each instance appeared partially or fully in multiple image sections. This pre-processing method was applied for both the Au/Ge and Pd/C HRTEM image datasets and individual image sections were subsequently uploaded to the Zooniverse platform to be presented to volunteers in two separate 'Particle Picking' workflows. Figure 6.15 illustrates an exemplary preparation of the 'Particle Picking' dataset with the respective python code provided in Appendix 10.7.

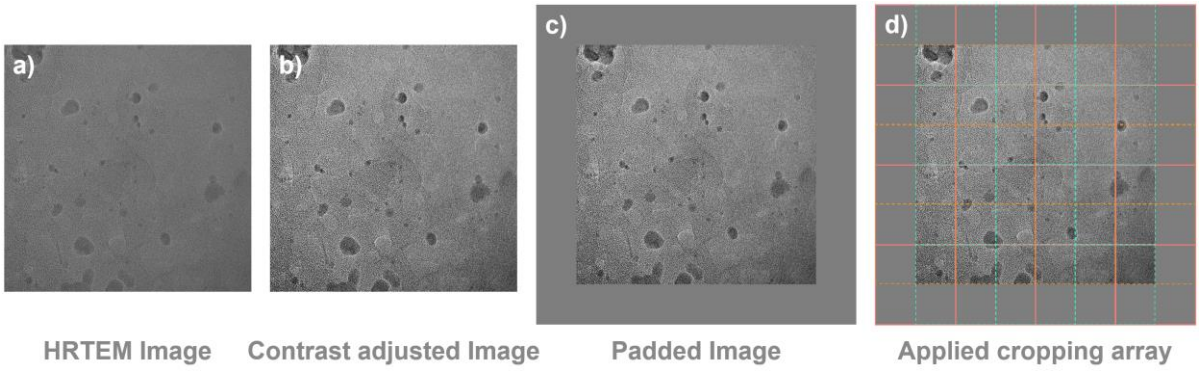


Figure 6.15: Schematic illustration of pre-processing process for the 'Particle Picking' datasets shown for an exemplary Pd/C HRTEM image. **a)** Original HRTEM image. **b)** HRTEM image after contrast enhancement. **c)** Contrast enhanced image padded to an image size of 2736 x 2736. **d)** Application of 684 x 684 image cropping with 342 x 342 overlap to obtain individual images that were subsequently presented to volunteers for particle picking. Individually coloured matrices indicate areas respective cropped sections extracted.

6.2.1.2.2 Particle Cleaning Dataset

Once respective volunteers have completed the 'Particle Picking' workflow, the (x, y) image coordinates were used to generate centralised cropped sections of 256 x 256 pixels in padded images (Figure 6.15c). This size was primarily chosen as it provided a good field of view and context for classification of individual instances, but also because this size would potentially allow for a more direct comparison to the SegNet module of *nNPipe* (compare Section 5.1). In a second step, a green circle enclosing the picked instance was added to each image section to act as a 'focal area' on which volunteers were asked to focus their attention on during classification. Prepared datasets were subsequently uploaded to the Zooniverse platform and linked to the 'Particle Cleaning' workflow (see below), where volunteers were asked to classify presented images as 'Single Nanoparticle', 'Nanoparticle Cluster', 'Artifact' and 'Nothing/Empty'. Hence, this workflow was not only used to classify previously picked instances, but also to remove potential outliers irrelevant for the training dataset. The preparation of the 'Particle Cleaning' dataset is described in Figure 6.16 for an overview. Respective code for data generation is provided in Appendix 10.7.

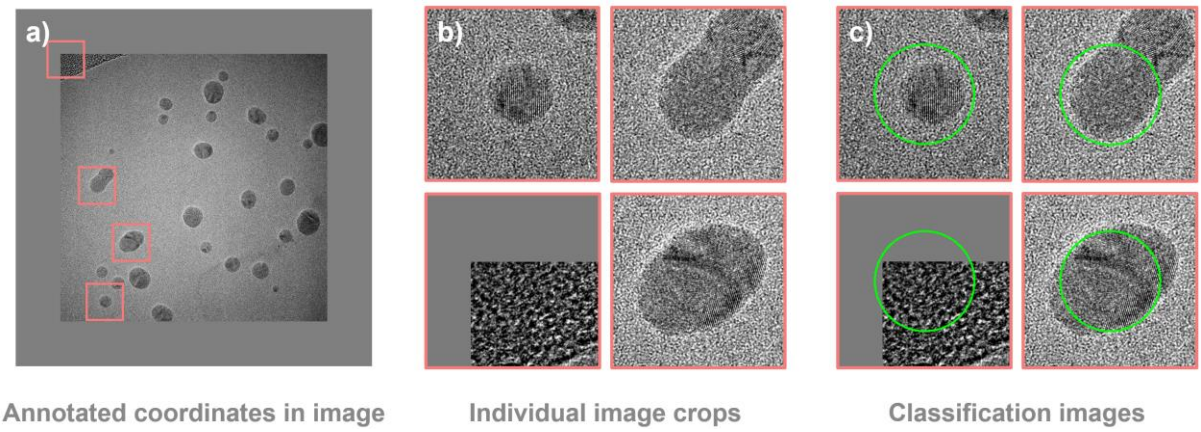


Figure 6.16: Pre-processing of the 'Particle Cleaning' dataset based on an exemplary image of the Au/Ge dataset. **a)** (x, y) Image coordinates obtained from the 'Particle Picking' workflow allow for centralised sections of 256 x 256 pixels as shown in **b)**. **c)** Cropped instances with green circles added in the centre of cropped sections to highlight an area of interest to facilitate classification for volunteers. From left to right and top to bottom, the correct classes in this example would be 'Nanoparticle', 'Cluster', 'Artifact' and 'Nanoparticle'.

6.2.1.3 Annotation Workflows

This section briefly outlines individual workflows, including tutorials to introduce volunteers to the respective tasks, available tools for annotation, as well as additional instructions and resources.

6.2.1.3.1 Particle Picking Workflow

In this workflow, volunteers contributing to the project were asked to place marks on the centroids of individual nanoparticles or on nanoparticles that form a cluster. A tutorial was prepared to introduce volunteers to the image type, to provide different examples for correct differentiation and to the usage of the 'marker tool' for annotation. Figure 6.17 displays the panels of the tutorial.

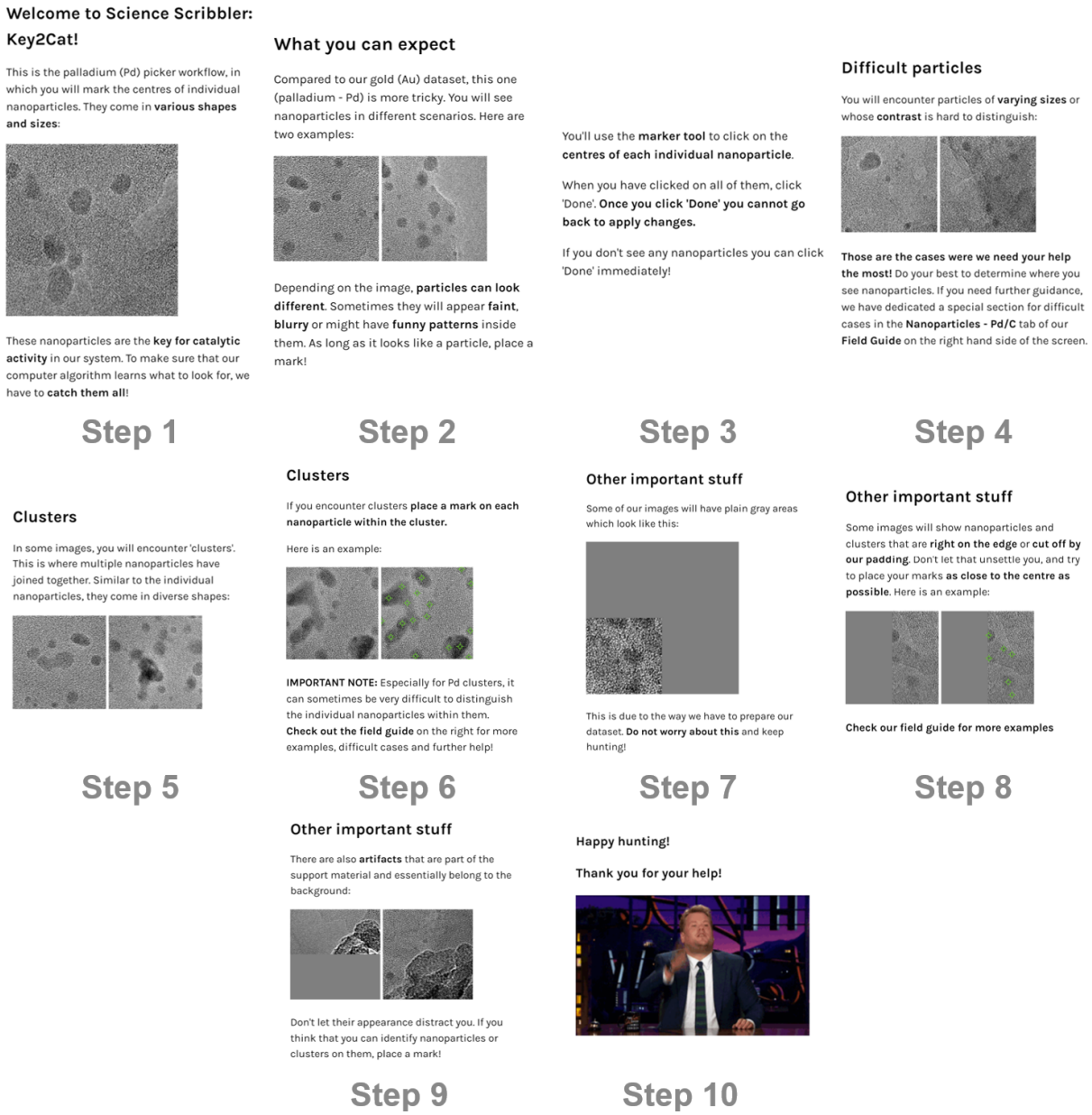


Figure 6.17: Tutorial for the 'Particle Picker' workflow as introduced for the Pd/C image dataset. Steps indicate the order of individual panels as they were displayed to introduce volunteers to the task. Note that the tutorial and respective tasks were formulated in a formal fashion to increase participation. A similar workflow was prepared for the Au/Ge dataset.

It is important to note that only few exemplary unmarked-marked image pairs were provided in the tutorial to avoid any introduction of bias, but sufficient to make the task clear. Volunteers were furthermore asked to place marks on instances they were not certain about, as wrong particle annotations could be classified as irrelevant in the subsequent 'Particle Cleaning' workflow. For cases in which a volunteer was unsure or confused, an additional 'Field Guide' provided further examples of potential imaging scenarios that could be encountered in addition to a community board that allowed direct contact to researchers for questions. Figure 6.18 shows the workflow view as typically encountered. Each image was annotated by 5 different volunteers.

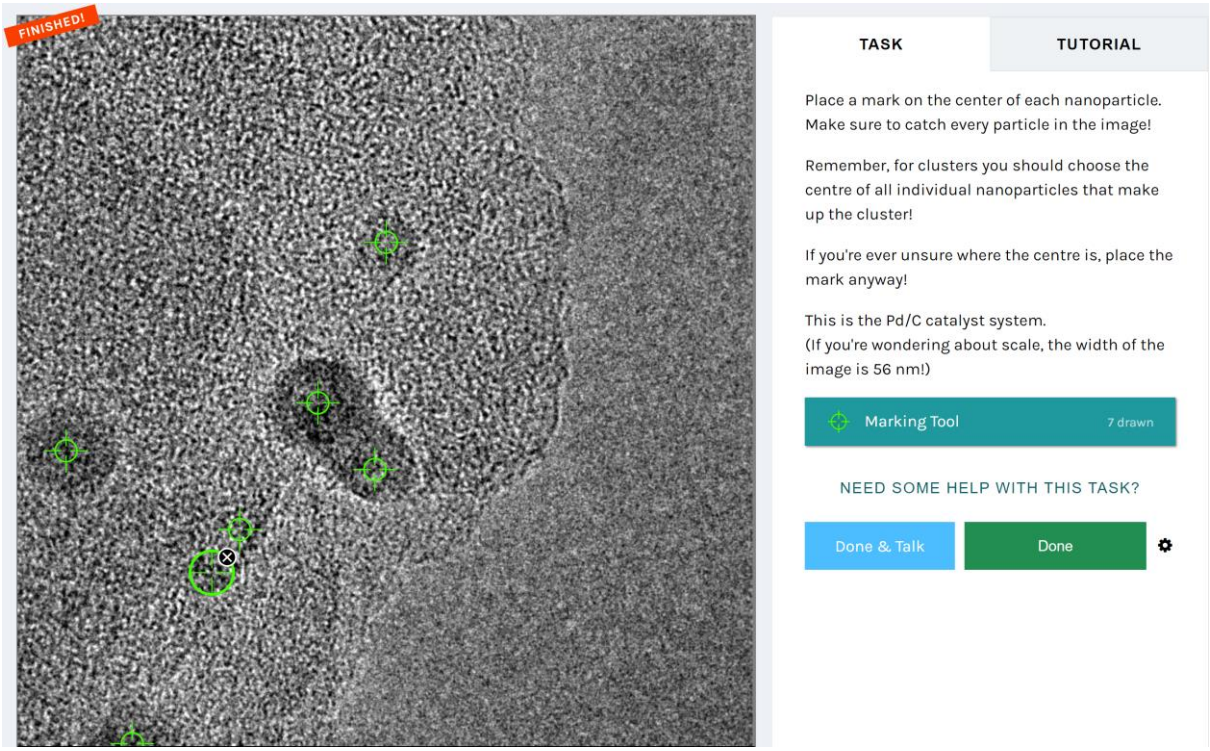


Figure 6.18: Typical view of the 'Particle Picking' workflow for the Pd/C dataset as encountered by a volunteer. The Marking Tool can be used to annotate centroids of individual nanoparticles and nanoparticles constituting a cluster. Instructions on the right hand side summarise the task previously introduced in the tutorial. Once a volunteer has finished, the 'Done' button can be clicked to view the next image. This workflow would continue indefinitely and each annotation would be saved immediately. This way, volunteers could freely decide on the time spent for annotation.

6.2.1.3.2 Particle Cleaning Workflow

As for the 'Particle Picking' workflow, the 'Particle Cleaning' workflow was introduced by a tutorial. In this case, volunteers were asked to classify previously picked instances into different categories to distinguish individual nanoparticles, clusters, artifacts, or empty areas (compare 6.2.1.2.2). Figure 6.19 displays the tutorial panels used for task introduction.

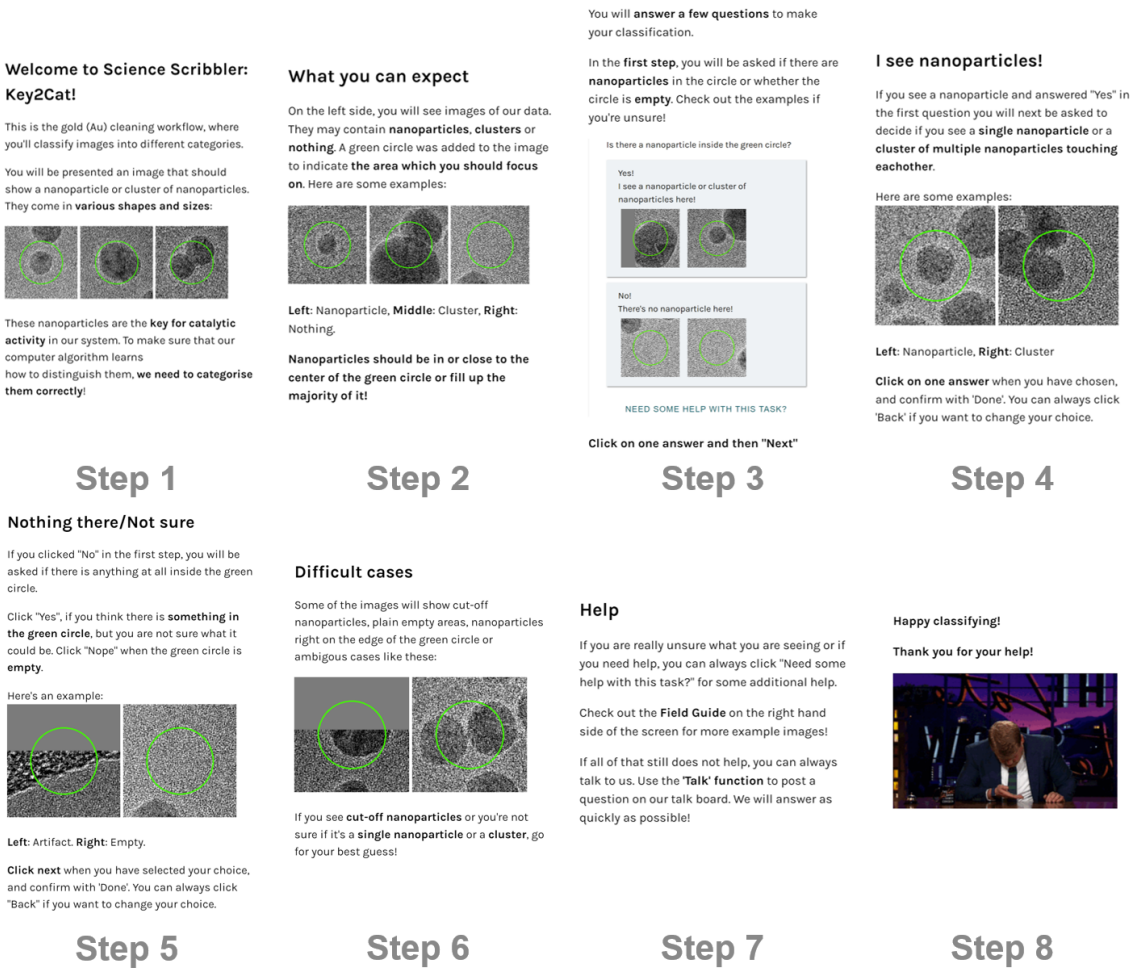


Figure 6.19: Tutorial explaining the tasks of the 'Particle Cleaner' workflow with the Au/Ge dataset selected as an example. Step numbers indicate the order in which individual panels were displayed to volunteers. The language used was chosen to be less formal to increase participation.

By analogy with the 'Particle Picking' workflow, only a few examples were provided with respective labels to avoid the introduction of a bias and a field guide and community board were given for cases in which the volunteers were unsure or confused. Once the tutorial was passed, a decision tree was prompted with which classification was conducted. The steps were formulated as follows:

Decision 1: It there a nanoparticle inside the green circle?

Option 1: Yes! I see a nanoparticle or cluster of nanoparticles here!

Option 2: No! There's no nanoparticle here!

Decision 2 with Option 1 previously selected: Is this a single nanoparticle or a cluster of nanoparticles?

Option 1: There's a single nanoparticle here! → **Class: Nanoparticle**

Option 2: There's a cluster of nanoparticles here! → **Class: Cluster**

Decision 2 with Option 2 previously selected: Is there anything here?

Option 1: Yes! There's something here, but it's not a nanoparticle! → **Class: Artifact**

Option 2: Nope! Nothing here at all! → **Class: Nothing/Empty**

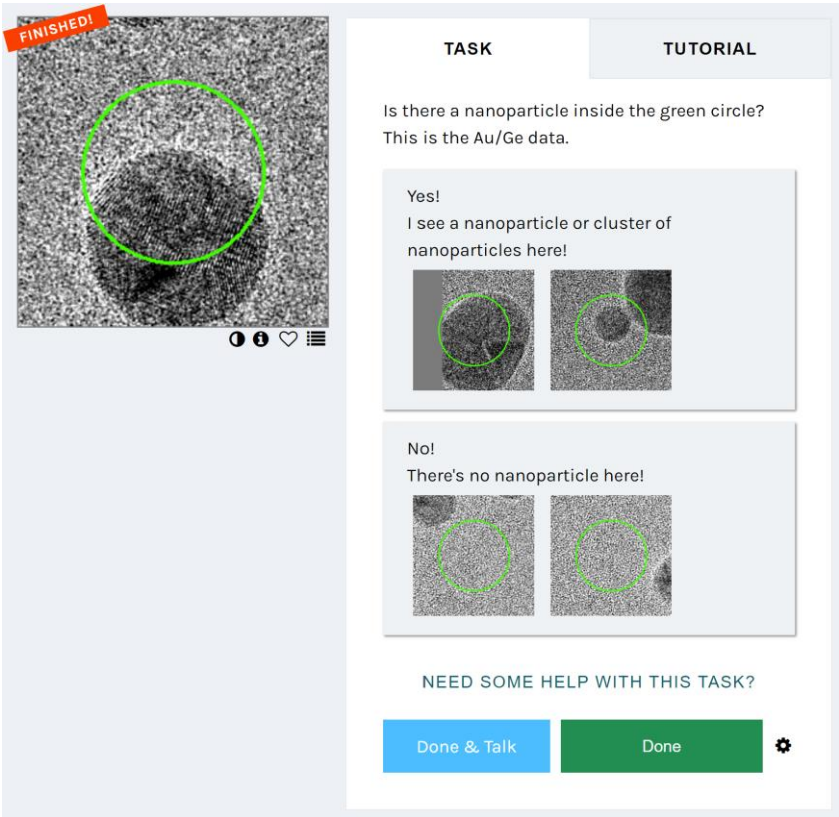


Figure 6.20: Typical view of the 'Particle Cleaning' workflow for the Au/Ge dataset as t encountered by a volunteer. Two questions are prompted with two different options respectively to classify the image presented on the right hand side. Pictures shown in the answer options act as additional decision support. If a nanoparticle or cluster is within the green circle, but slightly shifted into any direction as in the example shown, volunteers were asked to classify the instance as 'nanoparticle' or 'cluster' nonetheless. The label 'Nothing/Empty' should be only given when the majority of the area enclosed by the green circle would not show any instance.

As an alternative, it would have been possible to prompt the four class options within one single decision, but this strategy was not pursued as the task would have become less convenient due to introducing the necessity of scrolling on either the website or the mobile application. Note that it was possible for volunteers to change their decision even if an option has already been selected. Only classifications for which the decision was 'confirmed' at the end of the decision tree, were considered for analysis. Finally, Figure 6.20 shows an image of the 'Particle Cleaner' workflow as typically encountered. As for the 'Particle Picking' workflow, each image was classified five times by different volunteers respectively.

6.2.2 Preliminary Results

At the time of writing, the Science Scribbler: Key2Cat project is ongoing, whereby the last workflow, 'Particle Cleaning – Pd/C' has begun. Thus, the results presented herein are inherently preliminary for a community project and thus emphasise the potential for a new strategy of preparing (re)training data for DL-based analysis methods in (S)TEM imaging.

6.2.2.1 Particle Picking

After completion of the 'Particle Picking' workflows, registered results were downloaded from *zooniverse.org*. Relevant data was extracted with dedicated scripts that were also used for the dataset preparation of the 'Particle Cleaning' workflow (Appendix 10.7) and stored in separate .csv files. Individual picks were subsequently linked to corresponding image sections (compare to Section 6.2.1.2.1) and reconstructed to the original, contrast enhanced HRTEM images. An example of this process is shown in Figure 6.21.

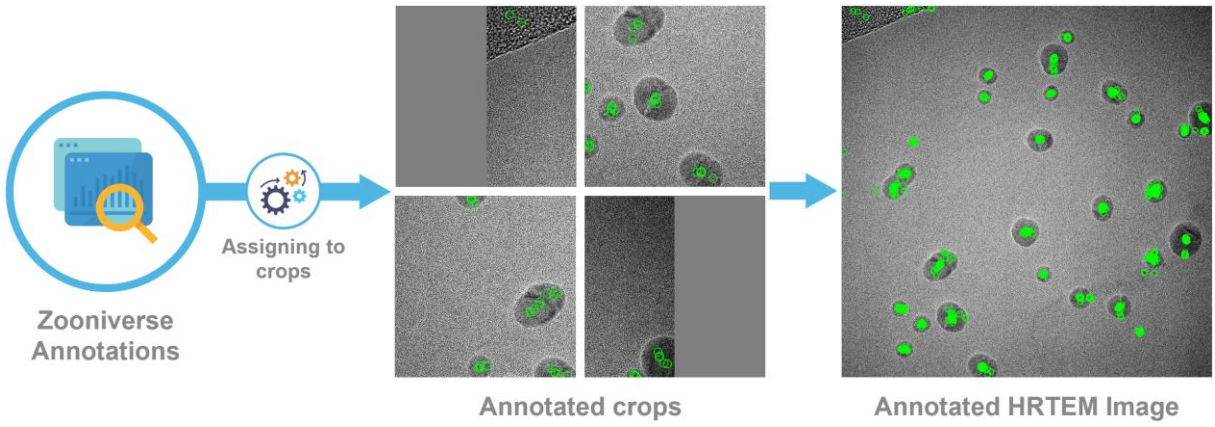


Figure 6.21: Schematic workflow of reconstructing annotations to corresponding contrast enhanced HRTEM images. Once the 'Particle Picking' workflows were completed for a material system, annotations were downloaded and processed into a .csv file containing most relevant data. Annotations were assigned to corresponding image sections, from which the full image was reconstructed. Note that annotations were given in pairs of (x, y) coordinates, but green circles were used in this illustration for improved visibility.

Since each image section was annotated five times by different volunteers and individual sections of 684 x 684 pixels overlapped by 50 %, up to 20 annotations could be aggregated per instance after reconstruction. However, since instances may be cut in specific sections as shown in Figure 6.21, annotations would spread over centroids after reconstruction.

Subsequently, the clustering algorithm HDBSCAN [427] was used to calculate centroids of annotation for clusters in reconstructed HRTEM images (Figure 6.21). This algorithm can be understood as an

extension of the DBSCAN [428], where hierarchical clustering is employed first, followed by a technique for extraction of flat clusters based on their stability [429]. HDBSCAN was chosen as it was successfully used for annotation clustering in previous Science Scribbler projects, while further details of its functionality are given in the documentation [429]. Initial results obtained for both, the Au/Ge and the Pd/C system are shown in Figures 6.22 and 6.23 respectively. Note that the only parameter given for the HDBSCAN algorithm was the minimum cluster size, which was empirically set to 11, as this value represented the majority of 20 picks. This may be subject to change in future work.

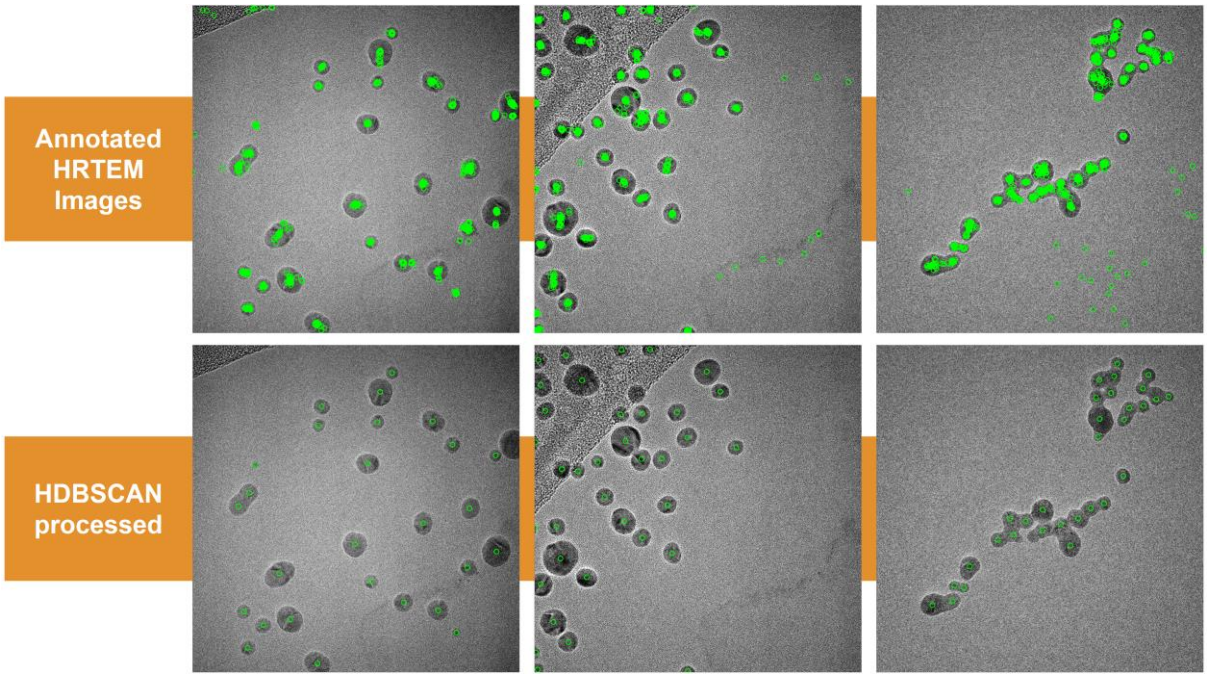


Figure 6.22: Typical images of annotated and contrast enhanced HRTEM images from the Au/Ge system (top row) and after processing with the HDBSCAN algorithm (bottom row) with the minimum sample per cluster empirically set to 11. Images were selected to illustrate common images, but also more difficult particle instances such as nanoparticles on lacy carbon or nanoparticle clusters.

The results shown in Figure 6.22 unambiguously prove that annotations given by mostly non-expert volunteers can be processed with the HDBSCAN algorithm to attain centroids of nanoparticle instances with high precision. Despite that some nanoparticle instances were only visible on the edge of the HRTEM image or on 'artifacts' such as lacy carbon, annotations were given correctly for all of these with the exception of a single nanoparticle. Moreover, even complicated cluster shapes as shown in the right column of Figure 6.22 were annotated correctly and the centroids of the constituent nanoparticles could be obtained for all instances. Misidentified selections were furthermore filtered out effectively as shown in the middle and right examples.

Similar results are reported for the Pd/C system shown in Figure 6.23, where precise results on centroid annotations were achieved although the catalyst system is morphologically more diverse and may be thus more difficult to interpret.

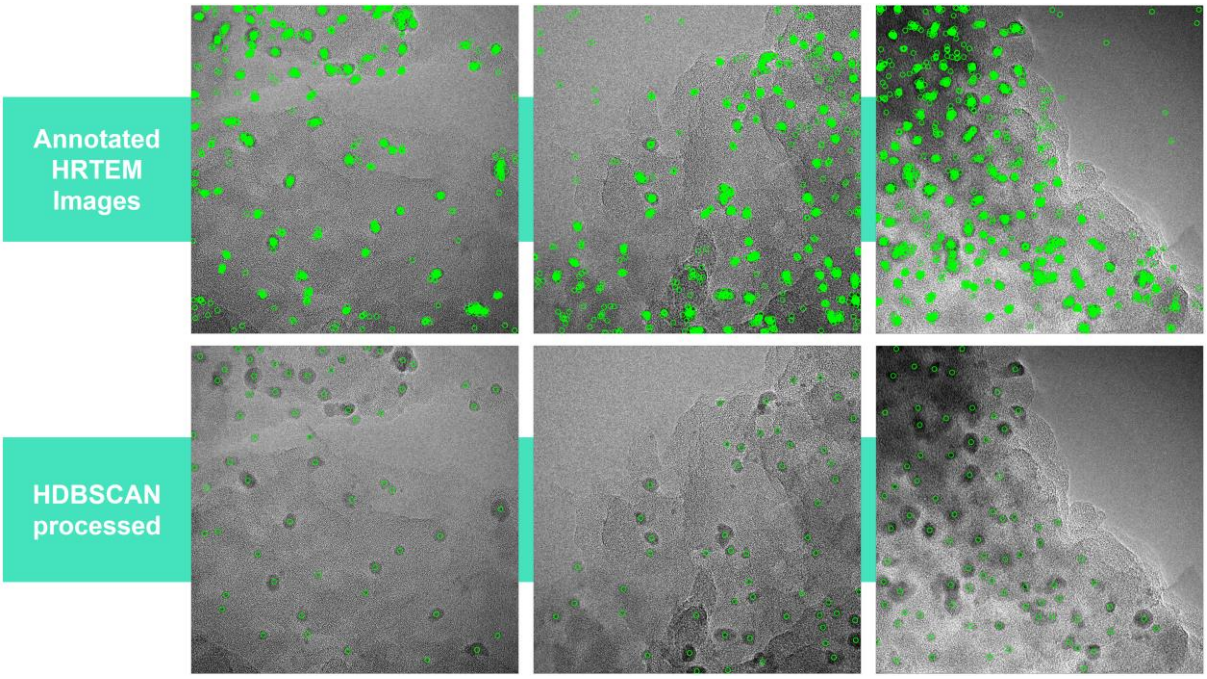


Figure 6.23: Selected images of annotated and contrast enhanced HRTEM images of the Pd/C system (top row) and after processing with the HDBSCAN algorithm (bottom row) with the minimum sample per cluster empirically set to 11. Images were selected to illustrate different imaging scenarios, including contrast variations, different nanoparticle sizes as well as thick layers of support material.

The examples shown on the left-hand side and middle of Figure 6.23 illustrate typical imaging scenarios for the Pd/C dataset where individual clusters and nanoparticles were encountered with different sizes, contrast and distributions. Centroids obtained after processing the annotations with the HDBSCAN algorithm show that nanoparticle instances were successfully distinguished by volunteers, although the Pd/C dataset is more complex. Most notably, the annotations remain precise even for the most difficult cases which incorporate nanoparticles buried amongst thick layers of amorphous carbon, exhibiting a blurred contrast (right hand side, Figure 6.23).

6.2.2.2 Particle Cleaning

Classes obtained from the 'Particle Cleaning' workflow (compare 6.2.1.2.2) were post-processed into numerical labels of: 0 – Nanoparticle, 1 – Cluster, 2 – Artifact, 3 – Nothing/Empty. Because each nanoparticle annotation was inspected five times, the resulting class for each annotation was given by the highest count. Subsequently, classified annotations were linked to respective annotation clusters

obtained by the HDBSCAN algorithm (see 6.2.2.1), for which a class was assigned by a majority principle. An example of the full classification process is shown for a typical Au/Ge HRTEM image in Figure 6.24. It is noted that the post-processing strategy is currently still under investigation and thus subject to further change in the future.

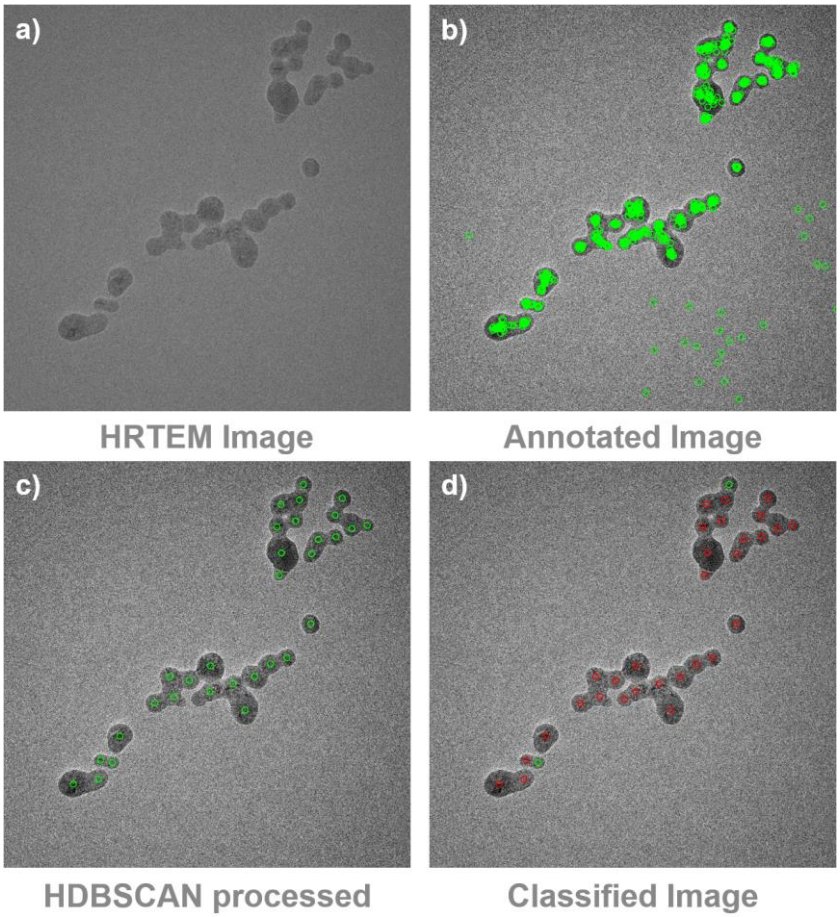


Figure 6.24: Full annotation and classification workflow of a single Au/Ge HRTEM image based on data obtain from zooniverse.org. **a)** Raw HRTEM image. **b)** Contrast-enhanced HRTEM image with reconstructed annotations obtained from the 'Particle Picking' workflow. **c)** Image from **b)** post-processed using the HDBSCAN algorithm. **d)** Annotation centroids with assigned classes obtained by majority counts from the 'Particle Cleaning' workflow. For **d)**, red circles refer to the assigned class of 'cluster' while green circles refer to 'single nanoparticle' respectively.

Considering Figure 6.24c and 6.24d, it is clear that annotated centroids have been successfully identified as 'Cluster' in most cases. This is an important result as it suggests that not only individual centroids of nanoparticles shown in HRTEM images can be assigned with high precision by non-expert volunteers, but also that individual picks can be classified reliably. However, as the treatment of this data is currently still under development, this result is yet to be confirmed for other imaging scenarios and alternative datasets.

6.2.3 Discussion

At the current time, the results described from an ongoing study are implicitly preliminary and thus of restricted significance. However, the initial insights described Section 6.2.2 are nonetheless promising for a potentially new strategy of (re)training dataset generation. As has been shown for both materials systems (Figure 6.22 and 6.23), annotations obtained from the 'Particle Picker' workflow allowed extraction of nanoparticle instance centroids with high precision. Most strikingly, volunteers were able to distinguish individual nanoparticles and clusters for varying imaging conditions, varying contrast and varying complexity of individual shapes. The subsequent post-processing *via* the HDBSCAN algorithm reliably calculated centroids arising from clusters of annotations while misidentified cases were filtered out in all cases.

Annotations were subsequently inspected by volunteers for classification and it was shown that the correct classes were assigned in the majority (93.75 %) of annotation centroids shown in Figure 6.24. These results give a first indication that it is possible to present HRTEM images of varying complexity to an audience of mostly non-expert volunteers for precise annotation. However, the annotation task had to be simplified when compared to the manual labelling of bounding boxes or binary segmented maps. Furthermore, clear instructions must be given to assure appropriate precision.

In the next steps, the post-processing algorithms for both, 'Particle Picking' and 'Particle Cleaning' will be further refined to ensure the generation of labels with improved precision. Based on these, (semi-) automated image processing approaches will be explored with the goal being to obtain images containing bounding box labels or a binary segmented map for both HRTEM image datasets. The use of these could be multifold:

i) Training of the *nNPipe* method from scratch,

ii) Retraining the original *nNPipe* method and

iii) Comparison of the ground truth datasets as annotated by experts.

Based on the results achieved, an entirely new strategy of training dataset preparation may be developed for (re)training DL-based methods for the analysis of heterogeneous catalyst materials.

6.3 Chapter Summary

In this chapter, retraining and adaption methods for both *nNPipe* and the preparation of training datasets have been discussed. Results outlined in Section 6.1 discussed the application of a retraining method targeted at researchers willing to use a pretrained CNN for differing materials systems. It was shown that 10 % of a new experimental image dataset may be sufficient to retrain a network to high inference performance. For morphologically diverse catalyst systems, at least 25 % are required to cover a majority of possible new imaging scenarios. For both the Au/Ge and Pd/C material systems, inference errors were significantly reduced after retraining with experimental subsets. Section 6.2 discussed a potential new strategy for the generation of training datasets. As an alternative to computationally aided methods or expert annotation, 1612 mostly non-expert volunteers were asked to annotate HRTEM image datasets by simplified workflows on the Zooniverse platform. Preliminary results indicated that volunteers annotated centroids of nanoparticle instances with high precision, whilst individual classes could be assigned correctly.

7 Advanced Applications

In this last of three experimental chapters, advanced applications of *nNPipe* are described for time-resolved imaging. Specifically, the implementation of a real-time, live analysis setup for HRTEM imaging is discussed in Section 7.1, while Section 7.2 describes DL-based analysis of datasets acquired during *in situ* heating of the Pd/C heterogeneous catalyst.

7.1 Live Image Analysis

In this section, the setup for a real-time, live image analysis method is described. It allows users to apply *nNPipe's* object detection module during both, sample viewing and image acquisition and gives access to bounding box information for individual frames. This can be envisioned to be of particular relevance where higher sample analysis throughput is expected, enabling faster catalyst optimisation, and where direct feedback on the quality of images produced *during* experimentation might be decisive for sample analysis. The findings and setup presented are the result of a close collaboration with Dr. Aakash M. Varambhia, Dr. Manfred E. Schuster and Dr. Doğan Özkaya during an internship with Johnson Matthey plc.

7.1.1 Setup Structure and Functionality

In the following, the setup structure for live image analysis is described for any (S)TEM instrument. Successful application is only dependent on the two software packages GMS 3 [430] (formerly known as Digital Micrograph, Gatan Inc.) and Python 3.8 (or later) with selected libraries. Two different versions of the setup are described with different respective advantages: Version A, applicable to any (S)TEM instrument operating the above software packages and using file storage and Version B, which is specifically targeted to (S)TEM instruments operating with GMS 3 versions where direct Python integration has been enabled, allowing for direct processing *via* byte streams. Figure 7.1 illustrates Version A of the live imaging setup schematically, while a detailed pseudo-code is provided in Section 10.8 of the Appendix. The working principle can be briefly described as follows:

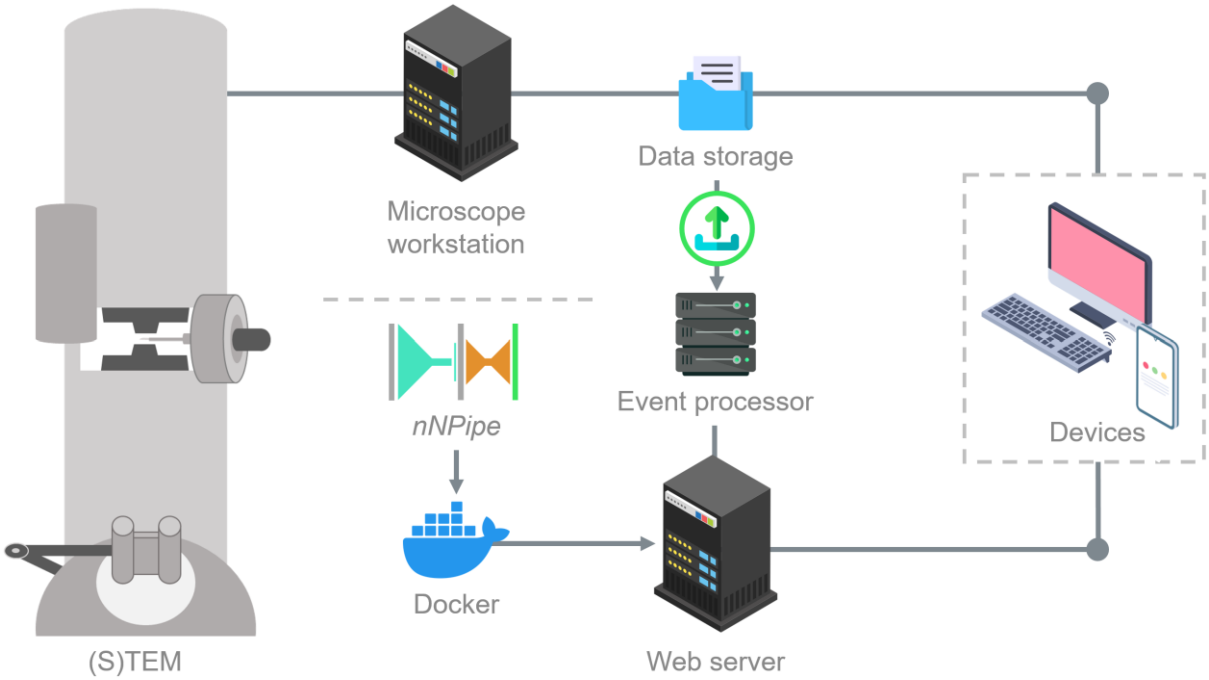


Figure 7.1: Schematic overview of the live imaging setup (Version A). A (S)TEM instrument is connected to a microscope workstation, running GMS 3. The 'GMS Uploader' writes individual frames into a dedicated data storage folder, which are subsequently read by the 'GMS Listener', which is part of the event processor. The REST API client (part of the event processor) subsequently passes the read data to the web server, where the YOLOv5x module of *nNPipe* analyses the image. Results are directly displayed in a web app hosted by the web server. Multiple devices can have access to an analysis session allowing the display of results remotely and in real-time. The Docker logo may sometimes feature a whale with a grin.

Step 1: First, the 'GMS Uploader (Version A)' writes the current frame of the sample view into a pre-defined data storage folder to which a 'GMS Listener' has access. The 'event processor' (Figure 7.1), contains the 'GMS Listener' and REST API [431] client. Individual images are read successively from the data storage folder by the 'GMS Listener', with a buffer of two images imposed to avoid conflicts in data handling. Depending on preferences, users can select between the YOLOv5x object detection module of *nNPipe* (compare Section 5.1) or a customised, non DL-based image processing in the 'GMS Listener', but the former is assumed in subsequent descriptions. Once an image is processed by the 'GMS Listener' it is passed to the REST API client to conduct Step 2 (compare Appendix 10.8).

Step 2: The REST API [431] client requests a fixed session on the web server for the entire data analysis process in which multiple datasets can be analysed. Subsequently, the read frame is resized to 320 x 320 pixels, encoded as a base64 data string, and sent to the web server *via* a custom 'get' function (compare Appendix 10.8) which essentially requests a data analysis module (in this case YOLOv5x object detection) on the web server (Figure 7.1) and awaits a server response for successful processing. Note that resizing to 320 x 320 pixels was chosen to increase the speed of data transfer between REST

API [431] client and REST API [431] server. Individual analysis modules can be installed with corresponding requirements on Docker [432] containers as illustrated in Figure 7.1.

Step 3: The web server (Figure 7.1) is where the REST API [431] server, a web-based analysis application (web app) and individual analysis modules – in this case YOLOv5x object detection – are hosted. At startup, the YOLOv5x neural network is warmed up with empty arrays and set as a global variable to pre-allocate working memory into which individual frames received from the REST API [431] client will be streamed (compare Appendix 10.8). This ensures quick response times and direct deployment when images are analysed. Once a data string is received, the web server converts it into a grayscale '.png' array (frame) for compatibility with YOLOv5x and stores it virtually within memory. Lastly, the received frame is processed by the YOLOv5x module which only generates a dataframe of bounding boxes, equally storing these in memory. Those are overlaid on the virtually stored image and displayed on the web app alongside relevant statistics, effectively generating the YOLOv5x object detection output. At this point, the processing loop is completed, and the next frame is expected by the REST API [431] server. The image overlay, processing and display of corresponding bounding box statistics is intellectual property of Johnson Matthey plc and thus not detailed further.

The operating user can additionally send acquired data to the web server for direct processing using the web app (Figure 7.1, 'Devices'), whilst also accessing the shared web server memory to store results. Furthermore, third-party devices can monitor the results displayed in the web app without interfering which is advantageous if experiments need to be observed remotely in real-time.

Version B of the setup mainly differs in how individual frames are passed by the 'GMS Uploader'. Instead of a write/read mechanism implemented in a data storage folder, frames are directly streamed to the REST API [431] client, allowing for significantly faster data transfer and hence higher frame rates compared to Version A. Nonetheless, the user can still access a data storage folder for direct processing in the web app as described for Version A. Note that only Version A was successfully tested on a probe-corrected JEOL ARM200F in HRTEM mode at ePSIC, Harwell Research Campus. The instruments acceleration voltage was set to 200 kV, while operating in TEM-S mode at spot size 3 and CL 1 aperture of 100 μm and the Au/Ge sample inserted (compare Section 4.1). For inference, the

Au/Ge YOLOv5x module as described in Section 5.1 was used. Respective data transfer and frame rates were not measured for this particular setup, but showed an approximate response rate of 0.5 frames/s on the web app. However, Version B was tested independently (not connected to a TEM instrument) and rates close to real-time video were observed. Thus, it is assumed that further code optimisation (Appendix 10.8) will fully enable real-time processing for Version B.

7.1.2 Experimental Setup Test

A typical view of the web app during processing of frames from the Au/Ge sample is shown in Figure 7.2, while Figure 7.3 shows photographs taken during live imaging experiment.

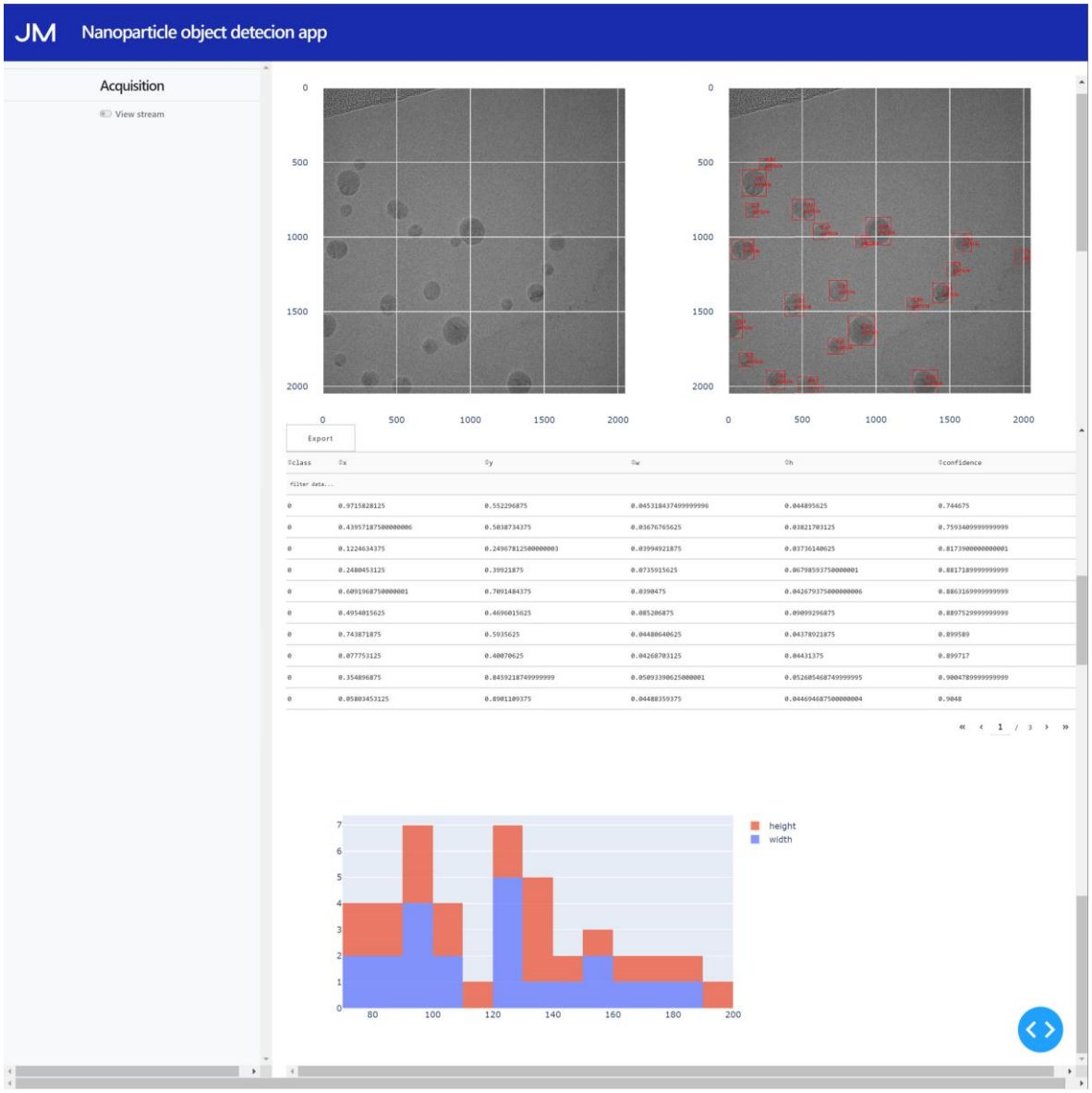


Figure 7.2: Typical web app view when running the live imaging setup during sample view or image acquisition. Note that the view shown was assembled from images displayed at different scrolling positions of the web app as indicated on the right-hand side. The raw image is shown on the top left, the YOLOv5x bounding boxes are overlaid on the top right alongside corresponding statistics shown in the middle and lower part of the view. The view can be customised.

The experimental live imaging setup was run for *ca.* 5 minutes in which the response of the web app was tested. During that time the Au/Ge was manually moved and varying beam convergence angles and magnifications. Although the processing frame rate was only around 0.5 frames/s, it was possible to track the sample movement and calculated results in all devices displaying the web app, highlighting

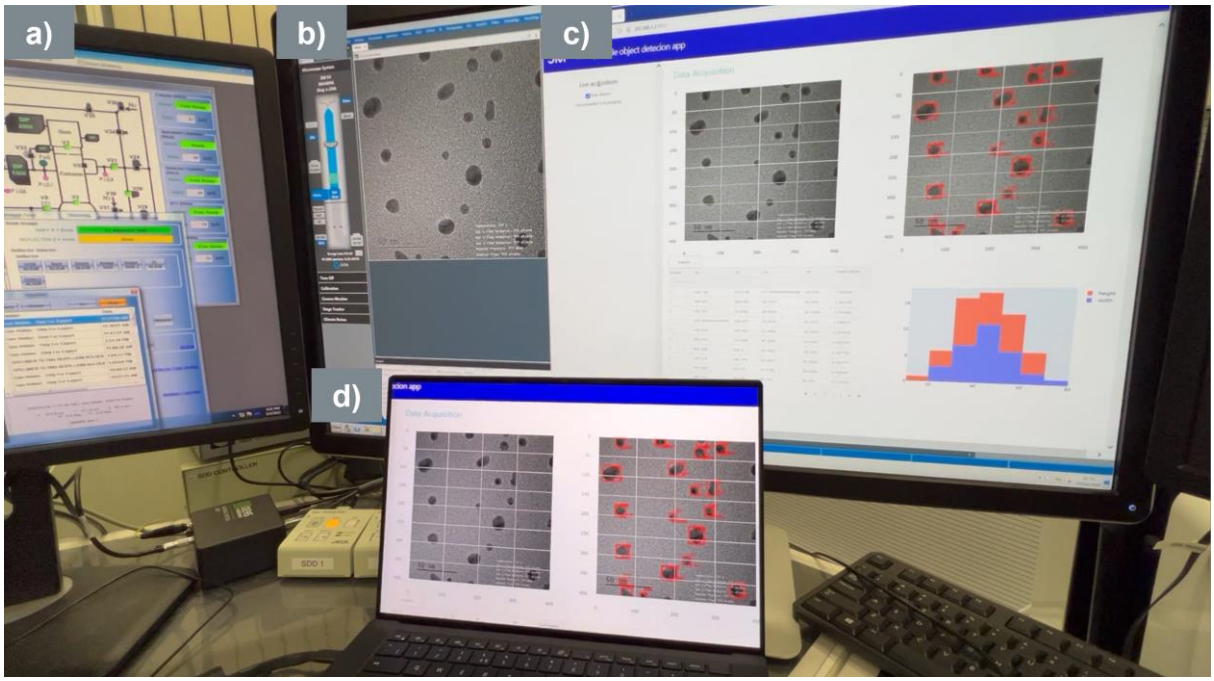


Figure 7.3: Photograph of the experimental live imaging setup tested at ePSIC, Harwell Research Campus, Harwell. **a)** JEOL ARM 200F Operation panel. **b)** Microscope workstation operating GMS 3 with the Au/Ge sample displayed in the sample view. **c)** Web app showing the currently read frame along with YOLOv5x object detection output and relevant statistics. In this case the microscope workstation was also acting as device accessing the data shown in the web app (compare Figure 7.1). **d)** External device viewing the same experimental session in the web app to demonstrate remote view functionality.

future potential to quantitatively track sample changes *in operando*. Unexpectedly, the Au/Ge YOLOv5x module generated precise results even for magnifications and brightness settings which were not included in the original training dataset (compare Section 4.2) which further demonstrates generalisation capabilities.

7.1.3 Discussion

Analytical EM workflows are commonly defined before an actual experiment is conducted and are dependent on the dynamics and sample features expected to be observed [9]. Hence, data acquisition is often linear and selective and analysing increasingly large datasets (see 2.1.8) becomes a hulking task. As it has been already demonstrated in Chapter 5 and 6, DL-based approaches are a key enabler towards a time-efficient and highly precise analysis of large data sets.

This live imaging setup marks an important step for (S)TEM development: Instead of limiting DL-based analysis to post acquisition analysis, it can be used as an integral part *during* operation. Effectively, it allows experimentalists to track sample changes with higher precision, select sample positions based on direct quantitative information and to receive analytical results immediately. Similar analytical routines could be thought of as running in the background during experimental operation and additionally record statistics throughout the entire experimental session.

As discussed, the experimental setup was run using Version A and thus, the effective frame rate shown in the web app was limited to *ca.* 0.5 frames/s. However, it should be noted that the setup itself is the first successfully tested version with further optimisation expected in the near future.

Another important aspect to consider is the analysis output itself. In this implementation users only have access to bounding box information that approximates the shape and size of catalyst particles to the first-order. While YOLOv5x can run inference at up to 83.3 frames/s [433], semantic segmentation processed by *nNPipe's* SegNet module is expected to fall between 1 – 3 frames/s, depending on the number of particles per given frame and material dependent post-processing (compare Table 5.1). Hence, it is likely to be only feasible as a post-acquisition option. Further optimisation of *nNPipes* post-processing routines, working on a smaller field of view, employing multiple GPUs or using customised CNNs with fewer parameters (compare Section 3.4.5) are options that would increase the inference speed for semantic segmentation and are part of ongoing research.

7.2 Analysis of time-resolved *in situ* Datasets using *nNPipe*

In this section, *nNPipe* is applied to the analysis of *in situ* heating data of the morphologically diverse heterogeneous catalyst Pd/C. It is demonstrated how retraining strategies can be leveraged to adapt the analytical capabilities of *nNPipe* to new imaging conditions which can be useful in both, live imaging and post-processing analytical scenarios. Pd/C is investigated at four independent positions in a temperature range between 25 °C and 800 °C and the observed sample behaviour is discussed. Lastly, time-resolved *in situ* datasets are analysed using *nNPipe*, highlighting the method's rapid and precise access to material dynamics. Note that the application of *nNPipe* on *in situ* data forms the core of this chapter with extensive analysis of observed phenomena reserved for future work.

7.2.1 Methods

7.2.1.1 Data Acquisition

The sample preparation method, as well as instrument and detector settings used are described in Section 4.1.2. Herein, focus is put on an *in situ* HRTEM imaging strategy across selected temperature and sample positions.

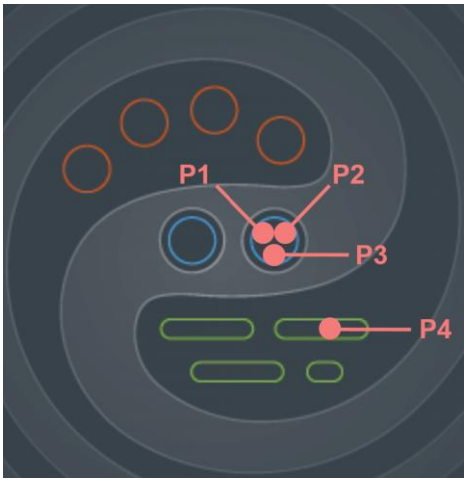


Figure 7.4: Schematic of a DENS Solution wildfire chip with representative sample positions used in in the in situ experiment marked. Adapted from [436]. Note that marked positions do not precisely show actual positions used.

At the beginning of the *in situ* experiment, four spatially separated and independent sample positions were selected. Positions P1-P3 were found on the same SiN_x window with P1 and P2 located on the same, sizeable piece of amorphous carbon and P3 was fully detached. Position P4 was set on an entirely separate SiN_x window. Figure 7.4 shows a schematic overview of the DENS Solutions wildfire heating chip with the distribution of sample positions marked. Those four positions were selected for different purposes: On P1, movies of $45 - 70$ s (100 frames/s) were acquired directly after each heating step and as soon as suitable imaging conditions were re-established after adjusting the sample drift and focal shift. Secondly, HRTEM images were acquired at varying magnifications at each temperature step on P1-P3 to ensure similar sample behaviour is observed at independent sample positions. HRTEM images of P4 were only

acquired at the beginning of the experiment and after application of the entire *in situ* heating procedure. Position P4 was hence considered as an 'absolute reference' as beam exposure was significantly less compared to P1-P3. Furthermore, the size of the amorphous carbon was small and consequently offered an overview of an enclosed system.

The *in situ* heating procedure used was as follows: 25 °C – 300 °C – 400 °C – 500 °C – 600 °C – 700 °C – 800 °C, with heating ramp time of 30 s per step. Dynamics and processes were observed to proceed comparably slowly within the temperature range of 25 °C – 300 °C. Since the *nNPipe* analysis on datasets exhibiting fast(er) dynamic processes was a core aim of this research this temperature range was omitted. Depending on how quickly defined sample positions were located and favourable image conditions were established, HRTEM imaging at varying magnifications between positions P1-P3 took between 40 – 60 minutes.

7.2.1.2 Datasets

A total of three datasets were considered for automated analysis using *nNPipe*:

i) Individual 2048 x 2048 pixel HRTEM images acquired at a magnification of 300,000X from P1-P3. These images were partly included in the retraining dataset of *nNPipe* and were representative of observed changes of catalyst particles across different temperatures.

ii) Manually aligned 2048 x 2048 pixel HRTEM images as described in i). Although individual images were acquired to capture the same field of view within P1-P3, slight deviations may induce differences in nanoparticle measurements, thus supporting the inclusion of this dataset. Manual alignment was conducted using the TrakEM2 plugin of the FIJI software using both, overlapping red/green channels or semi-transparent frames with respect to images taken at lower temperatures respectively. Figure 7.5 illustrates a direct comparison of the datasets described in i) and ii).

iii) A 5 *s* 2048 x 2048 pixel HRTEM movie acquired at 300,000X and 100 frames/s on sample position P1 to test and demonstrate the analytical capabilities of *nNPipe* on time-resolved datasets.

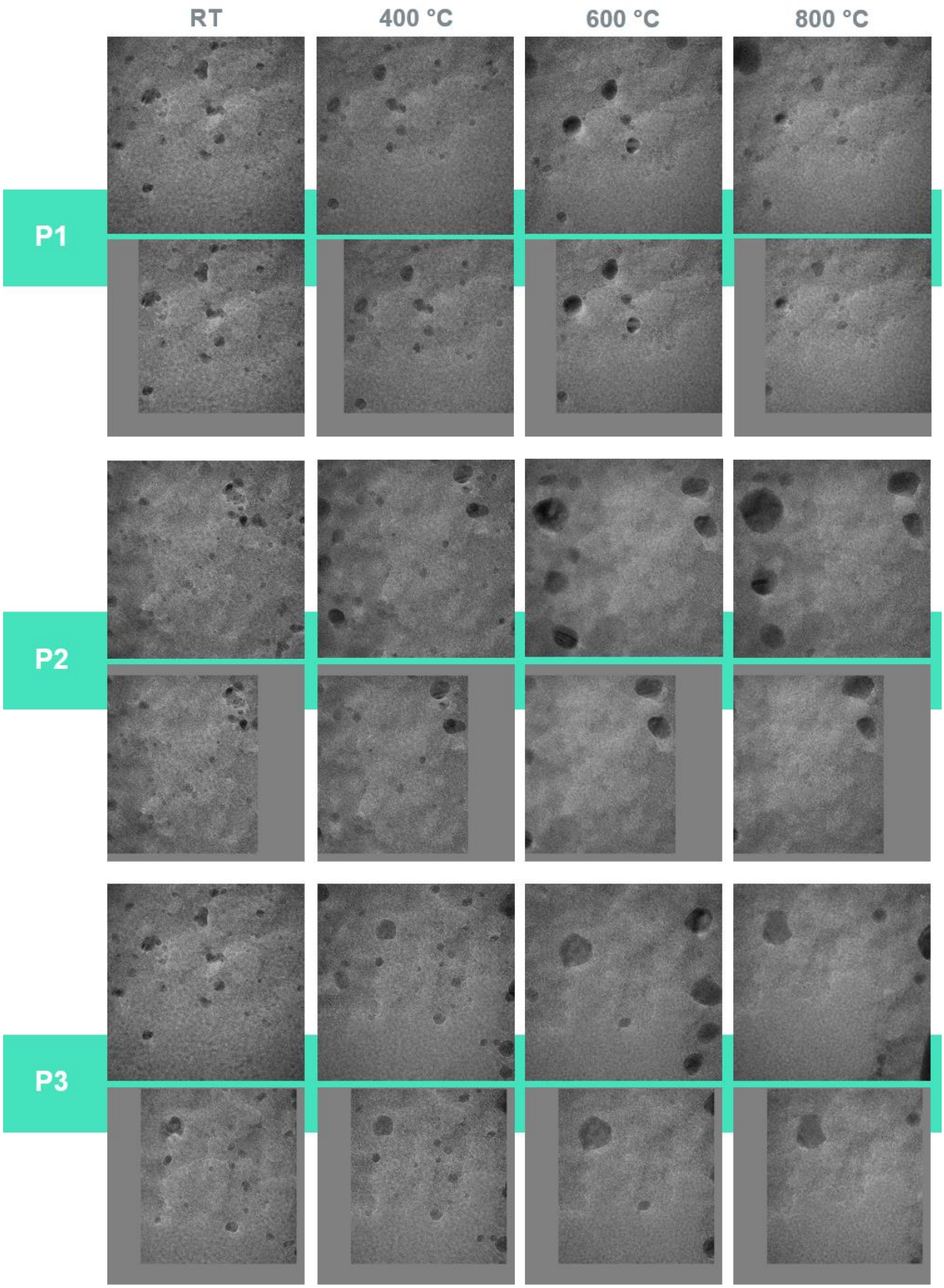


Figure 7.5: Example HRTEM images acquired at different positions (P1-P3) and temperatures of the Pd/C sample during *in situ* heating at a magnification of 300,000X. Top rows show raw acquisitions while bottom rows show aligned images for each position respectively. After manual alignment, the same field of view was located by substituting all image pixels that were not part of the unified field of view across different temperatures with the grayscale value 128. Image scales were not added on purpose for improved sample comparability (similar images with added scale are in 4.1.2). RT refers to room temperature.

7.2.1.3 nNPipe Retraining

Successful retraining strategies as discussed in Section 6.1 of this work were the basis for adapting *nNPipe* for *in situ* analysis. However, instead of retraining the original simulated Pd/C system in *nNPipe* (Chapter 5), additional analytical capabilities gained after retraining on experimental HRTEM datasets (Section 6.1) were utilised.

Retraining was conducted on the *nNPipe* retrained on Pd/C HRTEM images with an ESS of 24 images and augmented to 10 % of the original training data respectively for the best F1-Score performance (compare Section 6.1). This state was chosen as it reflected a feasible retraining strategy and therefore a more realistic and reproducible starting point. Following the same rationale, a limited retraining dataset of 20 images consisting of 10 frames extracted from *in situ* movies at P1, as well as 10 HRTEM images at sample positions P1-P3 acquired at different temperatures respectively. It is evident that the majority of retraining images selected originate from sample position P1. This was empirically chosen as the majority of the data to be analysed was indeed obtained at P1 (HRTEM images and movies) whilst inclusion of images taken at different temperatures and positions P2 and P3 were intended to increase generalisation capabilities. Ground truth images for training were obtained by manual annotation following the same methods as described in 6.1. Note that ground truth images were not created for all image types analysed by *nNPipe* as this would have been too time consuming. Similarly, F1-Score performances are not provided in this context.

The retraining dataset was augmented using similar strategies as described in Section 6.1.2 and Appendix 10.6. In this case however, ranges of histogram adaptions, brightness and blurring were given smaller values to match *in situ* data and the application of dead pixels was omitted. The final retraining dataset size was set to 10 % of the original training dataset (Section 4.2.2 and Chapter 5), and individual modules were *fine-tuned* for 50 and 80 epochs at a learning rate of 10^{-4} respectively. Weights of best performing states were selected based on highest mAP@0.50 for the YOLOv5x module and based on lowest Cross-Entropy-Loss for the validation dataset for the SegNet module. Appendix 10.8 includes code for the retraining dataset augmentation. Other code used to generate this retraining dataset is similar to that in Appendix 10.6.

7.2.2 Results - *in situ* Heating

In the following section, results obtained from *nNPipe* driven analysis on aforementioned datasets i) and ii) are discussed. In contrast to previous results where emphasis was put on post-acquisition analysis, the output of both variants, *i.e.* only the YOLOv5x module and the full *nNPipe* pipeline are considered here to emphasise the analytical potential for live imaging applications (see Section 7.1).

7.2.2.1 Object Detection Output

The retrained *nNPipe* was applied to datasets i) and ii) to track sample changes with increasing temperature. Figure 7.6 shows the output of the YOLOv5x model, as might be used during live image analysis (see 7.1).

The results unambiguously demonstrate an overall high inference performance of the YOLOv5x object detection module after retraining. This is particularly true for the raw images shown in the top rows of each sample position, where single nanoparticle or cluster instances are missed only occasionally as seen for P2 – 400 °C or P3 – 800 °C for example. A slightly higher number of false classifications is observed for the aligned data. For instance, P2 – RT contains a cluster bounding box whose borders are incorrectly inferred which also holds true a nanoparticle shown on the top right across different temperatures. Three falsely inferred clusters can be observed for P3-RT. Nonetheless, the large majority of catalyst particles are detected correctly, despite the fact that aligned images were not included in the retraining dataset. Exclusion of training images containing plain grey areas can be assumed as source for these few false object detections in the current context and suggest that such images need to be taken into account for training in the future.

An important observation as shown in Figure 7.5 is confirmed quantitatively by the results in Figure 7.6: Small nanoparticle instances decrease in number with increasing temperature and time held at high temperatures. Instances that were classified as clusters at room temperature appear to transform into large nanoparticles which subsequently grow with increasing temperature as indicated by the different colours for the classes of ‘nanoparticle’ and ‘cluster’ in respective datasets. Consequently, the width w and height h of the inferred *in situ* images were used to calculate bounding box areas and throughout the applied heating procedure.

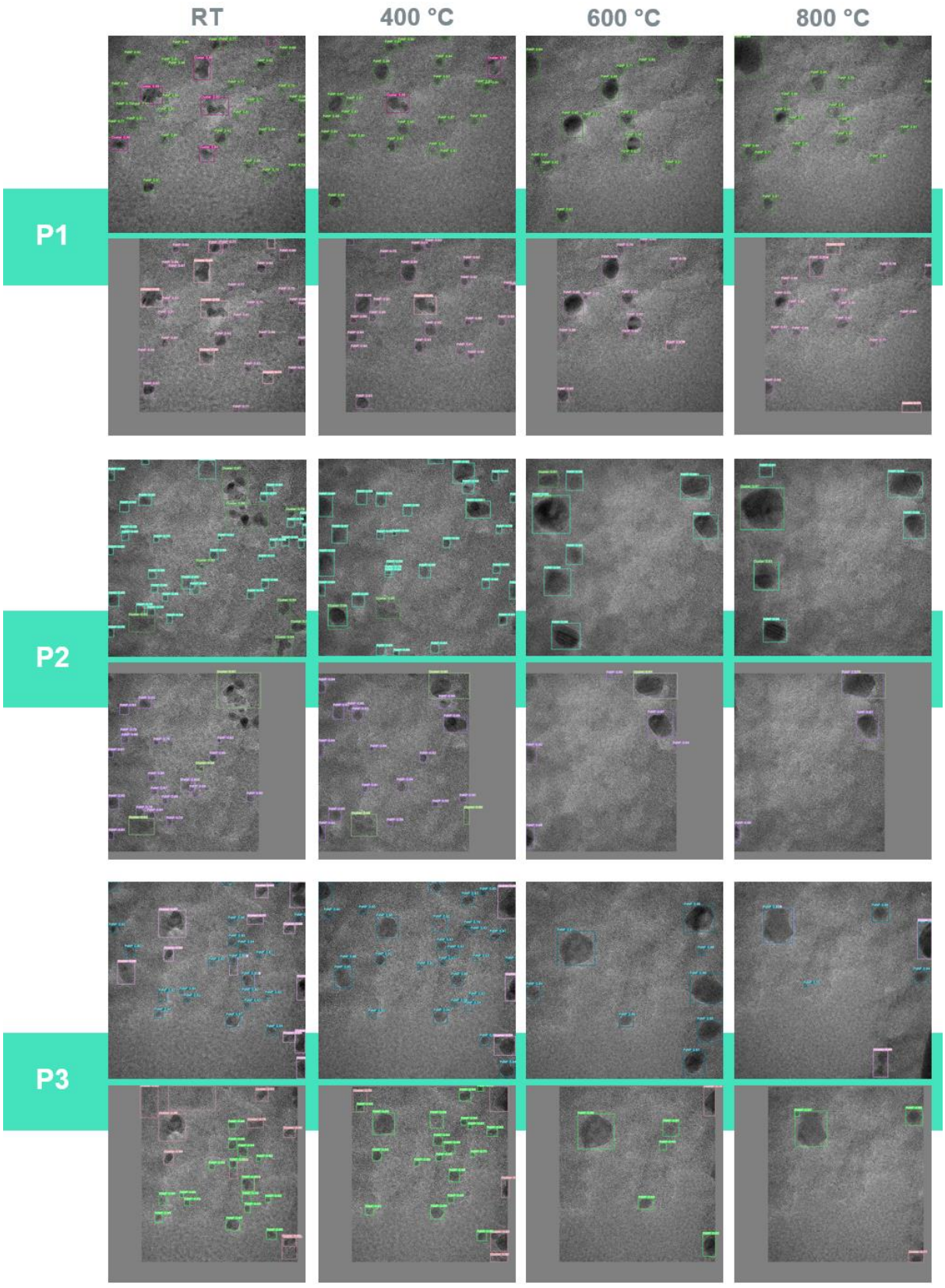


Figure 7.6: Example HRTEM images shown in Figure 7.5 after inference of the YOLOv5x module in *nNPipe*. Top rows show inference for raw images while bottom rows show aligned images for each sample position respectively. Bounding box colours were assigned randomly by the YOLOv5x module and only serve to distinguish nanoparticles and clusters in different datasets and positions respectively. RT refers to room temperature.

Figure 7.7 summarises nanoparticle and cluster counts per image, as well as plots of corresponding mean bounding box area for the considered heating range.

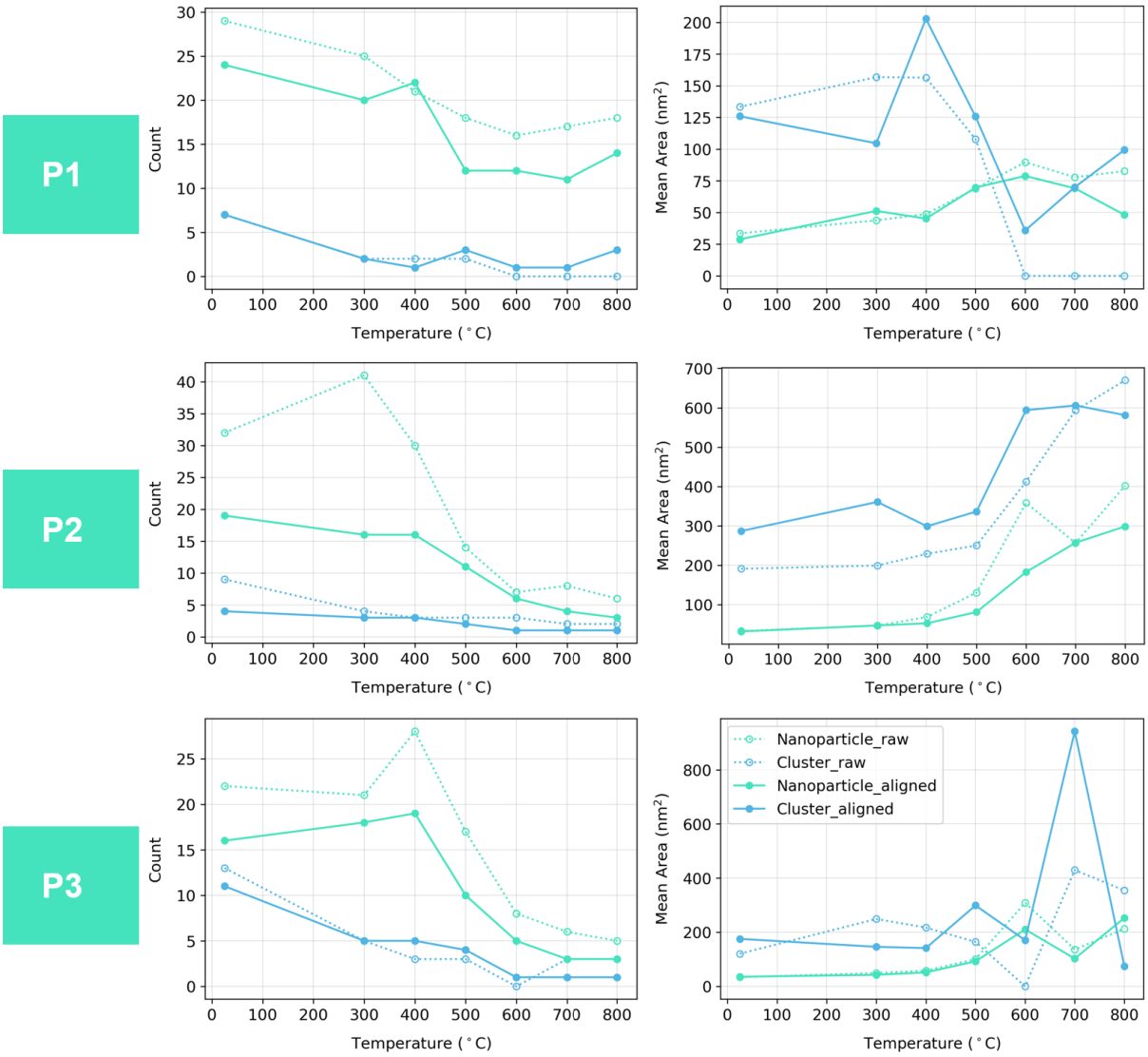


Figure 7.7: Point plots displaying the number of nanoparticles and clusters detected at a given sample position and temperature (left column) together with corresponding mean nanoparticle area (right column) calculated for raw and aligned datasets respectively. The legend given in the bottom right plot applies to all plots displayed.

Considering nanoparticle instances in the plots shown in Figure 7.7 a clear correlation can be inferred. Increasing temperature and progressing time reduces the number of nanoparticles while the mean nanoparticle area increases at all sample positions. This appears to occur in different phases: Between room temperature and 300 °C, the nanoparticle count decreases and subsequently increases to up to 400 °C for two of three sample positions respectively. Within this range, the mean nanoparticle area increases constantly. Examining Figure 7.5 and 7.6, this can be understood with the vanishing of smallest nanoparticles and clusters transforming into large nanoparticles. The latter is confirmed by

considering the decreasing cluster count within the temperature range of room temperature - 400 °C (Figure 7.7). When temperatures above 400 °C are used, a significantly steeper growth in the nanoparticle area is observed, in parallel with a significant decrease in nanoparticle counts for all three cases. Whilst the mean nanoparticle area steadily increases up to 800 °C at P2, a decrease is observed at P1 and P3 from 600 °C – 700 °C. Finally, the cluster count appears to either decrease further or remains at a relatively low level for temperatures above 400 °C. A more mixed picture is observed for the mean cluster area, from which no clear trend can be deduced. Within the temperature range above 400 °C mean cluster areas increase in two of three cases. Potentially, this could indicate a dominating growth in the $x - y$ plane following the flake-like geometry of the amorphous carbon support, as their count does not change significantly. On the other hand, it should be noted that the number of inferred clusters is lowest as soon as 600 °C is reached (Figure 7.6), whilst the ones referred are mostly found at image edges, where YOLOv5x inference errors may become more likely for aligned images (see above). Hence, the full *nNPipe* output is needs to be considered.

7.2.2.2 Full nNPipe Output

In similar fashion to previously discussed results, the full *nNPipe* output considers nanoparticle instances only. Figure 7.8 shows the segmented 2048 x 2048 pixel output of *nNPipe*, in accordance with panels in Figure 7.5 and 7.6. Note that an additional 'FillingTransform' step was added after step i) in the post-processing routine for improved performance (Table 10.2). Furthermore, a limitation of the *nNPipe* method became apparent during analysis. Several nanoparticles identified at temperatures of 600 °C and higher exceeded 256 x 256 pixels in size, which is the image array size required by the SegNet module. To infer affected particles at a set magnification of 300,000X correctly, an optimised post-processing routine is necessary. Alternatively, HRTEM images acquired at lower magnifications coiuld be considered, but this would necessitate retraining with a training dataset generated from a large ESS to yield reliable results from automated analysis. At the time of writing, a suitable strategy is part of ongoing research.

Notably, the results shown in Figure 7.8 underline successful retraining of both *nNPipe* modules as individual nanoparticles are accurately segmented and support findings discussed for the YOLOv5x

output, despite limitations. As seen for position P1 in particular, the number of nanoparticles decreases with increasing temperature and time, while the area of individual nanoparticles increases.

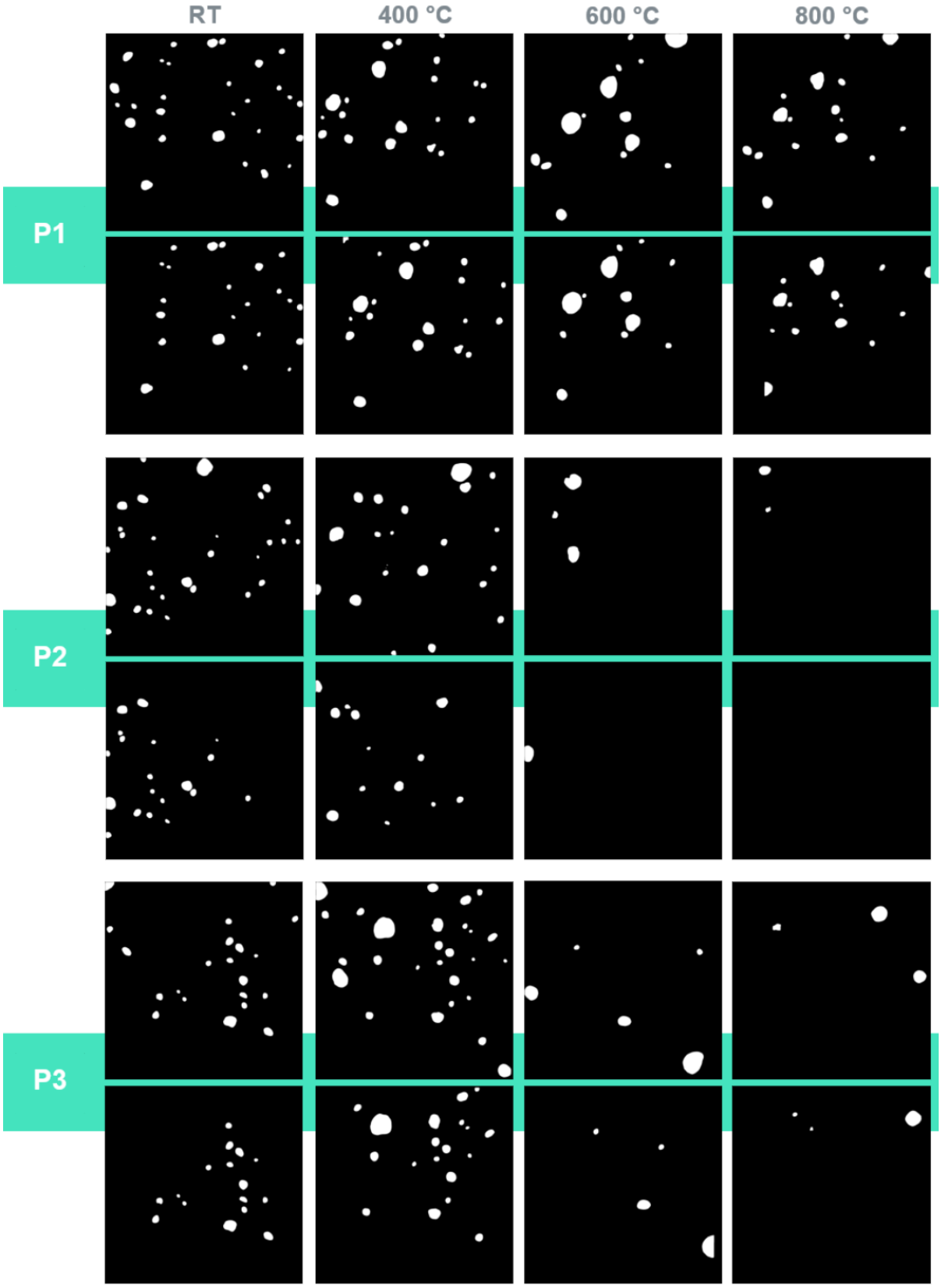


Figure 7.8: Example binary maps of 2048 x 2048 pixels generated using the *nNPipe* method. As with Figure 7.5 and 7.6, the top rows show the output of raw experimental images while the bottom rows show output generated from aligned HRTEM images respectively. RT refers to room temperature.

Similar sample transformations were observed at positions P2 and P3 in Figures 7.5 and 7.6 but were not replicated in the SegNet module as a number of nanoparticles exceeded the maximum area of 256 x 256 pixels.

The ParticleSpy [416] package was used to measure mean nanoparticle areas using the 'particle_analysis' function. Figure 7.9 summarises point plots displaying nanoparticle count and corresponding mean nanoparticle area for all three sample positions and across the applied heating temperatures. In this case, respective results were summarised within one plot for more direct comparison.

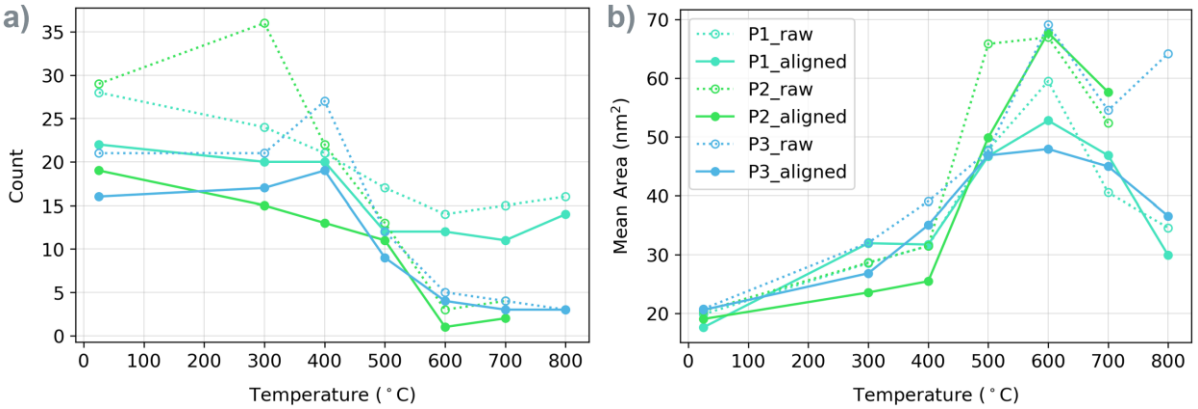


Figure 7.9: Point plots showing metrics determined using ParticleSpy for all three sample positions P1-P3. **a)** Nanoparticle count plotted against applied heating temperatures. **b)** Mean nanoparticle area (nm^2) plotted against applied heating temperature. Dotted lines refer to metrics extracted from raw experimental *nNPipe* input images, whilst full lines refer to aligned input images. The legend shown on the top left of **b)** holds true for both graphs.

Consistent with the *nNPipe* results in Figure 7.7, congruent trends can be deduced from the results in Figure 7.9. For all three sample positions the mean nanoparticle area increases continuously between room temperature and 400 °C, while the nanoparticle count decreases, except at P3. The mean nanoparticle area growth at P1 and P3 is very similar, whilst the slope at P2 appears less steep. This may be potentially linked with the sample state and morphology at selected positions, because both P1 and P3 contain edge areas of the sample support. When temperatures higher than 400 °C are applied, growth rates of mean nanoparticle areas (7.9 b) increase significantly and nanoparticle counts decrease rapidly, with maxima and minima reached at 600 °C. This result supports the findings deduced from Figure 7.7, but have to be treated with caution due to the limitations of the SegNet module discussed above. However, given that the larger nanoparticles do not contribute to the statistics presented in Figure 7.9 especially at sample positions P2 and P3, an even steeper growth of the mean nanoparticle area

might be expected (also compare Figures 7.5 and 7.6). With exception of P1, a decreasing mean nanoparticle area for temperatures higher than 600 °C is therefore understood as arising from limitations in analytical performance.

Semantically segmented nanoparticles detected at position P1 mostly remained within the 256 x 256 pixel analytical range limit of the SegNet module and hence the most consistent output was obtained (Figure 7.9 – P1). Additional insight into growth kinetics can be obtained by considering histograms of nanoparticle areas determined at applied temperatures at sample positions P1, as shown in Figure 7.10.

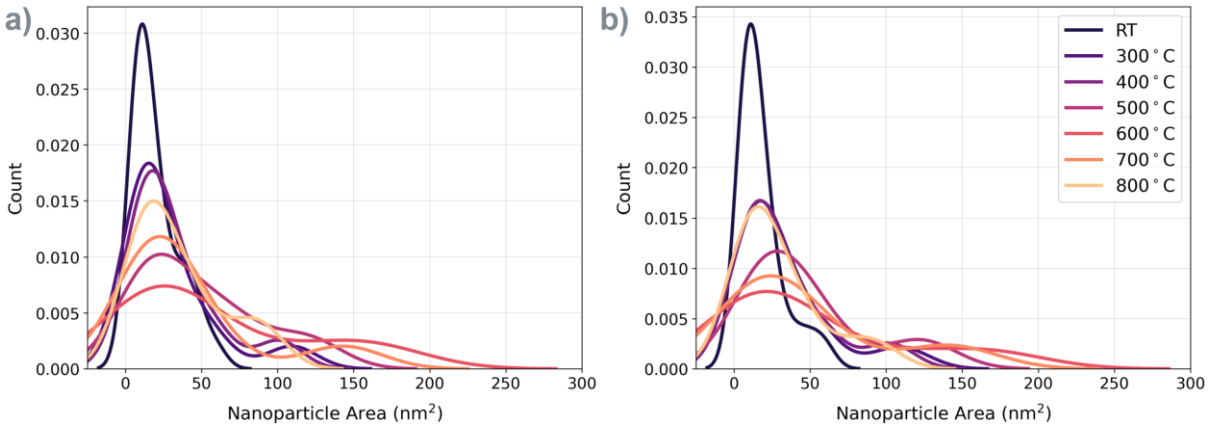


Figure 7.10: Histograms of nanoparticle area distributions at positions P1 and for all applied heating temperatures shown as KDE-plots. All KDE-plots were generated using a Gaussian kernel and Scott's rule [418]. **a)** Histograms obtained from raw experimental image input to nNPipe. **b)** Histograms obtained from aligned image input to nNPipe. The legend shown on the upper right hand side of **b)** holds true for both graphs shown and the lighter red colour is associated with higher temperatures.

While a sharper distribution can be stated for the initial nanoparticle area distribution at room temperature, broadening towards larger nanoparticle areas is observed for increasing temperatures to up to 600 °C. A very distinct trend is identified when considering Figure 7.10b in particular: The broadening of the nanoparticle area distribution reveals a bimodal distribution with increasing temperature where the first peak remains within an interval of *ca.* 10 – 35 nm^2 as its peak height decreases, while a second shoulder emerges at *ca.* 100 – 150 nm^2 and slowly increases its peak height. In combination with observations and results provided in Figure 7.6 – 7.9, this result quantitatively shows *the growth of individual nanoparticles at the expense of smaller ones*, *i.e.* Ostwald ripening (see Section 2.3.4). Compared to other studies (see Section 2.1.3.1 and Section 2.3.4), the Ostwald ripening process onset temperature is relatively moderate, but can be clearly distinguished at as early as 300 °C. Given that observations of P2 and P3 reveal a highly comparable sample behaviour, it will be of crucial importance to extend the analytical capabilities of *nNPipe* to compare similar nanoparticle area

distributions to obtain a full understanding of the material physics revealed by this data. At temperatures higher than 600 °C, distributions appear narrow up again. This reflects the results shown Figure 7.8 and 7.9, where the nanoparticle area increased with decreasing nanoparticle count to up to 600 °C after which a dip was observed. As the dips seen for positions P2 and P3 were associated with limits in the analytical capabilities of *nNPipe*, the decrease of nanoparticle area at P1 might be interpreted as an exception. The newly identified, smaller nanoparticles (Figure 7.8 – P1 – 800 °C) that emerge at temperatures higher than 600 °C may be either linked to a specific physical phenomenon or to additional electron beam – sample interaction as the beam exposure is longest at P1. However, these conclusions should be regarded as preliminary until optimised analytical performance or confirmation through additional experimental methods is available.

7.2.3 Results - Analysis of time-resolved *in situ* Datasets

In this section, the application of *nNPipe* is to an example dataset acquired during *in situ* heating of the morphologically diverse Pd/C heterogeneous catalyst. By analogy with Section 7.2.2, emphasis is put on the analysis method itself rather than on the determination of physical quantities. These are part of ongoing research which will be considered in the future.

Results obtained from the *in situ* datasets suggest a steep increase of mean nanoparticle area at 400 °C and higher. Consequently, an *in situ* movie acquired at position P1 at 400 °C was analysed using *nNPipe*. The full duration of the movie was 46.206 s and the first 5 s (507 frames) were saved as individual 2048 x 2048 pixels HRTEM images as exemplars. Figure 7.11 shows four exemplary and randomly selected frames alongside respective YOLOv5x and full *nNPipe* outputs.

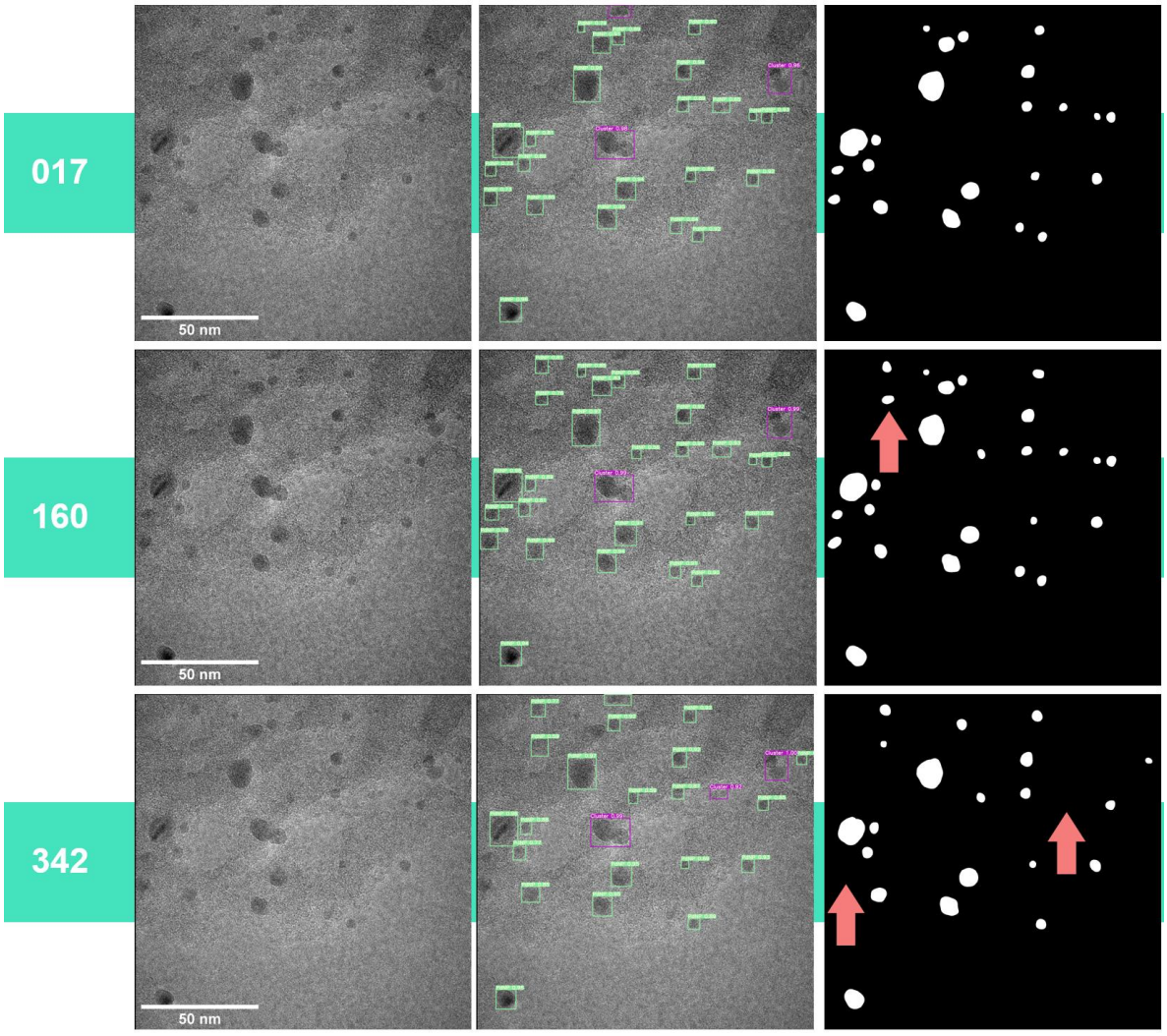


Figure 7.11: Example *in situ* HRTEM images at sample position P1 acquired at 400 °C (left column), alongside YOLOv5x output (middle column) and full *nNPipe* inference shown as binary maps (right column). Numbers given on the left hand side indicate the frame number and the scale bars shown in *in situ* images (right column) hold true for all images. Red arrows were added in the right column to indicate positions where inference was less consistent in the dataset.

A key achievement is immediately evident. Overall, *nNPipe* infers *in situ* movies consistently and accurately after being retrained on a dataset obtained from an ESS of only 20 images. This holds true for the entire 507 frames of the considered dataset. Closer inspection reveals few nanoparticles that are inferred with less consistency and examples are indicated by red arrows indicating missed nanoparticles in the binary maps in Figure 7.11. Although brightness and contrast changes have been taken into account during retraining, it appears that the YOLOv5x module misses nanoparticles in some cases, which are subsequently not segmented. An optimised analysis strategy, *i.e.* adaption of the object detection confidence threshold and post-processing routine are likely to improve generated output further. However, these developments must be approached with some caution as the sample transformations in the *in situ* heated sample may be genuinely responsible for changes in the output. For example, nanoparticles have been observed to change contrast rapidly, suggesting rotation or movement. However, if analysis were fully optimised for *in situ* imaging, differences in the output across frames would serve as indicator for sample changes and dynamics.

Nonetheless, the current *nNPipe* status enables rapid analysis of key material properties in time-resolved datasets. This is ultimately demonstrated by Figure 7.12 that shows scatter plots of nanoparticle counts and mean nanoparticle area measured for each individual frame using the ParticleSpy [416] package.

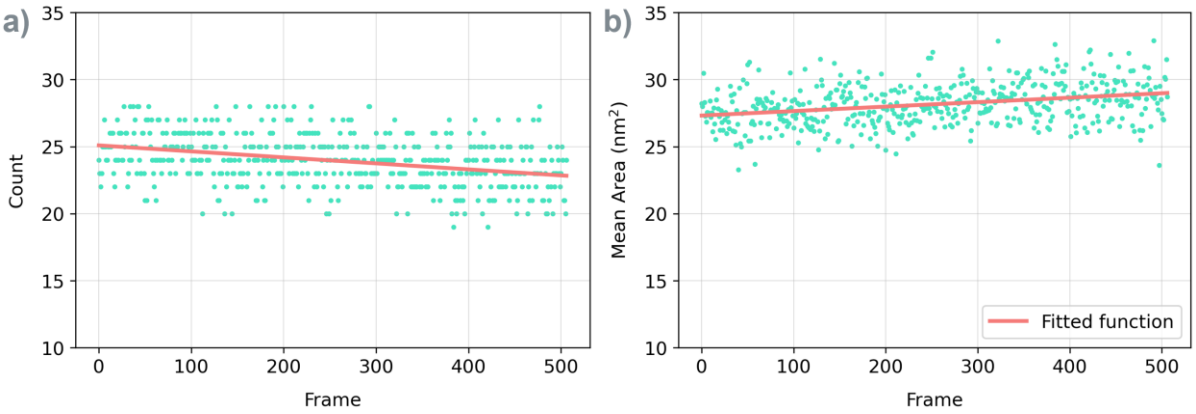


Figure 7.12: Scatter plots of measured material properties from individual frames of an *in* situ heating movie **a)** Nanoparticle count with respect to frame, alongside a linearly fitted function to indicate a decrease over time. **b)** Mean nanoparticle area per frame with a linearly fitted function to highlight an increase over time. The legend in **b)** holds true for both plots shown.

Scatter plots in Figure 7.12 show results that can vary across consecutive frames, due to varying inference on individual nanoparticles (see above). However those variations are limited and are

acceptable, especially when reconsidering the comparably small ESS from which the retraining subset was generated. Regardless of a varying output, trends similar to the ones discussed in 7.2.2 are observed; a decreasing number of infererred nanoparticles accompanied by an increasing mean nanoparticle area over time at 400 °C in the field of view (P1), *i.e.* Ostwald ripening. In contrast to previous analyses, those results were calculated on datasets with a time-resolution of 100 frames/s, giving access to the analysis of temperature-time dependent phenomena in heterogeneous catalyst materials. The total inference time of *nNPipe* to analyse these 507 images was *ca.* 1382.5 s (82.62 s YOLOv5x, 1299.9 s SegNet with post-processing) or *ca.* 23 minutes, which approximates the time required to manually segment a single 2048 x 2048 pixel *in situ* HRTEM image.

In a first-order analysis, the observed decrease of inferred nanoparticles and increasing mean nanoparticle area were assumed to be linear and consequently fitted using the optimise python-package of SciPy. The nanoparticle count was found to decrease at a rate of *ca.* $-4.479 \cdot 10^{-4}$ counts/ms or 1 count per 2.233 s whilst the mean nanoparticle area was found to increase at a rate of $3.325 \cdot 10^{-4}$ nm^2/ms. Although it has to be noted that these results reflect only a fraction the total *in situ* data acquired, it does allow for a deduction of nanoparticle growth rates related to Ostwald ripening at high time-resolution.

7.2.4 Discussion

The results presented substantiate the application potential of *nNPipe*. It was shown that elaborated retraining strategies (Chapter 6) allow for the transfer of *nNPipe*'s inference performance to experimental HRTEM datasets that are substantially different from the original training data. Given that an experimental subset of only 20 images was sufficient for a successful analysis of *in situ* heating datasets, it can be expected similar strategies can be applied for other *in situ* methods, such as gas-cell experiments.

Moreover, the generated output can be used in two-fold fashion, whereby the output generated by the YOLOv5x module is attractive for live imaging applications as discussed in 7.1. Although bounding boxes of inferred instances only approximate the morphology of nanoparticles and clusters, it was unambiguously shown that their number decreases while the mean nanoparticle area increases simultaneously at a given position. However, a few false detections had to be taken into account for aligned images, indicating a possible limitation of the object detection module. Parallels can be drawn comparing Figure 5.2 and Figure 6.6, where *nNPipe* erroneously inferred an image part showing lacy carbon as 'cluster' in its original training state, but generated significantly improved output once retrained. Hence, inclusion of adequate images in the retraining dataset is necessary for analysis of aligned datasets.

Binary maps generated by the full *nNPipe* method allowed for deduction of consistent trends in terms of nanoparticle number reduction and mean area growth over temperature and time, particularly for images acquired at sample position P1. In contrast to sample positions P2 and P3, new small nanoparticles appeared to emerge at 600 °C and higher, for which further investigation is required. For HRTEM images acquired at higher temperatures an important limitation became apparent; instances that exceed the 256 x 256 pixel limit cannot be analysed by *nNPipe* directly. It is conceivable that an additional processing routine could be applied for similar cases which could include inference in patches or of down sampled images. Alternatively, retraining *nNPipe* for the inference of HRTEM images acquired at lower magnification may serve as a solution, but is likely to require a larger experimental subset of images for a range of magnifications.

The observed sample behaviour was finally also quantified as shown in Figure 7.10, clearly indicating a broadening particle size distribution with increasing temperature alongside a growth of large nanoparticles on the expense of small ones, *i.e.* Ostwald ripening (Section 2.3.4). Although this study mainly focuses on the use of *nNPipe* for *in situ* datasets, these findings are of key importance for the examined system. From an onset as early as 300 °C, the system appears to undergo Ostwald ripening, effectively decreasing the surface-to-volume ratio due to increasing mean nanoparticle size and thus decreasing the number of active sites for heterogeneous catalysis. Furthermore, this process is accelerated from 400 °C onwards, for which a dedicated movie analysis was conducted (see below).

Within a temperature range from room temperature – 600 °C nanoparticle growth occurs in two phases and proportionally there is a decrease in the nanoparticle number at a given position. Subsequently, it was only at positions P2 and P3 where growth of nanoparticles appeared to continue. Similar sample transformations can be seen for sample position P4 shown in Figure 7.13, for which images were only acquired at the beginning and end of the heating procedure.

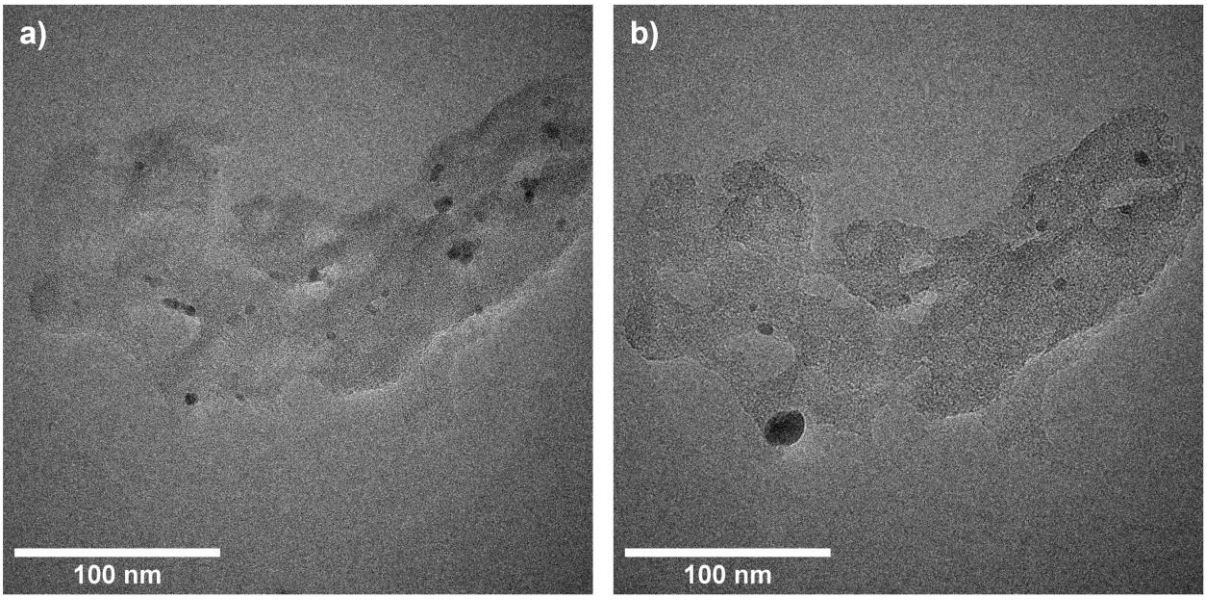


Figure 7.13: HRTEM images of in situ heated Pd/C catalyst at sample position P4, both at 150,000X magnification. **a)** Image acquired at room temperature at the beginning of the experiment. **b)** Image acquired at 800 °C after completion of the heating procedure. Note that the amorphous carbon support was fully isolated from the other Pd/C sample.

Although sample positions P1-P3 can be considered as fully independent, position P4 acts as proof for observed sample behaviour as it was exposed to the lowest electron dose, whilst also being a fully

isolated system. Additional analysis of images acquired at lower magnifications for P1-P3 completes the picture (Figure 7.14).

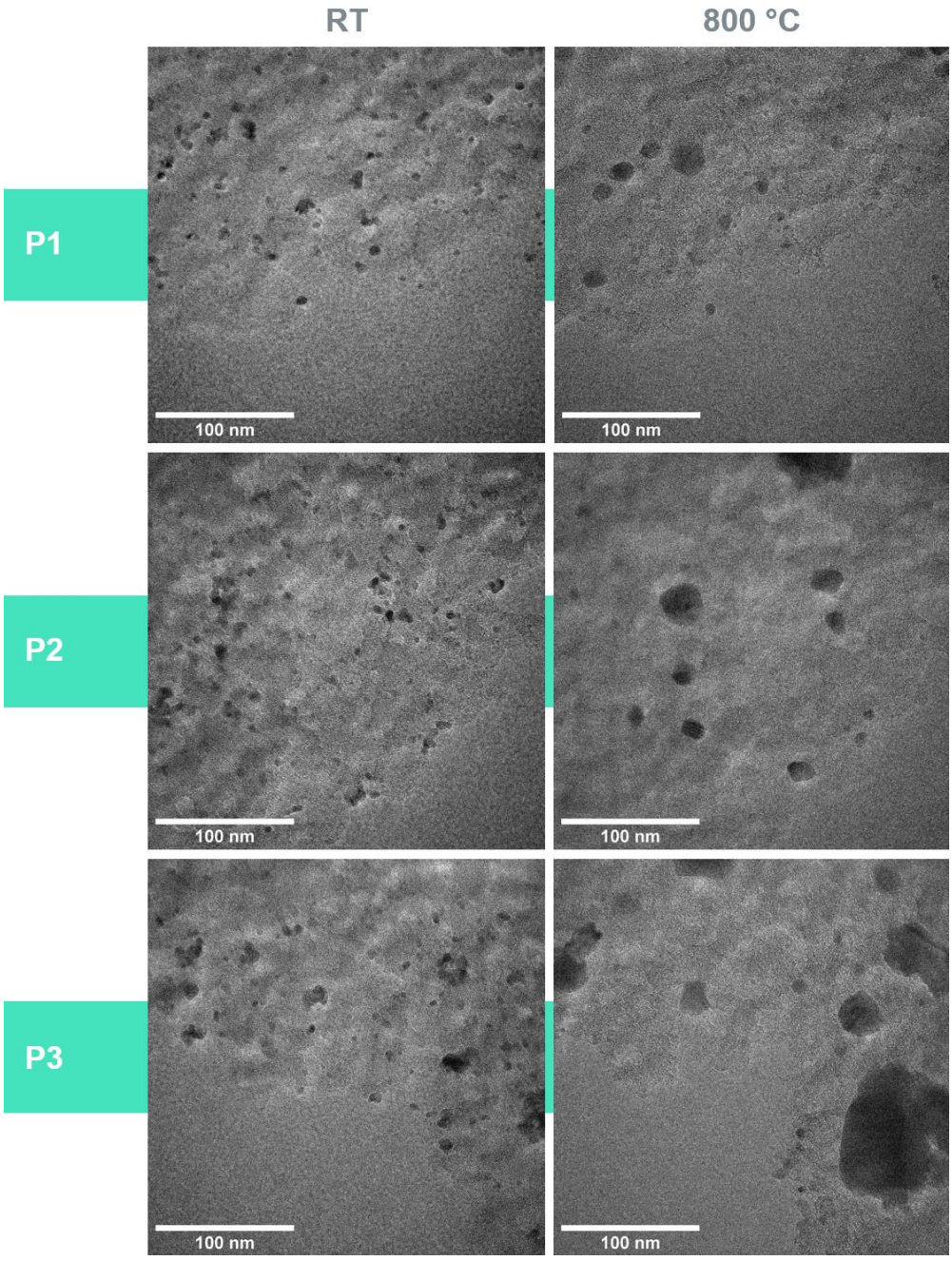


Figure 7.14: HRTEM images acquired at sample positions P1 – P3 at room temperature (left column) and 800 °C (right column) at 150,000X magnification.

The coarsening of nanoparticles in heterogeneous catalyst systems is actively researched (see Section 2.3.4). While the presented analysis suggests the growth mechanism of Ostwald ripening, particle migration might pose an alternative option. However, this was not observed in the experiments reported and is therefore not considered in this discussion.

A key enabler for tracking nanoparticle growth at high time-resolution has been demonstrated for the analysis of a movie acquired at sample position P1 at 400 °C. *nNPipe* generates accurate results across the whole time-resolved dataset considered, albeit with a few nanoparticles that were not resolved consistently, suggesting further optimisation as discussed.

Irrespective of this limitation, it was possible to extract nanoparticle growth at $3.325 \cdot 10^{-4}$ nm^2/ms for this temperature. It is this analytical capability of *nNPipe* in particular, which opens up pathways for improved understanding of material reactions and processes at higher time-resolutions.

7.3 Chapter Summary

In this chapter, applications of *nNPipe* for time-resolved imaging have been discussed. In Section 7.1, a live imaging setup using the analytical capabilities of the YOLOv5x module of *nNPipe* was introduced. Its functionality has been demonstrated in experiments at ePSIC, Diamond Light Source and showed direct access to first-order quantitative information of a heterogeneous catalyst material *during* sample viewing. Compared to other DL-based applications that are often used post-acquisition, the live imaging setup has the potential for a paradigm change in the use of DL methods in EM.

Subsequently, elaborated retraining methods have been used to retrain *nNPipe* for the analysis of an *in situ* training dataset. A small experimental subset size was sufficient to transfer inference capabilities to an entirely new imaging scenario, ensuring high practicality. First-order results demonstrated nanoparticle growth in the Pd/C system with as a function of heating temperature and time for four independent sample positions and Ostwald ripening was confirmed. Whilst further investigation of the data and method optimisation is required, the analysis reported clearly demonstrated the potential of *nNPipe's* analytical capabilities. Lastly, an *in situ* movie acquired at 400 °C at 100 frames/s was automatically inferred using *nNPipe* revealing insights into nanoparticle growth of heterogeneous catalyst materials at high time-resolution.

8 Conclusion and Outlook

The growing need for automated and reliable analysis methods driven by an increasing amount of data generated through developments in detector technologies and new imaging methods is an area of active research in EM. In this work, a DL-based method termed '*nNPipe'* has been introduced for heterogeneous catalyst systems.

The *nNPipe* method uses two independent CNNs, namely YOLOv5x [391] and SegNet [199] to perform semantic segmentation on 2048 x 2048 pixel HRTEM images. In its original form, both neural networks were trained on computationally generated image simulations exclusively. The method was subsequently tested in a realistic automated imaging scenario to extract statistically significant material properties of the heterogeneous catalysts Au/Ge and Pd/C. Results showed that it is possible to train neural networks to high analytical precision using computationally generated training data, effectively enabling the generation of virtually unlimited training images. Simultaneously, the method limitations have been described particularly for cases where individual nanoparticles and particle clusters exhibit low contrast with respect to the material support, and a filtering strategy based on an Instance Difference (ID) criterion was introduced to yield accurate results in automated imaging.

Performance improvements have been thoroughly investigated by considering different retraining strategies in *transfer learning* and *fine-tuning* with emphasis on highest achievable performance, while keeping the retraining approach practicable. In this case, differently sized subsets of an original experimental dataset were considered for retraining and augmented to varying degrees to track the influence of performance improvements. *Fine-tuning* was shown to be more advantageous than *transfer learning*, which was rationalised by networks improving their encoding of image features. Notably, results show that any of the retraining datasets led to improved analytical capabilities. For heterogeneous catalyst systems with sample morphologies ideal for HRTEM imaging, an experimental subset size of only 10 % is sufficient to achieve similar analytical performance as for retraining on 100 % of the experimental images. In the case of morphologically diverse systems, experimental subset

sizes of at least 25 % are required for comparable performance to the ideal system, which was explained by a minimal amount of new image features required during retraining.

In conjunction with the results reported in Chapter 5, the (retrained) *nNPipe* approach addresses an important bottleneck in catalyst optimisation: Statistically significant metrics are obtained in fast and reliable manner, whilst the approach can be adapted to a variety of metal nanoparticle systems relevant in both research and industrial contexts. This may ultimately enable *routine*, (S)TEM-driven nanoparticle optimisation.

Contrasting to real-world image applications, the lack of annotated training data is a fundamental challenge for DL-based methods. For many common approaches, domain experts annotate training data manually, but may introduce inherent bias. This can be omitted by using computationally generated datasets which are suitable for achieving high analytical performance, but ultimately, they only approximate experimental images and hence may not cover all possible scenarios.

A third approach, part of an ongoing Zooniverse study, was explored as a potential alternative. Over 1600 volunteers with different occupations, ages and gender were asked to annotate cropped HRTEM images of heterogeneous catalysts in a simplified workflow after being introduced to a short tutorial. Pre-liminary results showed precise markings of nanoparticle centroids and correct classifications from volunteer annotations. Next steps may include a combination with other image processing techniques to establish the first steps of a pathway towards a new strategy for creating (re)training datasets.

In the last part of this work, advanced applications based on the *nNPipe* method were discussed. The first of two applications highlighted a live imaging setup, deployable on any (S)TEM instrument operating the GMS 3 [430] software. In contrast to the more common approach of analysing data post-acquisition, this live imaging setup represents a possible paradigm change. It enables researchers to have access to first-order quantitative information and corresponding statistics in real time during sample viewing. In the second application, developed retraining methods were utilised to retrain the *nNPipe* modules for accurate inference of time-resolved *in situ* datasets. Sample behaviour of the Pd/C catalyst system was investigated using four independent sample positions across a temperature range

from 25 °C to 800 °C and Ostwald ripening could be confirmed. A 500 frame *in situ* movie acquired at 400 °C and at 100 frames/s was inferred using *nNPipe* in 23 minutes; a duration approximating the manual segmentation of a single frame. These results allowed for the extraction of a mean nanoparticle growth rate of $3.325 \cdot 10^{-4}\,\mathrm{nm}^2/\mathrm{ms}$. Although specific limitations of the applied methods are acknowledged, this example shows analytical insight into the kinetics of heterogeneous catalyst materials at elevated temperatures and high temporal resolution.

Introduced advanced applications underline the relevance of this work to the field of heterogeneous catalysts. The live imaging setup can be envisioned to be fully integrated in (S)TEM instrument operation software - not only enabling immediate feedback on sample properties, but also as a first building block to a broad live-imaging toolkit. Large *in situ* datasets can be subsequently analysed using a retrained *nNPipe*, not only giving access to crucial time-resolved data for *in situ* studies, but also allowing for a *complete* picture of catalyst properties and functionality in different environments, opening up entirely new strategies for catalyst optimisation. Outside of catalysts, the *nNPipe* approach can impact entirely different material systems such as metal precipitates, virus and protein crystal research, or even biomedical imaging (see Section 2.2)

In conclusion, a novel DL-based method for the analysis of heterogeneous catalyst materials has been described together with adaption strategies for new datasets and advanced applications for time-resolved imaging. This work can be regarded as part of a wider transformation underway in (S)TEM imaging towards data-driven and an intelligent microscopy [9]. ML and DL methods will continue to be improved and will have an expanding impact in EM research [11]. Considering the various data sources and imaging modes accessible during (S)TEM operation, manifold developments can be anticipated. Higher generalisation, made possible by a growing amount of available and annotated training data and better performing neural networks, will extend automated analysis to a wider range of complex material systems imaged under challenging conditions. Improvements in inference speed and performance will allow efficient and real-time data processing, and consequently wider analytical access to chemical and physical phenomena. Once an interplay between neural network output and microscope control is established, routine alignment and calibration procedures, along with

standardised materials characterisation can be expected to not require any human intervention in the near future. For this transformation to occur, several aspects must be taken into account:

i) The growth of collaborations between computer/data scientists, ML engineers, EM researchers and companies. While EM researchers have a fundamental understanding of the challenges in (S)TEM imaging, knowledge of data-handling, best coding practices and customising neural networks for specific tasks is less widespread. Furthermore, well-performing and generalised solutions must be integrated in commonly used, user-friendly software packages, ultimately ensuring a widespread use of DL-based approaches.

ii) The creation of a materials characterisation-oriented platform to share DL-based methods and annotated datasets. As a newly emerging field, DL-based EM should rely on open-access code and datasets that set a standard of analytical capabilities. This will ensure comprehensible benchmarking and accessibility to state-of-the art methods for further developments. Despite the existence of platforms such as zenodo.org or github.com, a hybrid-model between a solution as encountered in the cryo-EM community with standardised datasets of biological specimen and zenodo.org, where contributions are published in an article-like fashion for referencing shall be suggested here.

iii) Continuous testing and critical assessment against traditional analysis methods in both, idealised and realistic application scenarios. Widespread acceptance and adoption of DL-based analytics is only realistic if their performance can be at least *at par* with established methods while faster data-processing will increase the advantage.

From the moment when DL will be fully accepted and established within the EM and closely linked research communities, a change for common experimental strategies can be expected. Researchers will experience a metaphorical removal of blinders since experiments will be less oriented towards specific phenomena, but towards the analysis of all events occurring during acquisition. Areas such as accurately processing low-dose images with poor SNR may only be realised by DL-based methods and as such might finally overcome a boundary to physical phenomena currently inaccessible by (S)TEM imaging.

A future to look forward to.

9 References

1. Armor, J.N. (2011) A history of industrial catalysis. *Catal. Today*, 163 (1), 3–9.

2. Friend, C.M., and Xu, B. (2017) Heterogeneous Catalysis: A Central Science for a Sustainable Future. *Acc. Chem. Res.*, 50 (3), 517–521.

3. McNaught, A.D., and Wilkinson, A. (2022) "catalyst." *IUPAC. Compendium of Chemical Terminology, 2nd ed. (the "Gold Book"), Online version.*

4. Schlögl, R. (2015) Heterogeneous Catalysis. *Angew. Chem. Int. Ed.*, 54 (11), 3465–3520.

5. Cesário, M.R., Barros, B., Aouad, S., Gennequin, C., Abi-Aad, E., and Macedo, D.A. (2022) Understanding heterogeneous catalysis: A brief study on performance parameters, in *Heterogeneous Catalysis - Materials and Applications* (eds.Cesario, M.R., and Macedo, D.A.), Elsevier, Radarweg 29, PO Box 211, 1000 AE Amsterdam, Netherlands, pp. 1–18.

6. Chee, S.W., Lunkenbein, T., Schlögl, R., and Cuenya, B.R. (2021) In situ and operando electron microscopy in heterogeneous catalysis—insights into multi-scale chemical dynamics. *J. Phys. Condens. Matter*, 33 (15), 153001.

7. Morishita, S., Ishikawa, R., Kohno, Y., Sawada, H., Shibata, N., and Ikuhara, Y. (2018) Attainment of 40.5 pm spatial resolution using 300 kV scanning transmission electron microscope equipped with fifth-order aberration corrector. *Microscopy*, 67 (1), 46–50.

8. Clough, R., and Kirkland, A.I. (2016) Direct Digital Electron Detectors, in *Advances in Imaging and Electron Physics*, vol. 198.

9. Spurgeon, S.R., Ophus, C., Jones, L., Petford-Long, A., Kalinin, S.V., Olszta, M.J., Dunin-Borkowski, R.E., Salmon, N., Hattar, K., Yang, W.-C.D., Sharma, R., Du, Y., Chiaramonti, A., Zheng, H., Buck, E.C., Kovarik, L., Penn, R.L., Li, D., Zhang, X., Murayama, M., and Taheri, M.L. (2021) Towards data-driven next-generation transmission electron microscopy. *Nat. Mater.*, 20 (3), 274–279.

10. LeCun, Y., Bengio, Y., and Hinton, G. (2015) Deep learning. *Nature*, 521 (7553), 436–444.

11. Treder, K.P., Huang, C., Kim, J.S., and Kirkland, A.I. (2022) Applications of deep learning in electron microscopy. *Microscopy*, 71 (Supplement_1), i100–i115.

12. Knoll, M., and Ruska, E. (1932) Das Elektronenmikroskop. *Z. Physik*, 78 (5–6), 318–339.

13. Haguenau, F., Hawkes, P.W., Hutchison, J.L., Satiat–Jeunemaître, B., Simon, G.T., and Williams, D.B. (2003) Key Events in the History of Electron Microscopy. *Microsc. Microanal.*, 9 (2), 96–138.

14. Tendeloo, G.V., Bals, S., Aert, S.V., Verbeeck, J., and Dyck, D.V. (2012) Advanced Electron Microscopy for Advanced Materials. *Adv. Mater.*, 24 (42), 5655–5675.

15. Rose, H.H. (2009) Historical aspects of aberration correction. *J. Electron. Microsc.*, 58 (3), 77–85.

16. Oxley, M.P., Lupini, A.R., and Pennycook, S.J. (2017) Ultra-high resolution electron microscopy. *Rep. Prog. Phys.*, 80 (2), 026101.

17. Ardenne, M. von (1938) Das Elektronen-Rastermikroskop. *Z. Phys.*, 109 (9–10), 553–572.

18. Ardenne, M. von (1940) Über ein Universal-Elektronenmikroskop für Hellfeld-, Dunkelfeld- und Stereobild-Betrieb. *Z. Phys.*, 115 (5–6), 339–368.

19. Smith, D.J. (2008) Ultimate resolution in the electron microscope? *Mater. Today*, 11, 30–38.

20. Smith, D.J. (2010) Progress & perspectives for atomic-resolution electron microscopy. *Mater. Today*, 12, 10–16.

21. Midgley, P.A., and Thomas, J.M. (2014) Multi-Dimensional Electron Microscopy. *Angew. Chem. Int. Ed.*, 53 (33), 8614–8617.

22. Hirsch, P.B., Horne, R.W., and Whelan, M.J. (1956) LXVIII. Direct observations of the arrangement and motion of dislocations in aluminium. *Philos. Mag.*, 1 (7), 677–684.

23. Taylor, G.I. (1934) The mechanism of plastic deformation of crystals. Part I.—Theoretical. *Proc. Roy. Soc. A.*

24. Orowan, E. (1934) Zur Kristallplastizität. III. *Z. Phys.*, 89 (9–10), 634–659.

25. Polanyi, M. (1934) Über eine Art Gitterstörung, die einen Kristall plastisch machen könnte. *Z. Phys.*, 89 (9–10), 660–664.

26. Campbell, G.H., McKeown, J.T., and Santala, M.K. (2014) Time resolved electron microscopy for in situ experiments. *Appl. Phys. Rev.*, 1 (4), 041101.

27. Press, O.U. (2021) in situ, adv. and adj. *Oxford English Dictionary*.

28. Ross, F.M. (2015) Opportunities and challenges in liquid cell electron microscopy. *Science*, 350 (6267), aaa9886.

29. Taheri, M.L., Stach, E.A., Arslan, I., Crozier, P.A., Kabius, B.C., LaGrange, T., Minor, A.M., Takeda, S., Tanase, M., Wagner, J.B., and Sharma, R. (2016) Current status and future directions for in situ transmission electron microscopy. *Ultramicroscopy*, 170, 86–95.

30. Chen, Q., Dwyer, C., Sheng, G., Zhu, C., Li, X., Zheng, C., and Zhu, Y. (2020) Imaging Beam-Sensitive Materials by Electron Microscopy. *Adv. Mater.*, 32 (16), 1907619.

31. Ross, F.M., and Minor, A.M. (2019) In Situ Transmission Electron Microscopy, in *Springer Handbook of Microscopy*, vol. 1, Springer, Cham, pp. 101–187.

32. King, W.E., Campbell, G.H., Frank, A., Reed, B., Schmerge, J.F., Siwick, B.J., Stuart, B.C., and Weber, P.M. (2005) Ultrafast electron microscopy in materials science, biology, and chemistry. *J. Appl. Phys.*, 97 (11), 111101.

33. Banhart, F. (2008) Introduction to In-Situ Electron Microscopy, in *In-situ Electron Microscopy At High Resolution* (eds.Banhart, F.), World Scientific Publishing Company, pp. 1–14.

34. Butler, E.P. (1979) In situ experiments in the transmission electron microscope. *Rep. Prog. Phys.*, 42 (5), 833.

35. Bahr, G.F., Zeitler, E.H., and Kobayashi, K. (1966) High-Voltage Electron Microscopy. *J. Appl. Phys.*, 37 (7), 2900–2907.

36. Cosslett, V.E. (1968) High-voltage electron microscopy. *Phys. Today*, 21 (7), 23–31.

37. Loretto, M.H., and Smallman, R.E. (1977) An assessment of high voltage electron microscopy (HVEM). An invited review. *Mater. Sci. Eng.*, 28 (1), 1–32.

38. Hashimoto, H., Takai, Y., Yokota, Y., Endoh, H., and Fukada, E. (1980) Direct Observations of the Arrangement of Atoms around Stacking Faults and Twins in Gold Crystals and the Movement of Atoms Accompanying Their Formation and Disappearance. *Jpn. J. Appl. Phys.*, 19 (1), L1.

39. Sinclair, R., Ponce, F.A., Yamashita, T., Smith, D.J., Camps, R.A., Freeman, L.A., Erasmus, S.J., Nixon, W.C., Smith, K.C.A., and Catto, C.J.D. (1982) Dynamic observation of defect annealing in CdTe at lattice resolution. *Nature*, 298 (5870), 127–131.

40. Sinclair, R., Yamashita, T., Parker, M.A., Kim, K.B., Holloway, K., and Schwartzman, A.F. (1988) The development of in situ high-resolution electron microscopy. *Acta. Cryst. A*, 44 (6), 965–975.

41. Sinclair, R. (2013) In situ high-resolution transmission electron microscopy of material reactions. *MRS Bulletin*, 38 (12), 1065–1071.

42. Gibson, J.M., McDonald, M.L., and Unterwald, F.C. (1985) Direct imaging of a novel silicon surface reconstruction. *Phys. Rev. Lett.*, 55 (17), 1765–1767.

43. Sinclair, R., and Parker, M.A. (1986) High-resolution transmission electron microscopy of silicon re-growth at controlled elevated temperatures. *Nature*, 322 (6079), 531–533.

44. Olson, G.L., and Roth, J.A. (1988) Kinetics of solid phase crystallization in amorphous silicon. *Mater Sci Rep*, 3 (1), 1–77.

45. Ohring, M. (2002) Chapter 11 - Interdiffusion, Reactions, and Transformations in Thin Films, in *Materials Science of Thin Films*, vol. 126, pp. 680–681.

46. Johnson, B.C., McCallum, J.C., and Aziz, M.J. (2015) 7 - Solid-Phase Epitaxy, in *Handbook of Crystal Growth*, vol. 2, pp. 327–329.

47. Haider, M., Rose, H., Uhlemann, S., Schwan, E., Kabius, B., and Urban, K. (1998) A spherical-aberration-corrected 200kV transmission electron microscope. *Ultramicroscopy*, 75 (1), 53–60.

48. Haider, M., Uhlemann, S., Schwan, E., Rose, H., Kabius, B., and Urban, K. (1998) Electron microscopy image enhanced. *Nature*, 392 (6678), 768–769.

49. Haider, M., Uhlemann, S., and Zach, J. (2000) Upper limits for the residual aberrations of a high-resolution aberration-corrected STEM. *Ultramicroscopy*, 81 (3–4), 163–175.

50. Kabius, B., Haider, M., Uhlemann, S., Schwan, E., Urban, K., and Rose, H. (2002) First application of a spherical-aberration corrected transmission electron microscope in materials science. *J. Electron. Microsc.*, 51 (suppl 1), S51–S58.

51. Nellist, P.D., and Pennycook, S.J. (1998) Subangstrom Resolution by Underfocused Incoherent Transmission Electron Microscopy. *Phys. Rev. Lett.*, 81 (19), 4156–4159.

52. Nellist, P.D., Chisholm, M.F., Dellby, N., Krivanek, O.L., Murfitt, M.F., Szilagyi, Z.S., Lupini, A.R., Borisevich, A., Jr., W.H.S., and Pennycook, S.J. (2004) Direct Sub-Angstrom Imaging of a Crystal Lattice. *Science*, 305 (5691), 1741–1741.

53. Erni, R., Rossell, M.D., Kisielowski, C., and Dahmen, U. (2009) Atomic-Resolution Imaging with a Sub-50-pm Electron Probe. *Phys. Rev. Lett.*, 102 (9), 096101.

54. Egerton, R.F., and Malac, M. (2004) EELS in the TEM. *J. Electron. Spectros. Relat. Phenomena*, 143 (2–3), 43–50.

55. Brydson, R. (2007) Electron Energy-loss Spectroscopy and Energy Dispersive X-Ray Analysis, in *Nanocharacterisation*, vol. 3, Royal Society of Chemistry.

56. Boyes, E.D., and Gai, P.L. (1997) Environmental high resolution electron microscopy and applications to chemical science. *Ultramicroscopy*, 67 (1–4), 219–232.

57. Gai, P.L. (2002) Developments in in situ Environmental Cell High-Resolution Electron Microscopy and Applications to Catalysis. *Top. Catal.*, 21 (4), 161–173.

58. Zhang, M., Olson, E.A., Twesten, R.D., Wen, J.G., Allen, L.H., Robertson, I.M., and Petrov, I. (2005) In situ transmission electron microscopy studies enabled by microelectromechanical system technology. *J. Mater. Res*, 20 (7), 1802–1807.

59. Creemer, J.F., Helveg, S., Hoveling, G.H., Ullmann, S., Molenbroek, A.M., Sarro, P.M., and Zandbergen, H.W. (2008) Atomic-scale electron microscopy at ambient pressure. *Ultramicroscopy*, 108 (9), 993–998.

60. Web of Science. *https://www.webofscience.com/, Access Date: 12.04.2022*.

61. Yu, Q., Legros, M., and Minor, A.M. (2015) In situ TEM nanomechanics. *MRS Bull.*, 40 (1), 62–70.

62. Kacher, J., Zhu, T., Pierron, O., and Spearot, D.E. (2019) Integrating in situ TEM experiments and atomistic simulations for defect mechanics. *Curr. Opin. Solid. State Mater. Sci.*, 23 (3), 117–128.

63. Legros, M. (2014) In situ mechanical TEM: Seeing and measuring under stress with electrons. *C. R. Phys.*, 15 (2–3), 224–240.

64. Li, X., and Minor, A.M. (2021) Precise measurement of activation parameters for individual dislocation nucleation during in situ TEM tensile testing of single crystal nickel. *Scr. Mater.*, 197, 113764.

65. Wang, J., Zeng, Z., Weinberger, C.R., Zhang, Z., Zhu, T., and Mao, S.X. (2015) In situ atomic-scale observation of twinning-dominated deformation in nanoscale body-centred cubic tungsten. *Nat. Mater.*, 14 (6), 594–600.

66. Hosson, J.Th.M.D. (2009) Advances in transmission electron microscopy: In situ straining and in situ compression experiments on metallic glasses. *Microsc. Res. Tech.*, 72 (3), 250–260.

67. Legros, M., Gianola, D.S., and Motz, C. (2010) Quantitative In Situ Mechanical Testing in Electron Microscopes. *MRS Bull.*, 35 (5), 354–360.

68. Espinosa, H.D., Bernal, R.A., and Filleter, T. (2012) In Situ TEM Electromechanical Testing of Nanowires and Nanotubes. *Small*, 8 (21), 3233–3252.

69. Nili, H., Kalantar-zadeh, K., Bhaskaran, M., and Sriram, S. (2013) In situ nanoindentation: Probing nanoscale multifunctionality. *Prog. Mater. Sci.*, 58 (1), 1–29.

70. Hÿtch, M.J., and Minor, A.M. (2014) Observing and measuring strain in nanostructures and devices with transmission electron microscopy. *MRS Bull.*, 39 (2), 138–146.

71. Wang, J., and Mao, S.X. (2016) Atomistic perspective on in situ nanomechanics. *Extreme Mech. Lett.*, 8, 127–139.

72. Li, X., Sun, M., Shan, C., Chen, Q., and Wei, X. (2018) Mechanical Properties of 2D Materials Studied by In Situ Microscopy Techniques. *Adv. Mater. Interfaces*, 5 (5), 1701246.

73. Bhowmick, S., Espinosa, H., Jungjohann, K., Pardoen, T., and Pierron, O. (2019) Advanced microelectromechanical systems-based nanomechanical testing: Beyond stress and strain measurements. *MRS. Bull.*, 44 (6), 487–493.

74. Cui, Y., Li, N., and Misra, A. (2019) An overview of interface-dominated deformation mechanisms in metallic nanocomposites elucidated using in situ straining in a TEM. *J. Mater. Res.*, 34 (9), 1469–1478.

75. Boniface, M., Plodinec, M., Schlögl, R., and Lunkenbein, T. (2020) Quo Vadis Micro-Electro-Mechanical Systems for the Study of Heterogeneous Catalysts Inside the Electron Microscope? *Top. Catal.*, 63 (15–18), 1623–1643.

76. Zhang, M., Efremov, M.Yu., Schiettekatte, F., Olson, E.A., Kwan, A.T., Lai, S.L., Wisleder, T., Greene, J.E., and Allen, L.H. (2000) Size-dependent melting point depression of nanostructures: Nanocalorimetric measurements. *Phys. Rev. B*, 62 (15), 10548–10557.

77. Heide, H.G. (1962) ELECTRON MICROSCOPIC OBSERVATION OF SPECIMENS UNDER CONTROLLED GAS PRESSURE. *J. Cell Biol.*, 13 (1), 147–152.

78. Park, J., Koo, K., Noh, N., Chang, J.H., Cheong, J.Y., Dae, K.S., Park, J.S., Ji, S., Kim, I.-D., and Yuk, J.M. (2021) Graphene Liquid Cell Electron Microscopy: Progress, Applications, and Perspectives. *ACS Nano*, 15 (1), 288–308.

79. Baker, R.T.K. (1979) In Situ Electron Microscopy Studies of Catalyst Particle Behavior. *Catal. Rev. - Sci. Eng.*, 19 (2), 161–209.

80. Packan, N.H., and Braski, D.N. (1979) Electron microscope in situ annealing study of voids induced by irradiation in aluminum. *J. Nucl. Mater.*, 34 (3), 307–314.

81. Kamino, T., Yaguchi, T., Sato, T., and Hashimoto, T. (2005) Development of a technique for high resolution electron microscopic observation of nano-materials at elevated temperatures. *J. Electron Microsc.*, 54 (6), 505–508.

82. Saka, H., Kamino, T., Ara, S., and Sasaki, K. (2008) In Situ Heating Transmission Electron Microscopy. *MRS Bull.*, 33 (2), 93–100.

83. Allard, L.F., Bigelow, W.C., Jose-Yacaman, M., Nackashi, D.P., Damiano, J., and Mick, S.E. (2009) A new MEMS-based system for ultra-high-resolution imaging at elevated temperatures. *Microsc. Res. Tech.*, 72 (3), 208–215.

84. Zhu, Y., Zhao, H., He, Y., and Wang, R. (2021) In-situ transmission electron microscopy for probing the dynamic processes in materials. *J. Phys. D: Appl. Phys.*, 54 (44), 443002.

85. COMSOL COMSOL Multiphysics. *https://www.comsol.com/, Access Date: 04.05.2022*.

86. Mele, L., Rossi, T., Riccio, M., Iervolino, E., Santagata, F., Irace, A., Breglio, G., Creemer, J.F., and Sarro, P.M. (2011) Electro-thermal analysis of MEMS microhotplates for the optimization of temperature uniformity. *Procedia Eng.*, 25, 387–390.

87. Morana, B., Santagata, F., Mele, L., Mihailović, M., Pandraud, G., Creemer, J.F., and Sarro, P.M. (2011) A silicon carbide MEMS microhotplate for nanomaterial characterization in TEM. 380–383.

88. Allard, L.F., Overbury, S.H., Bigelow, W.C., Katz, M.B., Nackashi, D.P., and Damiano, J. (2012) Novel MEMS-Based Gas-Cell/Heating Specimen Holder Provides Advanced Imaging Capabilities for In Situ Reaction Studies. *Microsc. Microanal.*, 18 (4), 656–666.

89. Prasad, M., Arya, D.S., and Khanna, V.K. (2015) Fabrication and reliability study of a double spiral platinum-based MEMS hotplate. *J. Micro Nanolithogr. MEMS MOEMS*, 14 (2), 025003–025003.

90. Mele, L., Konings, S., Dona, P., Evertz, F., Mitterbauer, C., Faber, P., Schampers, R., and Jinschek, J.R. (2016) A MEMS-based heating holder for the direct imaging of simultaneous in-situ heating and biasing experiments in scanning/transmission electron microscopes. *Microsc. Res. Techniq.*, 79 (4), 239–250.

91. Bataineh, K. (2017) Novel in –Situ heating transmission electron microscope holder for atomic resolution. *Mechanika*, 23 (2), 265–272.

92. Garza, H.H.P., Morsink, D., Xu, J., Sholkina, M., Pivak, Y., Pen, M., Weperen, S., and Xu, Q. (2017) MEMS-based nanoreactor for in situ analysis of solid–gas interactions inside the transmission electron microscope. *Micro Nano Lett.*, 12 (2), 69–75.

93. Omme, J.T. van, Zakhozheva, M., Spruit, R.G., Sholkina, M., and Garza, H.H.P. (2018) Advanced microheater for in situ transmission electron microscopy; enabling unexplored analytical studies and extreme spatial stability. *Ultramicroscopy*, 192, 14–20.

94. DENS Solutions. *https://denssolutions.com/, Access Date: 04.05.2022*.

95. Protochips Inc. *https://www.protochips.com/, Access Date: 04.05.2022*.

96. DENS Solutions wildfire. *https://denssolutions.com/products/wildfire/, Access Date: 04.05.2022*.

97. Protochips Fusion Select. *https://www.protochips.com/products/fusion/, Access Date: 04.05.2022*.

98. He, K., Robertson, A.W., Fan, Y., Allen, C.S., Lin, Y.-C., Suenaga, K., Kirkland, A.I., and Warner, J.H. (2015) Temperature Dependence of the Reconstruction of Zigzag Edges in Graphene. *ACS Nano*, 9 (5), 4786–4795.

99. Dave, S.H., Gong, C., Robertson, A.W., Warner, J.H., and Grossman, J.C. (2016) Chemistry and Structure of Graphene Oxide via Direct Imaging. *ACS Nano*, 10 (8), 7515–7522.

100. Hansen, T.W., DeLaRiva, A.T., Challa, S.R., and Datye, A.K. (2013) Sintering of Catalytic Nanoparticles: Particle Migration or Ostwald Ripening? *Acc. Chem. Res.*, 46 (8), 1720–1730.

101. Asoro, M.A., Ferreira, P.J., and Kovar, D. (2014) In situ transmission electron microscopy and scanning transmission electron microscopy studies of sintering of Ag and Pt nanoparticles. *Acta Mater.*, 81, 173–183.

102. Yao, X., Wei, Y., Wang, Z., and Gan, L. (2020) Revealing the Role of Surface Composition on the Particle Mobility and Coalescence of Carbon-Supported Pt Alloy Fuel Cell Catalysts by In Situ Heating (S)TEM. *ACS Catal.*, 10 (13), 7381–7388.

103. DENS climate nanoreactor. *https://denssolutions.com/products/climate/nano-reactor/, Access Date: 06.05.2022*.

104. Creemer, J.F., Helveg, S., Kooyman, P.J., Molenbroek, A.M., Zandbergen, H.W., and Sarro, P.M. (2010) A MEMS Reactor for Atomic-Scale Microscopy of Nanomaterials Under Industrially Relevant Conditions. *J. Microelectromechanical Syst.*, 19 (2), 254–264.

105. Dai, S., You, Y., Zhang, S., Cai, W., Xu, M., Xie, L., Wu, R., Graham, G.W., and Pan, X. (2017) In situ atomic-scale observation of oxygen-driven core-shell formation in Pt3Co nanoparticles. *Nat. Commun.*, 8 (1), 204.

106. Avanesian, T., Dai, S., Kale, M.J., Graham, G.W., Pan, X., and Christopher, P. (2017) Quantitative and Atomic-Scale View of CO-Induced Pt Nanoparticle Surface Reconstruction at Saturation Coverage via DFT Calculations Coupled with in Situ TEM and IR. *J. Am. Chem. Soc.*, 139 (12), 4551–4558.

107. Beck, A., Huang, X., Artiglia, L., Zabilskiy, M., Wang, X., Rzepka, P., Palagin, D., Willinger, M.-G., and Bokhoven, J.A. van (2020) The dynamics of overlayer formation on catalyst nanoparticles and strong metal-support interaction. *Nat. Commun.*, 11 (1), 3220.

108. Tauster, S.J., Fung, S.C., Baker, R.T.K., and Horsley, J.A. (1981) Strong Interactions in Supported-Metal Catalysts. *Science*, 211 (4487), 1121–1125.

109. Yokosawa, T., Alan, T., Pandraud, G., Dam, B., and Zandbergen, H. (2012) In-situ TEM on (de)hydrogenation of Pd at 0.5–4.5bar hydrogen pressure and 20–400°C. *Ultramicroscopy*, 112 (1), 47–52.

110. Chi, M., Wang, C., Lei, Y., Wang, G., Li, D., More, K.L., Lupini, A., Allard, L.F., Markovic, N.M., and Stamenkovic, V.R. (2015) Surface faceting and elemental diffusion behaviour at atomic scale for alloy nanoparticles during in situ annealing. *Nat. Commun.*, 6 (1), 8925.

111. Unocic, K.A., Shin, D., Unocic, R.R., and Allard, L.F. (2017) NiAl Oxidation Reaction Processes Studied In Situ Using MEMS-Based Closed-Cell Gas Reaction Transmission Electron Microscopy. *Oxid. Met.*, 88 (3–4), 495–508.

112. Wu, J., Shan, H., Chen, W., Gu, X., Tao, P., Song, C., Shang, W., and Deng, T. (2016) In Situ Environmental TEM in Imaging Gas and Liquid Phase Chemical Reactions for Materials Research. *Adv. Mater.*, 28 (44), 9686–9712.

113. Gao, W., Hood, Z.D., and Chi, M. (2016) Interfaces in Heterogeneous Catalysts: Advancing Mechanistic Understanding through Atomic-Scale Measurements. *Acc. Chem. Res*, 50 (4), 787–795.

114. Dai, S., Gao, W., Zhang, S., Graham, G.W., and Pan, X. (2017) Transmission electron microscopy with atomic resolution under atmospheric pressures. *MRS Commun.*, 7 (4), 798–812.

115. He, B., Zhang, Y., Liu, X., and Chen, L. (2020) In-situ Transmission Electron Microscope Techniques for Heterogeneous Catalysis. *ChemCatChem*, 12 (7), 1853–1872.

116. Ye, F., Xu, M., Dai, S., Tieu, P., Ren, X., and Pan, X. (2020) In Situ TEM Studies of Catalysts Using Windowed Gas Cells. *Catalysts*, 10 (7), 779.

117. Jonge, N. de, and Ross, F.M. (2011) Electron microscopy of specimens in liquid. *Nat. Nanotechnol.*, 6 (11), 695–704.

118. Williamson, M.J., Tromp, R.M., Vereecken, P.M., Hull, R., and Ross, F.M. (2003) Dynamic microscopy of nanoscale cluster growth at the solid–liquid interface. *Nat. Mater.*, 2 (8), 532–536.

119. Zeng, Z., Liang, W.-I., Chu, Y.-H., and Zheng, H. (2014) In situ TEM study of the Li–Au reaction in an electrochemical liquid cell. *Faraday Discuss.*, 176, 95–107.

120. Leenheer, A.J., Jungjohann, K.L., Zavadil, K.R., and Harris, C.T. (2016) Phase Boundary Propagation in Li-Alloying Battery Electrodes Revealed by Liquid-Cell Transmission Electron Microscopy. *ACS Nano*, 10 (6), 5670–5678.

121. Zheng, H., Smith, R.K., Jun, Y., Kisielowski, C., Dahmen, U., and Alivisatos, A.P. (2009) Observation of Single Colloidal Platinum Nanocrystal Growth Trajectories. *Science*, 324 (5932), 1309–1312.

122. Evans, J.E., Jungjohann, K.L., Browning, N.D., and Arslan, I. (2011) Controlled Growth of Nanoparticles from Solution with In Situ Liquid Transmission Electron Microscopy. *Nano Lett.*, 11 (7), 2809–2813.

123. Wu, J., Gao, W., Yang, H., and Zuo, J.-M. (2017) Dissolution Kinetics of Oxidative Etching of Cubic and Icosahedral Platinum Nanoparticles Revealed by in Situ Liquid Transmission Electron Microscopy. *ACS Nano*, 11 (2), 1696–1703.

124. Jonge, N. de, Peckys, D.B., Kremers, G.J., and Piston, D.W. (2009) Electron microscopy of whole cells in liquid with nanometer resolution. *Proc. Natl. Acad. Sci. U.S.A.*, 106 (7), 2159–2164.

125. Mirsaidov, U.M., Zheng, H., Casana, Y., and Matsudaira, P. (2012) Imaging Protein Structure in Water at 2.7 nm Resolution by Transmission Electron Microscopy. *Biophys. J.*, 102 (4), L15–L17.

126. Park, J., Park, H., Ercius, P., Pegoraro, A.F., Xu, C., Kim, J.W., Han, S.H., and Weitz, D.A. (2015) Direct Observation of Wet Biological Samples by Graphene Liquid Cell Transmission Electron Microscopy. *Nano Lett.*, 15 (7), 4737–4744.

127. Jonge, N. de, Houben, L., Dunin-Borkowski, R.E., and Ross, F.M. (2019) Resolution and aberration correction in liquid cell transmission electron microscopy. *Nat. Rev. Mater.*, 4 (1), 61–78.

128. Wu, F., and Yao, N. (2015) Advances in sealed liquid cells for in-situ TEM electrochemial investigation of lithium-ion battery. *Nano Energy*, 11, 196–210.

129. Hodnik, N., Dehm, G., and Mayrhofer, K.J.J. (2016) Importance and Challenges of Electrochemical in Situ Liquid Cell Electron Microscopy for Energy Conversion Research. *Acc. Chem. Res.*, 49 (9), 2015–2022.

130. Liao, H.-G., and Zheng, H. (2016) Liquid Cell Transmission Electron Microscopy. *Annu. Rev. Phys. Chem.*, 67 (1), 719–747.

131. Pu, S., Gong, C., and Robertson, A.W. (2020) Liquid cell transmission electron microscopy and its applications. *R. Soc. Open Sci.*, 7 (1), 191204.

132. Sun, M., Tian, J., and Chen, Q. (2021) The studies on wet chemical etching via in situ liquid cell TEM. *Ultramicroscopy*, 231, 113271.

133. Aires, F.J.C.S., and Bertolini, J.C. (2009) On the Use of Silicon Nitride in Catalysis. *Top Catal.*, 52 (11), 1492–1505.

134. Hullmann, D., Wendt, G., Šingliar, U., and Ziegenbalg, G. (2002) Propane dehydrogenation over supported platinum silicon nitride catalysts. *Appl. Catal. A: Gen.*, 225 (1–2), 261–270.

135. Xiong, H., Liu, Z., Chen, X., Wang, H., Qian, W., Zhang, C., Zheng, A., and Wei, F. (2022) In situ imaging of the sorption-induced subcell topological flexibility of a rigid zeolite framework. *Science*, 376 (6592), 491–496.

136. Marton, L. (1935) La microscopie electronique des objects biologiques. *Bull. Acad. R. Med. Belg.*, 21 (553), 606–617.

137. Koneti, S., Roiban, L., Dalmas, F., Langlois, C., Gay, A.-S., Cabiac, A., Grenier, T., Banjak, H., Maxim, V., and Epicier, T. (2019) Fast electron tomography: Applications to beam sensitive samples and in situ TEM or operando environmental TEM studies. *Mater. Charact.*, 151, 480–495.

138. Jinschek, J.R. (2014) Advances in the environmental transmission electron microscope (ETEM) for nanoscale in situ studies of gas–solid interactions. *Chem. Commun.*, 50 (21), 2696–2706.

139. Campbell, G.H., McKeown, J.T., and Santala, M.K. (2019) High-Speed Electron Microscopy, in *Springer Handbook of Microscopy*, vol. 1, Springer Cham., pp. 455–486.

140. Bostanjoglo, O., and Rosin, Th. (1977) Ultrasonically Induced Magnetic Reversals Observed by Stroboscopic Electron Microscopy. *Opt. Acta: Int. J. Opt.*, 24 (6), 657–664.

141. LaGrange, T., Armstrong, M.R., Boyden, K., Brown, C.G., Campbell, G.H., Colvin, J.D., DeHope, W.J., Frank, A.M., Gibson, D.J., Hartemann, F.V., Kim, J.S., King, W.E., Pyke, B.J., Reed, B.W., Shirk, M.D., Shuttlesworth, R.M., Stuart, B.C., Torralva, B.R., and Browning, N.D. (2006) Single-shot dynamic transmission electron microscopy. *Appl. Phys. Lett.*, 89 (4), 044105.

142. Zewail, A.H. (2006) 4D ULTRAFAST ELECTRON DIFFRACTION, CRYSTALLOGRAPHY, AND MICROSCOPY. *Annu. Rev. Phys. Chem.*, 57 (1), 65–103.

143. Flannigan, D.J., Park, S.T., and Zewail, A.H. (2010) Nanofriction Visualized in Space and Time by 4D Electron Microscopy. *Nano Lett.*, 10 (11), 4767–4773.

144. Flannigan, D.J., and Zewail, A.H. (2010) Optomechanical and Crystallization Phenomena Visualized with 4D Electron Microscopy: Interfacial Carbon Nanotubes on Silicon Nitride. *Nano Lett.*, 10 (5), 1892–1899.

145. Yurtsever, A., Veen, R.M. van der, and Zewail, A.H. (2012) Subparticle Ultrafast Spectrum Imaging in 4D Electron Microscopy. *Science*, 335 (6064), 59–64.

146. Feist, A., Echternkamp, K.E., Schauss, J., Yalunin, S.V., Schäfer, S., and Ropers, C. (2015) Quantum coherent optical phase modulation in an ultrafast transmission electron microscope. *Nature*, 521 (7551), 200–203.

147. Priebe, K.E., Rathje, C., Yalunin, S.V., Hohage, T., Feist, A., Schäfer, S., and Ropers, C. (2017) Attosecond electron pulse trains and quantum state reconstruction in ultrafast transmission electron microscopy. *Nat. Photonics*, 11 (12), 793–797.

148. Houdellier, F., Caruso, G.M., Weber, S., Kociak, M., and Arbouet, A. (2018) Development of a high brightness ultrafast Transmission Electron Microscope based on a laser-driven cold field emission source. *Ultramicroscopy*, 186, 128–138.

149. Flannigan, D.J., and Zewail, A.H. (2012) 4D Electron Microscopy: Principles and Applications. *Acc. Chem. Res.*, 45 (10), 1828–1839.

150. Baskin, J.S., and Zewail, A.H. (2014) Seeing in 4D with electrons: Development of ultrafast electron microscopy at Caltech. *C.R. Phys.*, 15 (2–3), 176–189.

151. Shorokhov, D., and Zewail, A.H. (2015) Perspective: 4D ultrafast electron microscopy—Evolutions and revolutions. *J. Chem. Phys.*, 144 (8), 080901.

152. Bostanjoglo, O., and Heinricht, F. (1987) Producing high-current nanosecond electron pulses with a standard tungsten hairpin gun. *J. Phys. E: Sci. Instrum.*, 20 (12), 1491.

153. Bostanjoglo, O., Elschner, R., Mao, Z., Nink, T., and Weingärtner, M. (2000) Nanosecond electron microscopes. *Ultramicroscopy*, 81 (3–4), 141–147.

154. Dömer, H., and Bostanjoglo, O. (2003) High-speed transmission electron microscope. *Rev. Sci. Instrum.*, 74 (10), 4369–4372.

155. Bostanjoglo, O., and Otte, D. (1995) High-speed electron microscopy of nanocrystallization in Al□Ni films by nanosecond laser pulses. *Phys. Status Solidi A*, 150 (1), 163–169.

156. Bostanjoglo, O., and Nink, T. (1997) Liquid motion in laser pulsed Al, Co and Au films. *Appl. Surf. Sci.*, 109, 101–105.

157. Nink, T., Galbert, F., Mao, Z.L., and Bostanjoglo, O. (1999) Dynamics of laser pulse-induced melts in Ni–P visualized by high-speed transmission electron microscopy. *Appl. Surf. Sci.*, 138, 439–443.

158. Nink, T., Mao, Z.L., and Bostanjoglo, O. (2000) Melt instability and crystallization in thin amorphous Ni–P films. *Appl. Surf. Sci.*, 154, 140–145.

159. LaGrange, T., Reed, B.W., Santala, M.K., McKeown, J.T., Kulovits, A., Wiezorek, J.M.K., Nikolova, L., Rosei, F., Siwick, B.J., and Campbell, G.H. (2012) Approaches for ultrafast imaging of transient materials processes in the transmission electron microscope. *Micron*, 43 (11), 1108–1120.

160. LaGrange, T., Reed, B.W., and Masiel, D.J. (2015) Movie-mode dynamic electron microscopy. *MRS Bull.*, 40 (1), 22–28.

161. Kim, J.S., LaGrange, T., Reed, B.W., Taheri, M.L., Armstrong, M.R., King, W.E., Browning, N.D., and Campbell, G.H. (2008) Imaging of Transient Structures Using Nanosecond in Situ TEM. *Science*, 321 (5895), 1472–1475.

162. Grapes, M.D., LaGrange, T., Woll, K., Reed, B.W., Campbell, G.H., LaVan, D.A., and Weihs, T.P. (2014) In situ transmission electron microscopy investigation of the interfacial reaction between Ni and Al during rapid heating in a nanocalorimeter. *APL Mater.*, 2 (11), 116102.

163. Santala, M.K., Reed, B.W., Raoux, S., Topuria, T., LaGrange, T., and Campbell, G.H. (2013) Irreversible reactions studied with nanosecond transmission electron microscopy movies: Laser crystallization of phase change materials. *Appl. Phys. Lett.*, 102 (17), 174105.

164. Moldovan, G., Li, X., and Kirkland, A. (2008) Can direct electron detectors outperform phosphor-CCD systems for TEM? *J. Phys.: Conf. Ser.*, 126 (1), 012089.

165. Meyer, R.R., and Kirkland, A.I. (2000) Characterisation of the signal and noise transfer of CCD cameras for electron detection. *Microsc. Res. Tech.*, 49 (3), 269–280.

166. Meyer, R.R., and Kirkland, A. (1998) The effects of electron and photon scattering on signal and noise transfer properties of scintillators in CCD cameras used for electron detection. *Ultramicroscopy*, 75 (1), 23–33.

167. Deptuch, G., Besson, A., Rehak, P., Szelezniak, M., Wall, J., Winter, M., and Zhu, Y. (2007) Direct electron imaging in electron microscopy with monolithic active pixel sensors. *Ultramicroscopy*, 107 (8), 674–684.

168. Faruqi, A.R., and McMullan, G. (2011) Electronic detectors for electron microscopy. *Q. Rev. Biophys.*, 44 (3), 357–390.

169. Ruskin, R.S., Yu, Z., and Grigorieff, N. (2013) Quantitative characterization of electron detectors for transmission electron microscopy. *J. Struct. Biol.*, 184 (3), 385–393.

170. Herrmann, K. -H., and Krahl, D. (1982) The detection quantum efficiency of electronic image recording systems. *J. Microsc.*, 127 (1), 17–28.

171. McMullan, G., Faruqi, A.R., Clare, D., and Henderson, R. (2014) Comparison of optimal performance at 300 keV of three direct electron detectors for use in low dose electron microscopy. *Ultramicroscopy*, 147, 156–163.

172. McMullan, G., Chen, S., Henderson, R., and Faruqi, A.R. (2009) Detective quantum efficiency of electron area detectors in electron microscopy. *Ultramicroscopy*, 109 (9), 1126–1143.

173. Direct Electron Celeritas Camera Product Brochure, https://www.directelectron.com/wp-content/uploads/2022/07/Celeritas-Product-Booklet.pdf, Access Date: 25.05.2022.

174. Mir, J.A., Clough, R., MacInnes, R., Gough, C., Plackett, R., Shipsey, I., Sawada, H., MacLaren, I., Ballabriga, R., Maneuski, D., O'Shea, V., McGrouther, D., and Kirkland, A.I. (2017) Characterisation of the Medipix3 detector for 60 and 80keV electrons. *Ultramicroscopy*, 182, 44–53.

175. McMullan, G., Cattermole, D.M., Chen, S., Henderson, R., Llopart, X., Summerfield, C., Tlustos, L., and Faruqi, A.R. (2007) Electron imaging with Medipix2 hybrid pixel detector. *Ultramicroscopy*, 107 (4–5), 401–413.

176. MerlinEM Medipix3 Product Brochure, https://quantumdetectors.com/products/merlinem/, Access Date: 25.05.2022.

177. Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018) Deep Learning for Computer Vision: A Brief Review. *Comput. Intell. Neurosc.*, 2018, 1–13.

178. Zhang, D., Zhu, Y., Liu, L., Ying, X., Hsiung, C.-E., Sougrat, R., Li, K., and Han, Y. (2018) Atomic-resolution transmission electron microscopy of electron beam–sensitive crystalline materials. *Science*, 359 (6376), 675–679.

179. Ilett, M., S'ari, M., Freeman, H., Aslam, Z., Koniuch, N., Afzali, M., Cattle, J., Hooley, R., Roncal-Herrero, T., Collins, S.M., Hondow, N., Brown, A., and Brydson, R. (2020) Analysis of complex, beam-sensitive materials by transmission electron microscopy and associated techniques. *Phil. Trans. R. Soc. A*, 378 (2186), 20190601.

180. Kalinin, S.V., Sumpter, B.G., and Archibald, R.K. (2015) Big–deep–smart data in imaging for guiding materials design. *Nat. Mater.*, 14 (10), 973–980.

181. Kalinin, S.V., Lupini, A.R., Dyck, O., Jesse, S., Ziatdinov, M., and Vasudevan, R.K. (2019) Lab on a beam—Big data and artificial intelligence in scanning transmission electron microscopy. *MRS Bulletin*, 44 (7), 565–575.

182. Dan, J., Zhao, X., and Pennycook, S.J. (2019) A machine perspective of atomic defects in scanning transmission electron microscopy. *InfoMat.*, 1 (3), 359–375.

183. Ge, M., Su, F., Zhao, Z., and Su, D. (2020) Deep learning analysis on microscopic imaging in materials science. *Mater. Today Nano*, 11, 100087.

184. Ede, J.M. (2021) Deep learning in electron microscopy. *Mach. Learn.: Sci. Technol.*, 2 (1), 011004.

185. Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., and Lew, M.S. (2016) Deep learning for visual understanding: A review. *Neurocomputing*, 187, 27–48.

186. Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2012) ImageNet Classification with Deep Convolutional Neural Networks. *Adv. Neural Inf. Process. Syst.*, 25, 1097–1105.

187. Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., and Pietikäinen, M. (2020) Deep Learning for Generic Object Detection: A Survey. *Int. J. Comput. Vis.*, 128 (2), 261–318.

188. Hao, S., Zhou, Y., and Guo, Y. (2020) A Brief Survey on Semantic Segmentation with Deep Learning. *Neurocomputing*, 406, 302–321.

189. Rawat, W., and Wang, Z. (2017) Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Comput.*, 29 (9), 2352–2449.

190. Jam, J., Kendrick, C., Walker, K., Drouard, V., Hsu, J.G.-S., and Yap, M.H. (2021) A comprehensive review of past and present image inpainting methods. *Comput. Vis. Image Underst.*, 203, 103147.

191. Fan, L., Zhang, F., Fan, H., and Zhang, C. (2019) Brief review of image denoising techniques. *Vis. Comput. Ind. Biomed. Art*, 2 (1), 7.

192. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014) Generative Adversarial Nets. *Adv. Neural Inf. Process. Syst.*, 27, 2672–2680.

193. Gui, J., Sun, Z., Wen, Y., Tao, D., and Ye, J. (2020) A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. *arXiv*, 2001.06937.

194. Ziatdinov, M., Dyck, O., Maksov, A., Li, X., Sang, X., Xiao, K., Unocic, R.R., Vasudevan, R., Jesse, S., and Kalinin, S.V. (2017) Deep Learning of Atomically Resolved Scanning Transmission Electron Microscopy Images: Chemical Identification and Tracking Local Transformations. *ACS Nano*, 11 (12), 12742–12752.

195. Maksov, A., Dyck, O., Wang, K., Xiao, K., Geohegan, D.B., Sumpter, B.G., Vasudevan, R.K., Jesse, S., Kalinin, S.V., and Ziatdinov, M. (2019) Deep learning analysis of defect and phase evolution during electron beam-induced transformations in WS2. *Npj Comput. Mater.*, 5 (12).

196. Ziatdinov, M., Dyck, O., Li, X., Sumpter, B.G., Jesse, S., Vasudevan, R.K., and Kalinin, S.V. (2019) Building and exploring libraries of atomic defects in graphene: Scanning transmission electron and scanning tunneling microscopy study. *Sci. Adv.*, 5 (9), eaaw8989.

197. Ziatdinov, M., Dyck, O., Jesse, S., and Kalinin, S.V. (2019) Atomic Mechanisms for the Si Atom Dynamics in Graphene: Chemical Transformations at the Edge and in the Bulk. *Adv. Funct. Mater.*, 29 (52), 1904480.

198. Long, J., and Darrell, T. (2015) Fully convolutional networks for semantic segmentation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

199. Badrinarayanan, V., Handa, A., and Cipolla, R. (2015) SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling. *arXiv*, 1505.07293.

200. Ronneberger, O., Fischer, P., and Brox, T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 9351, 234–241.

201. Reynolds, D.A., and Rose, R.C. (1995) Robust text-independent speaker identification using Gaussian mixture speaker models. 3, 72–83.

202. (2022) 2.1. Gaussian mixture models. *scikit learn Documentation*.

203. Rosenblatt, F. (1958) The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.*, 65 (6), 386–408.

204. Al-Khedher, M.A., Pezeshki, C., McHale, J.L., and Knorr, F.J. (2007) Quality classification via Raman identification and SEM analysis of carbon nanotube bundles using artificial neural networks. *Nanotechnology*, 18 (35), 355703.

205. Trujillo, M.C.R., Alarcón, T.E., Dalmau, O.S., and Ojeda, A.Z. (2017) Segmentation of carbon nanotube images through an artificial neural network. *Soft Comput.*, 21 (3), 611–625.

206. Frei, M., and Kruis, F.E. (2018) Fully automated primary particle size analysis of agglomerates on transmission electron microscopy images via artificial neural networks. *Powder Technol.*, 332, 120–130.

207. Kirschner, H., and Hillebrand, R. (2000) Neural networks for HREM image analysis. *Information Sciences*, 129 (1–4), 31–44.

208. Broek, W. van den, and Koch, C.T. (2012) Method for Retrieval of the Three-Dimensional Object Potential by Inversion of Dynamical Electron Scattering. *Phys. Rev. Lett.*, 109 (24), 245502.

209. Broek, W. van den, and Koch, C.T. (2013) General framework for quantitative three-dimensional reconstruction from arbitrary detection geometries in TEM. *Phys. Rev. B*, 87 (18), 184108.

210. Pennington, R.S., Broek, W. van den, and Koch, C.T. (2014) Third-dimension information retrieval from a single convergent-beam transmission electron diffraction pattern using an artificial neural network. *Phys. Rev. B*, 89 (20), 205409.

211. Heindl, E., Rau, W.D., and Lichte, H. (1996) The phase-shift method in electron-off-axis holography: using neural network techniques. *Ultramicroscopy*, 64 (1–4), 87–97.

212. Meyer, and Heindl (1998) Reconstruction of off-axis electron holograms using a neural net. *J. Microsc.*, 191 (1), 52–59.

213. Roest, L.I., Heijst, S.E. van, Maduro, L., Rojo, J., and Conesa-Boj, S. (2021) Charting the low-loss region in electron energy loss spectroscopy with machine learning. *Ultramicroscopy*, 222, 113202.

214. Jain, V., Murray, J.F., Roth, F., Turaga, S., Zhigulin, V., Briggman, K.L., Helmstaedter, M.N., Denk, W., and Seung, H.S. (2007) Supervised Learning of Image Restoration with Convolutional Networks. *2007 IEEE 11th International Conference on Computer Vision*, 1–8.

215. Turaga, S.C., Murray, J.F., Jain, V., Roth, F., Helmstaedter, M., Briggman, K., Denk, W., and Seung, H.S. (2010) Convolutional Networks Can Learn to Generate Affinity Graphs for Image Segmentation. *Neural Comput.*, 22 (2), 511–538.

216. Helmstaedter, M., Briggman, K.L., Turaga, S.C., Jain, V., Seung, H.S., and Denk, W. (2013) Connectomic reconstruction of the inner plexiform layer in the mouse retina. *Nature*, 500 (7461), 168–174.

217. Ogura, T., and Sato, C. (2001) An Automatic Particle Pickup Method Using a Neural Network Applicable to Low-Contrast Electron Micrographs. *J. Struct. Biol.*, 136 (3), 227–238.

218. Ogura, T., and Sato, C. (2004) Automatic particle pickup method using a neural network has high accuracy by applying an initial weight derived from eigenimages: a new reference free method for single-particle analysis. *J. Struct. Biol.*, 145 (1–2), 63–75.

219. LeCun, Y. (1987) Modèles connexionistes de l'apprentissage. PhD Thesis.

220. Kalinin, S.V., Dyck, O., Jesse, S., and Ziatdinov, M. (2021) Exploring order parameters and dynamic processes in disordered systems via variational autoencoders. *Sci. Adv.*, 7 (17), eabd5084.

221. Bepler, T., Morin, A., Rapp, M., Brasch, J., Shapiro, L., Noble, A.J., and Berger, B. (2019) Positive-unlabeled convolutional neural networks for particle picking in cryo-electron micrographs. *Nat. Methods*, 16 (11), 1153–1160.

222. Lei, H., and Yang, Y. (2021) CDAE: A Cascade of Denoising Autoencoders for Noise Reduction in the Clustering of Single-Particle Cryo-EM Images. *Front. Genet.*, 11, 627746.

223. Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998) Gradient-based learning applied to document recognition. *Proc. IEEE*, 86 (11), 2278–2324.

224. Förster, G.D., Castan, A., Loiseau, A., Nelayah, J., Alloyeau, D., Fossard, F., Bichara, C., and Amara, H. (2020) A deep learning approach for determining the chiral indices of carbon nanotubes from high-resolution transmission electron microscopy images. *Carbon*, 169, 465–474.

225. Yoshioka, H., and Honda, T. (2021) Determination of the interface between amorphous insulator and crystalline 4H–SiC in transmission electron microscope image by using convolutional neural network. *AIP Adv.*, 11 (1), 015101.

226. Zhu, Y., Ouyang, Q., and Mao, Y. (2017) A deep convolutional neural network approach to single-particle recognition in cryo-electron microscopy. *BMC Bioinform.*, 18 (1), 348.

227. Chowdhury, A., Kautz, E., Yener, B., and Lewis, D. (2016) Image driven machine learning methods for microstructure recognition. *Comput. Mater. Sci.*, 123, 176–187.

228. Xu, W., and LeBeau, J.M. (2018) A deep convolutional neural network to analyze position averaged convergent beam electron diffraction patterns. *Ultramicroscopy*, 188, 59–69.

229. Wang, F., Gong, H., Liu, G., Li, M., Yan, C., Xia, T., Li, X., and Zeng, J. (2016) DeepPicker: A deep learning approach for fully automated particle picking in cryo-EM. *J. Struct. Biol.*, 195 (3), 325–336.

230. Zhang, J., Wang, Z., Chen, Y., Han, R., Liu, Z., Sun, F., and Zhang, F. (2019) PIXER: an automated particle-selection method based on segmentation using a deep neural network. *BMC Bioinform.*, 20 (1), 41.

231. Al-Azzawi, A., Ouadou, A., Max, H., Duan, Y., Tanner, J.J., and Cheng, J. (2020) DeepCryoPicker: fully automated deep neural network for single protein particle picking in cryo-EM. *BMC Bioinform.*, 21 (1), 509.

232. Weber, G.H., Ophus, C., and Ramakrishnan, L. (2018) Automated Labeling of Electron Microscopy Images Using Deep Learning. *2018 IEEE/ACM Machine Learning in HPC Environments (MLHPC)*, 26–36.

233. Simonyan, K., and Zisserman, A. (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv*, 1409.1556.

234. DeCost, B.L., Francis, T., and Holm, E.A. (2017) Exploring the microstructure manifold: Image texture representations applied to ultrahigh carbon steel microstructures. *Acta. Mater.*, 133, 30–40.

235. Azimi, S.M., Britz, D., Engstler, M., Fritz, M., and Mücklich, F. (2018) Advanced Steel Microstructural Classification by Deep Learning Methods. *Sci. Rep.*, 8 (1), 2128.

236. Dennler, N., Foncubierta-Rodriguez, A., Neupert, T., and Sousa, M. (2021) Learning-based defect recognition for quasi-periodic HRSTEM images. *Micron*, 146, 103069.

237. Luo, Q., Holm, E.A., and Wang, C. (2020) A transfer learning approach for improved classification of carbon nanomaterials from TEM images. *Nanoscale Adv.*, 3 (1), 206–213.

238. Sanchez-Garcia, R., Segura, J., Maluenda, D., Carazo, J.M., and Sorzano, C.O.S. (2018) Deep Consensus, a deep learning-based approach for particle pruning in cryo-electron microscopy. *IUCrJ*, 5 (6), 854–865.

239. Huang, G., Liu, Z., Maaten, L. van der, and Weinberger, K.Q. (2017) Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261–2269.

240. Roberts, G., Haile, S.Y., Sainju, R., Edwards, D.J., Hutchinson, B., and Zhu, Y. (2019) Deep Learning for Semantic Segmentation of Defects in Advanced STEM Images of Steels. *Sci. Rep.*, 9 (1), 12744.

241. Suveer, A., Gupta, A., Kylberg, G., and Sintorn, I.-M. (2019) Super-Resolution Reconstruction of Transmission Electron Microscopy Images Using Deep Learning. *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, 548–551.

242. Nanni, L., Luca, E. de, Facin, M.L., and Maguolo, G. (2020) Deep Learning and Handcrafted Features for Virus Image Classification. *J. Imaging*, 6 (12), 143.

243. He, K., Zhang, X., Ren, S., and Sun, J. (2016) Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

244. Kharin, A.Yu. (2020) Deep learning for scanning electron microscopy: synthetic data for the nanoparticles detection. *Ultramicroscopy*, 219, 113125.

245. Tegunov, D., and Cramer, P. (2019) Real-time cryo-electron microscopy data preprocessing with Warp. *Nat. Methods*, 16 (11), 1146–1152.

246. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014) Going Deeper with Convolutions. *arXiv*, 1409.4842.

247. George, B., Assaiya, A., Roy, R.J., Kembhavi, A., Chauhan, R., Paul, G., Kumar, J., and Philip, N.S. (2021) CASSPER is a semantic segmentation-based particle picking algorithm for single-particle cryo-electron microscopy. *Commun. Biol.*, 4 (1), 200.

248. Xu, M., Chai, X., Muthakana, H., Liang, X., Yang, G., Zeev-Ben-Mordehai, T., and Xing, E.P. (2017) Deep learning-based subdivision approach for large scale macromolecules structure recovery from electron cryo tomograms. *Bioinformatics*, 33 (14), i13–i22.

249. Modarres, M.H., Aversa, R., Cozzini, S., Ciancio, R., Leto, A., and Brandino, G.P. (2017) Neural Network for Nanoscience Scanning Electron Microscope Image Recognition. *Sci. Rep.*, 7 (1), 13282.

250. Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016) You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788.

251. Wagner, T., Merino, F., Stabrin, M., Moriya, T., Antoni, C., Apelbaum, A., Hagel, P., Sitsel, O., Raisch, T., Prumbaum, D., Quentin, D., Roderer, D., Tacke, S., Siebolds, B., Schubert, E., Shaikh, T.R., Lill, P., Gatsogiannis, C., and Raunser, S. (2019) SPHIRE-crYOLO is a fast and accurate fully automated particle picker for cryo-EM. *Commun. Biol.*, 2 (1), 218.

252. Wagner, T., and Raunser, S. (2020) The evolution of SPHIRE-crYOLO particle picking and its application in automated cryo-EM processing workflows. *Commun. Biol.*, 3 (1), 61.

253. Yokoyama, Y., Terada, T., Shimizu, K., Nishikawa, K., Kozai, D., Shimada, A., Mizoguchi, A., Fujiyoshi, Y., and Tani, K. (2020) Development of a deep learning-based method to identify “good” regions of a cryo-electron microscopy grid. *Biophys. Rev.*, 12 (2), 349–354.

254. Li, Y., Cash, J.N., Tesmer, J.J.G., and Cianfrocco, M.A. (2020) High-Throughput Cryo-EM Enabled by User-Free Preprocessing Routines. *Structure*, 28 (7), 858–869.

255. Yao, R., Qian, J., and Huang, Q. (2019) Deep-learning with synthetic data enables automated picking of cryo-EM particle images of biological macromolecules. *Bioinformatics*, 36 (4), 1252–1259.

256. Cherukara, M.J., Nashed, Y.S.G., and Harder, R.J. (2018) Real-time coherent diffraction inversion using deep generative networks. *Sci. Rep.*, 8 (1), 16520.

257. Madsen, J., Liu, P., Kling, J., Wagner, J.B., Hansen, T.W., Winther, O., and Schiøtz, J. (2018) A Deep Learning Approach to Identify Local Structures in Atomic-Resolution Transmission Electron Microscopy Images. *Adv. Theory Simul.*, 1 (8), 1800037.

258. Horwath, J.P., Zakharov, D.N., Mégret, R., and Stach, E.A. (2020) Understanding important features of deep learning models for segmentation of high-resolution transmission electron microscopy images. *Npj Comput. Mater.*, 6 (1), 108.

259. Yao, L., Ou, Z., Luo, B., Xu, C., and Chen, Q. (2020) Machine Learning to Reveal Nanoparticle Dynamics from Liquid-Phase TEM Videos. *ACS Cent. Sci.*, 6 (8), 1421–1430.

260. Groschner, C.K., Choi, C., and Scott, M.C. (2021) Machine Learning Pipeline for Segmentation and Defect Identification from High-Resolution Transmission Electron Microscopy Data. *Microsc. Microanal.*, 1–8.

261. Ragone, M., Yurkiv, V., Song, B., Ramsubramanian, A., Shahbazian-Yassar, R., and Mashayek, F. (2020) Atomic column heights detection in metallic nanoparticles using deep convolutional learning. *Comput. Mater. Sci.*, 180, 109722.

262. Lee, J., Jeong, C., and Yang, Y. (2021) Single-atom level determination of 3-dimensional surface atomic structure via neural network-assisted atomic electron tomography. *Nat. Commun.*, 12 (1), 1962.

263. Han, Y., Jang, J., Cha, E., Lee, J., Chung, H., Jeong, M., Kim, T.-G., Chae, B.G., Kim, H.G., Jun, S., Hwang, S., Lee, E., and Ye, J.C. (2021) Deep learning STEM-EDX tomography of nanocrystals. *Nat. Mach. Intell.*, 3 (3), 267–274.

264. Sanchez-Garcia, R., Segura, J., Maluenda, D., Sorzano, C.O.S., and Carazo, J.M. (2020) MicrographCleaner: A python package for cryo-EM micrograph cleaning using deep learning. *J. Struct. Biol.*, 210 (3), 107498.

265. Buchholz, T.-O., Jordan, M., Pigino, G., and Jug, F. (2019) Cryo-CARE: Content-Aware Image Restoration for Cryo-Transmission Electron Microscopy Data. *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, 502–506.

266. Bepler, T., Kelley, K., Noble, A.J., and Berger, B. (2020) Topaz-Denoise: general deep denoising models for cryoEM and cryoET. *Nat Commun.*, 11 (1), 5208.

267. Palovcak, E., Asarnow, D., Campbell, M.G., Yu, Z., and Cheng, Y. (2020) Enhancing the signal-to-noise ratio and generating contrast for cryo-EM images with convolutional neural networks. *IUCrJ*, 7, 1142–1150.

268. Avramov, T.K., Vyenielo, D., Gomez-Blanco, J., Adinarayanan, S., Vargas, J., and Si, D. (2019) Deep Learning for Validating and Estimating Resolution of Cryo-Electron Microscopy Density Maps. *Molecules*, 24 (6), 1181.

269. Pfab, J., Phan, N.M., and Si, D. (2021) DeepTracer for fast de novo cryo-EM protein structure modeling and special studies on CoV-related complexes. *Proc. Natl. Acad. Sci. U.S.A*, 118 (2), e2017525118.

270. Matuszewski, D.J., and Sintorn, I.-M. (2019) Reducing the U-Net size for practical scenarios: Virus recognition in electron microscopy images. *Comput Methods Programs Biomed*, 178, 31–39.

271. Lin, R., Zhang, R., Wang, C., Yang, X.-Q., and Xin, H.L. (2021) TEMImageNet training library and AtomSegNet deep-learning models for high-precision atom segmentation, localization, denoising, and deblurring of atomic-resolution images. *Sci. Rep.*, 11 (1), 5386.

272. Wang, F., Eljarrat, A., Müller, J., Henninen, T.R., Erni, R., and Koch, C.T. (2020) Multi-resolution convolutional neural networks for inverse problems. *Sci. Rep.*, 10 (1), 5730.

273. Diakogiannis, F.I., Waldner, F., Caccetta, P., and Wu, C. (2020) ResUNet-a: A deep learning framework for semantic segmentation of remotely sensed data. *ISPRS J. Photogramm. Remote Sens.*, 162, 94–114.

274. Lee, C.-H., Khan, A., Luo, D., Santos, T.P., Shi, C., Janicek, B.E., Kang, S., Zhu, W., Sobh, N.A., Schleife, A., Clark, B.K., and Huang, P.Y. (2020) Deep Learning Enabled Strain Mapping of Single-Atom Defects in Two-Dimensional Transition Metal Dichalcogenides with Sub-Picometer Precision. *Nano Lett.*, 20 (5), 3369–3377.

275. Fakhry, A., Zeng, T., and Ji, S. (2017) Residual Deconvolutional Networks for Brain Electron Microscopy Image Segmentation. *IEEE Transactions on Medical Imaging*, 36 (2), 447–456.

276. Ding, G., Liu, Y., Zhang, R., and Xin, H.L. (2019) A joint deep learning model to recover information and reduce artifacts in missing-wedge sinograms for electron tomography and beyond. *Sci. Rep.*, 9 (1), 12803.

277. Wang, C., Ding, G., Liu, Y., and Xin, H.L. (2020) 0.7 Å Resolution Electron Tomography Enabled by Deep-Learning-Aided Information Recovery. *Adv. Intell. Syst.*, 2000152.

278. Wang, F., Henninen, T.R., Keller, D., and Erni, R. (2020) Noise2Atom: unsupervised denoising for scanning transmission electron microscopy images. *Appl. Microsc.*, 50 (1), 23.

279. Ede, J.M., and Beanland, R. (2020) Partial Scanning Transmission Electron Microscopy with Deep Learning. *Sci. Rep.*, 10 (1), 8332.

280. Haan, K. de, Ballard, Z.S., Rivenson, Y., Wu, Y., and Ozcan, A. (2019) Resolution enhancement in scanning electron microscopy using deep learning. *Sci. Rep.*, 9 (1), 12050.

281. Jamali, V., Hargus, C., Ben-Moshe, A., Aghazadeh, A., Ha, H.D., Mandadapu, K.K., and Alivisatos, A.P. (2021) Anomalous nanoparticle surface diffusion in LCTEM is revealed by deep learning-assisted analysis. *Proc. Natl. Acad. Sci. U.S.A.*, 118 (10), e2017616118.

282. Oktay, A.B., and Gurses, A. (2019) Automatic detection, localization and segmentation of nano-particles with deep learning in microscopy images. *Micron*, 120, 113–119.

283. Li, W., Field, K.G., and Morgan, D. (2018) Automated defect analysis in electron microscopic images. *Npj Comput. Mater.*, 4 (1), 36.

284. Aguiar, J.A., Gong, M.L., Unocic, R.R., Tasdizen, T., and Miller, B.D. (2019) Decoding crystallography from high-resolution electron imaging and diffraction datasets with deep learning. *Sci. Adv.*, 5 (10), eaaw1949.

285. Cherukara, M.J., Zhou, T., Nashed, Y., Enfedaque, P., Hexemer, A., Harder, R.J., and Holt, M.V. (2020) AI-enabled high-resolution scanning coherent diffraction imaging. *Appl. Phys. Lett.*, 117 (4), 044103.

286. Chatzidakis, M., and Botton, G.A. (2019) Towards calibration-invariant spectroscopy using deep learning. *Sci. Rep.*, 9 (1), 2126.

287. Ede, J.M., and Beanland, R. (2019) Improving Electron Micrograph Signal-to-Noise with an Atrous Convolutional Encoder-Decoder. *Ultramicroscopy*, 202, 18–25.

288. Xiao, Y., and Yang, G. (2017) A fast method for particle picking in cryo-electron micrographs based on fast R-CNN. *AIP Conf. Proc.*, 1836, 020080.

289. Xu, K., Wang, Z., Shi, J., Li, H., and Zhang, Q.C. (2019) A2-Net: Molecular Structure Estimation from Cryo-EM Density Volumes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33 (01), 1230–1237.

290. Si, D., Moritz, S.A., Pfab, J., Hou, J., Cao, R., Wang, L., Wu, T., and Cheng, J. (2020) Deep Learning to Predict Protein Backbone Structure from High-Resolution Cryo-EM Density Maps. *Sci. Rep.*, 10 (1), 4282.

291. Matsumoto, S., Ishida, S., Araki, M., Kato, T., Terayama, K., and Okuno, Y. (2021) Extraction of protein dynamics information from cryo-EM maps using deep learning. *Nat. Mach. Intell.*, 3 (2), 153–160.

292. Chen, M., Dai, W., Sun, S.Y., Jonasch, D., He, C.Y., Schmid, M.F., Chiu, W., and Ludtke, S.J. (2017) Convolutional neural networks for automated annotation of cellular cryo-electron tomograms. *Nat. Methods*, 14 (10), 983–985.

293. Zeng, X., and Xu, M. (2020) AITom: Open-source AI platform for cryo-electron tomography data analysis. *arXiv*, 1911.03044.

294. Zeng, T., Wu, B., and Ji, S. (2017) DeepEM3D: Approaching human-level performance on 3D anisotropic EM image segmentation. *Bioinformatics*, 33 (16), 2555–2562.

295. Haberl, M.G., Churas, C., Tindall, L., Boassa, D., Phan, S., Bushong, E.A., Madany, M., Akay, R., Deerinck, T.J., Peltier, S.T., and Ellisman, M.H. (2018) CDeep3M—Plug-and-Play cloud-based deep learning for image segmentation. *Nat. Methods*, 15 (9), 677–680.

296. Quan, T.M., Hildebrand, D.G.C., and Jeong, W.-K. (2021) FusionNet: A Deep Fully Residual Convolutional Neural Network for Image Segmentation in Connectomics. *Front. Comput. Sci.*, 3, 613981.

297. Ito, E., Sato, T., Sano, D., Utagawa, E., and Kato, T. (2018) Virus Particle Detection by Convolutional Neural Network in Transmission Electron Microscopy Images. *Food Environ. Virol.*, 10 (2), 201–208.

298. Devan, K.S., Walther, P., Einem, J. von, Ropinski, T., Kestler, H.A., and Read, C. (2019) Detection of herpesvirus capsids in transmission electron microscopy images using transfer learning. *Histochem. Cell Biol.*, 151 (2), 101–114.

299. Devan, K.S., Walther, P., Einem, J. von, Ropinski, T., Kestler, H., and Read, C. (2021) Improved automatic detection of herpesvirus secondary envelopment stages in electron microscopy by augmenting training data with synthetic labelled images generated by a generative adversarial network. *Cell. Microbiol.*, 23 (2), e13280.

300. Zhang, L., and Yan, W.Q. (2020) Deep Learning Methods for Virus Identification from Digital Images. *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 1–6.

301. Xiao, C., Chen, X., Xie, Q., Li, G., Xiao, H., Song, J., and Han, H. (2021) Virus identification in electron microscopy images by residual mixed attention network. *Comput. Methods Programs Biomed.*, 198, 105766.

302. Chamier, L. von, Laine, R.F., Jukkala, J., Spahn, C., Krentzel, D., Nehme, E., Lerche, M., Hernández-Pérez, S., Mattila, P.K., Karinou, E., Holden, S., Solak, A.C., Krull, A., Buchholz, T.-O., Jones, M.L., Royer, L.A., Leterrier, C., Shechtman, Y., Jug, F., Heilemann, M., Jacquemet, G., and Henriques, R. (2021) Democratising deep learning for microscopy with ZeroCostDL4Mic. *Nat. Commun.*, 12 (1), 2276.

303. Marks, L.D. (1994) Experimental studies of small particle structures. *Rep. Prog. Phys.*, 57 (6), 603.

304. Breiman, L. (2001) Random Forests. *Mach. Learn.*, 45 (1), 5–32.

305. Saaim, K.M., Afridi, S.K., Nisar, M., and Islam, S. (2022) In search of best automated model: Explaining nanoparticle TEM image segmentation. *Ultramicroscopy*, 233, 113437.

306. Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017) Focal Loss for Dense Object Detection. *arXiv*, 1708.02002.

307. Oxley, M.P., Yin, J., Borodinov, N., Somnath, S., Ziatdinov, M., Lupini, A., Jesse, S., Vasudevan, R.K., and Kalinin, S.V. (2020) Deep learning of interface structures from simulated 4D STEM data: cation intermixing vs. roughening. *Mach Learn Sci Technology*.

308. Rojo, J. (2018) Machine Learning tools for global PDF fits. *arXiv*, 1809.04392.

309. Gao, J., Harland-Lang, L., and Rojo, J. (2017) The Structure of the Proton in the LHC Precision Era. *arXiv*, 1709.04922.

310. Zhou, Z., Siddiquee, M.M.R., Tajbakhsh, N., and Liang, J. (2018) UNet++: A Nested U-Net Architecture for Medical Image Segmentation. *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support. DLMIA 2018, ML-CDS 2018*, 11045, 3–11.

311. Çiçek, Ö., Abdulkadir, A., Lienkamp, S.S., Brox, T., and Ronneberger, O. (2016) 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, 9901, 424–432.

312. Maiden, A.M., and Rodenburg, J.M. (2009) An improved ptychographical phase retrieval algorithm for diffractive imaging. *Ultramicroscopy*, 109 (10), 1256–1262.

313. Lehtinen, J., Munkberg, J., Hasselgren, J., Laine, S., Karras, T., Aittala, M., and Aila, T. (2015) Noise2Noise: Learning Image Restoration without Clean Data. *Proceedings of the 35th International Conference on Machine Learning, PMLR*, 80, 2965–2974.

314. Batson, J., and Royer, L. (2019) Noise2Self: Blind Denoising by Self-Supervision. *Proceedings of the 36th International Conference on Machine Learning, PMLR*, 97, 524–533.

315. Krull, A., Buchholz, T.-O., and Jug, F. (2019) Noise2Void - Learning Denoising from Single Noisy Images. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2124–2132.

316. Laine, S., Karras, T., Lehtinen, J., and Aila, T. (2019) High-Quality Self-Supervised Deep Image Denoising. *Advances in Neural Information Processing Systems (NeurIPS 2019)*, 32, 6970–6980.

317. Krull, A., Vičar, T., Prakash, M., Lalit, M., and Jug, F. (2020) Probabilistic Noise2Void: Unsupervised Content-Aware Denoising. *Front. Comput. Sci.*, 2, 5.

318. Prakash, M., Lalit, M., Tomancak, P., Krull, A., and Jug, F. (2020) Fully Unsupervised Probabilistic Noise2Void. *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, 154–158.

319. Ede, J.M. (2020) Warwick electron microscopy datasets. *Mach. Learn.: Sci. Technol.*, 1 (4), 045003.

320. Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017) Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv*, 1706.05587.

321. Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018) Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. *Computer Vision – ECCV 2018*, 11211, 833–851.

322. Trampert, P., Bourghorbel, F., Potocek, P., Peemen, M., Schlinkmann, C., Dahmen, T., and Slusallek, P. (2018) How should a fixed budget of dwell time be spent in scanning electron microscopy to optimize image quality? *Ultramicroscopy*, 191, 11–17.

323. Stevens, A., Yang, H., Carin, L., Arslan, I., and Browning, N.D. (2014) The potential for Bayesian compressive sensing to significantly reduce electron dose in high-resolution STEM images. *Microscopy*, 63 (1), 41–51.

324. Stevens, A., Luzi, L., Yang, H., Kovarik, L., Mehdi, B.L., Liyu, A., Gehm, M.E., and Browning, N.D. (2018) A sub-sampled approach to extremely low-dose STEM. *Appl. Phys. Lett.*, 112 (4), 043104.

325. Rotunno, E., Tavabi, A.H., Rosi, P., Frabboni, S., Tiemeijer, P., Dunin-Borkowski, R.E., and Grillo, V. (2021) Alignment of electron optical beam shaping elements using a convolutional neural network. *Ultramicroscopy*, 228, 113338.

326. Kühlbrandt, W. (2014) The Resolution Revolution. *Science*, 343 (6178), 1443–1444.

327. Cheng, Y. (2015) Single-Particle Cryo-EM at Crystallographic Resolution. *Cell*, 161 (3), 450–457.

328. Cheng, Y. (2018) Single-particle cryo-EM - how did it get here and where will it go. *Science*, 361 (6405), 876–880.

329. Frank, J. (1989) Image analysis of single macromolecules. *Electron. Microsc. Rev.*, 2 (1), 53–74.

330. Sigworth, F.J. (2016) Principles of cryo-EM single-particle image processing. *Microscopy*, 65 (1), 57–67.

331. Ogura, T., and Sato, C. (2004) Auto-accumulation method using simulated annealing enables fully automatic particle pickup completely free from a matching template or learning data. *J. Struct. Biol.*, 146 (3), 344–358.

332. Jiménez-Moreno, A., Střelák, D., Filipovič, J., Carazo, J.M., and Sorzano, C.O.S. (2021) DeepAlign, a 3D alignment method based on regionalized deep learning for Cryo-EM. *J. Struct. Biol.*, 213 (2), 107712.

333. Ramírez-Aportela, E., Mota, J., Conesa, P., Carazo, J.M., and Sorzano, C.O.S. (2019) DeepRes: a new deep-learning- and aspect-based local resolution method for electron-microscopy maps. *IUCrJ*, 6, 1054–1063.

334. Zhong, E.D., Bepler, T., Berger, B., and Davis, J.H. (2021) CryoDRGN: reconstruction of heterogeneous cryo-EM structures using neural networks. *Nat. Methods*, 18 (2), 176–185.

335. Malhotra, S., Träger, S., Peraro, M.D., and Topf, M. (2019) Modelling structures in cryo-EM maps. *Curr Opin Struc Biol*, 58, 105–114.

336. Kim, D.N., Gront, D., and Sanbonmatsu, K.Y. (2020) Practical Considerations for Atomistic Structure Modeling with Cryo-EM Maps. *J. Chem. Inf. Model.*, 60 (5), 2436–2442.

337. Arganda-Carreras, I., Turaga, S.C., Berger, D.R., Cireşan, D., Giusti, A., Gambardella, L.M., Schmidhuber, J., Laptev, D., Dwivedi, S., Buhmann, J.M., Liu, T., Seyedhosseini, M., Tasdizen, T., Kamentsky, L., Burget, R., Uher, V., Tan, X., Sun, C., Pham, T.D., Bas, E., Uzunbas, M.G., Cardona, A., Schindelin, J., and Seung, H.S. (2015) Crowdsourcing the creation of image segmentation algorithms for connectomics. *Front. Neuroanat.*, 9, 142.

338. Dorkenwald, S., McKellar, C., Macrina, T., Kemnitz, N., Lee, K., Lu, R., Wu, J., Popovych, S., Mitchell, E., Nehoran, B., Jia, Z., Bae, J.A., Mu, S., Ih, D., Castro, M., Ogedengbe, O., Halageri, A., Ashwood, Z., Zung, J., Brittain, D., Collman, F., Schneider-Mizell, C., Jordan, C., Silversmith, W., Baker, C., Deutsch, D., Encarnacion-Rivera, L., Kumar, S., Burke, A., Gager, J., Hebditch, J., Koolman, S., Moore, M., Morejohn, S., Silverman, B., Willie, K., Willie, R., Yu, S., Murthy, M., and Seung, H.S. (2020) FlyWire: Online community for whole-brain connectomics. *bioRxiv*, 2020.08.30.274225.

339. Ishii, S., Lee, S., Urakubo, H., Kume, H., and Kasai, H. (2020) Generative and discriminative model-based approaches to microscopic image restoration and segmentation. *Microscopy*, 69 (2), 79–91.

340. Campbell, C.T. (2013) The Energetics of Supported Metal Nanoparticles: Relationships to Sintering Rates and Catalytic Activity. *Acc. Chem. Res.*, 46 (8), 1712–1719.

341. Fechete, I., Wang, Y., and Védrine, J.C. (2012) The past, present and future of heterogeneous catalysis. *Catal. Today*, 189 (1), 2–27.

342. Liang, J., Liang, Z., Zou, R., and Zhao, Y. (2017) Heterogeneous Catalysis in Zeolites, Mesoporous Silica, and Metal–Organic Frameworks. *Adv. Mater.*, 29 (30), 1701139.

343. Védrine, J. (2017) Heterogeneous Catalysis on Metal Oxides. *Catalysts*, 7 (11), 341.

344. Chen, F., Jiang, X., Zhang, L., Lang, R., and Qiao, B. (2018) Single-atom catalysis: Bridging the homo- and heterogeneous catalysis. *Chinese J. Catal.*, 39 (5), 893–898.

345. Gates, B.C. (2019) Atomically Dispersed Supported Metal Catalysts: Seeing Is Believing. *Trends Chem.*, 1 (1), 99–110.

346. Schlögl, R., and Hamid, S.B.A. (2004) Nanocatalysis: Mature Science Revisited or Something Really New? *Angew. Chem. Int. Ed.*, 43 (13), 1628–1637.

347. Ouyang, G., Wang, C.X., and Yang, G.W. (2009) Surface Energy of Nanostructural Materials with Negative Curvature and Related Size Effects. *Chem. Rev.*, 109 (9), 4221–4247.

348. Vert, M., Doi, Y., Hellwich, K.-H., Hess, M., Hodge, P., Kubisa, P., Rinaudo, M., and Schué, F. (2012) Terminology for biorelated polymers and applications (IUPAC Recommendations 2012). *Pure Appl. Chem.*, 84 (2), 377–410.

349. Volokitin, Y., Sinzig, J., Jongh, L.J. de, Schmid, G., Vargaftik, M.N., and Moiseevi, I.I. (1996) Quantum-size effects in the thermodynamic properties of metallic nanoparticles. *Nature*, 384 (6610), 621–623.

350. Buffat, Ph., and Borel, J.-P. (1976) Size effect on the melting temperature of gold particles. *Phys. Rev. A.*, 13 (6), 2287–2298.

351. Armbrüster, M., Schlögl, R., and Grin, Y. (2014) Intermetallic compounds in heterogeneous catalysis—a quickly developing field. *Sci. Technol. Adv. Mat.*, 15 (3), 034803.

352. Ndolomingo, M.J., Bingwa, N., and Meijboom, R. (2020) Review of supported metal nanoparticles: synthesis methodologies, advantages and application as catalysts. *J. Mater. Sci.*, 55 (15), 6195–6241.

353. Jamkhande, P.G., Ghule, N.W., Bamer, A.H., and Kalaskar, M.G. (2019) Metal nanoparticles synthesis: An overview on methods of preparation, advantages and disadvantages, and applications. *J. Drug. Deliv. Sci. Tec.*, 53, 101174.

354. Campelo, J.M., Luna, D., Luque, R., Marinas, J.M., and Romero, A.A. (2009) Sustainable Preparation of Supported Metal Nanoparticles and Their Applications in Catalysis. *ChemSusChem*, 2 (1), 18–45.

355. White, R.J., Luque, R., Budarin, V.L., Clark, J.H., and Macquarrie, D.J. (2009) Supported metal nanoparticles on porous materials. Methods and applications. *Chem. Soc. Rev.*, 38 (2), 481–494.

356. Astruc, D. (2020) Introduction: Nanoparticles in Catalysis. *Chem. Rev.*, 120 (2), 461–463.

357. Fan, J., Cheng, Y., and Sun, M. (2020) Functionalized Gold Nanoparticles: Synthesis, Properties and Biomedical Applications. *Chem. Rec.*, 20 (12), 1474–1504.

358. Liang, S., Jasinski, J., Hammond, G.B., and Xu, B. (2015) Supported Gold Nanoparticle-Catalyzed Hydration of Alkynes under Basic Conditions. *Org. Lett.*, 17 (1), 162–165.

359. Sharma, A.S., Kaur, H., and Shah, D. (2016) Selective oxidation of alcohols by supported gold nanoparticles: recent advances. *RSC Adv.*, 6 (34), 28688–28727.

360. He, Y., Liu, J.-C., Luo, L., Wang, Y.-G., Zhu, J., Du, Y., Li, J., Mao, S.X., and Wang, C. (2018) Size-dependent dynamic structures of supported gold nanoparticles in CO oxidation reaction condition. *PNAS*, 115 (30), 7700–7705.

361. Astruc, D. (2007) Palladium Nanoparticles as Efficient Green Homogeneous and Heterogeneous Carbon−Carbon Coupling Precatalysts: A Unifying View. *Inorg. Chem.*, 46 (6), 1884–1894.

362. Rivera-Cárcamo, C., Gerber, I.C., Rosal, I. del, Guicheret, B., Contreras, R.C., Vanoye, L., Favre-Réguillon, A., Machado, B.F., Audevard, J., Bellefon, C. de, Philippe, R., and Serp, P. (2021) Control of the single atom/nanoparticle ratio in Pd/C catalysts to optimize the cooperative hydrogenation of alkenes. *Catal. Sci. Technol.*, 11 (3), 984–999.

363. Ament, K., Wagner, D.R., Götsch, T., Kikuchi, T., Kröhnert, J., Trunschke, A., Lunkenbein, T., Sasaki, T., and Breu, J. (2014) Enhancing the Catalytic Activity of Palladium Nanoparticles via Sandwich-Like Confinement by Thin Titanate Nanosheets. *ACS Catal.*, 11 (5), 2754–2762.

364. Cao, Y., Ran, R., Wu, X., Si, Z., Kang, F., and Weng, D. (2023) Progress on metal-support interactions in Pd-based catalysts for automobile emission control. *J. Environ. Sci.*, 125, 401–426.

365. Weckhuysen, B.M. (2010) Preface: recent advances in the in-situ characterization of heterogeneous catalysts. *Chem. Soc. Rev.*, 39 (12), 4557–4559.

366. Wang, L., Wang, L., Meng, X., and Xiao, F. (2019) New Strategies for the Preparation of Sinter-Resistant Metal-Nanoparticle-Based Catalysts. *Adv. Mater.*, 31 (50), 1901905.

367. Voorhees, P.W. (1985) The theory of Ostwald ripening. *J. Stat. Phys.*, 38 (1–2), 231–252.

368. Westen, T. van, and Groot, R.D. (2018) Effect of Temperature Cycling on Ostwald Ripening. *Cryst. Growth Des.*, 18 (9), 4952–4962.

369. Asoro, M.A., Kovar, D., and Ferreira, P.J. (2013) In situ Transmission Electron Microscopy Observations of Sublimation in Silver Nanoparticles. *ACS Nano*, 7 (9), 7844–7852.

370. Reimer, L., and Kohl, H. (2008) Electron Guns, in *Transmission Electron Microscopy*, vol. 5, Springer Science+Business Media, LLC, Springer Science+Business M|edia, LLC, 223 Spring Street, New York, NY, 10013, USA.

371. Williams, D.B., and Carter, C.B. (2016) Electron Sources, in *Transmission Electron Microscopy* (eds.Williams, D.B., and Carter, C.B.), Springer International Publishing Switzerland, Springer-Verlag GmbH Berlin Heidelberg, pp. 1–15.

372. Botton, G., and Prabhudev, S. (2019) Analytical Electron Microscopy, in *Springer Handbook of Microscopy* (eds.Hawkes, P.W., and Spence, J.C.H.), Springer Nature Switzerland, Gewerbestrasse 11, 6630 Cham., Switzerland, pp. 345–444.

373. Feist, A., Bach, N., Silva, N.R. da, Danz, T., Möller, M., Priebe, K.E., Domröse, T., Gatzmann, J.G., Rost, S., Schauss, J., Strauch, S., Bormann, R., Sivis, M., Schäfer, S., and Ropers, C. (2016) Ultrafast transmission electron microscopy using a laser-driven field emitter: Femtosecond resolution with a high coherence electron beam. *Ultramicroscopy*, 176, 63–73.

374. Browning, N.D., Bonds, M.A., Campbell, G.H., Evans, J.E., LaGrange, T., Jungjohann, K.L., Masiel, D.J., McKeown, J., Mehraeen, S., Reed, B.W., and Santala, M. (2012) Recent

developments in dynamic transmission electron microscopy. *Curr. Opin. Solid. State Mater. Sci.*, 16 (1), 23–30.

375. Kuwahara, M., Nambo, Y., Aoki, K., Sameshima, K., Jin, X., Ujihara, T., Asano, H., Saitoh, K., Takeda, Y., and Tanaka, N. (2016) The Boersch effect in a picosecond pulsed electron beam emitted from a semiconductor photocathode. *Appl. Phys. Lett.*, 109 (1), 013108.

376. Kirkland, A.I., Chang, S.L.-J., and Hutchison, J.L. (2019) Atomic Resolution Transmission Electron Microscopy, in *Springer Handbook of Microscopy*, vol. 1, Springer Cham., pp. 3–47.

377. Typke, D., and Dierksen, K. (1995) Determination of image aberrations in high-resolution electron microscopy using diffractogram and cross-correlation methods. *Optik*, 4 (99), 155–166.

378. Kirkland, E.J. (2020) *Advanced Computing in Electron Microscopy*, Springer Nature Switzerland AG, Gewerbestrasse 11, 6330 Cham, Switzerland.

379. Cowley, J.M., and Moodie, A.F. (1957) The scattering of electrons by atoms and crystals. I. A new theoretical approach. *Acta Cryst.*, 10 (10), 609–619.

380. Lobato, I., and Dyck, D.V. (2015) A complete comparison of simulated electron diffraction patterns using different parameterizations of the electron scattering factors. *Ultramicroscopy*, 155, 11–19.

381. Goodfellow, I., Bengio, Y., and Courville, A. (2016) *Deep Learning*, MIT Press, Cambridge, Massachusetts.

382. Géron, A. (2019) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

383. Aggarwal, C.C. (2018) *Neural Networks and Deep Learning*, Springer Cham, Gewerbestrasse 11, 6330 Cham, Switzerland.

384. Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1987) Learning Internal Representations by Error Propagation, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, MIT Press, pp. 318–362.

385. Glorot, X., and Bengio, Y. (2010) Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 9, 249–256.

386. Sawada, H., Sannomiya, T., Hosokawa, F., Nakamichi, T., Kaneyama, T., Tomita, T., Kondo, Y., Tanaka, T., Oshima, Y., Tanishiro, Y., and Takayanagi, K. (2008) Measurement method of aberration from Ronchigram by autocorrelation function. *Ultramicroscopy*, 108 (11), 1467–1475.

387. Shiojiri, M., Saito, Y., Okada, H., and Sasaki, H. (1979) Densities of Amorphous Thin Films. *Jpn J. Appl. Phys.*, 18 (10), 1931.

388. Wypych, G. (2014) 3.2.1 Carbon black, in *Databook of Antistatics*, ChemTec Publishing, pp. 245–271.

389. Lobato, I., and Dyck, D.V. (2015) MULTEM: A new multislice program to perform accurate and fast electron diffraction and imaging simulations using Graphics Processing Units with CUDA. *Ultramicroscopy*, 156, 9–17.

390. Lobato, I., Aert, S.V., and Verbeeck, J. (2016) Progress and new advances in simulating electron microscopy datasets using MULTEM. *Ultramicroscopy*, 168, 17–27.

391. Jocher, G. (2020) YOLOv5. *YOLOv5, https://github.com/ultralytics/yolov5, Access Date: 25.07.2022*.

392. Redmon, J., and Farhadi, A. (2018) YOLOv3: An Incremental Improvement. *arXiv*, 1804.02767.

393. Redmon, J., and Farhadi, A. (2017) YOLO9000: Better, Faster, Stronger. *Computer Vision and Pattern Recognition (CVPR)*, 6517–6525.

394. Badrinarayanan, V., Kendall, A., and Cipolla, R. (2016) SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *arXiv*.

395. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J., and Zisserman, A. (2010) The Pascal Visual Object Classes (VOC) Challenge. *Int J Comput Vision*, 88 (2), 303–338.

396. Gontard, L.C., Ozkaya, D., and Dunin-Borkowski, R.E. (2010) A simple algorithm for measuring particle size distributions on an uneven background from TEM images. *Ultramicroscopy*, 111 (2), 101–106.

397. Park, C., Huang, J.Z., Huitink, D., Kundu, S., Mallick, B.K., Liang, H., and Ding, Y. (2011) A multistage, semi-automated procedure for analyzing the morphology of nanoparticles. *IIE Transactions*, 44 (7), 507–522.

398. Mondini, S., Ferretti, A.M., Puglisi, A., and Ponti, A. (2012) Pebbles and PebbleJuggler : software for accurate, unbiased, and fast measurement and analysis of nanoparticle morphology from transmission electron microscopy (TEM) micrographs. *Nanoscale*, 4 (17), 5356–5372.

399. Park, C., Huang, J.Z., Ji, J.X., and Ding, Y. (2013) Segmentation, Inference and Classification of Partially Overlapping Nanoparticles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (3), 1–1.

400. Epicier, T., Sato, K., Tournus, F., and Konno, T. (2012) Chemical composition dispersion in bi-metallic nanoparticles: semi-automated analysis using HAADF-STEM. *J. Nanopart. Res.*, 14 (9), 1106.

401. Groom, D.J., Yu, K., Rasouli, S., Polarinakis, J., Bovik, A.C., and Ferreira, P.J. (2018) Automatic segmentation of inorganic nanoparticles in BF TEM micrographs. *Ultramicroscopy*, 194, 25–34.

402. Uusimaeki, T., Wagner, T., Lipinski, H.-G., and Kaegi, R. (2019) AutoEM: a software for automated acquisition and analysis of nanoparticles. *J. Nanopart. Res.*, 21 (6), 122.

403. Crozier, P.A., and Hansen, T.W. (2015) In situ and operando transmission electron microscopy of catalytic materials. *MRS Bull.*, 40 (1), 38–45.

404. Kumar, S., Varma, R.S., Zboril, R., and Gawande, M.B. (2019) Support Morphology-dependent Activity of Nanocatalysts, in *Nanoparticle Design and Characterization for Catalytic Applications in Sustainable Chemistry*, The Royal Society of Chemistry 2019, pp. 84–114.

405. Jones, L., MacArthur, K.E., Fauske, V.T., Helvoort, A.T.J. van, and Nellist, P.D. (2014) Rapid Estimation of Catalyst Nanoparticle Morphology and Atomic-Coordination by High-Resolution Z-Contrast Electron Microscopy. *Nano Lett.*, 14 (11), 6336–6341.

406. Michen, B., Geers, C., Vanhecke, D., Endes, C., Rothen-Rutishauser, B., Balog, S., and Petri-Fink, A. (2015) Avoiding drying-artifacts in transmission electron microscopy: Characterizing the size and colloidal state of nanoparticles. *Sci. Rep.*, 5 (1), 9793.

407. Zhao, Y., Tang, R., and Huang, R. (2015) Palladium Supported on Graphitic Carbon Nitride: An Efficient and Recyclable Heterogeneous Catalyst for Reduction of Nitroarenes and Suzuki Coupling Reaction. *Catal. Lett.*, 145 (11), 1961–1971.

408. Ward, M.R., Theobald, B., Sharman, J., Boyes, E.D., and Gai, P.L. (2017) Direct observations of dynamic PtCo interactions in fuel cell catalyst precursors at the atomic level using E(S)TEM. *J. Microsc.*, 269 (2), 143–150.

409. He, L., Ren, Y., Fu, Y., Yue, B., Tsang, S.C.E., and He, H. (2019) Morphology-Dependent Catalytic Activity of Ru/CeO2 in Dry Reforming of Methane. *Molecules*, 24 (3), 526.

410. Zafari, S., Eerola, T., Ferreira, P., Kälviäinen, H., and Bovik, A. (2019) Automated Segmentation of Nanoparticles in BF TEM Images by U-Net Binarization and Branch and Bound. *Computer Analysis of Images and Patterns - CAIP 2019*, 113–125.

411. Colliard-Granero, A., Batool, M., Jankovic, J., Jitsev, J., Eikerling, M.H., Malek, K., and Eslamibidgoli, M.J. (2021) Deep learning for the automation of particle analysis in catalyst layers for polymer electrolyte fuel cells. *Nanoscale*.

412. Zhang, F., Zhao, D., Xiao, Z., Wu, J., Geng, L., Wang, W., and Liu, Y. (2021) Rodlike nanoparticle parameter measurement method based on improved Mask R-CNN segmentation. *SIViP*, 15, 579–587.

413. He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017) Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2961–2969.

414. Frei, M., and Kruis, F.E. (2020) Image-based size analysis of agglomerated and partially sintered particles via convolutional neural networks. *Powder Technol.*, 360, 324–336.

415. Otsu, N. (1979) A Threshold Selection Method from Gray-Level Histograms. *IEEE Trans. Syst. Man Cybern. Syst.*, 9 (1), 62–66.

416. ePSIC ParticleSpy. *https://zenodo.org/record/5094360, Access Date: 17.12.2021*.

417. Bell, C.G., Treder, K.P., Kim, J.S., Schuster, M.E., Kirkland, A.I., and Slater, T.J.A. (2022) Trainable segmentation for transmission electron microscope images of inorganic nanoparticles. *J. Microsc.*

418. Scott, D.W. (2015) Multivariate Density Estimation: Theory, Practice, and Visualization, in *Chapter 6 Kernel Density Estimators* (eds.Scott, D.W.), John Wiley & Sons, Incorporated, pp. 137 – 216.

419. (2022) SciPy Documentation: Gaussian KDE. *SciPy Documentation: Gaussian KDE*.

420. Chollet, F. (2020) Transfer learning & fine-tuning. *Transfer learning & fine-tuning*.

421. Bellos, D., Basham, M., Pridmore, T., and French, A.P. (2021) A stacked dense denoising–segmentation network for undersampled tomograms and knowledge transfer using synthetic tomograms. *Mach. Vis. Appl.*, 32 (3), 75.

422. Microsoft Visual Object Tagging Tool (VOTT). *Visual Object Tagging Tool (VOTT), https://github.com/microsoft/VoTT, Access Date: 15.08.2022*.

423. Zooniverse.org (2022) About Zooniverse.

424. Darrow, M., Luengo, I., Basham, M., Spink, M., King, O., Pennington, A., Tun, W., and Spiers, H. Science Scribbler Project. *https://www.zooniverse.org/projects/msbrhonclif/science-scribbler, Access Date: 06.10.2022*.

425. Darrow, M., Basham, M., King, O., Pennington, A., Boyce, M., Hecksel, C., and Spiers, H. Science Scribbler: Virus Factory. *https://www.zooniverse.org/projects/markbasham/science-scribbler-virus-factory/about/team, Access Date: 06.10.2022*.

426. Desforges, M., Jones, C., Chernyavsky, I., Lewis, R., Tun, W., Darrow, M., and Basham, M. Science Scribbler: Placenta Profiles. *https://www.zooniverse.org/projects/msbrhonclif/science-scribbler-placenta-profiles, Access Date: 06.10.2022*.

427. Campello, R.J.G.B., Moulavi, D., and Sander, J. (2013) Density-Based Clustering Based on Hierarchical Density Estimates. *PAKDD 2013: Advances in Knowledge Discovery and Data Mining*, 7819, 160–172.

428. Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 226–231.

429. McInnes, L., Healy, J., and Astels, S. (2016) HDBSCAN Documentation. *https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html*, *Access Date: 10.09.2022*.

430. Gatan Inc. GMS 3. *https://www.gatan.com/products/tem-analysis/gatan-microscopy-suite-software*, *Access Date: 06.10.2022*.

431. Fielding, R.T. (2000) Architectural Styles and the Design of Network-based Software Architectures.

432. Merkel, and Dirk (2014) Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014 (239), 2.

433. Jocher, G. YOLOv5 github repository. *ultralytics/yolov5*.

434. Oudheusden, T. van, Jong, E.F. de, Geer, S.B. van der, Root, W.P.E.M.O. 't, Luiten, O.J., and Siwick, B.J. (2007) Electron source concept for single-shot sub-100 fs electron diffraction in the 100 keV range. *J. Appl. Phys.*, 102 (9), 093501.

435. Tonomura, A. (1987) Applications of electron holography. *Rev. Mod. Phys.*, 59 (3), 639–669.

436. DENS Solutions wildfire nanochip product page. *https://denssolutions.com/products/wildfire/nano-chip/*, *Access Date: 06.10.2022*.

10 Appendix

Herein, all relevant supplementary information relevant to the work is deposited. The content is organised in sections that correspond to respective chapters and sections in the main text.

10.1 Additions to Theoretical Considerations

This section introduces theoretical concepts that are of equal importance as concepts discussed in the main text, but are only introduces briefly for completeness sake.

10.1.1 Pulsed Electron Emission

Besides the core characterisation measures of pulsed electron sources discussed, two additional properties shall be explicitly highlighted in the following.

10.1.1.1 Transverse Coherence

The transverse (lateral) coherence describes the electron bunch coherence in directions perpendicular to the optical axis. It is given by [434]:

$$L_{\perp} \equiv \frac{\lambda}{2\pi\sigma_{\theta}} \tag{10.1}$$

where $L_{\perp}$ refers to the transversal coherence, λ to the De Broglie wavelength, and σ_{θ} to the root-mean-square (r.m.s) of the angular spread. Additionally, the more general property of transverse normalised r.m.s emittance for a focused beam can be considered [373]:

$$\varepsilon_{n,r.m.s.,x} = \beta\gamma\sigma_x\sigma_{\theta x} \tag{10.2}$$

in this case, $\beta = v/c$ refers to the ratio between the speed of the electrons (v) and the speed of light (c), γ to the Lorentz factor, σ_x to the r.m.s. spread in x-direction and $\sigma_{\theta x}$ to the $x\theta$-component of r.m.s. angular spread. Combining Eq. 10.1 and Eq. 10.2 yields [373,434]:

$$L_{\perp} = \frac{\hbar}{mc} \cdot \frac{\sigma_x}{\varepsilon_{n,r.m.s.,x}} \tag{10.3}$$

with $\hbar$ as the Planck's constant divided by 2π and m as the electron mass.

10.1.1.2 Longitudinal Coherence

Similar importance is given for the longitudinal coherence length, which acts in the direction parallel to the optical beam axis and is given by [435]:

$$L_{\parallel} = \frac{\lambda^2}{\Delta\lambda} = \frac{2E}{\Delta E} \cdot \lambda \tag{10.4}$$

in which E is the kinetic energy of the electrons and ΔE is the energy spread.

10.2 Sample Model Generation

This section provides supplementary information for Section '4.2.1 Sample Model Generation' in the main text. Note that the formatting might vary slightly, as individual code lines were extracted as a metafile to be compatible with the presented document.

Code 10.1: Mathematica 12.1 notebook for computational Au/Ge sample models.

Full Au/Ge Model Builder V 1.2

A notebook generating Au/Ge models as input for MULTEM simulations.
Copyright: Kevin P Treder, Department of Materials, University of Oxford
Author: Kevin P Treder

Foreword

This notebook shall provide the full code for building models containing Au nanoparticles supported on amorphous Ge with relating explanations to used functions. The author claims the copyright of this notebook and as well as the concepts and ideas introduced to generate the resulting output. This notebook does not claim to work other purposes than generating models Au/Ge NP(nanoparticle) models. Furthermore, it is subject to ongoing optimization and development. All distances used in this notebook have to be given in Angstrom.

The base

The following functions are used to generate a list of {x,y,z} - coordinates of the amorphous Ge. The following function converts a density into an average distance between two atoms.

```
In[•]:= MeanDist[density_] := Block[{densitym, Numberm, AtDist},
          densitym = density * 10^6;
          Numberm = Surd[densitym / (12.0107 * 1.66054 * (10^(-24))), 3];
          AtDist = 1 / Numberm
         ]
```

Using MeanDist[], a rectangle as well as a box with the same projected size in the x-y- plane are generated. Using 'num' the box is filled with a random number of Ge atoms. The according densities found in GeDens were taken from Shiojiri et al., Jap. J. Appl. Phys., 18, 10, 1931-1936, 1979.

```
In[•]:= Layer[w_, h_] := Block[{GeDens, poly, box, num, gels},
          GeDens = MeanDist[RandomVariate[NormalDistribution[3.96, 0.08]]];
          poly = Region[Rectangle[{0, 0}, {w, w}]];
          box = Region[Cuboid[{0, 0, 0}, {w, w, h}]];
          num = Ceiling[w * w * h / (GeDens * 10^10)^3, 10];
          gels = RandomPoint[box, num];
          {gels, poly, box}]

In[•]:= AbsoluteTiming[GeSupport = Layer[1700, 15];]

Out[•]= {0.278777, Null}
```

Dropping NPs

The next step is to generate a number of realistic Au nanoparticles. Shapes that were observed in initial experimental data, mainly exhibited circular & elliptic particles as well as some particle structures that consisted of multiple particles due to sintering. The latter ones will be named "chains" in this notebook. In the following, chains will be generated by fusing multiple NPs together. Chains are subsequently "dropped" on the support material and occupy space that subsequently dropped NPs cannot occupy.

Crystal basis

```
In[•]:= FCCbasis = 0.5 * LatticeData["FaceCenteredCubic", "Basis"];

In[•]:= dAu = 4.0782; (*Au lattice constant in Angstrom*)
```

Generate an 'infinite' crystal from which NPs can be 'cut out'

```
In[•]:= Timing[FCCLattice = Tuples[Range[-100, 100], 3].FCCbasis;]

Out[•]= {2.12665, Null}

In[•]:= Timing[Audat = dAu * FCCLattice;]

Out[•]= {0.185763, Null}
```

Sizing functions and distribution functions

LongAxisDistNum[] allows particles to be in between 1 and 1.7 nm. It basically serves as sizing function.

```
In[•]:= LongAxisDistNum[] := Module[{s1},
          s1 = Abs[Round[RandomVariate[NormalDistribution[9.879, 3.074]], 0.01]] * 10;
          If[Between[s1, {10, 170}], s1, LongAxisDistNum[]]]
```

The gives a defined distribution for the aspect ratio between short and long axis of elliptical particles. The values were measured on experimental data.

```
In[•]:= AspectRatioGen[] := Module[{s1},
          s1 = Abs[Round[RandomVariate[NormalDistribution[1.116, 0.10057]], 0.01]];
          If[Between[s1, {1, 1.6}], s1, AspectRatioGen[]]]
```

NonZeroRandomInteger[] explains itself.

```
In[•]:= NonZeroRandomInteger[range_, dims_] := Module[{original}, original = RandomInteger[range, dims];
          If[Norm[original] == 0, NonZeroRandomInteger[range, dims], original]]
```

Initialize shape functions

The following functions allow the generation of different shapes. Although never seen in experimental data, cuboctahedra, icosahedra and decahedra are included as well for the sake of generalization abilities. However, those shapes will receive relatively low weights in the end distribution so they do not appear too often.

Cuboctahedron:

```
In[•]:= NPc8[size_] := Module[{shell, s = size, rotad, ls, rotm},
          rotad = RotationTransform[{{1, 0, 0}, {1, 1, 0}}];
          shell = PolyhedronData["Cuboctahedron", "BoundaryMeshRegion"];
          shell = Scale[shell, (s / 2)];
          shell = TransformedRegion[shell, rotad];
          ls = Pick[Audat, RegionMember[shell, Audat]];
          rotm = RotationTransform[Round[RandomReal[{0, 359}], 0.1] Degree, NonZeroRandomInteger[{-100, 100}, 3]];
          ls = rotm[ls];
          shell = TransformedRegion[shell, rotm];
          {ls, shell}]
```

Icosahedron:

```
In[•]:= NPicosa[size_] := Module[{shell, s = size, rotad1, rotad2, ls, rotm},
          rotad1 = RotationTransform[{{0.7694208842938134, -0.25, 0.}, {1, 0, 0}}];
          rotad2 = RotationTransform[30 Degree, {1, 0, 0}];
          (*Certain direction of the lattice have to match up with specific planes of the 'shell'*)
          shell = PolyhedronData["Icosahedron", "BoundaryMeshRegion"];
          shell = Scale[shell, (s / 2)]; (*s/2 the scaling factor is applied in all directions*)
          shell = TransformedRegion[shell, rotad1]; shell = TransformedRegion[shell, rotad2];
          ls = Pick[Audat, RegionMember[shell, Audat]];
          rotm = RotationTransform[Round[RandomReal[{0, 359}], 0.1] Degree, NonZeroRandomInteger[{-100, 100}, 3]];
          ls = rotm[ls];
          shell = TransformedRegion[shell, rotm];
          {ls, shell}]
```

Decahedron:

```
In[•]:= NPd10[size_] := Module[{shell, s = size, rotad1, rotad2, rotad3, ls, rotm},
          rotad1 = RotationTransform[{{{1/2, Sqrt[1/10 (5 - Sqrt[5])], 1/4 (1 + Sqrt[5])}, {1, 1, 0}}}];
          rotad2 = RotationTransform[{{0, 0, 1}, {1, 1, 0}}];
          shell = PolyhedronData["ElongatedPentagonalDipyramid", "BoundaryMeshRegion"];
          shell = Scale[shell, (s / 2)];
          shell = TransformedRegion[shell, rotad1]; shell = TransformedRegion[shell, rotad2];
          ls = Pick[Audat, RegionMember[shell, Audat]];
          rotm = RotationTransform[Round[RandomReal[{0, 359}], 0.1] Degree, NonZeroRandomInteger[{-100, 100}, 3]];
          ls = rotm[ls];
          shell = TransformedRegion[shell, rotm];
          {ls, shell}]
```

Ellipsoids:

```
In[•]:= NPellipsoid[size_] := Module[{s = size, s1, rkill, sctr, s2, ls, rottr, shell},
          s1 = Region[Ellipsoid[{0, 0, 0}, {1, 1, rkill = AspectRatioGen[]}]];
          s2 = Scale[s1, (s / 2)];
          ls = Pick[Audat, RegionMember[s2, Audat]];
          rottr = RotationTransform[Round[RandomReal[{0, 359}], 0.1] Degree, NonZeroRandomInteger[{-100, 100}, 3]];
          ls = rottr[ls];
          shell = TransformedRegion[s2, rottr];
          {ls, shell}]
```

Generate a chain(s) and place them on support

The chains are generated via the 'ConnectedParticles' function which is shown below. This function is not necessarily the fastest, but it reliably constructs NPchains as they occur in experimental images. Some of the input numbers came up due to testing the function multiple times. With the given input, the function seems to find a good compromise between a realistic model, high variability and acceptable computation time.

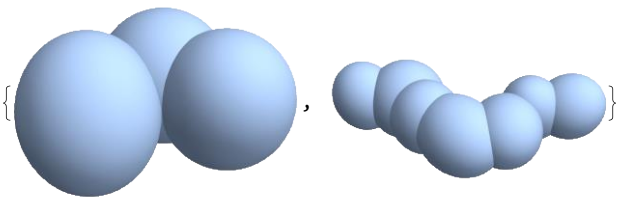
```
In[•]:= ConnectedParticles[ChainNP_] :=
 Module[{counter, dummycounter, SeedNP, SeedNPDummy, nxtNP, nxtNPDummy, AttachNP, sftP, firstrel, allint, rels, allints, plusZ},
  counter = {};
  dummycounter = {};
  SeedNP = NPellipsoid[Round[RandomReal[{85, 115}], 0.01]];
  SeedNPDummy = Region@Cuboid[First /@ RegionBounds[SeedNP[[2]]], Last /@ RegionBounds[SeedNP[[2]]]];
  AppendTo[counter, SeedNP];
  AppendTo[dummycounter, SeedNPDummy];
  (*the lists counter and dummycounter are now going to be built up*)
  While[Length[counter] < ChainNP,
   AttachNP = dummycounter[[-1]];
   nxtNP = NPellipsoid[Round[RandomReal[{85, 115}], 0.01]];
   (*nxtNPDummy=ConvexHullMesh[RandomPoint[nxtNP[[2]],3000]];*)
   nxtNPDummy = Region@Cuboid[First /@ RegionBounds[nxtNP[[2]]], Last /@ RegionBounds[nxtNP[[2]]]];
   sftP = Flatten[{RandomPoint[counter[[-1, 2]]][[1 ;; 2]], 0}] * RandomReal[{0.8, 2.2}];
   (*sftP picks a random point on the actual NP shell, but merely in the x-y plane so it will be easy to 'drop' in on the amorphous Ge*)
   nxtNPDummy = TransformedRegion[nxtNPDummy, TranslationTransform[sftP]];
   nxtNP[[1]] = sftP + # & /@ nxtNP[[1]];
   firstrel = N[Length@Pick[nxtNP[[1]], RegionMember[counter[[-1, 2]], nxtNP[[1]]]] / Length[nxtNP[[1]]]];
   (*How much Volume of new NP is in the old NP*)
   allint = Pick[nxtNP[[1]], RegionMember[#, nxtNP[[1]]]] & /@ counter[[1 ;; -2, 2]];
   (*check how many intersections with other particles there are*)
   rels = N[Length /@ allint / Length[nxtNP[[1]]]];
   (*check how any atoms of the new NP are in other parts of the chain*)
   If[! Between[firstrel, {0.1, 0.4}] ∨ AnyTrue[rels, # > 0.5 &],
    nxtNP = {}; nxtNPDummy = {};,
    allints = Pick[nxtNP[[1]], RegionMember[#, nxtNP[[1]]]] & /@ counter[[All, 2]];
    nxtNP[[1]] = Flatten[Table[nxtNP[[1]] = Complement[nxtNP[[1]], allints[[x]]], {x, 1, Length@allints}], 1];
    nxtNP[[2]] = TransformedRegion[nxtNP[[2]], TranslationTransform[sftP]];
    AppendTo[counter, nxtNP];
    AppendTo[dummycounter, nxtNPDummy];];];
  counter = {Flatten[counter[[All, 1]], 1], counter[[All, 1]], RegionUnion[counter[[All, 2]]]};
  plusZ = Abs[Min[Last[RegionBounds[counter[[3]]]]]];
  counter[[1]] = {0, 0, plusZ} + # & /@ counter[[1]];
  counter[[2]] = Map[({0, 0, plusZ} + # &), counter[[2]], {2}];
  counter[[3]] = TransformedRegion[counter[[3]], TranslationTransform[{0, 0, plusZ}]];
  counter
 ]
```

```
In[•]:= AbsoluteTiming[chaintest = ConnectedParticles[4];]
Out[•]= {31.6518, Null}
```

Now a quick test for appending more than one chain

```
In[•]:= chainlist = {};
In[•]:= AbsoluteTiming[Do[chain = ConnectedParticles[RandomInteger[{2, 7}]]; AppendTo[chainlist, chain], 2];]
Out[•]= {100.335, Null}
In[•]:= chainlist[[All, 3]]
```

Out[•]=



The following function places the chains on the Ge support. It is important that it does not 'hang' over the edges of the support, as otherwise things will not work out as intended for the subsequent MULTEM image simulation.

```
In[ ]:= PlaceChains[Chainls_, GeLs_, Sub_] := Module[{chainls = Chainls, gels = GeLs, sub = Sub, dropped, i, chaindummy, P, regP, drpP, sink},
 dropped = {};
 i = 1;
 While[Length[dropped] < Length[chainls],
  chaindummy = Region@Cuboid[First /@ RegionBounds[chainls[[i, 3]]], Last /@ RegionBounds[chainls[[i, 3]]]];
  P = Flatten[{RandomChoice[gels][[1 ;; 2]], Max[Part[RegionBounds[sub], 3]] + 50}];
  chaindummy = TransformedRegion[chaindummy, TranslationTransform[P]];
  regP = RegionNearest[sub, P];
  drpP = Flatten[Nearest[gels, regP]];
  sink = Flatten@{0, 0, (Differences[Last@RegionBounds[sub]] * 0.1 * RandomInteger[{1, 100}] / 100)};
  chaindummy = TransformedRegion[chaindummy, TranslationTransform[drpP - P - sink]];
  If[AnyTrue[{AnyTrue[Thread[IntegerPart[Volume[RegionIntersection[chaindummy, #], PrecisionGoal → 1]] & /@ dropped > 0], TrueQ],
     Max[RegionBounds[chaindummy][[1]]] > Max[RegionBounds[sub][[1]]],
     Max[RegionBounds[chaindummy][[2]]] > Max[RegionBounds[sub][[2]]],
     Min[RegionBounds[chaindummy][[1]]] < Min[RegionBounds[sub][[1]]],
     Min[RegionBounds[chaindummy][[2]]] < Min[RegionBounds[sub][[2]]]}, TrueQ],
   (chaindummy = {};),
   chainls[[i, 1]] = P + # & /@ chainls[[i, 1]];
   chainls[[i, 1]] = (drpP - P - sink) + # & /@ chainls[[i, 1]];
   chainls[[i, 2]] = Map[(P + # &), chainls[[i, 2]], {2}];
   chainls[[i, 2]] = Map[((drpP - P - sink) + # &), chainls[[i, 2]], {2}];
   chainls[[i, 3]] = TransformedRegion[chainls[[i, 3]], TranslationTransform[P]];
   chainls[[i, 3]] = TransformedRegion[chainls[[i, 3]], TranslationTransform[drpP - P - sink]];
   AppendTo[dropped, chaindummy];
   i = i + 1;]];
 {chainls, dropped}
]
```

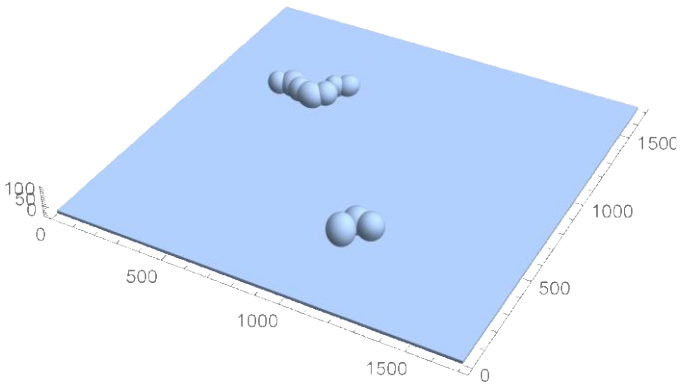
```
In[ ]:= AbsoluteTiming[FXchains = PlaceChains[chainlist, GeSupport[[1]], GeSupport[[3]]];]
```

```
Out[ ]= {17.1231, Null}
```

Here you can check where the dummies were dropped. Change FXchains[[2]] to FXchains[[1,All,3]] to see a more realistic model.

```
In[ ]:= Show[GeSupport[[3]], FXchains[[1, All, 3]], Axes → True]
```

```
Out[ ]=
```



Drop the rest of the NPs

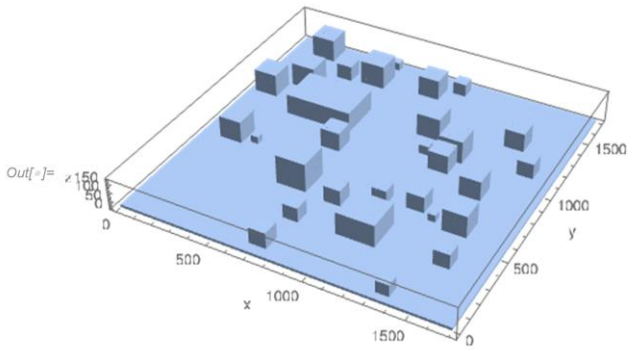
Here exclusively single NPs are dropped on the support material. DropNPv2 is the second version of the original function while 'xs' stands for a shortened version as in comparison to the Pd/C model, from which the whole approach originated. One has to note that the following function only drops placeholders for the actual NP shells. It specifies the number, size and position and makes sure the particles do not intersect.

```
In[•]:= DropNPv2xs[GeLs_, Sub_, number_, chaindummies_ : {}] :=
 Module[{gels = GeLs, sub = Sub, num = number, cbls, posls, counter, szls, P, cube, sz, regP, drpP, sink},
  cbls = {}; posls = {}; counter = {}; szls = {};
  If[chaindummies != {},
   cbls = chaindummies;
   posls = Flatten[RandomInteger[{0, 1}, Length[chaindummies]]]; counter = posls; szls = posls;];
  While[Length[counter] < num,
   P = Flatten[{RandomChoice[gels][[1 ;; 2]], Max[Part[RegionBounds[sub], 3]] + 50}];
   cube = TransformedRegion[DiscretizeRegion[Cuboid[{-# / 2, -# / 2, 0}, {# / 2, # / 2, #}] &@(LongAxisDistNum[] * 1.1)], TranslationTransform[P]];
   sz = Abs[Round[Last[Last[CoordinateBounds[cube]]] - First[Last[CoordinateBounds[cube]]]]];
   regP = RegionNearest[sub, P];
   drpP = Flatten[Nearest[gels, regP]];
   sink = Flatten@{0, 0, (Differences[Last@RegionBounds[sub]] * 0.1 * RandomInteger[{1, 100}] / 100)};
   cube = TransformedRegion[cube, TranslationTransform[drpP - P - sink]];
   If[AnyTrue[{AnyTrue[Thread[IntegerPart[Volume[RegionIntersection[cube, #], PrecisionGoal → 1]] & /@ cbls > 0], TrueQ],
      Max[RegionBounds[cube][[1]]] > Max[RegionBounds[sub][[1]]],
      Max[RegionBounds[cube][[2]]] > Max[RegionBounds[sub][[2]]],
      Min[RegionBounds[cube][[1]]] < Min[RegionBounds[sub][[1]]],
      Min[RegionBounds[cube][[2]]] < Min[RegionBounds[sub][[2]]]}, TrueQ],
    (cube = {};),
    (AppendTo[cbls, cube];
     AppendTo[posls, (drpP - sink)];
     AppendTo[counter, sz];
     AppendTo[szls, sz];)];];
  If[chaindummies ≠ {},
   cbls = Drop[cbls, {1, Length[chaindummies]}];
   posls = Drop[posls, {1, Length[chaindummies]}];
   szls = Drop[szls, {1, Length[chaindummies]}]];
  {cbls, posls, (szls * 1 / 1.1)}]
```

```
In[•]:= NPDrop = DropNPv2xs[GeSupport[[1]], GeSupport[[3]], 30, FXchains[[2]]];
```

```
In[•]:= Show[GeSupport[[3]], NPDrop[[1]], FXchains[[2]], Axes → True, Boxed → True, AxesLabel → {"x", "y", "z"}]
```

Out[•]=



Actually generate them

Here the NPs are actually generated. As mentioned earlier, probabilities for individual geometries to appear here can be set in the GenNPs[] function. Once generated, the particles need to be shifted to the earlier defined placeholders. Necessary properties are directly put in via the NPDrop function.

```
GenNPs[proplist_] := Module[{gen, prpls = proplist},
  gen = ReleaseHold@RandomChoice[{0.05, 0.05, 0.05, 0.85} -> {Hold[NPc8[#]], Hold[NPicosa[#]], Hold[NPd10[#]], Hold[NPellipsoid[#]]}] & /@
    (prpls[[3]]);
  gen]
```

```
AuNP = GenNPs[NPDrop];
```

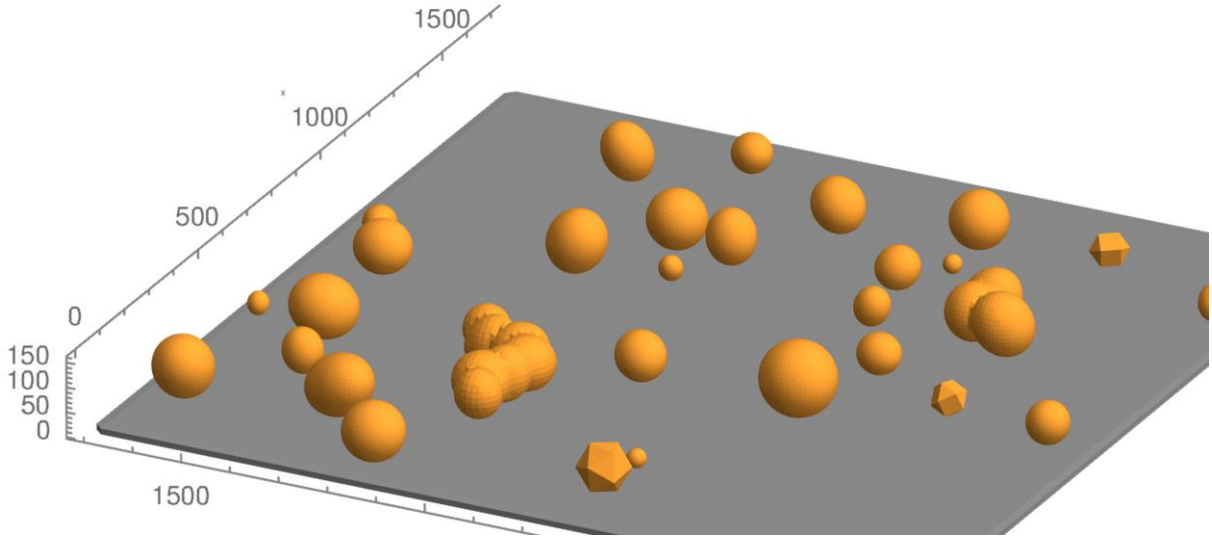
```
Shifter[NP_, NPshell_, position_] := Module[{np = NP, nps = NPshell, sft, shell},
  sft = Abs[Min[RegionBounds[nps][[3]]]];
  np = Table[np[[x]] + {0, 0, sft}, {x, 1, Length[np]}];
  np = Table[np[[x]] + position, {x, 1, Length[np]}];
  shell = TransformedRegion[nps, TranslationTransform[{0, 0, sft}]];
  shell = TransformedRegion[shell, TranslationTransform[position]];
  {np, shell}]
```

```
TotalAuNPs = MapThread[Shifter, {AuNP[[All, 1]], AuNP[[All, 2]], NPDrop[[2]]}];
```

```
TotalAuNPs[[All, 2]];
```

```
GeSupport[[3]], FXchains[[1, All, 3]]
```

```
AuGeModel = Show[Region[GeSupport[[3]], BaseStyle → Gray], Region[#, BaseStyle → RGBColor["#E3902D"]] & /@ FXchains[[1, All, 3]],
  Region[#, BaseStyle → RGBColor["#E3902D"]] & /@ TotalAuNPs[[All, 2]], ImageSize → Full, ViewPoint -> {-100, -50, 50},
  Axes → True, AxesLabel → {"x", "y", "z"}, AxesStyle → Thick, TicksStyle → Large, FontFamily → "Helvetica"]
```



Formatting for MULTEM

This last function below formats the generated coordinate lists into a MULTEM format. In particular, assigning the correct element number and the root-mean-square displacement place an important role.

```
MULTEMRdY[element_, list_, rms_, o1_, o2_, o3_] := Module[{ls = list, e = element, r = rms},
  ls = Prepend[#, e] & /@ ls;
  ls = Append[#, rms] & /@ ls;
  ls = Append[#, o1] & /@ ls;
  ls = Append[#, o2] & /@ ls;
  ls = Append[#, o3] & /@ ls;
  ls]
```

Lets briefly calculate the rms for Au and Ge atoms respectively. As stated in the Trueblood et al., Acta Cryst. A52, 1996; one way to calculate the B-factor can be calculated via:
B = 8Pi^2*U, whereby is related to the mean square displacement by: U = <u^2>.
The following values for B are given:
B(Au, 293 K, fcc) = 0.57 (V. F. Sears and S. A. Shelley, Acta Cryst. A47, 1991).
B(Ge, 293 K, Diamond) = 0.57 (V. F. Sears and S. A. Shelley, Acta Cryst. A47, 1991).

It has to be added, that the mean square displacement will not play a significant role, since the resolution this method is intended to work is too low for this effect to have a large influence.

Set your directory

```
SetDirectory["/mydirectory/"]
```

```
ModelGenerator[loops_] :=
 Do[Module[{GeLayer, GetChain, AllChains, AuChain, FXDchains, NPdrp, AuNPs, AuNPsonGe, AuonGelst, GeinAuNPs, GeinAuNPls, GelsCleans1,
    GeinChains, GeinChainsls, GelsTrue, LstShift, AuNPsonGeCC, AuNPsonGeCCID, FXDchainsCC, FXDchainsCCID, FXDcnsNPsCC, FXDcnsNPsCCID,
    GeList, AuList, AuM, GeM, Modellist},
   GeLayer = Layer[RandomInteger[{16, 17}] * 100, RandomInteger[{10, 22}]];
   (*numbers in RandomInteger define size and thickness of support*)
   GetChain = RandomChoice[{0.95, 0.05} -> {"On", "Off"}];
   If[GetChain == "On",
    (*ON*)
    AllChains = {};
    Do[AuChain = ConnectedParticles[RandomInteger[{2, 6}]]; AppendTo[AllChains, AuChain];, RandomInteger[{1, 3}]];
    FXDchains = PlaceChains[AllChains, GeLayer[[1]], GeLayer[[3]]];
    NPdrp = DropNPv2xs[GeLayer[[1]], GeLayer[[3]], RandomInteger[{10, 25}], FXDchains[[2]]];,
    (*OFF*)
    NPdrp = DropNPv2xs[GeLayer[[1]], GeLayer[[3]], RandomInteger[{10, 25}]];];
   AuNPs = GenNPs[NPdrp];
   AuNPsonGe = MapThread[Shifter, {AuNPs[[All, 1]], AuNPs[[All, 2]], NPdrp[[2]]}];
   If[GetChain == "On",
    (*ON*)
    AuonGelst = Flatten[Join[AuNPsonGe[[All, 1]], FXDchains[[1, All, 1]]], 1];
    GeinAuNPs = RegionMember[#, GeLayer[[1]]] & /@ AuNPsonGe[[All, 2]];
    GeinAuNPls = Flatten[Pick[GeLayer[[1]], #] & /@ GeinAuNPs, 1];
    GelsCleans1 = Complement[GeLayer[[1]], GeinAuNPls];
    GeinChains = RegionMember[#, GelsCleans1] & /@ FXDchains[[1, All, 3]];
    GeinChainsls = Flatten[Pick[GelsCleans1, #] & /@ GeinChains, 1];
    GelsTrue = Complement[GelsCleans1, GeinChainsls];,
    (*OFF*)
    AuonGelst = Flatten[AuNPsonGe[[All, 1]], 1];
    GeinAuNPs = RegionMember[#, GeLayer[[1]]] & /@ AuNPsonGe[[All, 2]];
    GeinAuNPls = Flatten[Pick[GeLayer[[1]], #] & /@ GeinAuNPs, 1];
    GelsTrue = Complement[GeLayer[[1]], GeinAuNPls];];
   LstShift = Abs[Part[#, 1] & /@ CoordinateBounds[{Part[#, 1] & /@ CoordinateBounds[GelsTrue], Part[#, 1] & /@ CoordinateBounds[AuonGelst]}]] +
     {1, 1, 1};
   If[GetChain == "On",
    (*ON*)
    AuNPsonGeCC = Table[(LstShift + # &) /@ AuNPsonGe[[x, 1]], {x, 1, Length[AuNPsonGe[[All, 1]]]}];
    AuNPsonGeCCID = Flatten[Table[Prepend[#, x] & /@ AuNPsonGeCC[[x]], {x, 1, Length[AuNPsonGeCC]}], 1];
    FXDcnsNPsCC = Map[(LstShift + # &), FXDchains[[1, All, 2]], {3}];
    FXDcnsNPsCCID = Table[Flatten[Table[Prepend[#, x] & /@ FXDcnsNPsCC[[u, x]], {x, 1, Length[FXDcnsNPsCC[[u]]]}], 1],
      {u, 1, Length[FXDcnsNPsCC]}];
    FXDcnsNPsCCID = Table[Prepend[#, x] & /@ FXDcnsNPsCCID[[x]], {x, 1, Length[FXDcnsNPsCCID]}];
    FXDcnsNPsCCID = Flatten[FXDcnsNPsCCID, 1];
    GeList = LstShift + # & /@ GelsTrue;
    AuList = LstShift + # & /@ AuonGelst;,
    (*OFF*)
    AuNPsonGeCC = Table[(LstShift + # &) /@ AuNPsonGe[[x, 1]], {x, 1, Length[AuNPsonGe[[All, 1]]]}];
    AuNPsonGeCCID = Flatten[Table[Prepend[#, x] & /@ AuNPsonGeCC[[x]], {x, 1, Length[AuNPsonGeCC]}], 1];
    GeList = LstShift + # & /@ GelsTrue;
    AuList = LstShift + # & /@ AuonGelst;];
   AuM = MULTEMRdY[79, AuList, 0.085, 1, 0, 0];
   GeM = MULTEMRdY[32, GeList, 0.085, 1, 0, 0];
   Modellist = Join[AuM, GeM];
   If[GetChain == "On",
    Export["Model" <> ToString[n] <> ".h5", {"/Dataset1" -> Modellist, "/Dataset2" -> AuNPsonGeCCID, "/Dataset3" -> FXDcnsNPsCCID}];,
    Export["Model" <> ToString[n] <> ".h5", {"/Dataset1" -> Modellist, "/Dataset2" -> AuNPsonGeCCID}];];
   ClearSystemCache[];], {n, 1, loops}]
```

```
In[ ]:= ModelGenerator[500]; // AbsoluteTiming
```

Code 10.2: Mathematica 12.1 notebook for computational Pd/C sample models.

Full Pd/C Model Builder V 1.7

A notebook generating Au/Ge models as input for
MULTEM simulations.
Copyright: Kevin P Treder, Department of Materials, University of Oxford
Author: Kevin P Treder

Foreword

The following content presented in this notebook shall provide the full code for the generation of model of Pd nanoparticles on active carbon as an input for MULTEM. Functions introduced stepwise and explained in detail. The author claims copyright of this notebook as well as the ideas and concepts introduced to generate the resulting output. This notebook does not claim to work on other purposes other than the one state above. However, it is published and distributed in the hope that it will help individual with simulations, research or serve as an example. This notebook is subject to ongoing development. Relevant sizes in this notebook should be given in Angstrom.

Active carbon, the base

As a start, the supporting material in the model - active carbon - needs to be generated in the form of a list containing {x,y,z} - coordinates. As a start the shape of a 'ground' layer is generated. Active carbon looks somewhat arbitrary on a 2d-projection. A list of country-shapes ("Countrygons") is generated:

```
In[•]:= Needs["Developer`"];

In[•]:= CountryShapes = CountryData[#, {"Polygon", {"Polyconic", Center}}] & /@ CountryData["Countries"];
```

This list represents manual picks of CountryShapes that were not as well suited as others for their shape, number of islands or other properties.

```
In[•]:= Dellist = {{2}, {7}, {9}, {10}, {11}, {13}, {15}, {16}, {17}, {24}, {27}, {32}, {33}, {34}, {40}, {41}, {42}, {45}, {46}, {48}, {50}, {51}, {53},
        {54}, {59}, {63}, {67}, {69}, {71}, {72}, {73}, {75}, {77}, {78}, {80}, {83}, {86}, {88}, {89}, {90}, {92}, {94}, {96}, {98}, {101}, {102},
        {106}, {108}, {111}, {116}, {118}, {127}, {129}, {133}, {134}, {136}, {137}, {141}, {143}, {155}, {156}, {157}, {158}, {162}, {163}, {166},
        {168}, {169}, {170}, {173}, {174}, {175}, {177}, {178}, {184}, {186}, {188}, {189}, {191}, {193}, {198}, {199}, {200}, {203}, {204}, {206},
        {207}, {209}, {213}, {215}, {219}, {221}, {222}, {223}, {224}, {225}, {226}, {227}, {228}, {229}, {234}, {236}, {237}, {240}, {241}, {242}, {244}, {247}};

In[•]:= Countrygons = Delete[CountryShapes, Dellist];
```

Based on the selected shapes, a Countrygon is transformed into a MeshRegion[] and its edges smoothened so that the shape has higher resemblance to the contours of the active carbon. Furthermore, the size is normalized such that it can be rescaled close to the desired final size later on.

```
In[•]:= CountryPol[factor_] := Module[{shape, tr1, Centrd, norm, f = factor, Sized, Rotd, Pol, Shift, Poltr, Polonx},
         shape = BoundaryMeshRegion[RandomChoice[Countrygons]];
         Rotd = DiscretizeRegion@TransformedRegion[shape, RotationTransform[RandomInteger[{0, 359}] Degree]];
         Pol = ImageMesh[Erosion[Blur[#, RandomInteger[{12, 30}]] &@ColorNegate[Graphics[Rotd]], RandomInteger[{0, 3}]]];
         Pol = Part[Sort[ConnectedMeshComponents[Pol], Area[#1] > Area[#2] &], 1];
         Shift = {0, 0} - RegionCentroid[Pol];
         Poltr = Translate[Pol, Shift];
         Sized = RegionResize[Poltr, {f}];
         Polonx = Translate[Sized, ({Abs[Min[First[RegionBounds[Sized]]]], Abs[Min[Last[RegionBounds[Sized]]]]})];
         Polonx]
```

This function stacks multiple CountryPols upon each other to generate an even more random object.

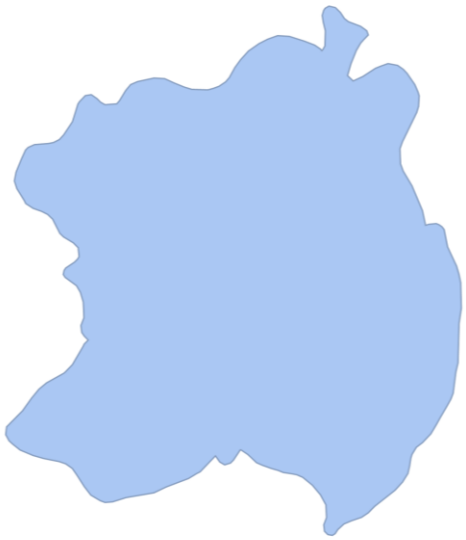
```
In[•]:= MultiCountryPol[Factor_, Number_] := Module[{f = Factor, num = Number, TotalPoly, BaseRadPoint, NewPol, NewPolCentre, Shift},
          TotalPoly = CountryPol[f];
          BaseRadPoint = RandomPoint[Disk[RegionCentroid[TotalPoly], (Max[RegionBounds[TotalPoly]] / 5)], 1];
          Do[
           NewPol = CountryPol[f];
           NewPolCentre = RegionCentroid[NewPol];
           Shift = Flatten@BaseRadPoint - NewPolCentre;
           NewPol = Translate[NewPol, Shift];
           TotalPoly = RegionUnion[TotalPoly, NewPol];
           BaseRadPoint = RandomPoint[Disk[RegionCentroid[TotalPoly], (Max[RegionBounds[TotalPoly]] / 5)], 1];, num];
          TotalPoly = Translate[TotalPoly, ({Abs[Min[First[RegionBounds[TotalPoly]]]], Abs[Min[Last[RegionBounds[TotalPoly]]]]})];
          TotalPoly]

In[•]:= AbsoluteTiming[pol1 = MultiCountryPol[500, 3];]

Out[•]= {0.736709, Null}

In[•]:= pol1

Out[•]=
```



Going 3D

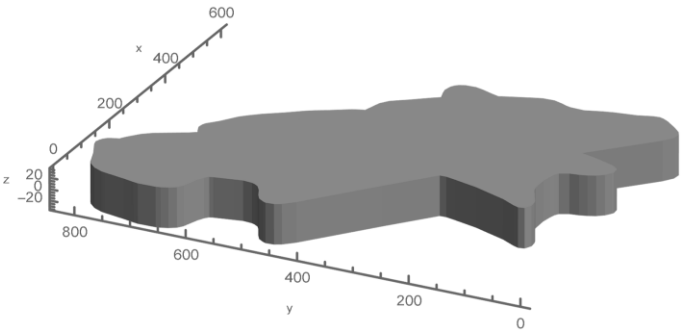
In the next step, one could imagine that a large enough box containing {x,y,z}-coordinates of carbon atoms is generated and the polygon created above is used as a 'cutting die' to create a 3d shape with the contour of active carbon. This operation is fast as it merely requires a comparison of x-y-coordinates that are within the polygon region, but has the disadvantage that it will form an orthogonal edge, which looks darker in the simulations than in experimental images.

```
In[•]:= Layer[poly_, h_] := Block[{pl = poly, bd, box, num, rm, sel, psel, pts, cls, pre, s},
          bd = Last /@ RegionBounds[pl];
          box = Region[Cuboid[{0, 0, 0}, {bd[[1]], bd[[2]], h}]];
          num = Ceiling[bd[[1]] * bd[[2]] * h / 2.27225^3, 10]; pts = RandomPoint[box, num];
          rm = RegionMember[pl];
          sel = rm[pts[[All, 1 ;; 2]]];
          psel = Developer`ToPackedArray[With[{True = 1, False = 0}, Evaluate[sel]], Tolerance → 0.01];
          cls = Pick[pts, psel, 1];
          pre = RegionProduct[pl, MeshRegion[{{0}, {h}}, Line[{1, 2}]]];
          s = Region@TransformedRegion[pre, TranslationTransform[{0, 0, -h / 2}]];
          cls = {0, 0, -h / 2} + # & /@ cls;
          {cls, pl, s}]

In[•]:= tlayer = Layer[MultiCountryPol[800, RandomInteger[{1, 5}]], RandomInteger[{70, 100}]];
```

```
In[•]:= Show[Region[#, BaseStyle → Gray] & /@ tlayer[[3]], ImageSize → Large, ViewPoint -> {-100, -50, 50}, Axes → True, AxesLabel → {"x", "y", "z"},
  AxesStyle → Thick, TicksStyle → Medium, FontFamily → "Helvetica"]
```

Out[•]=



FastSandwich[] stacks a multiple of the above layers upon another. This forms the full support of the active carbon.

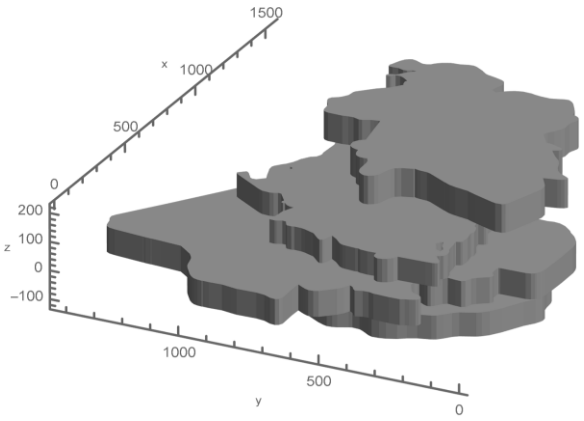
```
In[•]:= FastSandwich[CoreWidth_, Number_, MinH_, MaxH_] :=
 Module[{cw = CoreWidth, num = Number, minh = MinH, maxh = MaxH, CoreLayer, TotalCls, TotalRegs, UpRegs, DownRegs, UpDown, AllMinMax, StackLayer,
   StackCentre, StackBounds, PolyRadPoint, CoreRadPoint, ShiftVec},
  CoreLayer = Layer[MultiCountryPol[cw, RandomInteger[{1, 5}]], RandomInteger[{minh, maxh}]];
  TotalCls = {CoreLayer[[1]]};
  TotalRegs = CoreLayer[[3]];
  UpRegs = CoreLayer[[3]];
  DownRegs = CoreLayer[[3]];
  Do[
   UpDown = RandomChoice[{-1, 1}];
   If[Length[TotalRegs] > 1,
    AllMinMax = MinMax[Part[#, 3] & /@ RegionBounds /@ TotalRegs];,
    AllMinMax = MinMax[Part[#, 3] &@RegionBounds@TotalRegs];];
   If[ UpDown < 0,
    StackLayer = Layer[MultiCountryPol[(Max[RegionBounds[Last[DownRegs]]] * RandomReal[{0.5, 0.8}]), RandomInteger[{1, 5}]], RandomInteger[{minh, maxh}]];
    StackCentre = Flatten@{RegionCentroid[StackLayer[[3]]][[1 ;; 2]], 0};
    StackBounds = RegionBounds[StackLayer[[3]]];
    PolyRadPoint = RandomPoint[Disk[RegionCentroid[Last[DownRegs]][[1 ;; 2]], (Max[RegionBounds[Last[DownRegs]]] / 3)]];
    CoreRadPoint = Flatten[{PolyRadPoint, 0}];
    ShiftVec = Flatten[{Abs[StackCentre - CoreRadPoint][[1 ;; 2]], StackBounds[[3, 1]] + AllMinMax[[1]]}];
    StackLayer[[1]] = ShiftVec + # & /@ StackLayer[[1]];
    StackLayer[[3]] = TransformedRegion[StackLayer[[3]], TranslationTransform[ShiftVec]];
    AppendTo[TotalCls, StackLayer[[1]]];
    TotalRegs = Delete[Flatten[Reap[Sow[TotalRegs]; Sow[StackLayer[[3]]];]], 1];
    DownRegs = Delete[Flatten[Reap[Sow[DownRegs]; Sow[StackLayer[[3]]];]], 1];,
    StackLayer = Layer[MultiCountryPol[(Max[RegionBounds[Last[UpRegs]]] * RandomReal[{0.5, 0.8}]), RandomInteger[{1, 5}]], RandomInteger[{minh, maxh}]];
    StackCentre = Flatten@{RegionCentroid[StackLayer[[3]]][[1 ;; 2]], 0};
    StackBounds = RegionBounds[StackLayer[[3]]];
    PolyRadPoint = RandomPoint[Disk[RegionCentroid[Last[UpRegs]][[1 ;; 2]], (Max[RegionBounds[Last[UpRegs]]] / 3)]];
    CoreRadPoint = Flatten[{PolyRadPoint, 0}];
    ShiftVec = Flatten[{Abs[StackCentre - CoreRadPoint][[1 ;; 2]], StackBounds[[3, 2]] + AllMinMax[[2]]}];
    StackLayer[[1]] = ShiftVec + # & /@ StackLayer[[1]];
    StackLayer[[3]] = TransformedRegion[StackLayer[[3]], TranslationTransform[ShiftVec]];
    AppendTo[TotalCls, StackLayer[[1]]];
    TotalRegs = Delete[Flatten[Reap[Sow[TotalRegs]; Sow[StackLayer[[3]]];]], 1];
    UpRegs = Delete[Flatten[Reap[Sow[UpRegs]; Sow[StackLayer[[3]]];]], 1];], num - 1];
  {Flatten[TotalCls, 1], TotalRegs}
 ]
```

```
In[•]:= AbsoluteTiming[Teststack = FastSandwich[1000, 4, 80, 100];]
```

Out[•]= {21.7131, Null}

```
In[•]:= Show[Region[#, BaseStyle → Gray] & /@ Teststack[[2]], ImageSize → Large, ViewPoint -> {-100, -50, 50}, Axes → True, AxesLabel → {"x", "y", "z"},
  AxesStyle → Thick, TicksStyle → Medium, FontFamily → "Helvetica"]
```

Out[•]=



Dropping Cluster

As the substrate is now prepared, NPs shall be dropped. However, it will be difficult to find suitable positions for the clusters once all NPs cover the majority of the supporting material. Thus, clusters will be dropped first.
Two main aspects need to be remarked:
i) There is a case where two particles are close enough to resemble an 'elongated' one. A clear border between two particles has to be ensured.
ii) There are cases where single NPs are dropped below the cluster. Generally, stacking on particles on individual supporting material planes needs to be avoided!

Building the lattice

```
In[•]:= FCCbasis = 0.5 * LatticeData["FaceCenteredCubic", "Basis"];

In[•]:= dPd = 3.98;

In[•]:= Timing[FCCLattice = Tuples[Range[-100, 100], 3].FCCbasis;]

Out[•]= {0.872359, Null}
```

Sizing & Distribution functions

```
In[•]:= LongAxisDistNum[] := Module[{s1},
          s1 = Abs[Round[RandomVariate[NormalDistribution[7.1, 2.52558541]], 0.01]] * 10;
          If[Between[s1, {10, 130}], s1, LongAxisDistNum[]]]

In[•]:= AspectRatioGen[] := Module[{s1},
          s1 = Abs[Round[RandomVariate[NormalDistribution[1.22588235, 0.16814066]], 0.01]];
          If[Between[s1, {1, 1.6}], s1, AspectRatioGen[]]]

In[•]:= NonZeroRandomInteger[range_, dims_] := Module[{original}, original = RandomInteger[range, dims];
          If[Norm[original] == 0, NonZeroRandomInteger[range, dims], original]]
```

Initialize shape functions

Cuboctahedron:

```
In[•]:= NPc8[size_] := Module[{shell, s = size, rotad, ls, rotm},
          rotad = RotationTransform[{{1, 0, 0}, {1, 1, 0}}];
          shell = PolyhedronData["Cuboctahedron", "BoundaryMeshRegion"];
          shell = Scale[shell, (s / 2)];
          shell = TransformedRegion[shell, rotad];
          ls = Pick[Pddat, RegionMember[shell, Pddat]];
          rotm = RotationTransform[Round[RandomReal[{0, 359}], 0.1] Degree, NonZeroRandomInteger[{-100, 100}, 3]];
          ls = rotm[ls];
          shell = TransformedRegion[shell, rotm];
          {ls, shell}]
```

Icosahedron:

```
In[•]:= NPicosa[size_] := Module[{shell, s = size, rotad1, rotad2, ls, rotm},
          rotad1 = RotationTransform[{{0.7694208842938134, -0.25, 0.}, {1, 0, 0}}];
          rotad2 = RotationTransform[30 Degree, {1, 0, 0}];
          shell = PolyhedronData["Icosahedron", "BoundaryMeshRegion"];
          shell = Scale[shell, (s / 2)];
          shell = TransformedRegion[shell, rotad1]; shell = TransformedRegion[shell, rotad2];
          ls = Pick[Pddat, RegionMember[shell, Pddat]];
          rotm = RotationTransform[Round[RandomReal[{0, 359}], 0.1] Degree, NonZeroRandomInteger[{-100, 100}, 3]];
          ls = rotm[ls];
          shell = TransformedRegion[shell, rotm];
          {ls, shell}]
```

Decahedron:

```
In[•]:= NPd10[size_] := Module[{shell, s = size, rotad1, rotad2, rotad3, ls, rotm},
          rotad1 = RotationTransform[{{{1/2 Sqrt[1/10 (5 - Sqrt[5])], 1/4 (1 + Sqrt[5]), 0}, {1, 1, 0}}}];
          rotad2 = RotationTransform[{{0, 0, 1}, {1, 1, 0}}];
          shell = PolyhedronData["ElongatedPentagonalDipyramid", "BoundaryMeshRegion"];
          shell = Scale[shell, (s / 2)];
          shell = TransformedRegion[shell, rotad1]; shell = TransformedRegion[shell, rotad2];
          ls = Pick[Pddat, RegionMember[shell, Pddat]];
          rotm = RotationTransform[Round[RandomReal[{0, 359}], 0.1] Degree, NonZeroRandomInteger[{-100, 100}, 3]];
          ls = rotm[ls];
          shell = TransformedRegion[shell, rotm];
          {ls, shell}]
```

Ellipsoids:

```
In[•]:= NPellipsoid[size_] := Module[{s = size, s1, rkill, sctr, s2, ls, rottr, shell},
          s1 = Region[Ellipsoid[{0, 0, 0}, {1, 1, rkill = AspectRatioGen[]}]];
          s2 = Scale[s1, (s / 2)];
          ls = Pick[Pddat, RegionMember[s2, Pddat]];
          rottr = RotationTransform[Round[RandomReal[{0, 359}], 0.1] Degree, NonZeroRandomInteger[{-100, 100}, 3]];
          ls = rottr[ls];
          shell = TransformedRegion[s2, rottr];
          {ls, shell}]
```

Build a cluster

```
In[•]:= ConnectedParticles[ChainNP_] := Module[{counter, dummycounter, SeedNP, SeedNPDummy, nxtNP, nxtNPDummy, AttachNP, sftP, firstrel, allint, rels, allints, plusZ},
         counter = {};
         dummycounter = {};
         SeedNP = NPellipsoid[Round[RandomReal[{85, 115}], 0.01]];
         SeedNPDummy = Region@Cuboid[First /@ RegionBounds[SeedNP[[2]]], Last /@ RegionBounds[SeedNP[[2]]]];
         AppendTo[counter, SeedNP];
         AppendTo[dummycounter, SeedNPDummy];
         While[Length[counter] < ChainNP,
          AttachNP = dummycounter[[-1]];
          nxtNP = NPellipsoid[Round[RandomReal[{85, 115}], 0.01]];
          nxtNPDummy = Region@Cuboid[First /@ RegionBounds[nxtNP[[2]]], Last /@ RegionBounds[nxtNP[[2]]]];
          sftP = Flatten[{RandomPoint[counter[[-1, 2]]][[1 ;; 2]], 0}] * RandomReal[{1.0, 2.0}];
          nxtNPDummy = TransformedRegion[nxtNPDummy, TranslationTransform[sftP]];
          nxtNP[[1]] = sftP + # & /@ nxtNP[[1]];
          firstrel = N[Length@Pick[nxtNP[[1]], RegionMember[counter[[-1, 2]], nxtNP[[1]]]] / Length[nxtNP[[1]]]];
          allint = Pick[nxtNP[[1]], RegionMember[#, nxtNP[[1]]]] & /@ counter[[1 ;; -2, 2]];
          rels = N[Length /@ allint / Length[nxtNP[[1]]]];
          If[! Between[firstrel, {0.2, 0.6}] ∨ AnyTrue[rels, # > 0.4 &],
           nxtNP = {}; nxtNPDummy = {};,
           allints = Pick[nxtNP[[1]], RegionMember[#, nxtNP[[1]]]] & /@ counter[[All, 2]];
           nxtNP[[1]] = Flatten[Table[nxtNP[[1]] = Complement[nxtNP[[1]], allints[[x]]], {x, 1, Length@allints}], 1];
           nxtNP[[2]] = TransformedRegion[nxtNP[[2]], TranslationTransform[sftP]];
           AppendTo[counter, nxtNP];
           AppendTo[dummycounter, nxtNPDummy];];];
         counter = {Flatten[counter[[All, 1]], 1], counter[[All, 1]], RegionUnion[counter[[All, 2]]]};
         plusZ = Abs[Min[Last[RegionBounds[counter[[3]]]]]];
         counter[[1]] = {0, 0, plusZ} + # & /@ counter[[1]];
         counter[[2]] = Map[({0, 0, plusZ} + # &), counter[[2]], {2}];
         counter[[3]] = TransformedRegion[counter[[3]], TranslationTransform[{0, 0, plusZ}]];
         counter
        ]
```

```
In[•]:= chainlist = {};
```

```
In[•]:= AbsoluteTiming[Do[chain = ConnectedParticles[RandomInteger[{2, 6}]]; AppendTo[chainlist, chain], 2];]
```

```
Out[•]= {20.5743, Null}
```

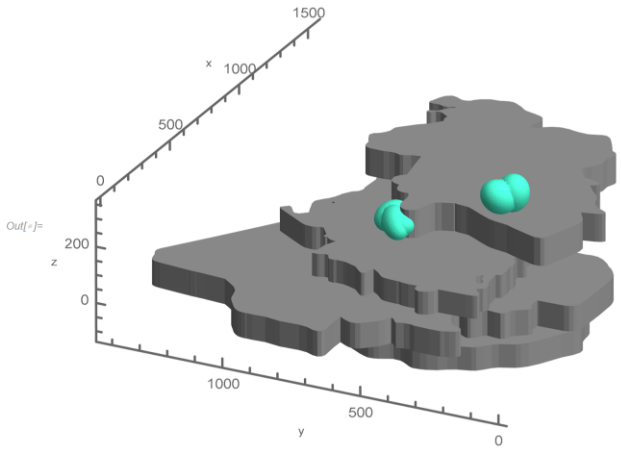
```
PlaceChains[Chainls_, CLs_, Sub_] := Module[{chainls = Chainls, cls = CLs, sub = Sub, dropped, i, chaindummy, allregP, vecls, angles, nPos, P, regP, drpP, sink},
 dropped = {};
 i = 1;
 While[Length[dropped] < Length[chainls],
  chaindummy = Region@Cuboid[First /@ RegionBounds[chainls[[i, 3]]], Last /@ RegionBounds[chainls[[i, 3]]]];
  P = Flatten[{RandomChoice[cls][[1 ;; 2]], Max[Last /@ Part[RegionBounds /@ sub, All, 3]] + 50}];
  chaindummy = TransformedRegion[chaindummy, TranslationTransform[P]];
  allregP = RegionNearest[#, P] & /@ sub;
  vecls = Round[P - # & /@ allregP, 0.01];
  angles = VectorAngle[{0, 0, 1}, #] & /@ vecls;
  nPos = Norm@Position[Norm /@ vecls, Min[Norm /@ Pick[vecls, # != Min[angles] & /@ angles]]];
  regP = RegionNearest[sub[[nPos]], P];
  drpP = Flatten[Nearest[cls, regP]];
  sink = Flatten@{0, 0, (Differences[Last@RegionBounds[sub[[nPos]]]] * 0.1 * RandomInteger[{1, 100}] / 100)};
  chaindummy = TransformedRegion[chaindummy, TranslationTransform[drpP - P - sink]];
  If[AnyTrue[{AnyTrue[Thread[IntegerPart[Volume[RegionIntersection[chaindummy, #], PrecisionGoal → 1]] & /@ dropped > 0], TrueQ],
     Max[RegionBounds[chaindummy][[1]]] > Max[Part[RegionBounds /@ sub, All, 1]],
     Max[RegionBounds[chaindummy][[2]]] > Max[Part[RegionBounds /@ sub, All, 2]],
     Min[RegionBounds[chaindummy][[1]]] < Min[Part[RegionBounds /@ sub, All, 1]],
     Min[RegionBounds[chaindummy][[2]]] < Min[Part[RegionBounds /@ sub, All, 2]]}, TrueQ],
   (chaindummy = {};),
   chainls[[i, 1]] = P + # & /@ chainls[[i, 1]];
   chainls[[i, 1]] = (drpP - P - sink) + # & /@ chainls[[i, 1]];
   chainls[[i, 2]] = Map[(P + # &), chainls[[i, 2]], {2}];
   chainls[[i, 2]] = Map[((drpP - P - sink) + # &), chainls[[i, 2]], {2}];
   chainls[[i, 3]] = TransformedRegion[chainls[[i, 3]], TranslationTransform[P]];
   chainls[[i, 3]] = TransformedRegion[chainls[[i, 3]], TranslationTransform[drpP - P - sink]];
   AppendTo[dropped, chaindummy];
   i = i + 1;]];
 {chainls, dropped}
]
```

```
AbsoluteTiming[PdFXchains = PlaceChains[chainlist, Teststack[[1]], Teststack[[2]]];]
```

```
{9.54821, Null}
```

```
Show[Region[#, BaseStyle → Gray] & /@ Teststack[[2]], Region[#, BaseStyle → RGBColor["#44E3BE"]] & /@ PdFXchains[[1, All, 3]], ImageSize → Large,
 ViewPoint -> {-100, -50, 50}, Axes → True, AxesLabel → {"x", "y", "z"}, AxesStyle → Thick, TicksStyle → Medium, FontFamily → "Helvetica"]
```

Out[ ]=



```
In[•]:= DropNPs[CLs_, Regs_, number_, ExNPls_ : {{}, {}, {}}, rt_ : {}, chains_ : {}] :=
 Module[{cls = CLs, regs = Regs, num = number, cbls, cblsLong, posls, counter, szls, P, cube, sz, allregP, vecls, angles, nPos, regP, drpP, sink, cubeLong, intersects},
  cbls = ExNPls[[1]]; posls = ExNPls[[2]]; counter = {}; szls = ExNPls[[3]];
  If[chains ≠ {},
   cbls = Join[chains[[2]], ExNPls[[1]]];
   (*This list ensures that NPs cannot be put below each other at different levels of the support, even at rotations*)
   posls = Join[Flatten[RandomInteger[{0, 1}, Length[chains[[2]]]]], ExNPls[[2]]];
   counter = {};
   szls = Join[Flatten[RandomInteger[{0, 1}, Length[chains[[2]]]]], ExNPls[[3]]];];
  While[Length[counter] < num,
   P = Flatten[{RandomChoice[cls][[1 ;; 2]], Max[Ceiling[Part[RegionBounds /@ regs, All, 3]]] + 50}];
   cube = TransformedRegion[DiscretizeRegion[Cuboid[{-# / 2, -# / 2, 0}, {# / 2, # / 2, #}] &@(LongAxisDistNum[] * 1.1)], TranslationTransform[P]];
   sz = Abs[Round[Last[Last[CoordinateBounds[cube]]] - First[Last[CoordinateBounds[cube]]]]];
   allregP = RegionNearest[#, P] & /@ regs;
   vecls = Round[P - # & /@ allregP, 0.01];
   angles = VectorAngle[{0, 0, 1}, #] & /@ vecls;
   nPos = Norm@Position[Norm /@ vecls, Min[Norm /@ Pick[vecls, # == Min[angles] & /@ angles]]];
   regP = RegionNearest[regs[[nPos]], P];
   drpP = Flatten[Nearest[cls, regP]];
   sink = Flatten@{0, 0, (Differences[Last@RegionBounds[regs[[nPos]]]] * 0.2 * RandomInteger[{1, 100}] / 100)};
   cube = TransformedRegion[cube, TranslationTransform[drpP - P - sink]];
   If[SameQ[rt, {}],
    (cblsLong = Region@Scale[#, {1.2, 1.2, 50}] & /@ cbls;
     cubeLong = Region@Scale[cube, {1.2, 1.2, 50}];),
    (cblsLong = Region@Scale[#, Abs[rt[{1.2, 1.2, 50}]]] & /@ cbls;
     cubeLong = Region@Scale[cube, Abs[rt[{1.2, 1.2, 50}]]];)]; (*The "1.2" are used to ensure distance in xy a little further*)
   intersects = RegionIntersection[cubeLong, #] & /@ cblsLong;
   If[AnyTrue[Thread[IntegerPart[Volume[intersects]] > 0], TrueQ],
    (cube = {};),
    (AppendTo[cbls, cube];
     AppendTo[posls, (drpP - sink)];
     AppendTo[counter, sz];
     AppendTo[szls, sz];)];];
  If[chains ≠ {},
   cbls = Drop[cbls, {1, Length[chains[[2]]]}]; posls = Drop[posls, {1, Length[chains[[2]]]}]; szls = Drop[szls, {1, Length[chains[[2]]]}]];
  {cbls, posls, (szls * 1 / 1.1)}
 ]

In[•]:= NPdrop1 = DropNPs[Teststack[[1]], Teststack[[2]], 25, {{}, {}, {}}, {}, PdFXchains]; // AbsoluteTiming

Out[•]= {119.79, Null}

In[•]:= AllNPs[CList_, Regls_, ExNPls_, number_, chains_ : {}] :=
 Module[{cls = CList, regs = Regls, exnpls = ExNPls, num = number, rtxm, rtxp, rtym, rtyp, allrts, apprts, cbls, posls, counter, szls, rton, rtoff, clsrt,
   regsrt, npdrp, cubes, pos, sizes},
  rtxm = RotationTransform[-90 Degree, {1, 0, 0}];
  rtxp = RotationTransform[90 Degree, {1, 0, 0}];
  rtym = RotationTransform[-90 Degree, {0, 1, 0}];
  rtyp = RotationTransform[90 Degree, {0, 1, 0}];
  allrts = {{rtxm, rtxp}, {rtxp, rtxm}, {rtym, rtyp}, {rtyp, rtym}};
  apprts = allrts[[RandomSample[{1, 2, 3, 4}]]][[1 ;; RandomChoice[{1, 2, 3, 4}]]]];
  Do[
   If[chains ≠ {},
    cbls = Join[chains[[2]], exnpls[[1]]];
    posls = Join[Flatten[RandomInteger[{0, 1}, Length[chains[[2]]]]], exnpls[[2]]];
    szls = Join[Flatten[RandomInteger[{0, 1}, Length[chains[[2]]]]], exnpls[[3]]];,
    cbls = exnpls[[1]]; posls = exnpls[[2]]; szls = exnpls[[3]];];
   rton = apprts[[k, 1]]; rtoff = apprts[[k, 2]];
   clsrt = rton[cls];
   regsrt = rton /@ regs;
   exnpls = {TransformedRegion[#, rton] & /@ cbls, rton /@ posls, szls};
   npdrp = DropNPs[clsrt, regsrt, num, exnpls, rton, {}];
   If[chains ≠ {},
    cbls = Drop[npdrp[[1]], {1, Length[chains[[2]]]}];
    posls = Drop[npdrp[[2]], {1, Length[chains[[2]]]}];
    szls = Drop[npdrp[[3]], {1, Length[chains[[2]]]}];,
    cbls = npdrp[[1]]; posls = npdrp[[2]]; szls = npdrp[[3]]];
   cbls = TransformedRegion[#, rtoff] & /@ cbls;
   posls = rtoff /@ posls;
   exnpls = {cbls, posls, szls};, {k, 1, Length[apprts]}];
  {cbls, posls, szls}
 ]

In[•]:= NPTot = AllNPs[Teststack[[1]], Teststack[[2]], NPdrop1, 3, PdFXchains]; // AbsoluteTiming

Out[•]= {61.3128, Null}
```

```
GenNPs[proplist_] := Module[{gen, prpls = proplist},
  gen = ReleaseHold@RandomChoice[{0.05, 0.05, 0.05, 0.85} -> {Hold[NPc8[#]], Hold[NPicosa[#]], Hold[NPd10[#]], Hold[NPellipsoid[#]]}] & /@ (prpls[[3]]);
  gen]
```

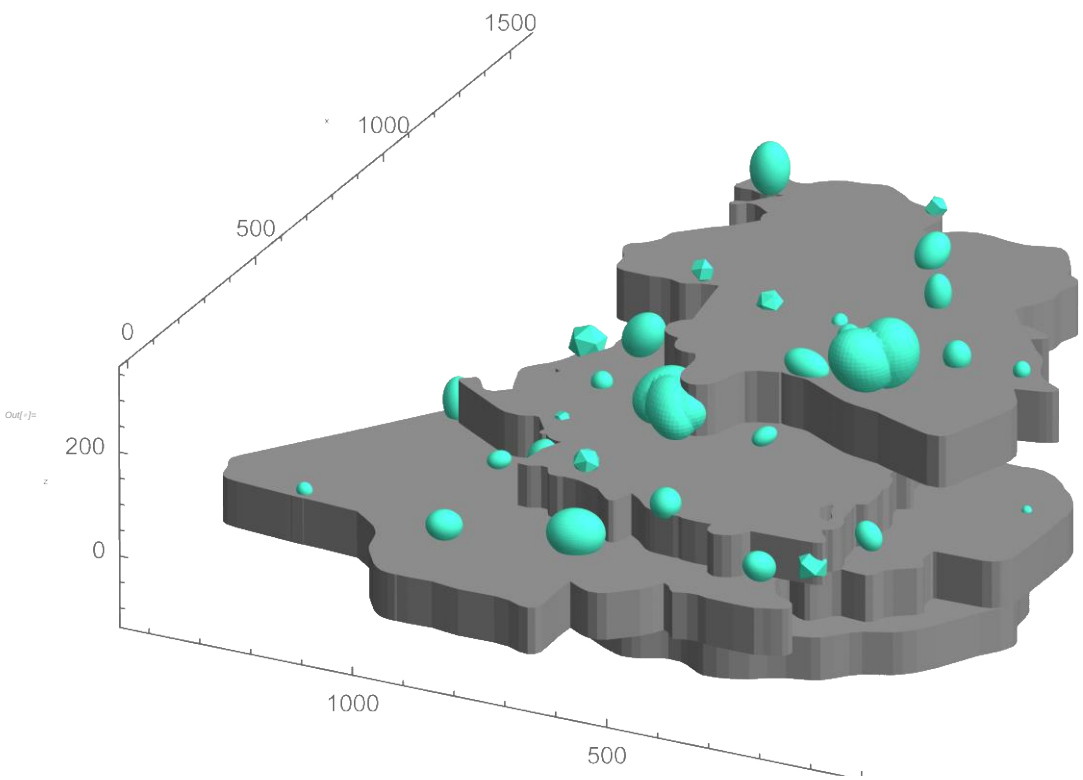
```
PdNP = GenNPs[NPTot]; // AbsoluteTiming
```

```
{151.519, Null}
```

```
Shifter[NP_, NPshell_, position_] := Module[{np = NP, nps = NPshell, sft, shell},
  sft = Abs[Min[RegionBounds[nps][[3]]]];
  np = Table[np[[x]] + {0, 0, sft}, {x, 1, Length[np]}];
  np = Table[np[[x]] + position, {x, 1, Length[np]}];
  shell = TransformedRegion[nps, TranslationTransform[{0, 0, sft}]];
  shell = TransformedRegion[shell, TranslationTransform[position]];
  {np, shell}]
```

```
PdNPonC = MapThread[Shifter, {PdNP[[All, 1]], PdNP[[All, 2]], NPTot[[2]]}];
```

```
PdCModel = Show[Region[#, BaseStyle → Gray] & /@ Teststack[[2]], Region[#, BaseStyle → RGBColor["#44E3BE"]] & /@ PdFXchains[[1, All, 3]],
  Region[#, BaseStyle → RGBColor["#44E3BE"]] & /@ PdNPonC[[All, 2]], ImageSize → Full, ViewPoint -> {-100, -50, 50}, Axes → True, AxesLabel → {"x", "y", "z"},
  AxesStyle → Thick, TicksStyle → Large, FontFamily → "Helvetica"]
```




```
MatLabRdY[element_, list_, rms_, o1_, o2_, o3_] := Module[{ls = list, e = element, r = rms},
  ls = Prepend[#, e] & /@ ls;
  ls = Append[#, rms] & /@ ls;
  ls = Append[#, o1] & /@ ls;
  ls = Append[#, o2] & /@ ls;
  ls = Append[#, o3] & /@ ls;
  ls]
```

Lets briefly calculate the rms for Au and Ge atoms respectively. As stated in the Trueblood et al., Acta Cryst. A52, 1996; one way to calculate the B-factor can be calculated via:
B = 8Pi^2*U, whereby is related to the mean square displacement by: U = <u^2>.
The following values for B are given:
B(Au, 293 K, fcc) = 0.57 (V. F. Sears and S. A. Shelley, Acta Cryst. A47, 1991).
B(Ge, 293 K, Diamond) = 0.57 (V. F. Sears and S. A. Shelley, Acta Cryst. A47, 1991).

Note that the automated function below does in some cases run into rendering instabilities model. Usually, it is sufficient to simply restart the code at the last loop successfully finished.

Set export directory

```
SetDirectory["/mydirectory"]
```

```
MultiModelGen[loops_] :=
 Module[{CStack, Cluster, NPchain, FXDClusters, NPdrp, NPTotal, PdNPs, PdTotal, CinNPs, ClsClean, CinClusters, CinCLusters, LstShift, PdTotalS,
   PdTotalSID, FXDClustersS, FXDClustersSID, CList, PdList, PdM, CM, Modellist},
  ParallelDo[
   CStack = FastSandwich[RandomInteger[{10, 11}] * 100, RandomInteger[{2, 5}], 40, 80];
   Cluster = {};
   Do[NPchain = ConnectedParticles[RandomInteger[{2, 5}]]; AppendTo[Cluster, NPchain], 2];
   FXDClusters = PlaceChains[Cluster, CStack[[1]], CStack[[2]]];
   NPdrp = DropNPs[CStack[[1]], CStack[[2]], RandomInteger[{10, 20}], {{}, {}, {}}, {}, FXDClusters];
   NPTotal = AllNPs[CStack[[1]], CStack[[2]], NPdrp, RandomInteger[{2, 3}], FXDClusters];
   PdNPs = GenNPs[NPTotal];
   PdNPs = MapThread[Shifter, {PdNPs[[All, 1]], PdNPs[[All, 2]], NPTotal[[2]]}];
   PdTotal = Flatten[Join[PdNPs[[All, 1]], FXDClusters[[1, All, 1]]], 1];
   CinNPs = RegionMember[#, CStack[[1]]] & /@ PdNPs[[All, 2]];
   CinNPs = Flatten[Pick[CStack[[1]], #] & /@ CinNPs, 1];
   ClsClean = Complement[CStack[[1]], CinNPs];
   CinClusters = RegionMember[#, ClsClean] & /@ FXDClusters[[1, All, 3]];
   CinClusters = Flatten[Pick[ClsClean, #] & /@ CinClusters, 1];
   ClsClean = Complement[ClsClean, CinClusters];
   LstShift = Abs[Part[#, 1] & /@ CoordinateBounds[{Part[#, 1] & /@ CoordinateBounds[ClsClean], Part[#, 1] & /@ CoordinateBounds[PdTotal]}]] + {1, 1, 1};
   PdTotalS = Table[(LstShift + # &) /@ PdNPs[[x, 1]], {x, 1, Length[PdNPs[[All, 1]]]}];
   PdTotalSID = Flatten[Table[Prepend[#, x] & /@ PdTotalS[[x]], {x, 1, Length[PdTotalS]}], 1];
   FXDClustersS = Map[(LstShift + # &), FXDClusters[[1, All, 2]], {3}];
   FXDClustersSID = Table[Flatten[Table[Prepend[#, x] & /@ FXDClustersS[[u, x]], {x, 1, Length[FXDClustersS[[u]]]}], 1], {u, 1, Length[FXDClustersS]}];
   FXDClustersSID = Table[Prepend[#, x] & /@ FXDClustersSID[[x]], {x, 1, Length[FXDClustersSID]}];
   FXDClustersSID = Flatten[FXDClustersSID, 1];
   CList = LstShift + # & /@ ClsClean;
   PdList = LstShift + # & /@ PdTotal;
   PdM = MatLabRdY[46, PdList, 0.085, 1, 0, 0];
   CM = MatLabRdY[6, CList, 0.08, 1, 0, 0];
   Modellist = Join[PdM, CM];
   Export["Model" <> ToString[n] <> ".h5", {"/Dataset1" → Modellist, "/Dataset2" → PdTotalSID, "/Dataset3" → FXDClustersSID}];
   ClearSystemCache[];, {n, 1, loops}]]
```

```
MultiModelGen[500]
```

```
ClearSystemCache[]
```

10.3 Multislice Simulations and Training Data

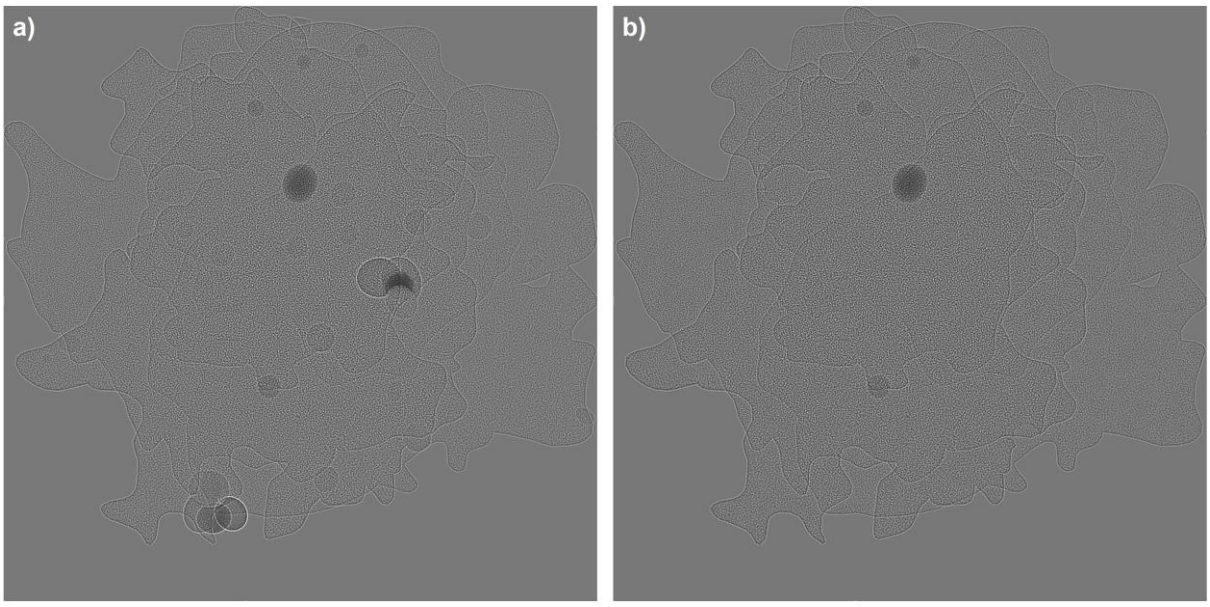


Figure 10.1: Comparison of an HRTEM multislice simulation before and after the application of the GLD criterion. **a)** Image of a Pd/C sample before application of the GLD criterion. **b)** Resimulated after application of the GLD criterion.

Code 10.3: MATLAB script to execute single HRTEM image simulations through the MULTEM implementation.

```
function [output] = hrtem_series(atoms, num_series, abr, sig_val)

output = cell(num_series,1);


%%%%%%%%%%%%%%%%% Load multem default parameter %%%%%%%$$%%%%%%%%%
input_multislice = multem_default_values();     % Load default values;

%%%%%%%%%%%%%%%%%%%% Set system configuration %%%%%%%%%%%%%%%%%%%%
system_conf.precision = 1;              % eP_Float = 1, eP_double = 2
system_conf.device = 2;              % eD_CPU = 1, eD_GPU = 2
system_conf.cpu_nthread = 1;
system_conf.gpu_device = 1;

%%%%%%%%%%%%%%%%%%%% Set simulation experiment %%%%%%%%%%%%%%%%%%%%
% eTEMST_STEM=11, eTEMST_ISTEM=12, eTEMST_CBED=21, eTEMST_CBEI=22, eTEMST_ED=31,
eTEMST_HRTEM=32, eTEMST_PED=41, eTEMST_HCTEM=42, eTEMST_EWFS=51, eTEMST_EWRS=52,
% eTEMST_EELS=61, eTEMST_EFTEM=62, eTEMST_ProbeFS=71, eTEMST_ProbeRS=72, eTEMST_PPFS=81,
eTEMST_PPRS=82,eTEMST_TFFS=91, eTEMST_TFRS=92
input_multislice.simulation_type = 32;

%%%%%%%%%%%%%% Electron-Specimen interaction model %%%%%%%%%%%%%%%%
input_multislice.interaction_model = 1;       % eESIM_Multislice = 1, eESIM_Phase_Object = 2,
eESIM_Weak_Phase_Object = 3
input_multislice.potential_type = 6;         % ePT_Doyle_0_4 = 1, ePT_Peng_0_4 = 2,
ePT_Peng_0_12 = 3, ePT_Kirkland_0_12 = 4, ePT_Weickenmeier_0_12 = 5, ePT_Lobato_0_12 = 6

%%%%%%%%%%%%%%%%%%%%%%% Potential slicing %%%%%%%%%%%%%%%%%%%%%%%%%
input_multislice.potential_slicing = 2;       % ePS_Planes = 1, ePS_dz_Proj = 2, ePS_dz_Sub =
3, ePS_Auto = 4

%%%%%%%%%%%%%% Electron-Phonon interaction model %%%%%%%%%%%%%%%%%%
```

```
input_multislice.pn_model = 3;            % ePM_Still_Atom = 1, ePM_Absorptive = 2,
ePM_Frozen_Phonon = 3
input_multislice.pn_coh_contrib = 0;
input_multislice.pn_single_conf = 0;         % 1: true, 0:false (extract single
configuration)
input_multislice.pn_nconf = 10;           % true: specific phonon configuration, false:
number of frozen phonon configurations
input_multislice.pn_dim = 110;            % phonon dimensions (xyz)
input_multislice.pn_seed = 300183;          % Random seed(frozen phonon)

% %%%%%%%%%%%%%%%%%%%%%% Specimen information %%%%%%%%%%%%%%%%%%%%%%
% % na = 16; nb = 16; nc = 20; ncu = 2; rms3d = 0.085;
%
% atoms(:, 2:4) = atoms(:, 2:4) - min(atoms(:, 2:4));
% box = max(atoms(:, 2:4));
%
% input_multislice.spec_atoms = atoms;
% input_multislice.spec_lx = box(1);
% input_multislice.spec_ly = box(2);
% input_multislice.spec_lz = box(3);
% input_multislice.spec_dz = 2;

c = 2;

%%%%%%%%%%%%%%%%%%%%% Specimen thickness %%%%%%%%%%%%%%%%%%%%%%%%%
input_multislice.thick_type = 1;           % eTT_Whole_Spec = 1, eTT_Through_Thick = 2,
eTT_Through_Slices = 3
input_multislice.thick = c:c:1000;          % Array of thickes (�)

%%%%%%%%%%%%%%%%%%%%% x-y sampling %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
input_multislice.nx = 2048;
input_multislice.ny = 2048;
input_multislice.bwl = 0;              % Band-width limit, 1: true, 0:false

%%%%%%%%%%%%%%%%%%%% Microscope parameters %%%%%%%%%%%%%%%%%%%%%%%%
input_multislice.E_0 = 300;             % Acceleration Voltage (keV)
input_multislice.theta = 0.0;            % Till ilumination (�)
input_multislice.phi = 0.0;             % Till ilumination (�)

%%%%%%%%%%%%%%%%%%%%% Illumination model %%%%%%%%%%%%%%%%%%%%%%%%%%
input_multislice.illumination_model = 2;       % 1: coherente mode, 2: Partial coherente
mode, 3: transmission cross coefficient, 4: Numerical integration
input_multislice.temporal_spatial_incoh = 1;     % 1: Temporal and Spatial, 2: Temporal, 3:
Spatial

%%%%%%%%%%%%%%%%%%%%%%%% condenser lens %%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%% source spread function %%%%%%%%%%
ssf_sigma = il_mrad_2_sigma(input_multislice.E_0, 0.02); % mrad to standard deviation
input_multislice.obj_lens_ssf_sigma = ssf_sigma;     % standard deviation: For parallel
ilumination(�^-1); otherwise (�)
input_multislice.obj_lens_ssf_npoints = 4;        % # of integration points. It will be only
used if illumination_model=4

%%%%%%%%%%%%%%%%%%%%%%%% Objective lens %%%%%%%%%%%%%%%%%%%%%%%%%%
input_multislice.obj_lens_m = 0;          % Vortex momentum
input_multislice.obj_lens_c_10 = abr.C0;      % Defocus (�)
```

```
input_multislice.cond_lens_c_21 = abr.B2;     % [B2] Axial coma (?)
input_multislice.cond_lens_phi_21 = abr.B2_theta; % [phi_B2] Azimuthal angle of axial coma
(?)
input_multislice.cond_lens_c_23 = 0.0;       % [A2] 3-fold astigmatism (?)
input_multislice.cond_lens_phi_23 = 0.0;      % [phi_A2]    Azimuthal angle of 3-fold
astigmatism (?)

input_multislice.obj_lens_c_30 = abr.C30;     % Third order spherical aberration (mm)
input_multislice.obj_lens_c_50 = 0.00;       % Fifth order spherical aberration (mm)
input_multislice.obj_lens_c_12 =abr.A2;      % Twofold astigmatism (?)
input_multislice.obj_lens_phi_12 = abr.A2_theta;  % Azimuthal angle of the twofold
astigmatism (?)
input_multislice.obj_lens_c_23 = abr.A3;      % Threefold astigmatism (?)
input_multislice.obj_lens_phi_23 = abr.A3_theta;  % Azimuthal angle of the threefold
astigmatism (?)
input_multislice.obj_lens_inner_aper_ang = 0.0;  % Inner aperture (mrad)
input_multislice.obj_lens_outer_aper_ang = 0.0;  % Outer aperture (mrad)

%%%%%%%%% defocus spread function %%%%%%%%%%%
dsf_sigma = il_iehwgd_2_sigma(sig_val);      % from defocus spread to standard deviation
input_multislice.obj_lens_dsf_sigma = dsf_sigma;  % standard deviation (?)
input_multislice.obj_lens_dsf_npoints = 5;     % # of integration points. It will be only
used if illumination_model=4

%%%%%%%%% zero defocus reference %%%%%%%%%%%
input_multislice.obj_lens_zero_defocus_type = 3;  % eZDT_First = 1, eZDT_User_Define = 2,
eZDT_Last = 3, eZDT_User_Define = 4
input_multislice.obj_lens_zero_defocus_plane = 0;

tic;
for i = 1:num_series
 %%%%%%%%%%%%%%%%%%%%%% Specimen information %%%%%%%%%%%%%%%%%%%%%%
% na = 16; nb = 16; nc = 20; ncu = 2; rms3d = 0.085;

% s = strcat(atoms_filepaths,sprintf('%02d', i),'.mat');
% load(s);
% atoms = test_atoms;

%atoms(:, 4) = -atoms(:, 4);
%atoms(:, 2:4) = atoms(:, 2:4) - min(atoms(:, 2:4));
box = max(atoms(:, 2:4));

input_multislice.spec_atoms = atoms;
input_multislice.spec_lx = max(box);
input_multislice.spec_ly = max(box);
input_multislice.spec_lz = box(3);
input_multislice.spec_dz = 2;


clear il_MULTEM;
output_multislice = il_MULTEM(system_conf, input_multislice);

output{i} = output_multislice;
end
toc;
```

```
% figure(1);
% for i=1:length(output_multislice.data)
%   imagesc(output_multislice.data(i).m2psi_tot);
%   title(strcat('Total intensity - Thick = ', num2str(output_multislice.thick(i))));
%   axis image;
%   colormap gray;
%   pause(0.25);
% end
end
```

Code 10.4: MATLAB looping script for repeated exectution of HRTEM multislice simulation over multiple models.

```
  model_folder = sprintf('yourmodelfolder/Run_%d',m); %Path to models
  cd(model_folder);
  modelfiles = dir('*.h5');

  output_folder= sprintf('yourmodelfolder/Run_%d',m);
  num_models = size(modelfiles,1);
  sim_loops = 2;

  script_folder = '/yourscriptfolder';
  all_output = cell(num_models, 1);

  for q = 1:sim_loops %desired number of loops
    cd(script_folder);
      for i = 1:num_models
         abr.C0 = (-300 -(-1600)).*rand(1)+(-1600); %aberration values relevant for
hrtem_series_nNPipe.m
         abr.C30 = (0.5-(-0.05)).*rand(1)+(-0.05);
         abr.A2 = 250*rand(1);
         abr.A2_theta = rand*360;
         abr.B2 = 500*rand(1);
         abr.B2_theta = rand*360;
         abr.A3 = 1200*rand(1);
         abr.A3_theta = rand*360;

         sig_val = (160 - 90).*rand(1) + 90;

         model = h5read([modelfiles(i).folder, filesep, modelfiles(i).name], '/Dataset1')';
         output = hrtem_series(model, 1, abr, sig_val);
         all_output{i,1} = output;
         all_output{i,2} = abr;
         all_output{i,3} = sig_val;
      end
    cd(output_folder);
    save(sprintf('SimRun_%d',q),'all_output', '-v7.3');
  end
```

Code 10.5: MATLAB script to apply GLD criterion on individual image simulations.

```
for g = 1
  %Mapping folders and models
  modelfolder = sprintf('/modelfolder/Run_%d',g);
  cd(modelfolder)
  models = dir('*.h5');
  refimg_folder = 'experimental_ref_images/Ref_images_for_MaskClean';
  %The following have been used as histogram reference:
  %002, 004, 006, 007, 021, 029, 030, 033, 042, 047, 049, 054,
  %056, 060
  cd(refimg_folder);
  refimgs = dir('*.tif');
  simfolder = sprintf('simrum_folder/Run_%d',g);
  cd(simfolder);

  for h = 1
    %Load the first batch of MULTEM simulations
    sprintf('We are at Run: %d, Sim: %d', g,h)
    simfile = sprintf('SimRun_%d.mat', h);
    load(simfile)
    clean_imask = cell(size(models,1),4);

    %Work through all models and simulations
    for i = 1:size(models,1)
      sprintf('Working on model: %d', i)
      pdc_data = pdc_dataloader(models(i));
      %Load a reference image for the image histogram.
      rand_ref = randi(size(refimgs,1));
      r_im = imread([refimgs(rand_ref).folder, '/', refimgs(rand_ref).name]);


      box = max(pdc_data{1,1}(:,2:4));
      l = max(box(1:2));
      scale = 12.134/(10/all_output{i,1}{1,1}.dx);

      %Get img, img_n and support mask
      %noise values are saved for the resimulation
      [img, img_n, sup_mask, noise] = noisemask_pd(all_output{i,1}{1,1}.data.m2psi_tot,
scale, l, r_im, pdc_data{1,end});

      %initialize label
      label = imbinarize(zeros(size(img_n,1)));

      %Test for every NP
      allidx.nps = [];
      for j = 1:max(pdc_data{1,2}(:,1))
        index = find(pdc_data{1,2}(:,1) == j);
        np_coor = pdc_data{1,2}(index(1):index(end),2:4);
        [graylevel_diff, np_mask] = np_quantrast_pd(np_coor, img_n, l, sup_mask);

        if graylevel_diff < 0.0250
          label(np_mask) = 0;
          pdc_data{1,1}(index(1):index(end),:) = [];
          pdc_data{1,2}(index(1):index(end),:) = [];
        else
          label(np_mask) = 1;
```

```
            allidx.nps = [allidx.nps, j];
          end
        end

        %If clusters were assigned
        if size(pdc_data,2) > 3
          allidx.cluster = [];
          for k = 1:max(pdc_data{1,3}(:,1))
            cl_index = find(pdc_data{1,3}(:,1) == k);
            all_npncl_coor = pdc_data{1,3}(cl_index(1):cl_index(end),2:5);
            cluster_coor = pdc_data{1,3}(cl_index(1):cl_index(end),3:5);

            [cluster_maskdil, cluster_mask] = clustermask_pd(cluster_coor, l, img_n, sup_mask);

            %The individual NPs will be tested against the cluster
            %surrounding instead of the individual surrounding.

            clusterdiffs = cell(max(all_npncl_coor(:,1)),1);

            for m = 1:max(all_npncl_coor(:,1))
              npncl_index = find(all_npncl_coor(:,1) == m);
              npncl_coor = all_npncl_coor(npncl_index(1):npncl_index(end),2:4);
              [npncl_graylevel_diff, npncl_mask] = np_quantrast_pd(npncl_coor, img_n, l,
sup_mask);
              clusterdiffs{m,1} = npncl_graylevel_diff;
            end

            if any(cell2mat(clusterdiffs) < 0.0250)
              label(cluster_maskdil) = 0;

pdc_data{1,1}((length(pdc_data{1,2})+cl_index(1)):(length(pdc_data{1,2})+cl_index(end)),:) =
[];
              pdc_data{1,3}(cl_index(1):cl_index(end),:)=[];
            else
              label(cluster_maskdil) = 1;
              allidx.cluster = [allidx.cluster, k];
            end
          end
        end

        %Now, a new model list should have been generated, which
        %only contains particles that can be seen well. The label
        %is already generated, just a new simulation is lacking.

        newmodel = pdc_data{1,1};
        abr = all_output{i,2};
        sig_val = all_output{i,3};
        newimg = hrtem_series(newmodel, 1, abr, sig_val);
        newimg = mat2gray(newimg{1,1}.data.m2psi_tot);
        newimg = imresize(newimg, scale);
        newimg = imhistmatch(newimg,r_im);
        newimg = newimg + noise.brtval;

        %Generate noise, but save values for regeneration.
        newimg_n = mat2gray(sim_noise(newimg, noise.img_mean, noise.beam_profile,
noise.bg_bp_r, noise.im_bg_r, noise.snr, 0.0));

        %clean_imask{i,1} = img;
```

```
      %clean_imask{i,2} = img_n;
      clean_imask{i,1} = newimg;
      clean_imask{i,2} = newimg_n;
      clean_imask{i,3} = double(label);
      clean_imask{i,4} = allidx;

      clear pdc_data
    end
    file = sprintf('SimRun_%d_Cclean_250',h);
    save(file, 'clean_imask','-v7.3');
    clear all_output
    clear clean_imask

  end
end
```

Code 10.6: MATLAB function 'sim_noise'. This function was written by Dr. Chen Huang.

```
function [img] = sim_noise(input_img, img_mean, beam_profile, bg_bp_r, im_bg_r, gn_snr,
pepper_pct)
  % input_img: input image.
  % img_mean: mean of the output image before adding noise.
  % beam_profile: the non-uniform illumination.
  % bg_bp_r: ratio uniform and non-uniform illumination (backgound).
  % im_bg_r: ratio of image contrast and background.
  % gn_snr: signal-to-noise ratio of additive white Gaussian noise.
  % pepper_pct: percentage of the pepper noise (dead pixels).

  img = mat2gray(input_img);

  background = ones(size(input_img));
  beam_profile = mat2gray(beam_profile);

  background = background * bg_bp_r + beam_profile;
  background = background/max(background(:));

  img = img * im_bg_r + background;
  img = img/max(img(:));

  img = awgn(img, gn_snr, 'measured');

  img = round(img / mean(img(:)) * img_mean);
  img = poissrnd(img);

  img = add_pepper_noise(img, pepper_pct, 0);

end
```

Code 10.7: MATLAB function 'add_pepper_noise'. This function was written by Dr. Chen Huang.

```
function [output_img] = add_pepper_noise(input_img, pepper_pct, dark_level)

  if nargin > 2
    pepper_value = dark_level;
  else
    pepper_value = min(input_img(:));
  end

  num_of_pixel = numel(input_img);
  num_of_pepper = round(pepper_pct * num_of_pixel);

  output_img = input_img;
  output_img(randperm(num_of_pixel, num_of_pepper)) = pepper_value;

end
```

Code 10.8: MATLAB function 'noisemask_pd' for the introduction of noise onto image simulations.

```
function [img, img_n, sup_mask, noise] = noisemask(sim, sc, boxlength, ref, all_np)

  img = mat2gray(sim);
  img = imresize(img, sc);
  img = imhistmatch(img,ref);
  noise.brtval = (0.1 - (-0.1)).*rand(1) + (-0.1);
  img = img + noise.brtval;

  %Generate noise, but save values for regeneration.
  noise.img_mean = randi([15,30],1)*10;
  noise.beam_profile = beam_bg(size(img,1), randi([5,25])*1000);
  noise.bg_bp_r = randi([5,50],1)/100;
  noise.im_bg_r = randi([3,10],1);
  noise.snr = randi([9,20],1)*10;

  %Generate noisy image
  img_n = sim_noise(img, noise.img_mean, noise.beam_profile, noise.bg_bp_r, noise.im_bg_r,
noise.snr, 0.0);
  img_n = mat2gray(img_n);

  %create support map
  allnpmask = zeros(size(img_n,1));

allnpmask(sub2ind(size(allnpmask),floor((all_np(:,2)/boxlength)*size(img_n,1)),floor((all_np(
:,1)/boxlength)*size(img_n,1)))) = ones(length(all_np),1);
  allnpmask = imbinarize(imclose(allnpmask,strel('disk',3)));
  sup_mask = imcomplement(allnpmask);
end
```

Code 10.9: MATLAB function 'beam_bg' for beam-like brightness variations.

```
function [im_beam] = beam_bg(sizeim, sigmf)
% Size of image:
 N = 128;
% A grid is needed for a two dimensional plot
 x1 = linspace(-(N/2),(N/2),N);
 x2 = x1;
 [X1,X2] = meshgrid(x1,x2);
% Create the background with a multidimensional Gaussian distribuition
 mu=[randi([-25,25],1);randi([-25,25],1)];
 sigma = [sigmf 0;0 sigmf];
 const= 1/((2*pi)^2/2)*sqrt(det(sigma));
 u=[X1(:)-mu(1) X2(:)-mu(2)];
 beam = const*exp(-0.5*diag(u*inv(sigma)*u'));
 beam = reshape(beam, size(X1));
% Resize the array to a desired size
 im_beam = imresize(beam, (sizeim/N),'bilinear');
% Normalize to 1 and apply desired factor
 im_beam=mat2gray(im_beam);
end
 %figure (1)
 %imshow(background)
 %surfc(nX1,nX2,background,'LineStyle', 'none')
```

Code 10.10: MATLAB function 'pdc_dataloader'.

```
function [sampledata] = pdc_dataloader(model)
   info = h5info([model.folder, filesep, model.name]);

   if size(info.Datasets,1) == 3
    sampledata = cell(1,4);
    %atoms
    sampledata{1,1} = h5read([model.folder, filesep, model.name], '/Dataset1')';
    %pdnp
    sampledata{1,2} = h5read([model.folder, filesep, model.name], '/Dataset2')';
    %pdcluster
    sampledata{1,3} = h5read([model.folder, filesep, model.name], '/Dataset3')';
    %all pd coordinates
    sampledata{1,4} =
sampledata{1,1}(1:(size(sampledata{1,2},1)+size(sampledata{1,3},1)),2:4);
   else
    sampledata = cell(1,3);
    %atoms
    sampledata{1,1} = h5read([model.folder, filesep, model.name], '/Dataset1')';
    %pdnp
    sampledata{1,2} = h5read([model.folder, filesep, model.name], '/Dataset2')';
    %all pd coordinates
    sampledata{1,3} = sampledata{1,1}(1:size(sampledata{1,2},1),2:4);
   end
end
```

Code 10.11: MATLAB function 'np_quantrast_pd'.

```
function [grayleveldiff, np_maskdil] = np_quantrast_pd(np_coor, img_n, boxlength, sup_mask)

  np_coor_sam = [max(floor((np_coor(:,1)/boxlength)*size(img_n,1)),1),
max(floor((np_coor(:,2)/boxlength)*size(img_n,1)),1)];
  np_mask = zeros(size(img_n,1));

np_mask(sub2ind(size(np_mask),floor((np_coor(:,2)/boxlength)*size(img_n,1)),floor((np_coor(:,
1)/boxlength)*size(img_n,1)))) = ones(length(np_coor),1);

  np_maskdil = imbinarize(imclose(np_mask,strel('disk',3)));
  np_maskinner = imerode(np_maskdil, strel('disk',6));

  min_point = min(np_coor_sam);
  max_point = max(np_coor_sam);

  bounds = [min_point(2)-50, max_point(2) + 50, min_point(1)-50, max_point(1) + 50];
  bounds = min(bounds, size(img_n,1));
  bounds = max(bounds, 1);

  masksur = imbinarize(zeros(size(img_n,1)));
  masksur(bounds(1):bounds(2), bounds(3):bounds(4)) = 1;
  sup_masksur = sup_mask;
  sup_masksur(imcomplement(masksur)) = 0;

  grayleveldiff = mean(img_n(sup_masksur)) - mean(img_n(np_maskinner));
end
```

Code 10.12: MATLAB function 'clustermask_pd'.

```
function [cluster_maskdil, cmask_local] = clustermask_pd(coor, boxlength, img_n, sup_mask)

  cluster_coor_sam = [max(floor((coor(:,1)/boxlength)*size(img_n,1)),1),
max(floor((coor(:,2)/boxlength)*size(img_n,1)),1)];
  cluster_mask = zeros(size(img_n,1));

cluster_mask(sub2ind(size(cluster_mask),floor((coor(:,2)/boxlength)*size(img_n,1)),floor((coo
r(:,1)/boxlength)*size(img_n,1)))) = ones(length(coor),1);
  cluster_maskdil = imbinarize(imclose(cluster_mask,strel('disk',3)));

  min_point = min(cluster_coor_sam);
  max_point = max(cluster_coor_sam);

  bounds = [min_point(2)-50, max_point(2) + 50, min_point(1)-50, max_point(1) + 50];
  bounds = min(bounds, size(img_n,1));
  bounds = max(bounds, 1);

  cluster_masksur = imbinarize(zeros(size(img_n,1)));
  cluster_masksur(bounds(1):bounds(2), bounds(3):bounds(4)) = 1;
  cmask_local = sup_mask;
  cmask_local(imcomplement(cluster_masksur)) = 0;
end
```

Code 10.13: MATLAB function 'crop_center'.

```
function [ im_new ] = crop_center( im , w_and_h)

  if(length(w_and_h)==1)
    w = round(w_and_h(1)) - 1;
    h = w;
  else
    w = round(w_and_h(1)) - 1;
    h = round(w_and_h(2)) - 1;
  end

  im_w = size(im, 2);
  im_h = size(im, 1);

  y0 = ceil((im_w - w)/2);
  x0 = ceil((im_h - h)/2);

  im_new = im(y0:y0+w, x0:x0+h);

end
```

Code 10.14: Mathematica notebook used for training the SegNet semantic segmention CNN. This notebook was originally written by Dr Chen Huang and adjusted to the needs of the work presented. The notebooks was used for both material systems.

SegNet training module for nNPipe

Copyright: Dr Chen Huang, The Rosalind Franklin Institute, Diamond Light Source, Harwell Research Campus.
Author: Dr Chen Huang

System setup

Cuda setup

```
In[•]:= Needs["CUDALink`"]

In[•]:= CUDADriverVersion[]

Out[•]= 470.74.0

In[•]:= CUDAResourcesInformation[]

In[•]:= CUDAResourcesInstall[Update → True]

In[•]:= SystemInformation[]

In[•]:= CUDAInformation[];
```

Network building

Encoder

```
In[•]:= encChain[index_, nLayers_, nChannels_] := Module[{tags, names, layers}, tags = Table[ToString[index] <> "_" <> ToString[j], {j, nLayers}];
          names = Append[Flatten[{"conv" <> #, "conv" <> # <> "bn", "relu" <> #} & /@ tags], "pool" <> ToString[index]];
          layers =
           Append[Flatten@Table[{ConvolutionLayer[nChannels, {3, 3}, "PaddingSize" → {1, 1}], BatchNormalizationLayer[], ElementwiseLayer[Ramp]},
              {nLayers}], PoolingLayer[{2, 2}, 2]];
          AssociationThread[names → layers]]

        encoder = NetChain[Join[encChain[1, 2, 64], encChain[2, 2, 128], encChain[3, 3, 256], encChain[4, 3, 512], encChain[5, 3, 512]],
          "Input" → NetEncoder[{"Image", {256, 256}, "Grayscale"}]]
```

Decoder

```
In[•]:= decChain[index_, nLayersmax_, nLayersmin_, nChannels_] := Module[{tags, names, layers, nlayers}, nlayers = nLayersmax - nLayersmin + 1;
          tags = Table[ToString[index] <> "_" <> ToString[j] <> "_" <> "D", {j, nLayersmax, nLayersmin, -1}];
          names = Flatten@Append[{"deconv" <> ToString[index], "deconv" <> ToString[index] <> "pad"},
             Flatten[{"conv" <> #, "conv" <> # <> "bn", "relu" <> #} & /@ tags]];
          layers = Flatten@Append[{DeconvolutionLayer[nChannels, {3, 3}, "Stride" → 2, "PaddingSize" → 1], PaddingLayer[{{0, 0}, {0, 1}, {0, 1}}]},
             Flatten@Table[{ConvolutionLayer[nChannels, {3, 3}, "PaddingSize" → {1, 1}], BatchNormalizationLayer[], ElementwiseLayer[Ramp]}, {nlayers}]];
          AssociationThread[names → layers]]

        decChain2[index_, nLayersmax_, nLayersmin_, nChannels_] := Module[{tags, names, layers, nlayers}, nlayers = nLayersmax - nLayersmin + 1;
          tags = Table[ToString[index] <> "_" <> ToString[j] <> "_" <> "D", {j, nLayersmax, nLayersmin, -1}];
          names = Flatten[{"conv" <> #, "conv" <> # <> "bn", "relu" <> #} & /@ tags];
          layers = Flatten@Table[{ConvolutionLayer[nChannels, {3, 3}, "PaddingSize" → {1, 1}], BatchNormalizationLayer[], ElementwiseLayer[Ramp]},
             {nlayers}];
          AssociationThread[names → layers]]

        decoder = NetChain[Join[decChain[5, 3, 1, 512], decChain[4, 3, 2, 512], decChain2[4, 1, 1, 256], decChain[3, 3, 2, 256],
           decChain2[3, 1, 1, 128], decChain[2, 2, 2, 128], decChain2[2, 1, 1, 64], decChain[1, 2, 2, 64]]]
```

Network chain with sigmoid layer

```
In[•]:= chain =
         NetChain[{encoder, decoder, ConvolutionLayer[1, {1, 1}, "PaddingSize" → 0],
           ElementwiseLayer[LogisticSigmoid, "Output" → NetDecoder[{"Image", "Grayscale"}]]}]
```

Data setup

Data location

```
In[•]:= datadir = SetDirectory["path_to_all_training_images/Segmentation"]
Out[•]= /media/rob/hdd1/kevin/AuGe_Neural_Network/Segmentation
```

Make datasets

```
In[•]:= loadFiles[dir_] := Map[File[#] &, FileNames["*tif", dir, Infinity]];
```

Network Input

```
In[•]:= dataFeature = loadFiles[FileNameJoin[{datadir, "img_n"}]];
```

Network Output (options)

```
In[•]:= dataLabel = loadFiles[FileNameJoin[{datadir, "label"}]]; dataNoiseless = loadFiles[FileNameJoin[{datadir, "img"}]];
In[•]:= Length /@ {dataFeature, dataLabel, dataNoiseless}
Out[•]= {480739, 480739, 480739}
```

Network training

Choose network type

```
In[•]:= netType = "feature_to_label";
```

Training setup

```
In[•]:= trainingInfo = <|"feature_to_label" → <|"GPUIdx" → 2, "trainingInput" → "dataFeature", "trainingOutput" → "dataLabel"|>|> // Dataset
```

Out[•]=

	GPUIdx	trainingInput	trainingOutput
feature_to_label	2	dataFeature	dataLabel

```
In[•]:= Ceiling[480739 * 0.1]
Out[•]= 48074
In[•]:= 480739 - 48074
Out[•]= 432665
In[•]:= dataset = RandomSample[Thread@Rule[ToExpression[trainingInfo[netType, "trainingInput"]], ToExpression[trainingInfo[netType, "trainingOutput"]]]];
        trainingData = Take[dataset, 432665];
        testData = Take[dataset, 48074];
```

Check if saving directory exists. If not true, create one.

```
In[•]:= netDir = FileNameJoin[{datadir, "nets", netType}]
In[•]:= DirectoryQ[netDir]
Out[•]= True
In[•]:= GPUIdx = trainingInfo[netType, "GPUIdx"]
Out[•]= 2
```

Pre-training check

This part checks if img_n and label are mapped to each other correctly.

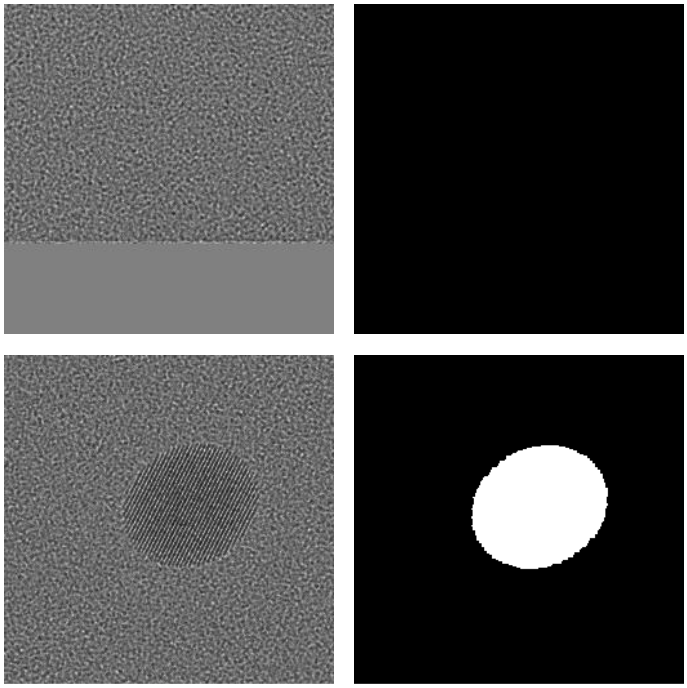
This part checks if img_n and label are mapped to each other correctly.

```
In[ ]:= RandomSample[trainingData, 3]
        RandomSample[testData, 3]
        Length /@ {trainingData, testData}
```

```
Out[ ]= {432 665, 48 074}
```

```
In[ ]:= Import[#] & /@ {Keys[#], Values[#]} & /@ RandomSample[trainingData, 2] // GraphicsGrid
```

Out[ ]=



Start training

```
In[ ]:= trained = NetTrain[chain, trainingData, All, ValidationSet → testData, MaxTrainingRounds → 200, TargetDevice → {"GPU", GPUIdx},
          BatchSize → 16, LearningRate → 0.001, TrainingProgressCheckpointing → {"Directory", netDir, "Interval" → Quantity[60, "Minutes"]}]
```

```
In[ ]:= Iconize[trained]
```

```
Out[ ]= NetTrainResultsObject[···]
```

Table 10.1: Hyperparameters applied during training for both CNNs and material systems.

	Object Detection module		Segmentation module	
Hyperparameters	lr0:	0.001	Learning rate:	0.001
	lrf:	0.2	Optimiser:	ADAM
	momentum:	0.937	Batch-size:	16
	weight_decay:	0.0005	Train/Val split:	90/10 %
	warmup_epochs:	3.0		
	warmup_momentum:	0.8		
	warmup_bias_lr:	0.1		
	box:	0.05		
	cls:	0.5		
	cls_pw:	1.0		
	obj:	1.0		
	obj_pw:	1.0		
	iou_t:	0.2		
	anchor_t:	4.0		
	fl_gamma:	0.0		
	hsv_h:	0.0		
	hsv_s:	0.0		
	hsv_v:	0.0		
	degrees:	0.0		
	translate:	0.1		
	scale:	0.5		
	shear:	0.0		
	perspective:	0.0		
	flipud:	0.0		
	fliplr:	0.0		
	mosaic:	0.0		
	mixup:	0.0		
	Epochs:	100		
	Train/Val split:	90/10 %		
	Optimiser:	ADAM		
	Batch-size:	8		

Code 10.15: MATLAB script ‘objectdect_label_nNPipe’ for the generartion of object detection labels based on generated computational models and pruned images simulations.

```
for j = 1
  modelfolder = sprintf('modelfolder/Run_%d',j);
  simfolder = sprintf('simulationfolder/Run_%d',j);
  for k = 1
    cd(simfolder);
    sprintf('We are at: Run %d, Sim %d',j,k)
    pack_file = sprintf('SimRun_%d_Cclean_250.mat',k);
    load(pack_file);
    sprintf('Pack%d loaded',k)

    cd(modelfolder)
    models = dir('*.h5');
    %change '3' in dependency whether you have vaccuum
    od_label = cell(size(models,1),3);

    for m = 1:size(models,1)
      sprintf('Working on Model %d now.',m)
      tic
      info = h5info([models(m).folder, filesep, models(m).name]);
```

```
      if size(info.Datasets,1) == 3
        atoms = h5read([models(m).folder, filesep, models(m).name], '/Dataset1')';
        pdnp = h5read([models(m).folder, filesep, models(m).name], '/Dataset2')';
        pdcluster = h5read([models(m).folder, filesep, models(m).name], '/Dataset3')';
      else
        atoms = h5read([models(m).folder, filesep, models(m).name], '/Dataset1')';
        pdnp = h5read([models(m).folder, filesep, models(m).name], '/Dataset2')';
      end

      box = max(atoms(:,2:4));
      l = max(box(1:2));
      sampsz = size(clean_imask{m,2},1);

      all_coor = atoms(:,2:4);
      all_coor_sam=[max(floor((all_coor(:,1)/l)*sampsz),1),
max(floor((all_coor(:,2)/l)*sampsz),1)];

      if size(info.Datasets,1) == 3
        c_atoms = atoms(length(pdnp)+length(pdcluster)+1:end,2:4);
      else
        c_atoms = atoms(length(pdnp)+1:end,2:4); %length(pdnp)+1 because the lists were
joined as {PdNP,Catoms} in MMA12.
      end
      c_atoms_sam = [max(floor((c_atoms(:,1)/l)*sampsz),1),
max(floor((c_atoms(:,2)/l)*sampsz),1)];

      np_data = cell(max(pdnp(:,1)),2);
      for n = clean_imask{m,4}.nps
        index = find(pdnp(:,1) == n);
        np_coor = pdnp(index(1):index(end),2:4);
        np_coor_sam = [max(floor((np_coor(:,1)/l)*sampsz),1),
max(floor((np_coor(:,2)/l)*sampsz),1)];
        np_mask = zeros(sampsz);

np_mask(sub2ind(size(np_mask),floor((np_coor(:,2)/l)*sampsz),floor((np_coor(:,1)/l)*sampsz)))
= ones(length(np_coor),1);
        np_mask = imbinarize(imclose(np_mask,strel('disk',3)));

        %define boxes
        centre = round(mean(np_coor_sam));
        dx = max(np_coor_sam(:,1))-min(np_coor_sam(:,1));
        dy = max(np_coor_sam(:,2))-min(np_coor_sam(:,2));
        corner = max(floor([centre(1)-(dx/2), centre(2)-(dy/2)]),1);
        np_data{n,1} = [corner(1), corner(2), dx, dy];
        %rectangle data for matlab: x,y are in top left corner.
        np_data{n,2} = [centre(1)/sampsz, centre(2)/sampsz, dx/sampsz, dy/sampsz];
        %Bbox with x,y in centre yolo style label
      end
      np_data = np_data(~all(cellfun('isempty',np_data),2),:);

      %As the foundation pair of bboxes and the image is defined,
      %augmentation can follow.

      %Rotation and concatenate cells
      for r = [90,180,270]
        np_data = rotateLabel(np_data, r, clean_imask{m,2});
      end
```

```
      %Horizontal flip
      bboxes = cell2mat(np_data(:,1));
      tform = affine2d([-1 0 0;0 1 0; 0 0 1]);
      rout = affineOutputView(size(clean_imask{m,2}),tform);
      bboxHF = bboxwarp(bboxes, tform, rout);
      %this is the matlab rectangle format.
      bboxHFyolo = [bboxHF(:,1) + (bboxHF(:,3)/2), bboxHF(:,2) + (bboxHF(:,4)/2),
bboxHF(:,3), bboxHF(:,4)]/size(clean_imask{m,2},1);
      %and the yolo format accordingly.
      np_dataHF = cat(2, num2cell(bboxHF, 2), num2cell(bboxHFyolo, 2));

      %Rotation and concatenate cells with horizontally flipped boxes
      for r = [90,180,270]
        np_dataHF = rotateLabel(np_dataHF, r, clean_imask{m,2});
      end

      od_label{m,1} = cat(2, np_data, np_dataHF);

      clustercheckpoint = exist('pdcluster', 'var');
      if clustercheckpoint == 1
        cluster_data = cell(max(pdcluster(:,1)),2);
        for o = clean_imask{m,4}.cluster
          cl_index = find(pdcluster(:,1) == o);
          cluster_coor = pdcluster(cl_index(1):cl_index(end),3:5);
          cluster_coor_sam = [max(floor((cluster_coor(:,1)/l)*sampsz),1),
max(floor((cluster_coor(:,2)/l)*sampsz),1)];
          cluster_mask = zeros(sampsz);

cluster_mask(sub2ind(size(cluster_mask),floor((cluster_coor(:,2)/l)*sampsz),floor((cluster_co
or(:,1)/l)*sampsz))) = ones(length(cluster_coor),1);
          cluster_mask = imbinarize(imclose(cluster_mask,strel('disk',3)));

          %define boxes
          dx = max(cluster_coor_sam(:,1))-min(cluster_coor_sam(:,1));
          dy = max(cluster_coor_sam(:,2))-min(cluster_coor_sam(:,2));
          centre = round([min(cluster_coor_sam(:,1))+(dx/2),
min(cluster_coor_sam(:,2))+(dy/2)]);
          corner = max(floor([centre(1)-(dx/2), centre(2)-(dy/2)]),1);
          cluster_data{o,1} = [corner(1), corner(2), dx, dy];
          %rectangle data for matlab: x,y are in top left corner.
          cluster_data{o,2} = [centre(1)/sampsz, centre(2)/sampsz, dx/sampsz, dy/sampsz];
          %Bbox with x,y in centre yolo style label
        end
        cluster_data = cluster_data(~all(cellfun('isempty',cluster_data),2),:);

        %Same procedure as for NPs
        %Rotation and concatenate cells
        for r = [90,180,270]
          cluster_data = rotateLabel(cluster_data, r, clean_imask{m,2});
        end

        %Horizontal flip
        c_bboxes = cell2mat(cluster_data(:,1));
        c_tform = affine2d([-1 0 0;0 1 0; 0 0 1]);
        c_rout = affineOutputView(size(clean_imask{m,2}),c_tform);
        c_bboxHF = bboxwarp(c_bboxes, c_tform, c_rout);
        %this is the matlab rectangle format.
```

```
        c_bboxHFyolo = [c_bboxHF(:,1) + (c_bboxHF(:,3)/2), c_bboxHF(:,2) + (c_bboxHF(:,4)/2),
c_bboxHF(:,3), c_bboxHF(:,4)]/size(clean_imask{m,2},1);
        %and the yolo format accordingly.
        cluster_dataHF = cat(2, num2cell(c_bboxHF, 2), num2cell(c_bboxHFyolo, 2));

        %Rotation and concatenate cells with horizontally flipped boxes
        for r = [90,180,270]
          cluster_dataHF = rotateLabel(cluster_dataHF, r, clean_imask{m,2});
        end

        od_label{m,2} = cat(2, cluster_data, cluster_dataHF);

      elseif clustercheckpoint == 0
        od_label{m,2} = [];
      end

      clear atoms
      clear pdnp
      if clustercheckpoint == 1
        clear pdcluster
      end
    end
    cd(simfolder)
    file = sprintf('Obdect_Boxes_c250_%d',k);
    save(file,'od_label','-v7.3');
    clear clean_imask
  end
end
```

Code 10.16: MATLAB script 'YOLO_labelgen_nNPipe' for the to convert object detection labels into a YOLO compatible format.

```
for j = 1
  model_folder = sprintf('modelfolder/Run_%d',j);
  cd(model_folder)
  models = dir('*.h5');
  sim_folder = sprintf('simulationsfolder/Run_%d',j);
  img_folder = 'mypath/images';
  label_folder = 'mypath/labels';
  sizedat_folder = 'mypath/size_data';

  for k = 1
    cd(sim_folder)
    sim_file = sprintf('SimRun_%d_Cclean_250.mat',k);
    load(sim_file);
    label_file = sprintf('Obdect_Boxes_c250_%d.mat',k);
    load(label_file);
    parfor m = 1:size(models,1)
      %The 'NoFlip' scenario
      for rot = [0, 90, 180, 270]
        cd(img_folder)
        img_name = sprintf('R%dM%dS%d_NF%d.tif',j,m,k,rot);
        tform = affine2d([cosd(rot) sind(rot) 0; -sind(rot) cosd(rot) 0; 0 0 1]);
        img = imwarp(clean_imask{m,2},tform);
        imwrite(img, img_name, 'tif', 'Compression', 'none');
        cd(label_folder)
        labelfile = sprintf('R%dM%dS%d_NF%d.txt',j,m,k,rot);
```

```
        lb_assign = (rot/45) + 2;
        fileID = fopen(labelfile, 'w');
        if isempty(od_label{m,1})
          sprintf('No NPs in model %d',m);
        elseif ~isempty(od_label{m,1})
          cd(label_folder)
          for o = 1:size(od_label{m,1},1)
            IDnp = [0, od_label{m,1}{o,lb_assign}];
            fprintf(fileID, '%d %f %f %f %f\n', IDnp);
          end
        end
        if isempty(od_label{m,2})
          sprintf('No clusters in model %d',m);
        elseif ~isempty(od_label{m,2})
          for p = 1:size(od_label{m,2},1)
            IDc = [1, od_label{m,2}{p,lb_assign}];
            fprintf(fileID, '%d %f %f %f %f\n', IDc);
          end
        end
        fclose(fileID);
      end
      %The 'Horizontal flip' scenario
      for rot2 = [0, 90, 180, 270]
        cd(img_folder)
        img_hf_name = sprintf('R%dM%dS%d_HF%d.tif',j,m,k,rot2);
        hftform = affine2d([-1 0 0;0 1 0; 0 0 1]);
        img_hf = imwarp(clean_imask{m,2},hftform);
        tform = affine2d([cosd(rot2) sind(rot2) 0; -sind(rot2) cosd(rot2) 0; 0 0 1]);
        img_hf = imwarp(img_hf,tform);
        imwrite(img_hf, img_hf_name, 'tif', 'Compression', 'none');
        cd(label_folder)
        labelfile_hf = sprintf('R%dM%dS%d_HF%d.txt',j,m,k,rot2);
        lb_assign_hf = (rot2/45) + 10;
        fileID = fopen(labelfile_hf, 'w');
        if isempty(od_label{m,1})
          sprintf('No NPs in model %d',m);
        elseif ~isempty(od_label{m,1})
          for o = 1:size(od_label{m,1},1)
            IDnp = [0, od_label{m,1}{o,lb_assign_hf}];
            fprintf(fileID, '%d %f %f %f %f\n', IDnp);
          end
        end
        if isempty(od_label{m,2})
          sprintf('No clusters in model %d',m);
        elseif ~isempty(od_label{m,2})
          for p = 1:size(od_label{m,2},1)
            IDc = [1, od_label{m,2}{p,lb_assign_hf}];
            fprintf(fileID, '%d %f %f %f %f\n', IDc);
          end
        end
        fclose(fileID);
      end
    end
    clear od_label
    clear clean_imask
  end
end
```

Code 10.17: MATLAB script 'train_val_split_yolov5_nNPipe' to split the training data into a 90/10 % traning/test split.

```
image_folder = 'mypath/images';
label_folder = 'mypath/labels';
cd(image_folder)
training_files = dir('*.tif');
trainfiles = struct2cell(training_files)';
valid_files = randsample(training_files, round(size(training_files,1) * 0.1), false);
valfiles = struct2cell(valid_files)';

idx = find(ismember(trainfiles(:,1), valfiles(:,1)));
training_files(idx) = [];
trainfiles = struct2cell(training_files)';

%save training and val images

%train files
img_train_folder = 'mypath/train';
cd(image_folder)
parfor h = 1:size(training_files,1)
  imgtr = [training_files(h).folder, filesep, training_files(h).name];
  [status, msg] = copyfile(imgtr, img_train_folder);
end

%val files
img_val_folder = 'mypath/val';
cd(image_folder)
parfor i = 1:size(valid_files,1)
  imgv = [valid_files(i).folder, filesep, valid_files(i).name];
  [status, msg] = copyfile(imgv, img_val_folder);
end


label_train_folder = 'mypath/train';
cd(label_folder)
parfor j = 1:size(training_files,1)
  trfile = [label_folder, filesep, training_files(j).name];
  trfile = strrep(trfile, '.tif', '.txt');
  [status, msg] = copyfile(trfile, label_train_folder);
end


label_val_folder = 'mypath/val';
cd(label_folder)
parfor k = 1:size(valid_files,1)
  vfile = [label_folder, filesep, valid_files(k).name];
  vfile = strrep(vfile, '.tif', '.txt');
  [status, msg] = copyfile(vfile, label_val_folder);
end
```

Code 10.18: MATLAB script 'Seg_Traindata_Gen_nNPipe' to generate training and validation data for SegNet.

```
for j = 1
  modelfolder = sprintf('modelfolder/Run_%d',j);
  simfolder = sprintf('simulationfolder/Run_%d',j);

  for k = 1
    %Where to store data
    img_folder = sprintf('cleanimages_folder/SimRun_%d',j,k);
    img_n_folder = sprintf('noisyimages_folder/SimRun_%d',j,k);
    label_folder = sprintf('label_folder/SimRun_%d',j,k);
    cd(simfolder)
    pack_file = sprintf('SimRun_%d_Cclean_250.mat',k);
    load(pack_file);
    sprintf('Pack_%d loaded',k)

    cd(modelfolder)
    models=dir('*.h5');
    sprintf('Working on: Run_%d, Sim_Run_%d', j, k)
    tic
    for m = 1:size(models,1)
      traindat = clean_imask(m,:);

      %Load a model
      atoms = h5read([models(m).folder, filesep, models(m).name], '/Dataset1')';
      pdnp = h5read([models(m).folder, filesep, models(m).name], '/Dataset2')';

      box = max(atoms(:,2:4));
      l = max(box(1:2));

      %Add padding for image edges
      pad = [256, 256];
      sampsz = size(traindat{1,2},1);

      vac_val = 0.5;
      img = traindat{1,1};
      img_pad = padarray(img, pad, vac_val, 'both');
      img_n = traindat{1,2};
      img_n_pad = padarray(img_n, pad, vac_val, 'both');
      mask = traindat{1,3};
      mask_pad = padarray(mask, pad, 0, 'both');
      data = cell(max(traindat{1,4}.nps),1);

      %Generate images on model
      if ~isempty(traindat{1,4}.nps)
         for n = traindat{1,4}.nps
           index = find(pdnp(:,1) == n);
           np_coor = pdnp(index(1):index(end),2:4);
           np_coor_sam = [max(floor((np_coor(:,1)/l)*sampsz),1),
max(floor((np_coor(:,2)/l)*sampsz),1)];
           np_mask = zeros(sampsz);

np_mask(sub2ind(size(np_mask),floor((np_coor(:,2)/l)*sampsz),floor((np_coor(:,1)/l)*sampsz)))
= ones(length(np_coor),1);
           np_maskdil = imbinarize(imclose(np_mask,strel('disk',3)));
           np_maskinner = imerode(np_maskdil, strel('disk',6));
           np_mask_pad = padarray(np_maskdil, pad, 0, 'both');
```

```
      if mean(mask(np_maskdil)) > 0.85
        data{n,1} = cell(3,3);
      end

      if mean(mask(np_maskdil)) > 0.85
       data{n,1} = cell(3,3);
       [x, y] = find(np_mask_pad == 1);
       xy = [y, x];
       centre = floor(mean(xy));
       cut_area = [centre(1)-150, centre(2)-150, 300, 300];
       np_area_img = imcrop(img_pad, cut_area);
       np_area_img_n = imcrop(img_n_pad, cut_area);
       np_area_mask = imcrop(mask_pad, cut_area);
       win1 = randomCropWindow2d(size(np_area_img), [256 256]);
       win2 = randomCropWindow2d(size(np_area_img), [256 256]);
       theta1 = 90 * randi([0,1]);
       theta2 = 90 * randi([2,3]);
       data{n,1}{1,1} = imcrop(np_area_img, win1);
       data{n,1}{2,1} = imcrop(np_area_img_n, win1);
       data{n,1}{3,1} = double(imcrop(np_area_mask, win1));
       data{n,1}{1,2} = imrotate(imcrop(np_area_img, win2), theta1);
       data{n,1}{2,2} = imrotate(imcrop(np_area_img_n, win2), theta1);
       data{n,1}{3,2} = double(imrotate(imcrop(np_area_mask, win2), theta1));
       data{n,1}{1,3} = imrotate(imcrop(np_area_img, win2), theta2);
       data{n,1}{2,3} = imrotate(imcrop(np_area_img_n, win2), theta2);
       data{n,1}{3,3} = double(imrotate(imcrop(np_area_mask, win2), theta2));
      end
    end
  end
  data(:,2) = cell(size(data,1),1);
  killval = 1;

  %Write images
  for o = 1:size(data,1)
    while isempty(data{o,2}) == 1
      rwin = randomCropWindow2d(size(img_n_pad), [256 256]);
      testwindow = imcrop(mask_pad,rwin);
      if any(testwindow(:) == killval)
       data{o,2} = [];
      else
       data{o,2} = cell(3,1);
       phi = 90 * randi([0,3]);
       data{o,2}{1,1} = imrotate(imcrop(img_pad, rwin), phi);
       data{o,2}{2,1} = imrotate(imcrop(img_n_pad, rwin), phi);
       data{o,2}{3,1} = double(imrotate(imcrop(mask_pad, rwin), phi));
      end
    end
  end

  for p = 1:size(data,1)
    for c = 1:size(data{p,1},2)
      img_name = sprintf('R%dS%dM%dP%dimg_%d.tif',j,k,m,p,c);
      cd(img_folder);
      imwrite(data{p,1}{1,c},img_name,'tif','Compression','none');
      img_n_name = sprintf('R%dS%dM%dP%dimg_n_%d.tif',j,k,m,p,c);
      cd(img_n_folder);
      imwrite(data{p,1}{2,c},img_n_name,'tif','Compression','none');
```

```
            label_name = sprintf('R%dS%dM%dP%dlabel_%d.tif',j,k,m,p,c);
            cd(label_folder);
            imwrite(data{p,1}{3,c},label_name,'tif','Compression','none');
          end
        end

        for pp = 1:size(data,1)
           for cc = 1:size(data{pp,2},2)
             img_name = sprintf('R%dS%dM%dE%dimg_%d.tif',j,k,m,pp,cc);
             cd(img_folder);
             imwrite(data{pp,2}{1,cc},img_name,'tif','Compression','none');
             img_n_name = sprintf('R%dS%dM%dE%dimg_n_%d.tif',j,k,m,pp,cc);
             cd(img_n_folder);
             imwrite(data{pp,2}{2,cc},img_n_name,'tif','Compression','none');
             label_name = sprintf('R%dS%dM%dE%dlabel_%d.tif',j,k,m,pp,cc);
             cd(label_folder);
             imwrite(data{pp,2}{3,cc},label_name,'tif','Compression','none');
           end
        end
      end
      clear clean_imask
      toc
    end
end
```

Table 10.2: Post-processing routines applied on the 256 x 256 SegNet output for both considered material systems.

Au/Ge	Pd/C
i) Binarisation; thres. = 0.999 ii) Delete small components iii) Delete border components iv) Filling transform	i) Binarisation, thres. = 0.999 ii) Water-shedding iii) Dilation; $r_{Diskmatrix} = 7$ iv) Delete border components v) Filling transform vi) Erosion; $r_{Diskmatrix} = 7$ vii) Circularity Filtering; thres. = 0.8

Code 10.19: Mathematica 12.1 script for semantic segmentation of Au/Ge image sections and reconstruction of binary maps.

2k nNPipe Image Reconstruction – Au/Ge

A notebook written by Kevin P Treder, University of Oxford. Copyright Kevin P Treder.

In this notebook, results from the object detection by yolov5 are directly loaded into this notebook and subsequently processed by the SegNet module, effectively forming nNPipe.

Set directories, Import YOLO output

Image directory

```
ImgDir = "/mylabeldir/expX";
```

Boxes directory

```
ObDectDir = "/mylabeldir/expX";
```

```
In[•]:= DataImporter[ImDir_, BoxDir_, Pos_] :=
 Block[{imgls, imgfile, boxfile, imgname, boxfilename, img},
  imgls = FileNames[ImDir <> "*.tif"];
  imgname = imgls[[Pos]];
  boxfilename =
   BoxDir <> "/" <> StringTrim[StringSplit[imgname, "/"][[-1]], ".tif"] <> ".txt";
  img = Import[imgname];
  boxfile = Import[boxfilename, "Data"];
  {img, imgname, boxfile, boxfilename}
 ]
```

Load trained SegNet:

```
NetDir =
  "/myweightsdir/";

AuNet = Import[NetDir <> "/" <> "au_segnet_weights.wlnet"]
```

```
Out[•]= NetChain[ <image> ]
```

Input port: image
Output port: image

Inference and Reconstruction:

Load an image:

```
In[•]:= tfile = DataImporter[ImgDir, ObDectDir, 34];
```

The image needs to be padded, so particles on the image edges can be considered as well.

```
In[ ]:= tpadim = ImagePad[tfile[[1]], 128, GrayLevel[0.5]];
```

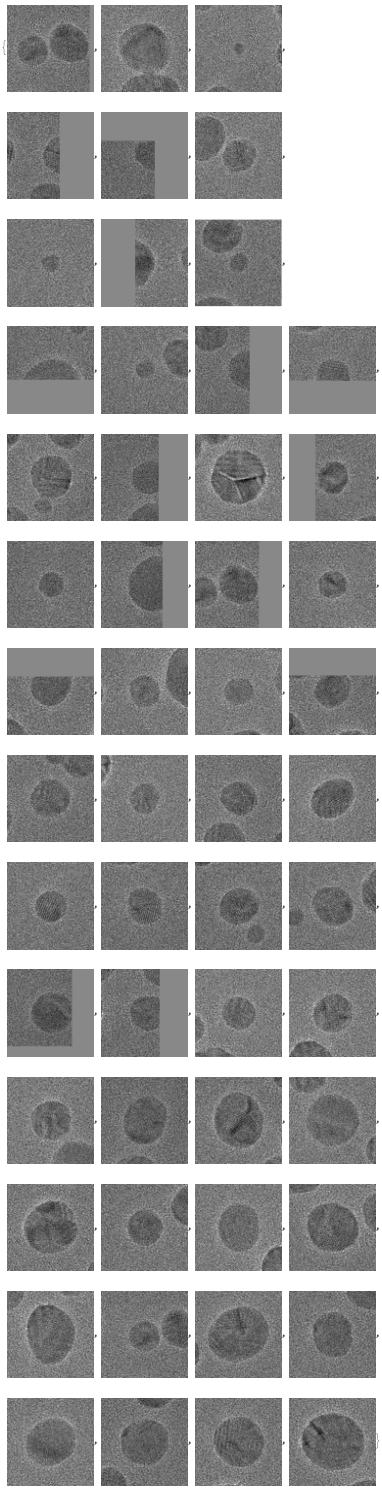
```
In[ ]:= tnpls = Pick[tfile[[3]], tfile[[3, All, 1]], 0];
```

```
In[ ]:= tnpcoors = Floor[Delete[#, 1] & /@ tnpls * 2048];
```

Let's have a look on the 256x256 output of the YOLOv5 module:

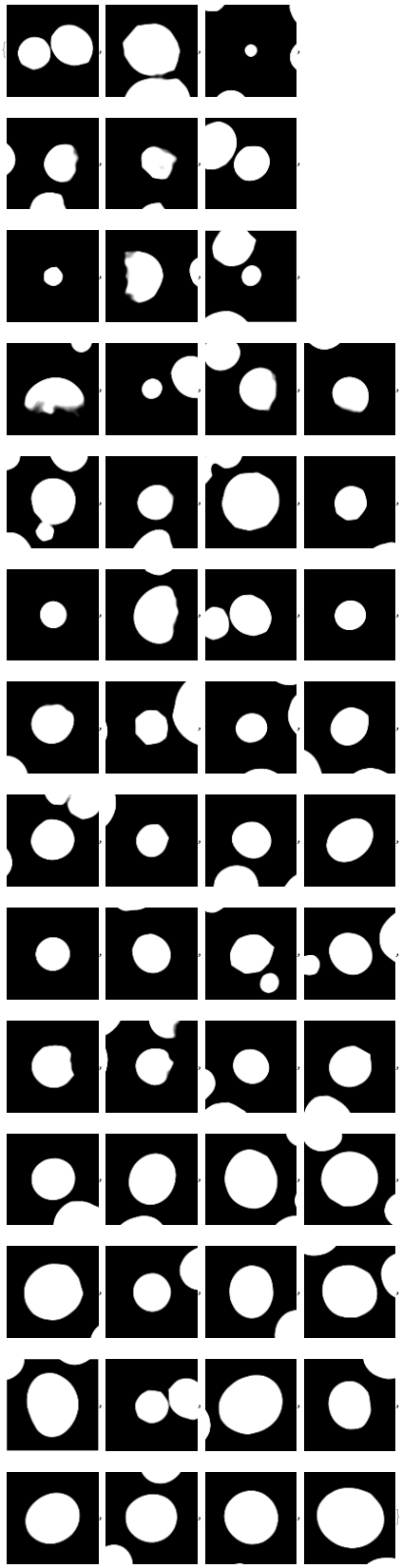
```
In[ ]:= tnpimgs =
         ImageTake[tpadim, IntegerPart[{#[[2]], #[[2]] + 255}],
            IntegerPart[{#[[1]], #[[1]] + 255}]] & /@ tnpcoors
```

Out[ ]=



```
In[ ]:= tgpuseg = AuNet[tnpimgs, TargetDevice → {"GPU", 1}, BatchSize → 8]
```

Out[ ]=

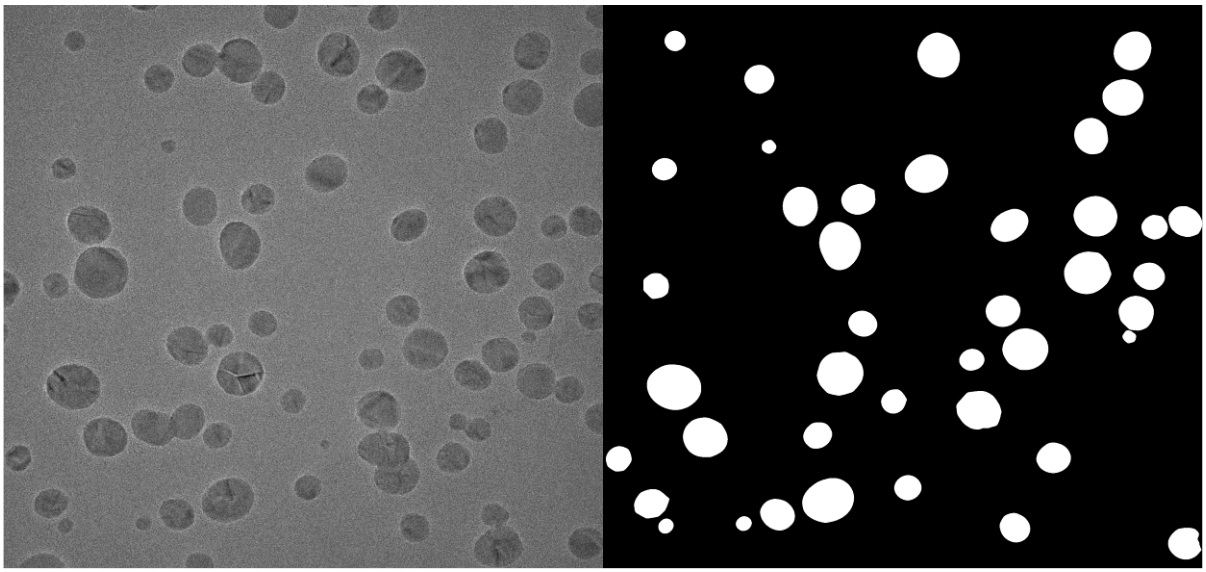


Getting a 2048 x 2048 binary map:

```
AssembleAU[Segs_, Coors_] :=
 Block[{s1, s2, s3, s4, s5, s6, s7, Coorsmod, b, NPLs, SegIM},
  s1 = Binarize[#, 0.999] & /@ Segs;
  s2 = DeleteSmallComponents[#, 1000] & /@ s1;
  s3 = DeleteBorderComponents /@ s2;
  s4 = FillingTransform /@ s3;
  s5 = RemoveBackground /@ s4;
  Coorsmod = Table[{Coors[[x, 1]], 2048 - Coors[[x, 2]]}, {x, 1, Length@Coors}];
  b = Image[ConstantArray[0, {2048, 2048}]];
  NPLs = Transpose[{s5, Coorsmod}];
  SegIM = ImageCompose[b, #1, #2] &[NPLs[[All, 1]], NPLs[[All, 2]]];
  SegIM = Binarize[SegIM, 0.99];
  SegIM = DeleteBorderComponents[SegIM]]
```

```
In[•]:= FullSeg = AssembleAU[tgpuseg, tnpcoors];
```

```
In[•]:= ImageCollage[{tfile[[1]], FullSeg}]
```



Process all images:

```
SetDirectory["/myoutputdirectory"]
```

```
In[•]:= NNSegmentation[IMPath_, LbPath_] :=
 Module[{imgs, file, padim, npls, npcoors, npimgs, gpuseg, fullim, imname},
  imgs = FileNames[IMPath <> "*.tif"];
  Do[
   file = DataImporter[IMPath, LbPath, n];
   padim = ImagePad[file[[1]], 128, GrayLevel[0.5]];
   npls = Pick[file[[3]], file[[3, All, 1]], 0];
   npcoors = Floor[Delete[#, 1] & /@ npls * 2048];
   npimgs =
    ImageTake[padim, IntegerPart[{#[[2]], #[[2]] + 255}],
       IntegerPart[{#[[1]], #[[1]] + 255}]] & /@ npcoors;
   gpuseg = AuNet[npimgs, TargetDevice → {"GPU", 1}, BatchSize → 8];
   fullim = AssembleAU[gpuseg, npcoors];
   imname = "0" <> ToString[n] <> "_NN.tif";
   Export[imname, fullim];, {n, 1, Length@imgs}];]
```

```
In[•]:= NNSegmentation[ImgDir, ObDectDir]; // AbsoluteTiming
```

Code 10.20: Mathematica 12.1 script for semantic segmentation of Au/Ge image sections and reconstruction of binary maps.

2k nNPipe Image Reconstruction – Pd/C

A notebook written by Kevin P Treder, University of Oxford. Copyright Kevin P Treder.

In this notebook, results from the object detection by yolov5 are directly loaded into this notebook and subsequently processed by the SegNet module, effectively forming nNPipe.

Set directories, Import YOLO output

Image directory:

```
ImgDir = "/myimagedir"
```

Boxes directory:

```
ObDectDir = "/mylabeldir/expX";
```

```
In[•]:= FileNames[ImgDir <> "*.tif";];
```

```
In[•]:= DataImporter[ImDir_, BoxDir_, Pos_] := Block[{imgls, imgfile, boxfile, imgname, boxfilename, img},
   imgls = FileNames[ImDir <> "*.tif"];
   imgname = imgls[[Pos]];
   boxfilename = BoxDir <> "/" <> StringTrim[StringSplit[imgname, "/"][[-1]], ".tif"] <> ".txt";
   img = Import[imgname];
   boxfile = Import[boxfilename, "Data"]; {img, imgname, boxfile, boxfilename}
  ]
```

Load the network:

```
NetDir =
  "/myweightsdir/";

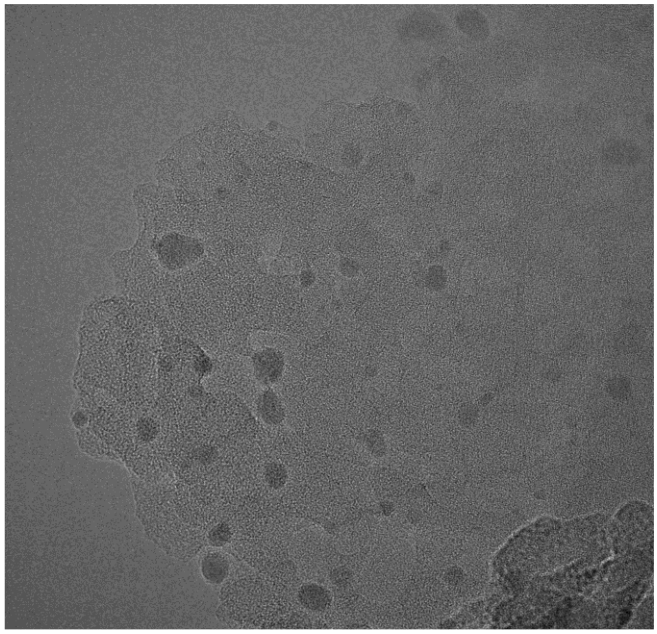
PdNet = Import[NetDir <> "/" <> "pd_segnet_weights.wlnet"];
```

Let's pick an image to analyse

```
In[•]:= tfile = DataImporter[ImgDir, ObDectDir, 12];
```

```
In[•]:= Show[tfile[[1]], ImageSize → 400]
```

Out[•]=



The image needs to be padded, so particles on the image edges can be considered as well.

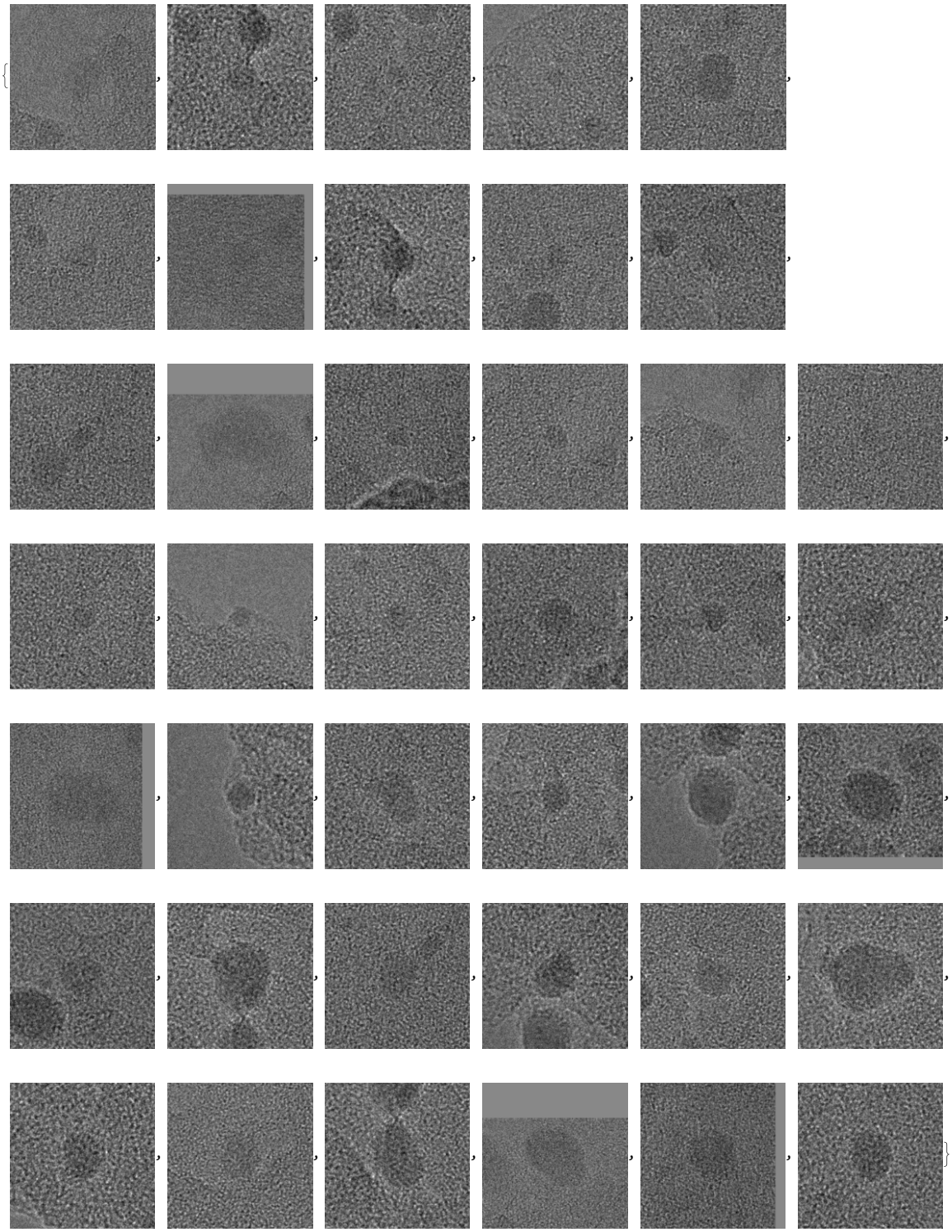
In[•]:=
```
tpadim = ImagePad[tfile[[1]], 128, GrayLevel[0.5]];
npls = Pick[tfile[[3]], tfile[[3, All, 1]], 0];
```

Reshaping coordinates

In[•]:=
```
tnpcoors = Floor[Delete[#, 1] & /@ npls * 2048];
```

Let's have a look on the 256x256 output of the YOLOv5 module:

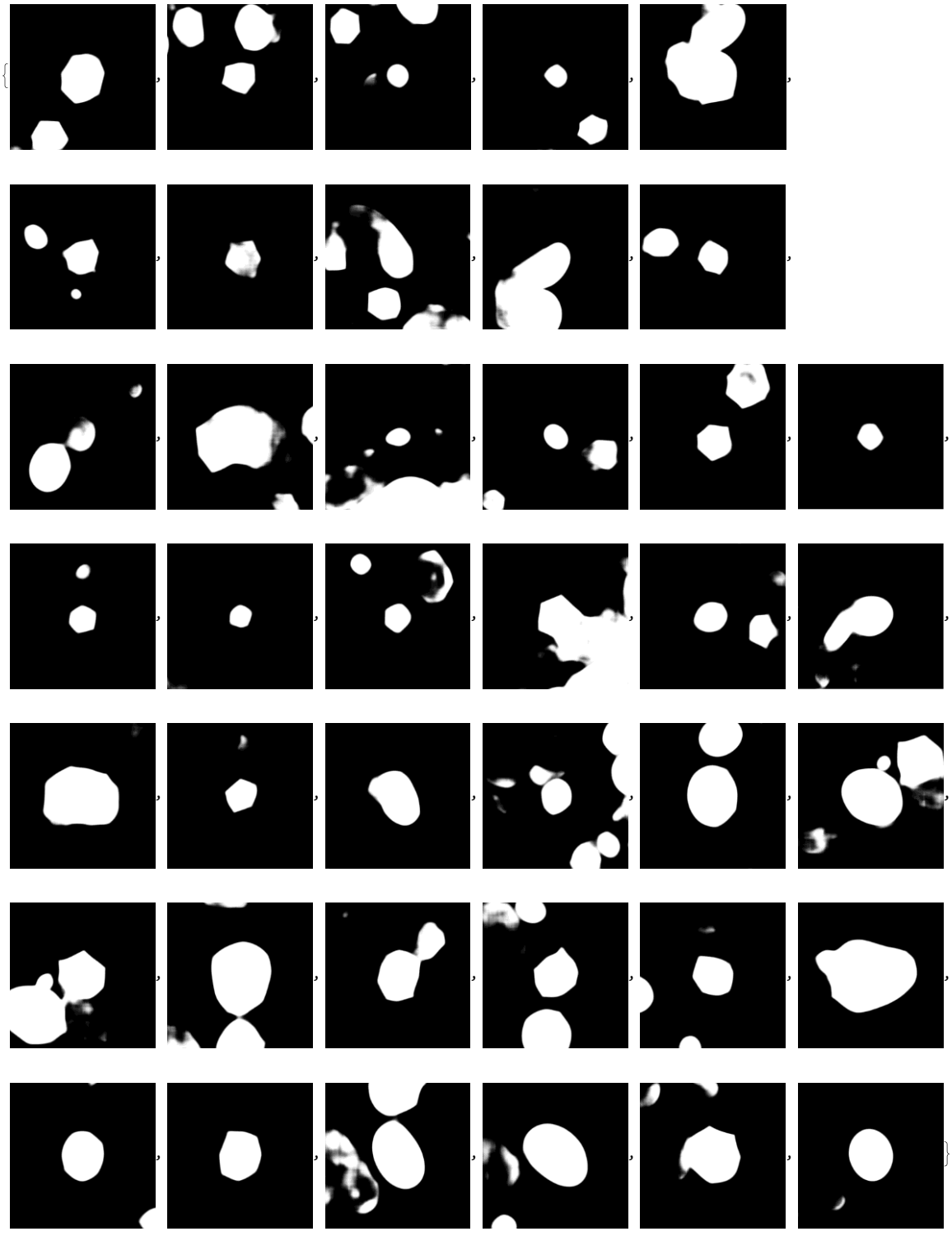
In[•]:=
```
tnpimgs = ImageTake[tpadim, IntegerPart[{#[[2]], #[[2]] + 255}], IntegerPart[{#[[1]], #[[1]] + 255}]] & /@ tnpcoors
```

Out[•]= 

Those can be processed by the SegNet module:

In[•]:=
```
tgpuseg = PdNet[tnpimgs, TargetDevice → {"GPU", 1}]
```

Out[ ]=

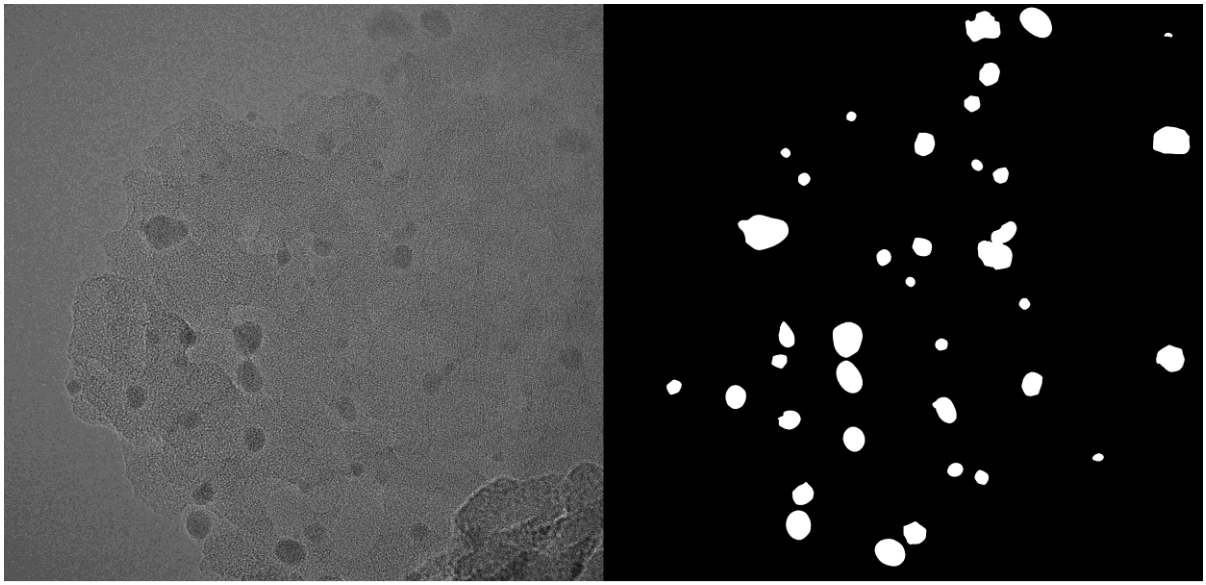


A post-processing routine is applied to reconstruct a 2kx2k binary map.

```
In[•]:= WaterShedSemble[Segs_, Coors_] :=
 Block[{s1, w1, w2, w3, wsegs, wsin, wcent, w4, w5, s2, cent, dels, s2c, centc, nf1, centls, s3, s4, s5, s6, s7, s8,
   coordsmod, coordsmodc, blanc, NPlist, FinIM},
  s1 = Binarize[#, 0.999] & /@ Segs;
  w1 = DistanceTransform[#] & /@ s1;
  w2 = ImageAdjust[#] & /@ w1;
  w3 = MaxDetect[#, 0.05] & /@ w2;
  wsegs = {};
  Do[wsin = ImageData[s1[[k]], "Bit"] × WatershedComponents[ColorNegate@w2[[k]], w3[[k]]];
   AppendTo[wsegs, wsin];, {k, 1, Length@s1}];
  w4 = Colorize /@ wsegs;
  w5 = Binarize /@ w4;
  cent = Values@ComponentMeasurements[#, "Centroid"] & /@ w5;
  dels = Position[cent, {}];
  s2c = Delete[w5, dels];
  centc = Delete[cent, dels];
  nf1 = Function[Nearest[#, {128, 128}]];
  centls = Flatten[nf1 /@ centc, 1];
  s3 = Table[SelectComponents[s2c[[x]], #Centroid == centls[[x]] &], {x, 1, Length@s2c}];
  s4 = Dilation[#, DiskMatrix[7]] & /@ s3;
  s5 = DeleteBorderComponents /@ s4;
  s6 = FillingTransform /@ s5;
  s7 = Erosion[#, DiskMatrix[7]] & /@ s6;
  s8 = Table[SelectComponents[s7[[x]], #Circularity > 0.8 &], {x, 1, Length[s7]}];
  s8 = RemoveBackground /@ s8;
  coordsmod = Table[{Coors[[x, 1]], 2048 - Coors[[x, 2]]}, {x, 1, Length@Coors}];
  coordsmodc = Delete[coordsmod, dels];
  blanc = Image[ConstantArray[0, {2048, 2048}]];
  NPlist = Transpose[{s8, coordsmodc}];
  FinIM = ImageCompose[blanc, #1, #2] &[NPlist[[All, 1]], NPlist[[All, 2]]];
  FinIM = Binarize[FinIM, 0.99];
  FinIM = DeleteBorderComponents[FinIM]];

In[•]:= FullSeg = WaterShedSemble[tgpuseg, tnpcoors];
```

Input/Output comparison:



Set output directory

```
SetDirectory["/myoutputdirectory"]
```

Now we can process the whole stack:

```
NNSegmentation[IMPath_, LbPath_] := Module[{imgs, file, padim, npls, npcoors, npimgs, gpuseg, fullim, imname},
  imgs = FileNames[IMPath <> "*.tif"];
  Do[
   file = DataImporter[IMPath, LbPath, n];
   padim = ImagePad[file[[1]], 128, GrayLevel[0.5]];
   npls = Pick[file[[3]], file[[3, All, 1]], 0];
   npcoors = Floor[Delete[#, 1] & /@ npls * 2048];
   npimgs = ImageTake[padim, IntegerPart[{#[[2]], #[[2]] + 255}], IntegerPart[{#[[1]], #[[1]] + 255}]] & /@ npcoors;
   gpuseg = PdNet[npimgs, TargetDevice → {"GPU", 1}, BatchSize → 8];
   fullim = WaterShedSemble[gpuseg, npcoors];
   imname = "0" <> ToString[n] <> "_NN.jpg";
   Export[imname, fullim];, {n, 1, Length@imgs}];]
```

```
In[•]:= NNSegmentation[ImgDir, ObDectDir]; // AbsoluteTiming
```

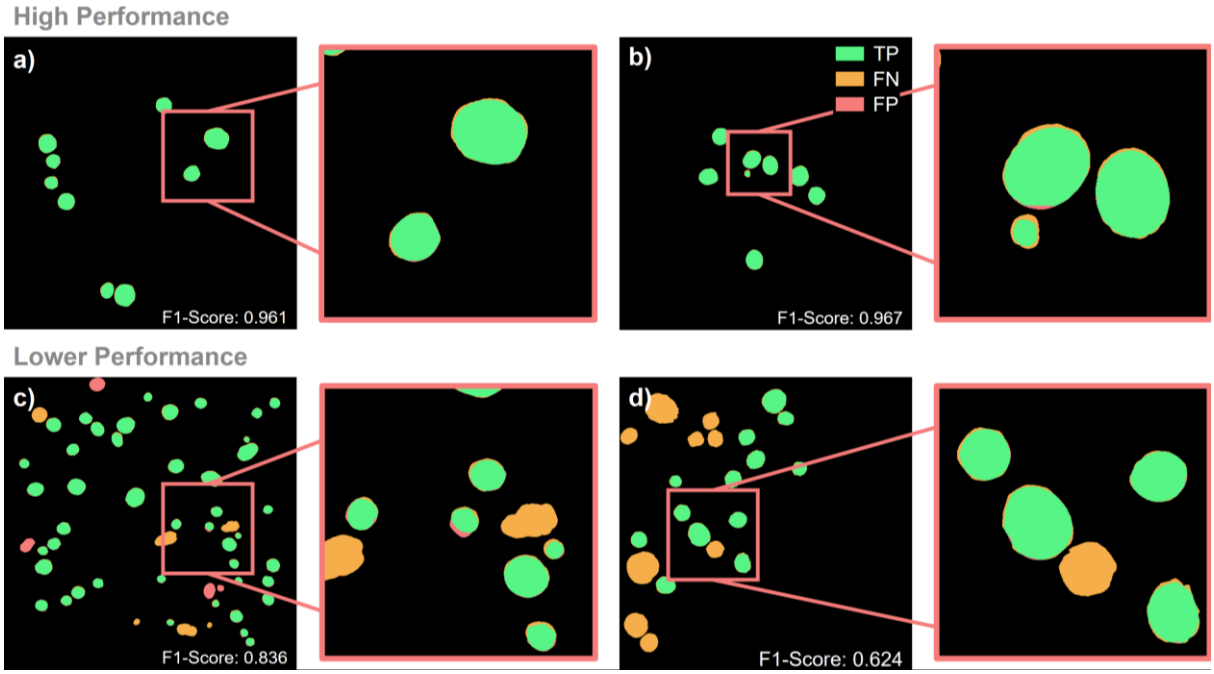


Figure 10.2: Supplement to Figure 5.2. Selected cases of colour-coded segmentation masks for performace assessment of nNPipe. Specific areas of are enlarged to highlight results discussed in the main text.

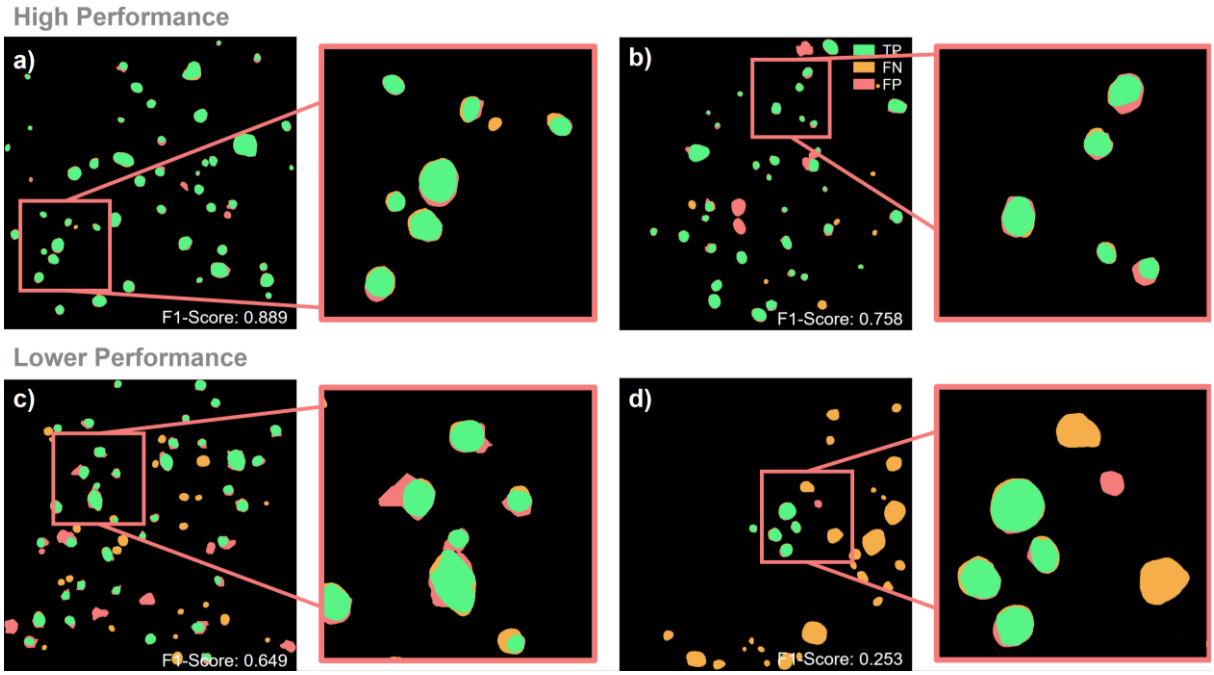


Figure 10.3: Supplement to Figure 5.3. Selected cases of colour-coded segmentation masks for performace assessment of *nNPipe*. Specific areas of are enlarged to highlight results discussed in the main text.

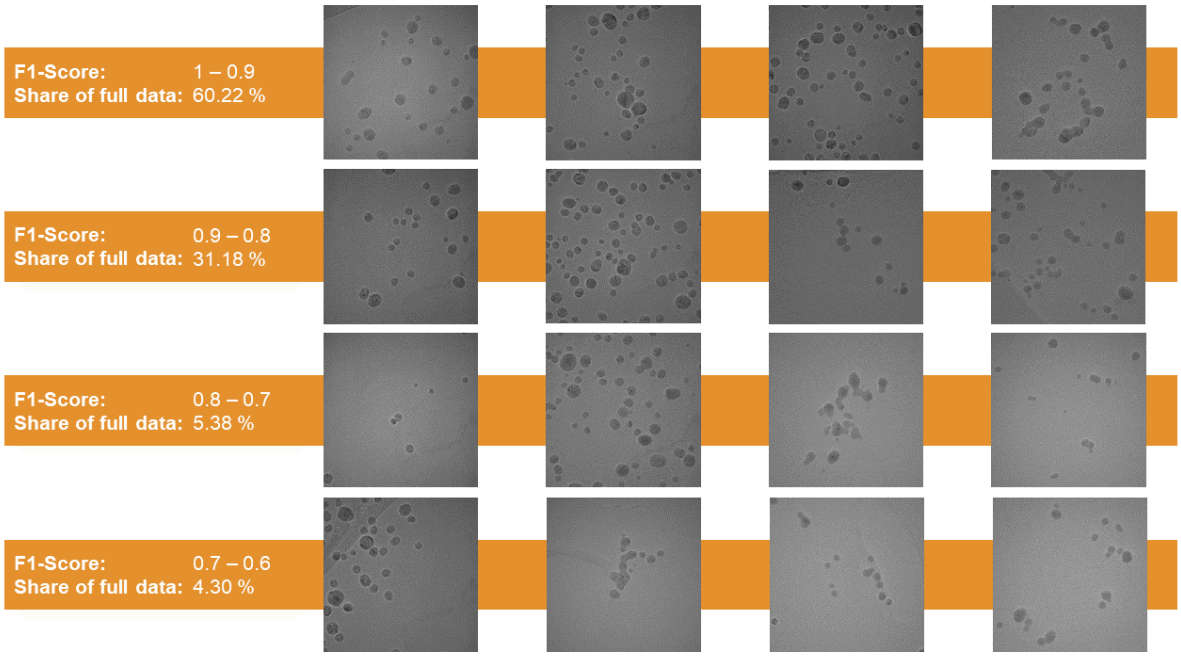


Figure 10.4: Examples of HRTEM images of the Au/Ge dataset that were inferred at different F1-Score intervals with respective shares of the full dataset.

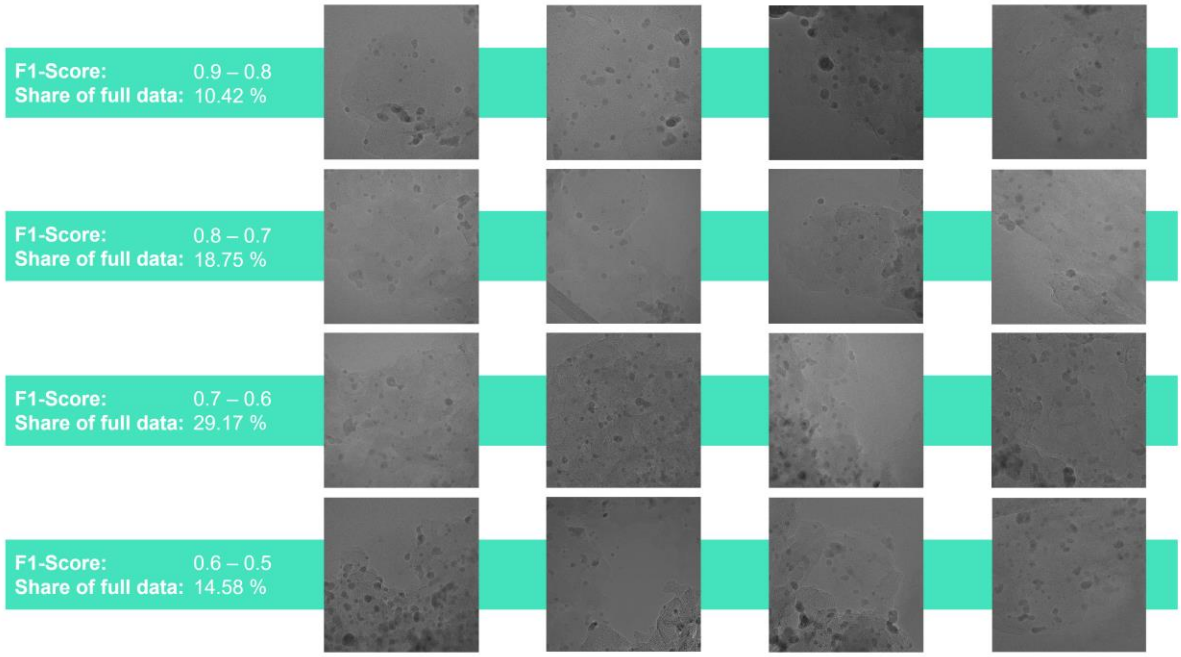


Figure 10.5: Exemplary HRTEM images of the Pd/C dataset inferred at different F1-Score intervals with respective shares of the full dataset.

10.6 Retraining *nNPipe*

Code 10.21: MATLAB code 'Multiply_Seed.m' to generated a 'Seed' for image augmentation, required for YOLOv5x.

```
origin_im_path = 'path2rawdata/raw';
origin_im_dir = dir([origin_im_path, '/*.jpg']);
origin_lb_path = 'path2YOLOlabel/full_yolo_label';
origin_label_dir = dir([origin_lb_path, '/*.txt']);

img_path = 'path/Seed'; %where images shall be stored
label_path = 'path/Seed'; %where YOLO labels shall be stored

sel_vector = []; %Enter index numbers of experimental images randomly selected


for i = sel_vector
  for rot = [0, 90, 180, 270]
    for hf = [0,1]

      img_name = sprintf([origin_im_dir(i).name(1:3), 'R%dHF%d', '.jpg'], rot, hf);
      label_name = sprintf([origin_label_dir(i).name(1:3), 'R%dHF%d', '.txt'], rot,
hf);

      img = imread([origin_im_dir(i).folder, '/', origin_im_dir(i).name]);
      lb = load([origin_label_dir(i).folder, '/', origin_label_dir(i).name]);

      if hf == 0
        %Rotate image and write it with adjusted name.
        cd(img_path)
        tform = affine2d([cosd(rot) sind(rot) 0; -sind(rot) cosd(rot) 0; 0 0 1]);
        rout = affineOutputView(size(img),tform);
        rot_im = imwarp(img, tform,'OutputView',rout);
        imwrite(rot_im, img_name, 'jpg', 'Quality', 100);

        %Rotate label:
        cd(label_path)
        rot_lb = rotyoloLabel(lb, rot, img);
        rot_lb_y = floor([rot_lb(:,1),
rot_lb(:,2)+(rot_lb(:,4)/2),rot_lb(:,3)+(rot_lb(:,5)/2), rot_lb(:,4),
rot_lb(:,5)]);
        rot_lb_y(:,2:end) = rot_lb_y(:,2:end)/2048;

        fileID = fopen(label_name, 'w');
          for j = 1:size(rot_lb_y,1)
            line = rot_lb_y(j,:);
            fprintf(fileID, '%d %f %f %f %f\n', line);
          end
        fclose(fileID);

      elseif hf == 1
        %First, we need to do a horizontal flip.
        hform = affine2d([-1 0 0;0 1 0; 0 0 1]);
        hout = affineOutputView(size(img),hform);
        img_hf = imwarp(img, hform,'OutputView',hout);

        %Now the label
        lb_hf = HFyoloLabel(lb, img);
        lb_hf = floor([lb_hf(:,1),
lb_hf(:,2)+(lb_hf(:,4)/2),lb_hf(:,3)+(lb_hf(:,5)/2), lb_hf(:,4), lb_hf(:,5)]);
        lb_hf(:,2:end) = lb_hf(:,2:end)/2048;

        %Now the same fun as without the horizontal flip

        %Rotate image and write it with adjusted name.
        cd(img_path)
        tform = affine2d([cosd(rot) sind(rot) 0; -sind(rot) cosd(rot) 0; 0 0 1]);
```

```
        rout = affineOutputView(size(img_hf),tform);
        rot_im_hf = imwarp(img_hf, tform,'OutputView',rout);
        imwrite(rot_im_hf, img_name, 'jpg', 'Quality', 100);

        %Rotate label:
        cd(label_path)
        rot_lb_hf = rotyoloLabel(lb_hf, rot, img_hf);
        rot_lb_y_hf = floor([rot_lb_hf(:,1),
rot_lb_hf(:,2)+(rot_lb_hf(:,4)/2),rot_lb_hf(:,3)+(rot_lb_hf(:,5)/2),
rot_lb_hf(:,4), rot_lb_hf(:,5)]);
        rot_lb_y_hf(:,2:end) = rot_lb_y_hf(:,2:end)/2048;

        fileID = fopen(label_name, 'w');
          for k = 1:size(rot_lb_y_hf,1)
            line = rot_lb_y_hf(k,:);
            fprintf(fileID, '%d %f %f %f %f\n', line);
          end
        fclose(fileID);
      end
    end
  end
end
```

Code 10.22: MATLAB code 'Obdect_augmentation.m' for nested augmentation of previously generated Seed.

```
seed_im_path = 'path2Seedimages/Seed';
seed_im_dir = dir([seed_im_path, '/*.jpg']);
seed_lb_path = 'path2Seedlabel/Seed';
seed_lb_dir = dir([seed_lb_path, '/*.txt']);

save_im_path = 'custompath/Augmented';
save_lb_path = 'custompath/Augmented';

%We loop over all images

for idx = 1:size(seed_im_dir,1)
  img = imread([seed_im_dir(idx).folder,'/', seed_im_dir(idx).name]);
  label = load([seed_lb_dir(idx).folder,'/', seed_lb_dir(idx).name]);
  old_name = erase(seed_im_dir(idx).name, '.jpg');
  %First the contrast adjustment. The value 0 has been added to skip
  %this step. Selection is done beforehand to ensure unique values.
  n_space = [0, 15:1:30];
  n = n_space(randperm(length(n_space)));
  for i = n(1:2) %change end number for more augmentations
    if i == 0
      img_s1 = img;
    elseif i ~= 0
      img_s1 = hist_adj_std(img, (i/10));
    end
  %Now Brightness adjustments:
    br_space = [-40:5:40];
    br = br_space(randperm(length(br_space)));
    for j = br(1:5)
      if j < 0
        br_print = sprintf('m%d', abs(j));
      elseif j >= 0
        br_print = sprintf('p%d', abs(j));
      end
      img_s2 = add_br(img_s1, j);
  %The next step adds blurring:
      bl_space = [0, 5:1:25];
      bl = bl_space(randperm(length(bl_space)));
      for k = bl(1:3)
        if k == 0
          img_s3 = img_s2;
```

```
        elseif k ~= 0
          img_s3 = im2uint8(imgaussfilt(img_s2,(k/10)));
        end
   %Lastly, adding dead pixels different levels of dead pixels:
        dp_space = 1:1:15;
        dp = dp_space(randperm(length(dp_space)));
        for l = [0, dp(1)]%:2)]
          img_s4 = add_pepper_noise(img_s3, l/100, 0);

          %Now, saving the image, and respective label files.
          new_img_name = sprintf([old_name,'HST%dBR%sBL%ddP%d.jpg'], i, br_print,
k, l);
          new_lb_name = sprintf([old_name,'HST%dBR%sBL%ddP%d.txt'], i, br_print, k,
l);
          %The imag
          cd(save_im_path)
          imwrite(img_s4, new_img_name, 'jpg', 'Quality', 100);
          %The label
          cd(save_lb_path)

          fileID = fopen(new_lb_name, 'w');
          for ww = 1:size(label,1)
            line = label(ww,:);
            fprintf(fileID, '%d %f %f %f %f\n', line);
          end
          fclose(fileID);
        end
      end
    end
  end
end
```

Code 10.23: MATLAB code 'Crop_augmented_imgs.m' to generate additional image sections of augmented images.

```
%Source data
aug_im_path = 'path2augmentedimages/Augmented';
aug_im_dir = dir([aug_im_path, '/*.jpg']);
aug_lb_path = 'path2augmentedlabels/Augmented';
aug_lb_dir = dir([aug_lb_path, '/*.txt']);

%Saving the data
save_im_path = 'custompath/Cropped';
save_lb_path = 'custompath/Cropped';

parfor i = 1:size(aug_im_dir,1)
    img = imread([aug_im_dir(i).folder,'/', aug_im_dir(i).name]);
    label = load([aug_lb_dir(i).folder,'/', aug_lb_dir(i).name]);

  for j = 1:2
    %Set cropping conditions
    inputSize=size(img);
    px_num = floor((randi([65 95],1)/100)*2048); % Anything between 95% and 65% of
the orignal 2048
    targetSize = [px_num px_num];
    win = randomCropWindow2d(inputSize, targetSize);

    mat_label = bbox_yolo2mat(label, inputSize);
    mat_label_short = mat_label(:,2:end);

    %Crop the img
    img_crp = imcrop(img,win);

    %Crop the label
    winXYWH = [win.XLimits(1) win.YLimits(1) diff(win.XLimits)+1
diff(win.YLimits)+1];
```

```
    [mat_label_crpd,indices] = bboxcrop(mat_label_short,winXYWH,
'OverlapThreshold', 0.4);

    mat_label_crpd = [mat_label(indices,1), mat_label_crpd];
    yolo_lb_crpd = [mat_label_crpd(:,1),
mat_label_crpd(:,2)+(mat_label_crpd(:,4)/2),
mat_label_crpd(:,3)+(mat_label_crpd(:,5)/2), mat_label_crpd(:,4),
mat_label_crpd(:,5)];
    yolo_lb_crpd(:,2:end) = min(yolo_lb_crpd(:,2:end)/px_num,1);

    old_name = erase(aug_im_dir(i).name, '.jpg');
    new_img_name = sprintf([old_name,'_crp%d.jpg'],j);
    new_lb_name =sprintf([old_name,'_crp%d.txt'],j);

    %Write it all.
    cd(save_im_path)
    imwrite(img_crp, new_img_name, 'jpg', 'Quality', 100);
    %The label
    cd(save_lb_path)

    fileID = fopen(new_lb_name, 'w');
    for ww = 1:size(yolo_lb_crpd,1)
      line = yolo_lb_crpd(ww,:);
      fprintf(fileID, '%d %f %f %f %f\n', line);
    end
    fclose(fileID);
  end
end
```

Code 10.24: MATLAB code 'gen_YOLO_gt.m', a pre-processing step to generate a training dataset for SegNet.

```
gt_im_path = 'Large_image_ground_truth/gt_4_retraining_norm';
gt_im_dir = dir([gt_im_path, '/*.tif']);
gt_lb_path = 'path2YOLOlabel/gt/gen_labels';
gt_label_dir = dir([gt_lb_path, '/*.txt']);

img_path = 'path/img'; %Make YOLO labels for gt images
label_path = 'path/label';

sel_vector = []; %Image indexes can be put here

for i = sel_vector
  for rot = [0, 90, 180, 270]
    for hf = [0,1]

      img_name = sprintf([gt_im_dir(i).name(1:3), 'R%dHF%d', '.tif'], rot, hf);
      label_name = sprintf([gt_label_dir(i).name(1:3), 'R%dHF%d', '.txt'], rot,
hf);

      img = imread([gt_im_dir(i).folder, '/', gt_im_dir(i).name]);
      lb = load([gt_label_dir(i).folder, '/', gt_label_dir(i).name]);

      if hf == 0
        %Rotate image and write it with adjusted name.
        cd(img_path)
        tform = affine2d([cosd(rot) sind(rot) 0; -sind(rot) cosd(rot) 0; 0 0 1]);
        rout = affineOutputView(size(img),tform);
        rot_im = imwarp(img, tform,'OutputView',rout);
        imwrite(rot_im, img_name, 'tif', 'Compression', 'none');

        %Rotate label:
        cd(label_path)
        rot_lb = rotyoloLabel(lb, rot, img);
```

```
        rot_lb_y = floor([rot_lb(:,1),
rot_lb(:,2)+(rot_lb(:,4)/2),rot_lb(:,3)+(rot_lb(:,5)/2), rot_lb(:,4),
rot_lb(:,5)]);
        rot_lb_y(:,2:end) = rot_lb_y(:,2:end)/2048;

        fileID = fopen(label_name, 'w');
          for j = 1:size(rot_lb_y,1)
            line = rot_lb_y(j,:);
            fprintf(fileID, '%d %f %f %f %f\n', line);
          end
        fclose(fileID);

      elseif hf == 1
        %First, we need to do a horizontal flip.
        hform = affine2d([-1 0 0;0 1 0; 0 0 1]);
        hout = affineOutputView(size(img),hform);
        img_hf = imwarp(img, hform,'OutputView',hout);

        %Now the label
        lb_hf = HFyoloLabel(lb, img);
        lb_hf = floor([lb_hf(:,1),
lb_hf(:,2)+(lb_hf(:,4)/2),lb_hf(:,3)+(lb_hf(:,5)/2), lb_hf(:,4), lb_hf(:,5)]);
        lb_hf(:,2:end) = lb_hf(:,2:end)/2048;

        %Now the same fun as without the horizontal flip

        %Rotate image and write it with adjusted name.
        cd(img_path)
        tform = affine2d([cosd(rot) sind(rot) 0; -sind(rot) cosd(rot) 0; 0 0 1]);
        rout = affineOutputView(size(img_hf),tform);
        rot_im_hf = imwarp(img_hf, tform,'OutputView',rout);
        imwrite(rot_im_hf, img_name, 'tif', 'Compression', 'none');

        %Rotate label:
        cd(label_path)
        rot_lb_hf = rotyoloLabel(lb_hf, rot, img_hf);
        rot_lb_y_hf = floor([rot_lb_hf(:,1),
rot_lb_hf(:,2)+(rot_lb_hf(:,4)/2),rot_lb_hf(:,3)+(rot_lb_hf(:,5)/2),
rot_lb_hf(:,4), rot_lb_hf(:,5)]);
        rot_lb_y_hf(:,2:end) = rot_lb_y_hf(:,2:end)/2048;

        fileID = fopen(label_name, 'w');
          for k = 1:size(rot_lb_y_hf,1)
            line = rot_lb_y_hf(k,:);
            fprintf(fileID, '%d %f %f %f %f\n', line);
          end
        fclose(fileID);
      end
    end
  end
end
```

Code 10.25: MATLAB code 'Gen_Seg_retrain_dat.m' used to generate training datasets for SegNet retraining.

```
Path to training data from YOLO set
train_path = 'path/yolov5/images/train';%original path to YOLO folder
trainims = dir([train_path, '/*.jpg']);
tracer = [];
for k = 1:size(trainims,1)
check = contains(trainims(k).name, '_crp');
tracer(end+1,1) = check;
end
nocrp_idx = find(tracer < 1);
trainims2k = trainims(nocrp_idx);
```

```
%Path to Ground Truth images
gt_im_path = '/path2_gt_img';
gtimgs = dir([gt_im_path, '/*.tif']);
gt_lb_path = '/path2_gt_YOLO_label'; %previously generated YOLO label for GT
gtlbs = dir([gt_lb_path, '/*.txt']);

%Where to save SegNet data
img_n_path = '/SegNet_files/img_n';
label_path = '/SegNet_files/label';

link_file = cell(size(gtimgs,1),2);

for i = 1:size(gtimgs,1)
  link_file{i,1} = gtimgs(i).name;

  log = [];
  for j = 1:size(trainims2k,1)
    check = contains(trainims2k(j).name, erase(gtimgs(i).name, '.tif'));
    log(end+1,1) = check;
  end
  logidx = find(log);
  sel_train_imgs = trainims2k(logidx);
  link_file{i,2} = sel_train_imgs;
end

parfor k = 1:size(link_file,1)

  if size(link_file{k,2},1) > 0
    gt_IMG = imread([gtimgs(k).folder, '/', link_file{k,1}]);
    gt_IMG_pad = padarray(gt_IMG, [200 200], 0, 'both');
    gt_LB = load([gtlbs(k).folder, '/', gtlbs(k).name]);
    gt_LB_mtlb = [gt_LB(:,1), floor(gt_LB(:,2:end)*2048)];
    gt_LB_rects = floor([gt_LB_mtlb(:,2)-175, gt_LB_mtlb(:,3)-175,
ones(1,size(gt_LB_mtlb,1))'*350, ones(1,size(gt_LB_mtlb,1))'*350]);
    gt_LB_rects_pad = [gt_LB_rects(:,1)+200, gt_LB_rects(:,2)+200,
gt_LB_rects(:,3), gt_LB_rects(:,4)];

    for l = 1:size(link_file{k,2},1)
      train_IMG = imread([link_file{k,2}(l).folder, '/', link_file{k,2}(l).name]);
      train_IMG_pad = padarray(train_IMG, [200 200], floor(mean(train_IMG(:))),
'both');

      for m = 1:size(gt_LB_rects_pad,1)
        gt_Lcrop = imcrop(gt_IMG_pad, gt_LB_rects_pad(m,:));
        train_Lcrop = imcrop(train_IMG_pad, gt_LB_rects_pad(m,:));

        %Set cropping conditions
        inputSize = size(gt_Lcrop);
        targetSize = [256 256];

        for n = 1:2
          win = randomCropWindow2d(inputSize, targetSize);
          gt_crop = imcrop(gt_Lcrop, win);
          gt_crop = mat2gray(gt_crop);
          train_crop = imcrop(train_Lcrop, win);
          train_crop = mat2gray(train_crop);

          img_name = sprintf([img_n_path,'/',
erase(link_file{k,2}(l).name,'.jpg'),'p%d_img_%d.tif'],m,n);
          label_name = sprintf([label_path,'/',
erase(link_file{k,2}(l).name,'.jpg'),'p%d_label_%d.tif'],m,n);

          imwrite(train_crop, img_name, 'tif', 'Compression', 'none');
          imwrite(gt_crop, label_name, 'tif', 'Compression', 'none');
        end
      end

      for o = 1:4
```

```
        targetS_rand = [256 256];
        win_rand = randomCropWindow2d(size(gt_IMG_pad), targetS_rand);

        rand_crop_train = imcrop(train_IMG_pad, win_rand);
        rand_crop_train = mat2gray(rand_crop_train);
        rand_crop_gt = imcrop(gt_IMG_pad, win_rand);
        rand_crop_gt = mat2gray(rand_crop_gt);

        if all(rand_crop_gt(:) == 0)
          %write the empty random crop
          img_e_name = sprintf([img_n_path,'/',
erase(link_file{k,2}(l).name,'.jpg'),'_e_img_%d.tif'],o);
          label_e_name = sprintf([label_path,'/',
erase(link_file{k,2}(l).name,'.jpg'),'_e_label_%d.tif'],o);

          imwrite(rand_crop_train, img_e_name, 'tif', 'Compression', 'none');
          imwrite(rand_crop_gt, label_e_name, 'tif', 'Compression', 'none');

        end
      end
    end
  end
end
```

Code 10.26: MATLAB code 'Dataset_trimm.m'. Exemplary code to cut redundant data from generated datasets.

```
fulldat_im_path = 'path2YOLOv5/yolov5/images/train';
fulldat_im_dir = dir([fulldat_im_path, '/*.jpg']);
fulldat_lb_path = 'path2YOLOv5/yolov5/labels/train';
fulldat_lb_dir = dir([fulldat_lb_path, '/*.txt']);

%All previously generated images and labels were initially stored in the
%train folder to adjust it to the desired dataset size.

trulist = [];
A = cell(size(fulldat_im_dir,1),2);

for i = 1:size(fulldat_im_dir,1)
A{i,1} = erase(fulldat_im_dir(i).name, '.jpg');
A{i,2} = erase(fulldat_lb_dir(i).name, '.txt');
trulist(end+1,:) = isequal(A{i,1}, A{i,2});
end
any(trulist < 1)

num = 4624;
del_numbers = [1:size(fulldat_im_dir)];
del_shuffle = del_numbers(randperm(length(del_numbers)));
del_ls = del_shuffle(1:num);
del_ls = sort(del_ls);

for u = del_ls
jpgfile = [fulldat_im_dir(u).folder, '/', fulldat_im_dir(u).name];
lbfile = [fulldat_lb_dir(u).folder, '/', fulldat_lb_dir(u).name];
delete(jpgfile)
delete(lbfile)
end

fulldat_im_dir2 = dir([fulldat_im_path, '/*.jpg']);
fulldat_lb_dir2 = dir([fulldat_lb_path, '/*.txt']);
```

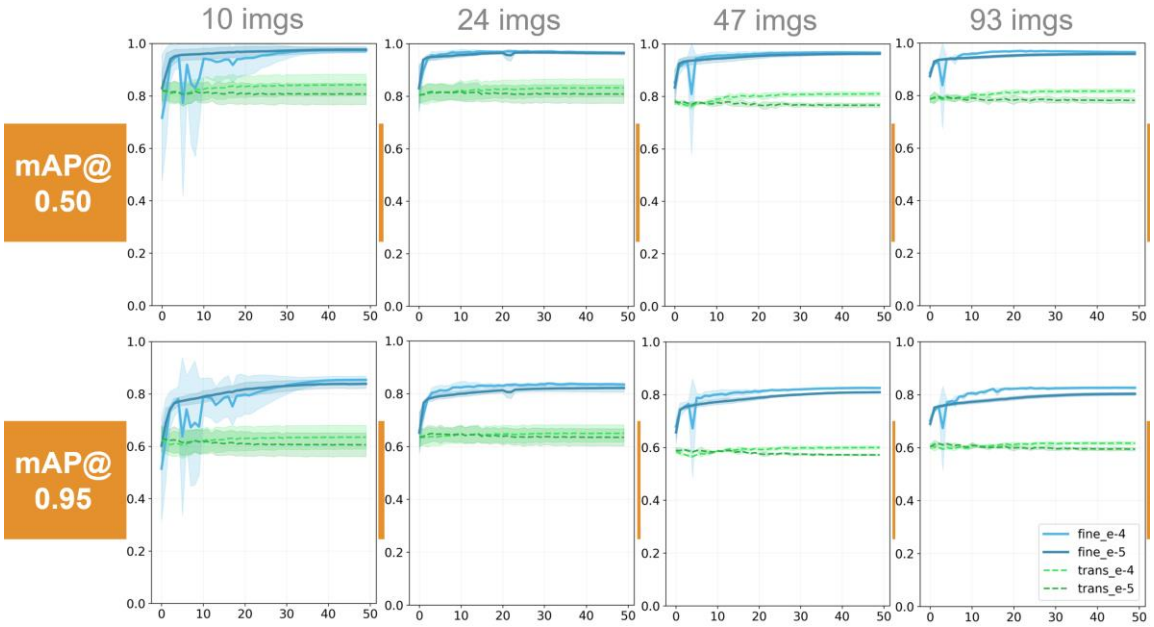


Figure 10.6: Comparison of mAP@0.50 and mAP@0.95 for *transfer learning* and *fine-tuning* at learning rates of 10^{-4} and 10^{-5} for the YOLOv5x module of the Au/Ge *nNPipe* respectively. Line-plots were obtained by calculating the mean values of three sets of randomly selected instances drawn from given experimental subsets. Vertical axis refer to of mAP@0.50 and mAP@0.95 values and the horizontal axis refers to epochs respectively. Although a mean standard deviation is given for each curve, it shall be noted that its significane is limted since it is based on three values per epoch.

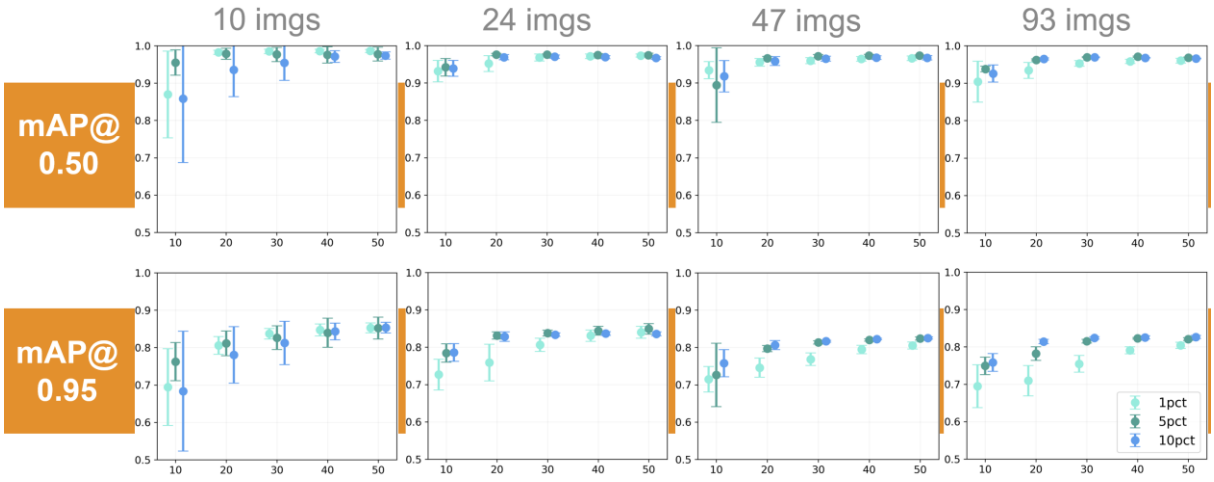


Figure 10.7: Comparison of mAP@0.50 and mAP@0.95 for different Au/Ge experimental subsets at varying degrees of augmentation obtained thorugh *fine-tuning* the YOLOv5x module at a learning rate of 10^{-4}. Each point represents the mean value of three sets of randomly selected instances of a given ESS which were averaged over 10 epochs to obtain a meaningful mean standard deviation. Vertical axis refer to of mAP@0.50 and mAP@0.95 values and the horizontal axis refers to epochs respectively. Invidual points were shifted around each 10 epochs to improve discernibility. The legend shown in the lower right corner holds true for all graphs presented in this figure.

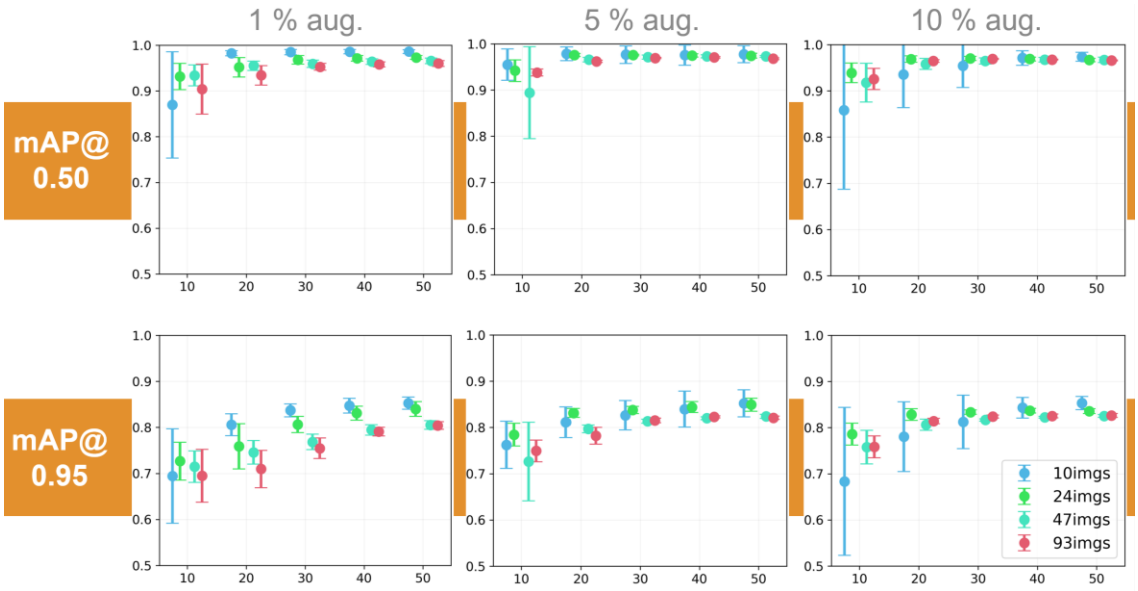


Figure 10.8: Comparison of mAP@0.50 and mAP@0.95 for different Au/Ge degrees of augmentation for varying experimental subsets obtained thorugh *fine-tuning* the YOLOv5x module at a learning rate of 10^{-4}. Each point represents the mean value of three sets of randomly selected instances of a given ESS which were averaged over 10 epochs to obtain a meaningful mean standard deviation. Vertical axis refer to of mAP@0.50 and mAP@0.95 values and the horizontal axis refers to epochs respectively. Invidual points were shifted around respective epochs to improve discernibility. The legend shown in the lorer right corner holds true for all graphs presented in this figure.

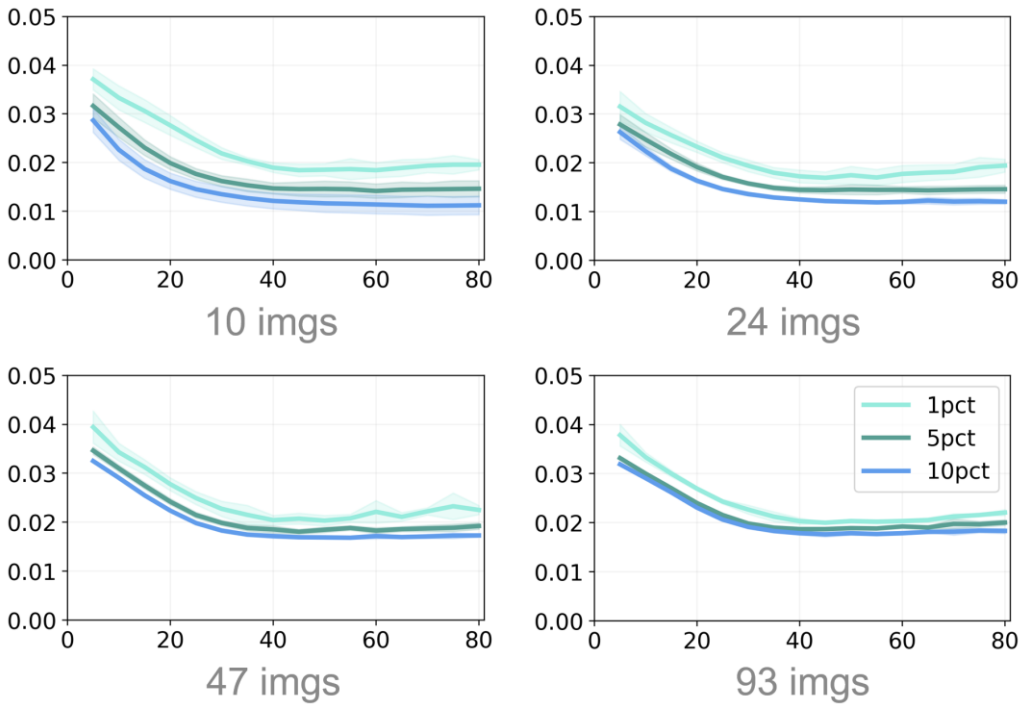


Figure 10.9: Line-plots of Cross-Entropy loss during *fine-tuning* the SegNet module of the Au/Ge *nNPipe* and measured on validation sets, shown for different experimental subsets at varying degrees of augmentation. The vertical axis refers to the Cross-Entropy loss and the horizontal axis to epochs. Each curve was calculated from three sets of randomly selected instances of given experimental subsets which were averaged to result in a mean retraining line-plot. Note that rolling mean standard deviations are included in all plots, but have limited significance since they were calculated on only three curves respectively.

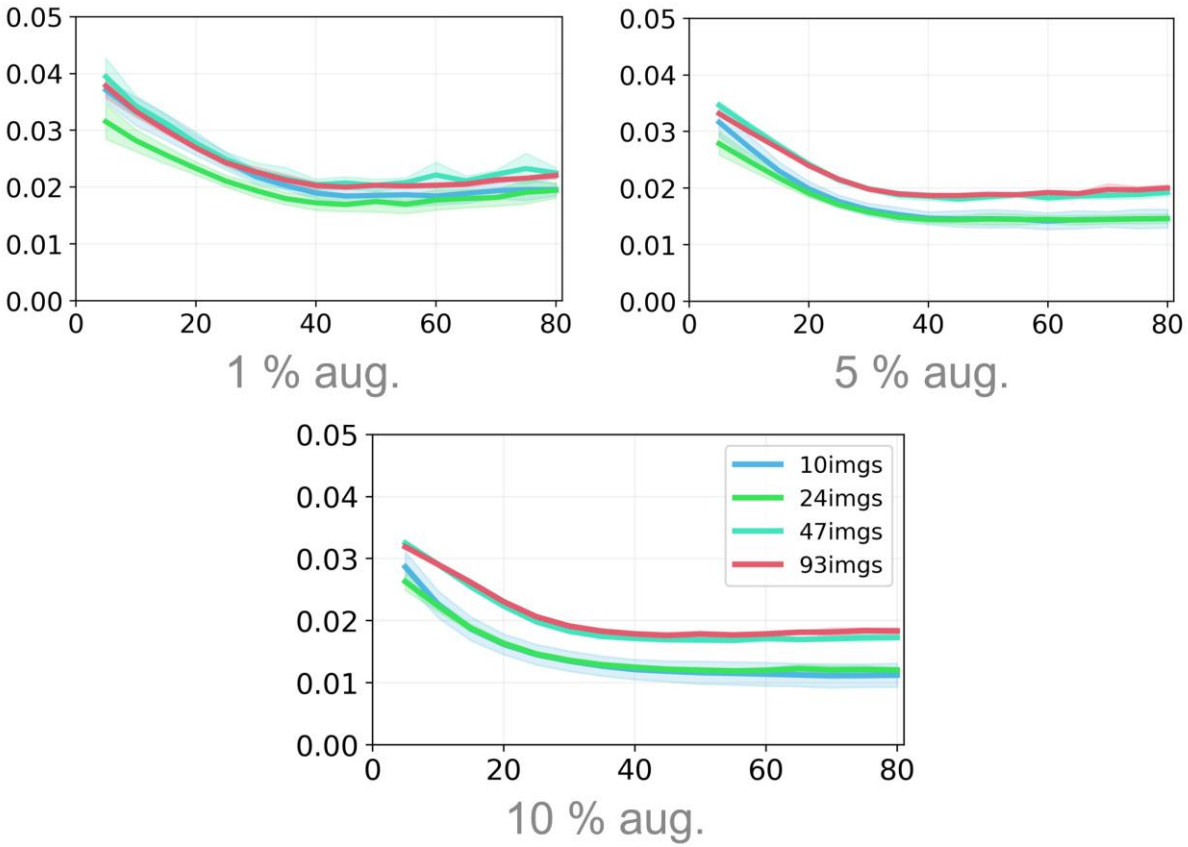


Figure 10.10: Line-plots of Cross-Entropy loss during *fine-tuning* the SegNet module of the Au/Ge *nNPipe* and measured on validation sets, shown for different degrees of augmentation and varying experimental subsets of the Au/Ge dataset. Each curve was calculated from three sets of randomly selected instances of given experimental subsets which were averaged to result in a mean retraining line-plot. Note that rolling mean standard deviations are included in all plots, but have limited significance since they were calculated on only three curves respectively.

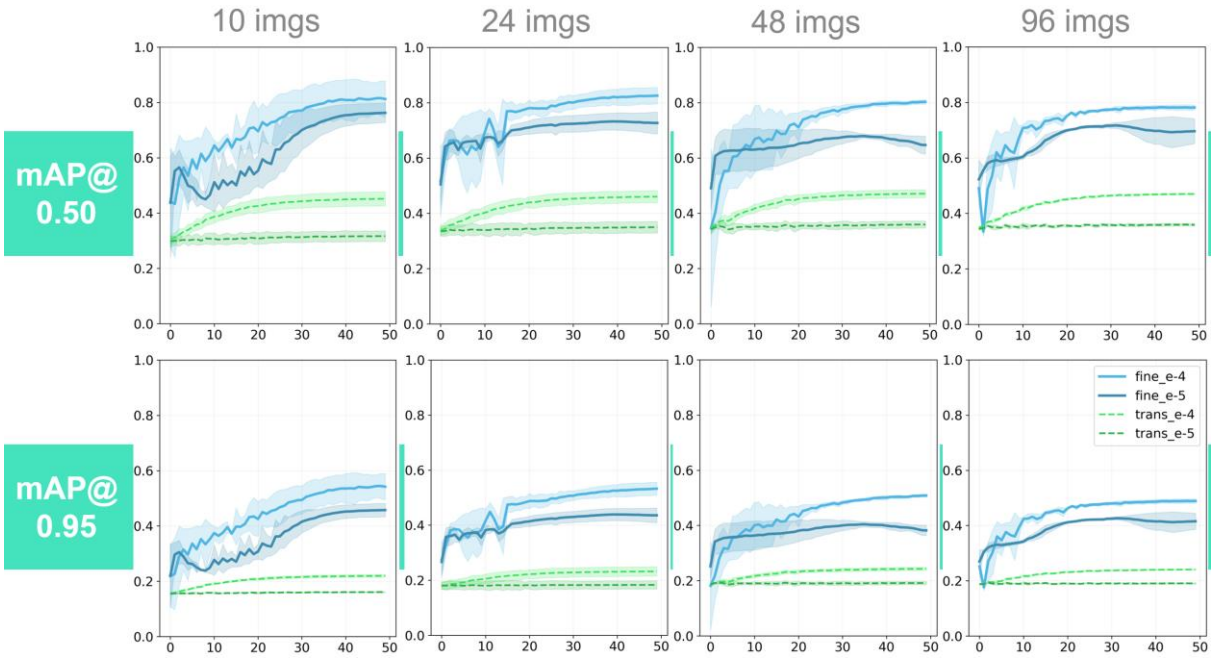


Figure 10.11: Comparison of mAP@0.50 and mAP@0.95 for *transfer learning* and *fine-tuning* at learning rates of 10^{-4} and 10^{-5} for the YOLOv5x module of the Pd/C *nNPipe* respectively. Line-plots were obtained by calculating the mean values of three sets of randomly selected instances drawn from given experimental subsets. Vertical axis refer to of mAP@0.50 and mAP@0.95 values and the horizontal axis refers to epochs respectively. Although a mean standard deviation is given for each curve, it shall be noted that its significane is limted since it is based on three values per epoch.

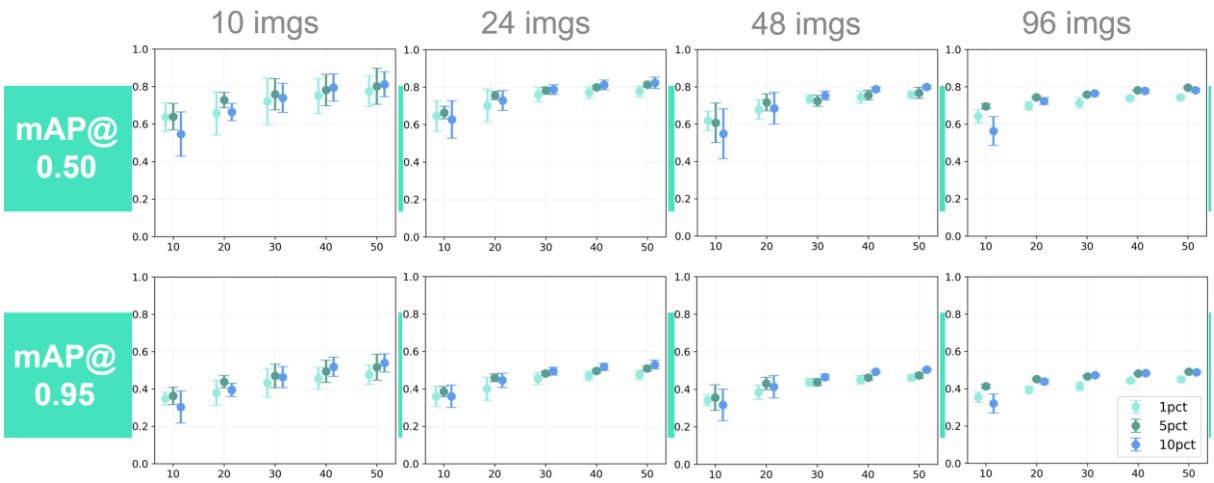


Figure 10.12: Comparison of mAP@0.50 and mAP@0.95 for different Pd/C experimental subsets at varying degrees of augmentation obtained thorugh *fine-tuning* the YOLOv5x module at a learning rate of 10^{-4}. Each point represents the mean value of three sets of randomly selected instances of a given ESS which were averaged over 10 epochs to obtain a meaningful mean standard deviation. Vertical axis refer to of mAP@0.50 and mAP@0.95 values and the horizontal axis refers to epochs respectively. Invidual points were shifted around respective epoch values to improve discernibility. The legend shown in the lower right corner holds true for all graphs presented in this figure.

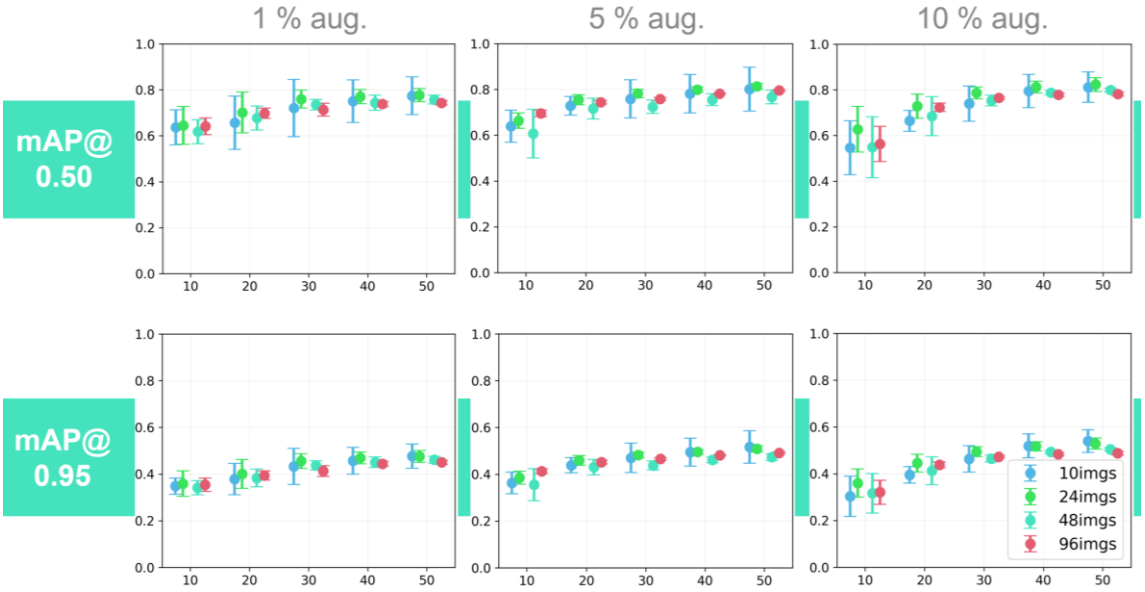


Figure 10.13: Comparison of mAP@0.50 and mAP@0.95 for different Pd/C degrees of augmentation for varying experimental subsets obtained thorugh *fine-tuning* the YOLOv5x module at a learning rate of 10^{-4}. Each point represents the mean value of three sets of randomly selected instances of a given ESS which were averaged over 10 epochs to obtain a meaningful mean standard deviation. Vertical axis refer to of mAP@0.50 and mAP@0.95 values and the horizontal axis refers to epochs respectively. Invidual points were shifted around respective epochs to improve discernibility. The legend shown in the lorer right corner holds true for all graphs presented in this figure.

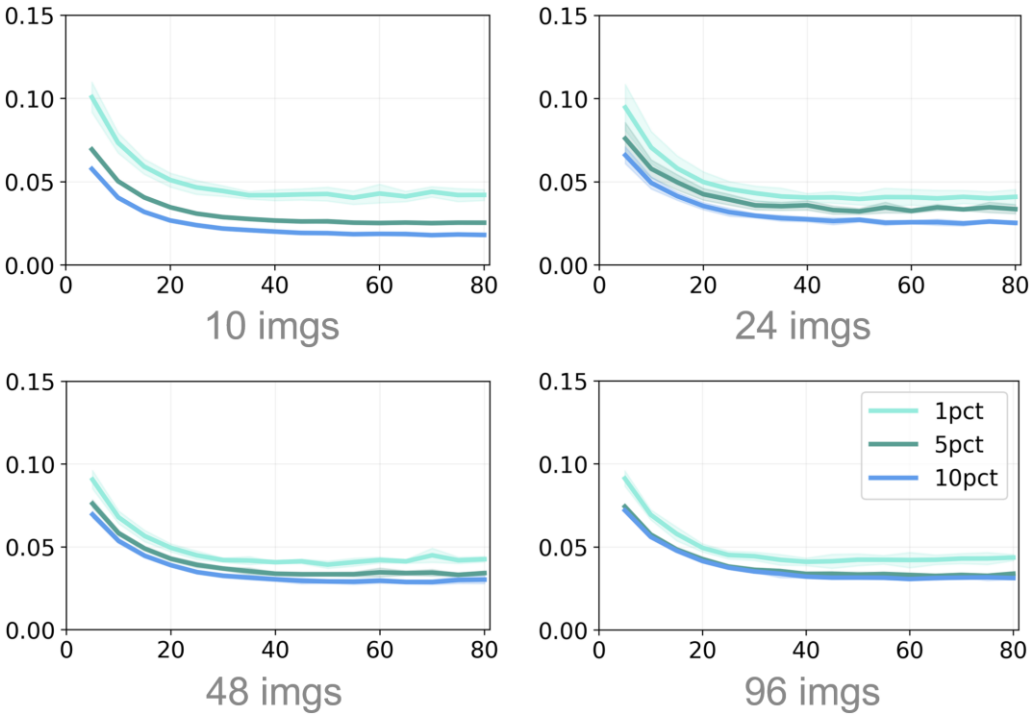


Figure 10.14: Line-plots of Cross-Entropy loss during *fine-tuning* the SegNet module of the Pd/C *nNPipe* and measured on validation sets, shown for different experimental subsets at varying degrees of augmentation. The vertical axis refers to the Cross-Entropy loss and the horizontal axis to epochs. Each curve was calculated from three sets of randomly selected instances of experimental subsets which were averaged to result in a mean retraining line-plot. Note that rolling mean standard deviations are included in all plots, but have limited significance since they were calculated on only three curves respectively.

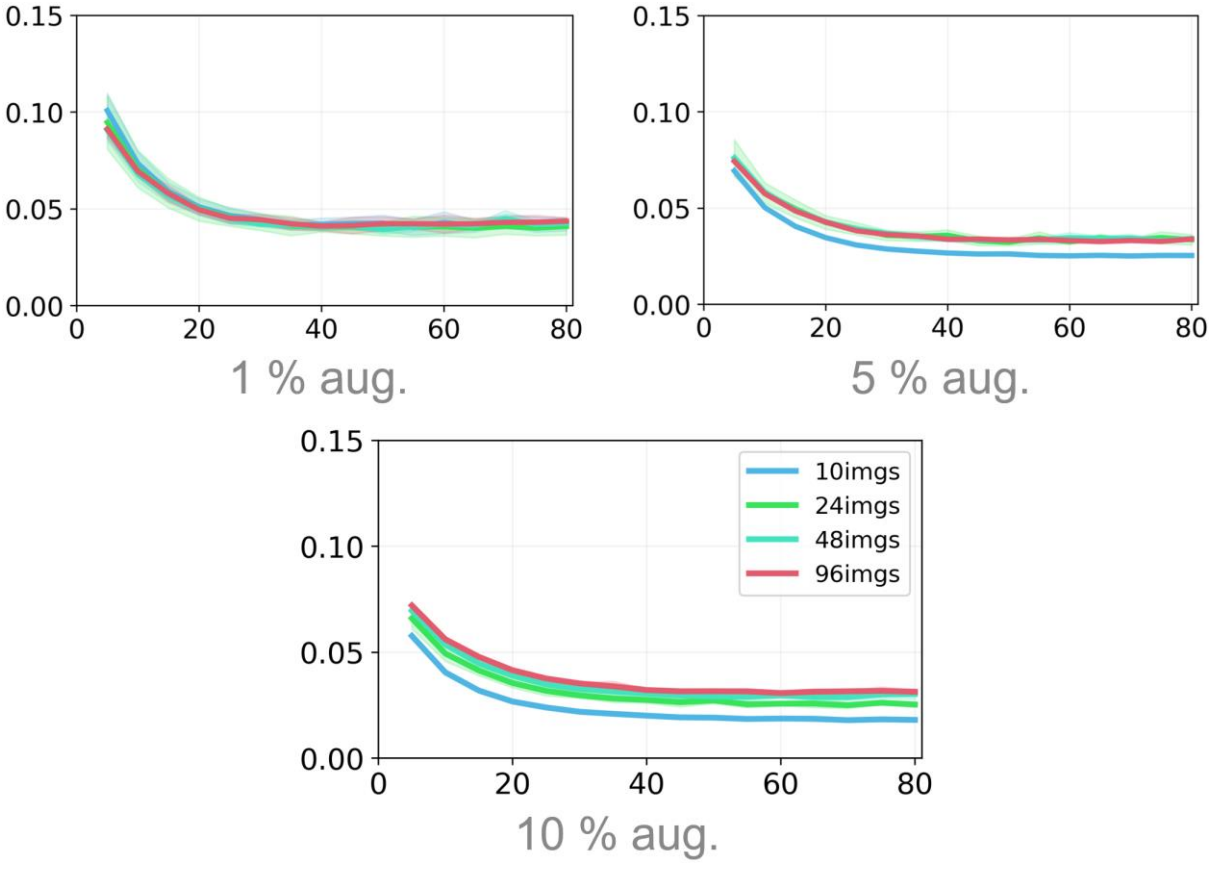


Figure 10.15: Line-plots of Cross-Entropy loss during *fine-tuning* the SegNet module of the Pd/C *nNPipe* and measured on validation sets, shown for different degrees of augmentation and varying experimental subsets of the Pd/C dataset. Each curve was calculated from three sets of randomly selected instances of given experimental subsets which were averaged to result in a mean retraining line-plot. Note that rolling mean standard deviations are included in all plots, but have limited significance since they were calculated on only three curves respectively.

10.7 Democratised Preparation of Training Data

In this section, all relevant code used for the preparation of Zooniverse dataset is deposited. The code has written by Dr. Michele Darrow, Dr. Olly King, Dr. Mark Basham and other collaborators of the Science Scribbler projects. Permission for depositon of respective code has been granted by all three of researchers mentioned above.

Code 10.27: python function ‚image_cropper.py' to prepare cropped datasets. Execution in terminal.

```
#!/usr/bin/env python

import argparse
import logging
import os
import os.path as op
import warnings

import h5py as h5
import numpy as np
import pandas as pd
from skimage.io import imsave
from skimage.util import img_as_ubyte

warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=FutureWarning)

LOGGING_FMT = "%(asctime)s - %(levelname)s - %(message)s"
LOGGING_DATE_FMT = "%d-%b-%y %H:%M:%S"

def init_argparse() -> argparse.ArgumentParser:
  """Custom argument parser for this program.
  Returns:
    argparse.ArgumentParser: An argument parser with the appropriate
    command line args contained within.
  """
  parser = argparse.ArgumentParser(
    usage="%(prog)s [path/to/3d/image/data/file.h5]
[path/to/output/directory] options...",
    description="Generates region images from a volume and fills the
appropriate CSV file with metadata."
  )
  parser.add_argument(
    "-v", "--version", action="version",
    version=f"{parser.prog} version 1.0.0"
  )
  parser.add_argument("input_file_path", metavar='Input file path',
type=str,
            help='the path to a file containing 3d image data.')
  parser.add_argument("output_dir", metavar='Output directory', type=str,
            help='the path to a directory for output.')
  parser.add_argument("--pad", metavar='Pad data flag', type=bool,
            nargs="?", default=True,
            help='Whether to pad the data with a median value.')
  parser.add_argument("--window", metavar='Window size', type=str,
            nargs="?", default="240x264",
```

```
            help='Size of the windows to extract (X,Y in pixels). e.g.
240x264')
  parser.add_argument("--overlap", metavar='Overlap size', type=str,
            nargs="?", default="120x132",
            help='Size of the overlap between windows (X,Y in pixels). e.g.
120x132')
  parser.add_argument("--zrange", metavar='z range', type=str,
            nargs="?", default="start-end",
            help='Range of the central slices to process (#start-#end).
e.g. 100-400')
  parser.add_argument("--zskip", metavar='z skip', type=int,
            nargs="?", default=1,
            help='Number of Z slices to skip.')
  parser.add_argument("--prefix", metavar='Data prefix', type=str,
            nargs="?", default="data",
            help='Name for the dataset and prefix for the naming of
images.')
  parser.add_argument("--fillcsv", metavar='CSV flag', type=bool,
            nargs="?", default=True,
            help='Whether to create a CSV file.')
  parser.add_argument("--csv_filename", metavar='CSV filename', type=str,
            nargs="?", default="manifest.csv",
            help='Filename for the CSV file output.')
  parser.add_argument("--transpose", metavar='Transpose data', type=str,
            nargs="?", default=None,
            help='Order of the axes for data transpose, if required. (e.g.
1,0,2)')
  return parser

def main():
  logging.basicConfig(
    level=logging.INFO, format=LOGGING_FMT,
    datefmt=LOGGING_DATE_FMT)
  parser = init_argparse()
  args = vars(parser.parse_args())
  logging.info('Loading data')
  source = args['input_file_path']
  # load all the data into a numpy array in memory
  with h5.File(source, "r") as f:
    data = f["/data"][()]

  # Transpose if required
  transpose = args['transpose']
  if transpose is not None:
    logging.info('Transposing the data')
    axes = tuple(map(int, transpose.split(',')))
    data = np.transpose(data, axes)

  logging.info(f"Data shape: {data.shape}")
  output = args['output_dir']
  logging.info(f"Creating output directory at {output}.")
  os.makedirs(output, exist_ok=True)

  window = args['window']
  overlap = args['overlap']
  zrange = args['zrange']
  window = tuple(map(int, window.split('x')))
  overlap = tuple(map(int, overlap.split('x')))
  zS, zE = zrange.split('-')
  zS = 0 if zS == 'start' else int(zS)
  zE = data.shape[0] if zE == 'end' else int(zE)
```

```
  # Pad the data
  pad_flag = args['pad']
  if pad_flag:
    # Pad the data
    logging.info("Padding the data")
    median = np.median(data[0, :, :])
    slices = data.shape[0]
    height = data.shape[1] + (2 * overlap[1])
    width = data.shape[2] + (2 * overlap[0])
    padded = np.zeros((slices, height, width))
    for i in range(slices):
      padded[i, :, :] = (np.pad(data[i, :, :],
      ((overlap[1], overlap[1]), (overlap[0], overlap[0])), 'constant',
constant_values=median))

    data = padded
    logging.info(f"Padded Data shape: {data.shape}")

  zskip = args['zskip']
  fillcsv = args['fillcsv']
  prefix = args['prefix']
  csv_list = []
  for z in range(zS, zE, zskip):
    logging.info(f"Processing slice: {z}")
    for y in range(0, data.shape[1], window[1] - overlap[1]):
      yS = min(y, data.shape[1] - window[1]); yE = yS + window[1]
      for x in range(0, data.shape[2], window[0] - overlap[0]):
        xS = min(x, data.shape[2] - window[0]); xE = xS + window[0]
        roi = data[z, yS:yE, xS:xE]
        fname = '{}_{}_{}x{}x{}x{}.png'.format(prefix, z, xS, xE, yS, yE)
        if np.issubdtype(roi.dtype, np.float):
          roi = img_as_ubyte(roi)
        imsave(op.join(output, fname), roi)
        if fillcsv:
          csv_list.append([fname, prefix, z, xS, xE, yS, yE, source,
transpose])

  csvfile = args['csv_filename']
  if fillcsv:
    columns = ['filename', 'prefix', 'slice', 'xstart', 'xend', 'ystart',
'yend', '#source', '#transpose']
    df = pd.DataFrame(csv_list, columns=columns)
    df.index.name = 'id'
    csvfile = op.join(output, csvfile)
    if op.isfile(csvfile):
      with open(csvfile, 'a') as f:
        df.to_csv(f, header=False)
    else:
      df.to_csv(csvfile)

if __name__ == "__main__":
  main()
```

Code 10.28: Python functions to convert annotated picks into 256x256 image sections for classification workflow.

```
import sys
```

```
import time
import os
import csv
import re
import logging
import json
import pandas as pd
import numpy as np
from skimage.io import imsave
from skimage import io
from skimage.transform import rescale
import os.path as op
import os
from datetime import datetime
import h5py as h5
from os.path import exists
from matplotlib import pyplot as plt
```

Loading the original dataset

In [2]:

```
# load all the data into a numpy array in memory


def load_and_pad_dataset(source, transpose):
  """
  Loads the HDF5 file with the CryoEM data, crops and then pads the
  data
  """

  with h5.File(source, "r") as f:
    data = f["/data"][()]

  if transpose is not None:
    logging.info('Transposing the data')
    axes = tuple(map(int, transpose.split(',')))
    data = np.transpose(data, axes)

  data = data[...].copy()
  # Remove 3 pixels from the top and 3 from the bottom
  # data = data[:, 3:-3, :]

  logging.info('Data shape: {}'.format(data.shape))

  t1, t2 = np.percentile(data, (1, 99))
  data[data < t1] = t1
  data[data > t2] = t2
  data = (data - t1) / (t2 - t1)

  median = np.median(data[0, :, :])
  logging.info("Median: {}".format(median))
  slices = data.shape[0]
  overlap_x, overlap_y = tuple(map(int, OVERLAP.split('x')))
  height = data.shape[1] + (2 * overlap_y)
  width = data.shape[2] + (2 * overlap_x)
  padded = np.zeros((slices, height, width))

  for i in range(slices):
    padded[i, :, :] = (np.pad(data[i, :, :],
     ((overlap_y, overlap_y), (overlap_x, overlap_x)),
```

```
     'constant', constant_values=median))
  logging.info('Padded Data shape: {}'.format(padded.shape))
  return padded
```

In [3]:

```
def save_dataframe_to_csv(dataframe, prefix, output_dir):
  csv_filename = "{}_{}.csv".format(prefix,
datetime.now().strftime("%Y%m%d-%H%M%S"))
  csvfile_path = op.join(output_dir, csv_filename)
  logging.info("Saving {} data to {}".format(prefix, csv_filename))
  dataframe.to_csv(csvfile_path)
  return csvfile_path
```

In [4]:

```
def wrangle_data(dataframe, output_path, database_file_path):
  """
  Takes the data from the Zooniverse export and puts the useful
  information into a data structure for generating images of
  annotated virus particles
  """

  def zipper(*args):
    """Packs args into a tuple"""
    return tuple(arg for arg in args)

  def unpacker(anno_dict_list):
    """Converts a list of annotation dicts into list of coordinate
tuples"""
    return [(anno_dict['x'], anno_dict['y']) for anno_dict in
anno_dict_list]

  # Remove the top row of the dataframe as it contains the plain text
headers
  dataframe = dataframe[1:]
  # Create a new column with the dictionary extracted from a json list
  dataframe['anno'] = (dataframe['annotations']
            .apply(json.loads)
            .apply(lambda x: x[0]['value']))
  # Extract the sub dictionary with the metadata
  dataframe['subj'] = (dataframe['subject_data']
            .apply(json.loads)
            .apply(lambda x: x[list(x.keys())[0]]))

  # Create a column with booleans, True if the classification is from the
first
  # workflow
  dataframe['particle_picker'] = (dataframe['subj']
                  .apply(lambda row: True if row.get('slice') else False))

  # Reduce data down to those rows with classifications from the first
workflow
  reduced = dataframe[dataframe.particle_picker]

  # Create a column with booleans, True if the annotations list is not
empty
  reduced['annotation_given'] = (reduced['anno']
                  .apply(lambda x: True if x else False))

  # Reduce data down to those rows with annotations given
  reduced = reduced[reduced.annotation_given]
```

```
  # Create the parent_data_roi tuple (z, xstart, ystart, xend, yend)
  reduced['parent_data_roi'] = (reduced
                .apply(lambda row: zipper(int(row['subj']['slice']),
                            int(row['subj']['xstart']),
                            int(row['subj']['ystart']),
                            int(row['subj']['xend']),
                            int(row['subj']['yend'])),
                            axis=1))
  # Make a column with a list of (x, y) coordinates from annotations
  reduced['coords'] = reduced.apply(lambda row: unpacker(row['anno']),
axis=1)
  # Combine the classification_id, parent_data_roi tuples and the
coordinates into a tuple
  reduced['roi_coord_tuples'] = (reduced
                .apply(lambda row: zipper(row['classification_id'],
                            row['parent_data_roi'],
                            row['coords'],
                            row['user_name']), axis=1))

  save_dataframe_to_csv(reduced, 'wrangled', output_path)

  # Now check the database file to see if the classifications have already
  # been uploaded, if they have, remove from reduced data if they haven't,
append the new
  # classifications to the database file
  ##database_frame = pd.read_csv(database_file_path,
  ##               names=['id', 'classification_id', 'time'])
  ##previous_ids = database_frame['classification_id']
  # Create boolean column with True when ID was in database file
  ##reduced['uploaded'] = reduced.classification_id.isin(previous_ids)
  # Reduce to just rows with new classification IDs
  ##reduced = reduced[~reduced.uploaded]
  # Append new classification IDs to the file
  ##logging.info("The following classification IDs will be added to the"
  ##       " database file {}".format(reduced.classification_id.tolist()))
  #new_df = pd.DataFrame({'classification_id': reduced.classification_id})
  #new_df['time'] = pd.Timestamp.now(tz='Europe/London')
  #new_df.to_csv(database_file_path, mode='a', header=False)
  # Just return the relavent data for cutting out virus particles
  return reduced['roi_coord_tuples'].tolist()
```

In [5]:

```
def cut_out_and_save(data, data_list, output):
  """
  Given datafile, roi in the datafile and coordinates of the virus particle
  positions, this method cuts out a window containing the particle,
upscales
  it and saves it as an image file

  Args:
    data: 3D datafile
    data_list: List of tuples with 3 members. Member 1 - classification_id,
    Member 2 - Tuple containing dimensions of the ROI the particle was
    picked from (z, xstart, ystart, xend, yend), Member 3 Coordinates
    of the centre of the particle (x, y).
    output: Path to a directory for image output
  """

  win_xsize, win_ysize = tuple(map(int, PARTICLE_WINDOW.split('x')))
  max_z, max_y, max_x = data.shape
  csv = []
```

```
  image_counter = 0
  headers = ['filename', 'parent_image_roi', 'picked_coordinates',
      'previous_classification_id','particle_roi_z', 'particle_roi_xstart',
      'particle_roi_xend', 'particle_roi_ystart', 'particle_roi_yend',
'user_name']
  # Cut out the data ROI
  for data_tuple in data_list:
    classification_id = data_tuple[0]
    data_roi = data_tuple[1]
    coordinates = data_tuple[2]
    logging.info('Data ROI {}'.format(data_roi))
    z, droi_xstart, droi_ystart, droi_xend, droi_yend = data_roi
    user_name = data_tuple[3]

    # Use the whole slice otherwise will be difficult to cut out particles
at
    # the edge. Add an offset to the particle ROI
    droi = data[z, :, :]
    # Cut out each particle in that data ROI
    for coordinate in coordinates:
      xpos, ypos = coordinate
      xpos = int(round(xpos)) + droi_xstart
      ypos = int(round(ypos)) + droi_ystart

      xstart = xpos - (win_xsize // 2); xend = xpos + (win_xsize // 2)
      ystart = ypos - (win_ysize // 2); yend = ypos + (win_ysize // 2)

      if (xstart < 0 or ystart < 0):
        logging.error('Particle ROI X or Y begins outside data, skipping')
        continue
      if (xend > max_x or yend > max_y):
        logging.error('Particle ROI X or Y ends outside data, skipping')
        continue

      logging.info("Cutting out particle window with xstart {}, xend {}, "
            "ystart {}, yend {}".format(xstart, xend, ystart, yend))
      particle_roi = droi[ystart:yend, xstart:xend]
      fname = '{}_{}_{}x{}x{}x{}.png'.format("particle", z, xstart, xend,
ystart, yend)


      ##This was added by KevT7 to avoid overwriting of crops
      fname_path = output + '\\' + fname
      file_exists = exists(fname_path)
      cntr = 2

      if (file_exists == True):

        while file_exists == True:
          fname_up = fname[:-4] + '_' + str(cntr) +'.png'
          fname_path = output + '\\' + fname_up
          file_exists = exists(fname_path)
          cntr = cntr+1

        fname = fname_up

      output_path = op.join(output, fname)

      ##
      csv.append([fname, data_roi, coordinate, classification_id,
            z, xstart, xend, ystart, yend, user_name])
```

```
      ##upscaled = rescale(particle_roi, 2)*255
      upscaled = particle_roi * 255
      #adding this line to convert to uint8
      upscaled = upscaled.astype(np.uint8)
      imsave(output_path, upscaled)
      image_counter += 1
      logging.info("File saved to {}".format(output_path))

  logging.info("{} images saved to {}".format(image_counter, output))
  return pd.DataFrame(csv, columns=headers)
```

In [6]:

```
h5_dataset_path = 'path2h5file'
source_data = h5_dataset_path + '\\' + h5_file_name

OVERLAP = '342x342'
PARTICLE_WINDOW = '256x256'
outputpath = 'outputpath'
```

In [7]:

```
pad_datastack = load_and_pad_dataset(source_data, '0,2,1')
```

In [8]:

```
plt.figure(figsize = (10,10))
plt.imshow(pad_datastack[1,:,:], cmap='gray')
```

Out[8]:

```
<matplotlib.image.AxesImage at 0x1f1a83a35c8>
```

In [9]:

```
workflow_export = pd.read_csv('path/Raw_Au_Ge_Picker.csv')
```

In [15]:

```
dat_list = wrangle_data(workflow_export, os.getcwd(), os.getcwd())
```

In [17]:

```
manifest = cut_out_and_save(pad_datastack, dat_list, outputpath)
```

In [18]:

```
manifest['#source_data'] = source_data
```

In [19]:

```
manifest_path = save_dataframe_to_csv(manifest, 'manifest', outputpath)
```

In [ ]:

```
    padded_data = load_and_pad_dataset(source_data, transpose)
    manifest = cut_out_and_save(padded_data, data_list, images_subdir)
    manifest['#source_data'] = source_data
    # Save manifest to image output folder
    manifest_path = save_dataframe_to_csv(manifest, 'manifest',
images_subdir)
    subject_set = create_subject_set(project, manifest_path)
    subjects = create_subjects(manifest_path, images_subdir, project)
    subject_set.add(subjects)
    if add_to_workflow:
```

10.8 Live Image Analysis

This section of the Appendix contains pseudo-code to the live imaging analysis setup described in Section 7.1, which was the result of a close collaboration with Dr. Aakash M. Varambhia, Dr. Manfred E. Schuster and Dr. Doğan Özkaya during an internship with Johnson Matthey plc. After this thesis has been submitted, the following pseudo-code will be protected under a time-limited publication embargo agreed between the Unversity of Oxford and Johnson Matthey plc.

Code 10.28: Pseudo code for the live imaging setup

Step 1

GMS Uploader (Version A)

```
#This part is run as a GMS3 script

data_storage = 'C:/path_to_preferred_filefolder/'
img = GetFrontImage() #returns front image (sample view) and its name

i = 0
stop_criterion = PressShiftKey() #returns True if shift key is pressed

while i > 0:
	if stop_criterion == False:
		i = i+1
		new_filename = FrontImageName + asstring(i)
		img = GetFrontImageDocument() #update front image for each frame
		ImageDocumentSavetoFile(img, new_filename, 'tiff', data_storage)

	elseif stop_criterion == True:
		Abort
```

GMS Uploader (Version B)

```
#This part is run as a GMS3 script and uses python integration

img = GetFrontImage()
stop_criterion = PressShiftKey()

while stop_criterion == False:
	img_array = img.GetNumArray.astype(int) #convert DM array to numpy
	img_array = client.process_on_server
	#outputs image or table, defined by client

	#Optional, displaying single result in GMS3:
	img_analysed = DM.CreateImage(img_array)
	img_analysed_name = 'test'
	ShowImage(img_analysed)
```

GMS Listener (Version A)

```
#Part of the event processor

while true:
	try:
		img_list = sort('*/.tif', key=get_key())
		#get_key() function was required to omit windows sorting of files
		img = img_list[-2]
		#recursive processing, imposing a buffer of two files
		delete_img_list(img_list[:-2])
		#earse data in storage folder apart from current ones in list
		PIL_img = ImageOps.grayscale(Image.open(img))
```

```
            #The following allows to run different modules stored on the REST
            #API server, depending on user preferences.

            if object_detection:
                  try:
                        process_image_on_server_OD(PIL_img, address, url)
                  except:
                        pass
            else:
                  try:
                        process_image_on_server_custom(PIL_img, address, url)
                  except:
                        pass

            PIL_img.close() #clear memory, optional, creates overhead

      except:
            pass
```

Step 2

REST API Client (Version A/B)

```
#Defining essential functions

def array_to_url(img, datatype=None):
      if datatype != None:
            img = image.setastype(datatype) #Preference dependent

      PIL_img = Image.fromarray(img)
      buff = BytesIO() #Create memory buffer and sage PIL object virtually
      prefix = b'data:image/png:base64'
      #added as a header as required for decoding on the server
      img_sting = (prefix + base64.b64encode(buff.getvalue()).decode('utf-8'))

      return img_string


#The client is part of the event processor and mainly runs this function:

def process_image_on_server(PIL_img, server_address, url):
      channel = requests.session() #requesting fixed session on webserver
      imsize = 320
      full_address = server_address + url
      PIL_img = PIL_img.resize(imsize, imsize)
      img_array = np.array(PIL_img)
      im_encoded = array_to_url(img_array)
      server_output = server.get(url, im_encoded, 'aspng', steam=True)
      #get function requests module in python on webserver, encoded image is
      #sent as datastring, stream=True to maintan the originally opened session
      server_response = server_output.content

      return server_response

#Lastly the server address and web app url have to be defined

server_address = 'myaddress:8000'
url = 'myurl'
```

Step 3

Web server (Version A/B)

```
#The web server contains the REST API Server, the web app and analysis models.
#Technically, those would be separated.

#At startup of the live imaging setup, the following function is called:

set_global_variable: object_detection_model
object_detection_model = load_particle_model()
#This function warms up the YOLOv5x module with empty arrays and selects a GPU
device. Mainly, a detectMultiBackend() function is used, similar to ultralytics.
This is done to stream individual frame into the same, preallocated memory.

#Defining an inference function:

def inference.predict(model, img, device=0):
       img = torch.from_numpy(img).todevice(device, dtype.float)
       img = #img/255 #ensuring correct scaling

       pred = model(img)
       pred = non_max_suppression(pred, conf_thres=0.5, iou_thresh=0.45)
       #Running non-maximum suppression over inferred bounding boxes. This
       #variable outputs a txt file with bounding box information.

       boxdf=extract_box_data()
       #Customised function to convert txt file into a pandas dataframe

       return boxdf

#REST API Server:

@server.route('link_to_GMS_Uploader', Methods['POST', 'GET])
#This defines the url to which the server listens, using the status 'GET' (shows
if server works, i.e. code 200, 404, etc.). The status 'POST' can be used to get
post results. Depending on the status, users can decide on whether results shall
be displayed.

def image_string_to_array(im_string):
       img_array =
np.array(Image.open(BytesIO(base64.b64decode(im_string[22:]))))
#This function decodes the image string with the 22 character header removed. The
image is read into memory.

def fetch_process_dat():
       r = request
       img = data_io.image_string_to_array(r.data)
       img = preprocess.bytescale(img) #rescale to [0, 255]

       #Optional:
       Shared_MEDICT['image'] = img
       #This function allows users to access the shared virtual memory of the
       #websever and web app and is important for general functionality.

       boxesdf = inference.predict(object_detection_model, img)
       #Generates dataframe of bounding boxes.

       img_overlay = annotate_image(img, boxesdf)
       #This part is processed on the web app. The bounding boxes are
       #superimposed on the virtually accessible image from shared memory,
       #effectively creating the object detection output as displayed in the
       #web app.
```



Wilhelm Conrad Röntgen, *ca.* 1900.

This scientist is not only famous for the discovery of X-ray radiation and winning the first Nobel Prize in Physics in 1901, but also for his beard.

This picture was added to act as a reminder of the Kirkland tradition.

Picture from:
https://de.wikipedia.org/wiki/Wilhelm_Conrad_R%C3%B6ntgen, Access Date: 07.10.2022